Executive Summary

Team SWIM-R has designed and constructed a prototype underwater robot to be used for investigation as well as data taking. Currently underwater robots costs thousands of dollars and are controlled from a large proprietary custom computer over a long heavy tether. SWIM-R is targeted at the amateur hobbyist, boat owners, and underwater researchers. Based on these needs, the SWIM-R prototype is wirelessly controlled from the user’s personal laptop and has been tested at a depth of 15 feet (the depth of Calvin College’s pool). The team spent $1200 out of the original budget of $1460 for this project and delivered a working prototype on time and under budget. A total of 1700 design hours were accumulated amongst the four members of Team SWIM-R through the two semesters of work, which is near the team’s original estimate of 1550 hours. During the Calvin College Senior Design Open House on May 4, 2013 the team demonstrated SWIM-R to hundreds of guests for a total of two hours, one hour more than the original goal for operating time. Demo videos of SWIM-R in the Calvin College pool can be found here¹.

¹ http://www.youtube.com/playlist?list=PLdFfWvWkG89HjXx2iPhQi9UKf3vVwdksOfD
Table of Contents

FINAL DESIGN REPORT .........................................................................................................................................................

Executive Summary ................................................................................................................................................................. ii

Table of Figures ......................................................................................................................................................................... viii

Table of Tables ........................................................................................................................................................................... x

1 Project Overview ................................................................................................................................................................. 1
  1.1 Calvin Senior Design ......................................................................................................................................................... 1
  1.2 Problem Statement ............................................................................................................................................................... 1
  1.3 Project Proposal .................................................................................................................................................................... 1
    1.3.1 Project Overview ........................................................................................................................................................... 1
    1.3.2 Differentiation ................................................................................................................................................................. 2
    1.3.3 Target Customers ............................................................................................................................................................... 3
  1.4 Tailoring to Sponsor Goals ................................................................................................................................................ 3
  1.5 Team Organization ............................................................................................................................................................... 3
    1.5.1 Team Member Bios ........................................................................................................................................................... 3
    1.5.2 Team Member Responsibilities ......................................................................................................................................... 5
    1.5.3 Team Leadership and Management .................................................................................................................................. 5
  1.6 Report Structure ................................................................................................................................................................. 7

2 Technical Description ............................................................................................................................................................. 8
  2.1 Computer ............................................................................................................................................................................... 8
  2.2 Float ..................................................................................................................................................................................... 8
  2.3 Sub ....................................................................................................................................................................................... 8
    2.3.1 Electrical Hardware .......................................................................................................................................................... 8
    2.3.2 Software ........................................................................................................................................................................... 9
    2.3.3 Mechanical .................................................................................................................................................................... 9

3 Requirements ......................................................................................................................................................................... 10
  3.1 System Features ................................................................................................................................................................. 10
  3.2 Customer Requirements ....................................................................................................................................................... 11
    3.2.1 Weight Limit ................................................................................................................................................................. 11
    3.2.2 Size Limit ....................................................................................................................................................................... 11
    3.2.3 Price ................................................................................................................................................................................ 11
    3.2.4 Water Type ................................................................................................................................................................. 12
    3.2.5 Water Proof ............................................................................................................................................................... 12
    3.2.6 Pressure Proof ............................................................................................................................................................ 12
    3.2.7 Removable Ethernet Cable ........................................................................................................................................... 12
  3.3 Mechanical Requirements .................................................................................................................................................... 12
    3.3.1 Water Proof ................................................................................................................................................................. 12
    3.3.2 Pressure Proof ............................................................................................................................................................. 12
    3.3.3 Access ........................................................................................................................................................................... 13
    3.3.4 Passive Ballast ............................................................................................................................................................ 13
    3.3.5 Sampling Containers ................................................................................................................................................... 13
3.3.6 Robotic Arm .......................................................... 13

3.4 Control Requirements .................................................................................. 14
  3.4.1 Basic Motor Control ................................................................. 14
  3.4.2 6-Degrees of Motion ................................................................. 14
  3.4.3 PID Control ........................................................................... 14
  3.4.4 Ambient Water Temperature .................................................. 14
  3.4.5 Compartment Humidity ............................................................ 14
  3.4.6 Return to Surface ................................................................. 14
  3.4.7 Dive Planning ................................................................. 15

3.5 User Interface Requirements ...................................................................... 15
  3.5.1 Inputs .................................................................................. 15
  3.5.2 First Person View ................................................................. 16
  3.5.3 Video Archiving ................................................................. 16
  3.5.4 Data Archiving ................................................................. 17
  3.5.5 Heads-Up Display ................................................................. 17
  3.5.6 Sound .............................................................................. 17
  3.5.7 Mobile Device Control ....................................................... 17

3.6 Communication Requirements .................................................................. 17
  3.6.1 Basic Commands ................................................................. 17
  3.6.2 Live Video Feed ................................................................. 17
  3.6.3 Live Data Streaming ............................................................... 17
  3.6.4 Data Verification ................................................................. 17
  3.6.5 Long Distance Control .......................................................... 18

3.7 Power Requirements ................................................................................. 18
  3.7.1 Battery Life ....................................................................... 18
  3.7.2 Battery Monitoring ............................................................... 18
  3.7.3 Emergency Battery ............................................................... 18
  3.7.4 Trickle Charge ................................................................... 18
  3.7.5 Solar Charge ..................................................................... 18
  3.7.6 “Plug and Play” Batteries ...................................................... 18
  3.7.7 Multiple Battery System ...................................................... 18

3.8 Sensor Requirements ................................................................................. 19
  3.8.1 Positioning ....................................................................... 19
  3.8.2 Depth .............................................................................. 19
  3.8.3 Case Humidity .................................................................. 19
  3.8.4 Ambient Water Temperature ............................................. 19
  3.8.5 Compartment Temperature ................................................ 19
  3.8.6 Proximity ......................................................................... 19
  3.8.7 pH .................................................................................. 19
  3.8.8 Modular Sensors ............................................................... 19
  3.8.9 Pressure Sensor ............................................................... 20

3.9 Environmental Requirements ..................................................................... 20
  3.9.1 Efficient Power Use ............................................................. 20
  3.9.2 RoHS Compliance ............................................................... 20
  3.9.3 Low Vibrations ................................................................. 20
  3.9.4 Wildlife Friendly ............................................................... 20
  3.9.5 Recyclable Materials .......................................................... 20
  3.9.6 Parts Are Fair Trade .......................................................... 20
6.2 Mechanical Hardware Design .................................................................. 43
  6.2.1 Float Enclosure .................................................................................. 43
  6.2.2 Tether and Waterproof Connectors .................................................. 43
6.3 Float Testing ............................................................................................. 44
6.4 Communication System .......................................................................... 45
  6.4.1 Communication from the Computer down to the R-PI .................. 45
  6.4.2 getCaseTemperature() .................................................................... 46
  6.4.3 getWaterTemperature() ................................................................... 46
  6.4.4 getBatteryLife() ................................................................................ 46
  6.4.5 getError() ........................................................................................ 46
  6.4.6 getDepth() ........................................................................................ 46
  6.4.7 getRoll() .......................................................................................... 46
  6.4.8 getPitch() ........................................................................................ 46
  6.4.9 getYaw() .......................................................................................... 46
  6.4.10 Data that is written ......................................................................... 46
  6.4.11 Communication from the R-PI to the Arduino ............................ 47
  6.4.12 R-PI Reading data from the Arduino ............................................ 48
  6.4.13 Connection Handling ...................................................................... 48
7 Sub ............................................................................................................. 51
  7.1 Hardware Design ................................................................................... 51
    7.1.1 Development Boards ....................................................................... 52
    7.1.2 Sensors ........................................................................................... 53
    7.1.3 Motors ............................................................................................ 56
    7.1.4 Camera ........................................................................................... 60
    7.1.5 Testing and Prototyping ................................................................. 61
    7.1.6 Power System ................................................................................ 62
  7.2 Software Design ...................................................................................... 65
    7.2.1 Raspberry Pi Software .................................................................... 65
    7.2.2 Arduino Software ............................................................................ 66
  7.3 Mechanical Design .................................................................................. 70
    7.3.1 Sub Hull .......................................................................................... 70
    7.3.2 Access Port ..................................................................................... 73
    7.3.3 Waterproof Ethernet Port ............................................................... 73
    7.3.4 Ballast ............................................................................................ 73
8 Prototypes ................................................................................................... 74
  8.1 SWIM-R 0.1 ........................................................................................... 74
    8.1.1 System ........................................................................................... 74
    8.1.2 Hardware ......................................................................................... 74
    8.1.3 Software ........................................................................................ 75
    8.1.4 Mechanical .................................................................................... 76
  8.2 SWIM-R 0.2 ........................................................................................... 76
    8.2.1 System ........................................................................................... 76
    8.2.2 Hardware ......................................................................................... 76
    8.2.3 Software ........................................................................................ 76
    8.2.4 Mechanical .................................................................................... 77
  8.3 SWIM-R 0.3 ........................................................................................... 77
    8.3.1 System ........................................................................................... 77
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.3.2 Hardware</td>
<td>77</td>
</tr>
<tr>
<td>8.3.3 Software</td>
<td>77</td>
</tr>
<tr>
<td>8.3.4 Mechanical</td>
<td>77</td>
</tr>
<tr>
<td>8.4 SWIM-R 1.0</td>
<td>78</td>
</tr>
<tr>
<td>8.4.1 System</td>
<td>78</td>
</tr>
<tr>
<td>8.4.2 Hardware</td>
<td>78</td>
</tr>
<tr>
<td>8.4.3 Software</td>
<td>78</td>
</tr>
<tr>
<td>8.4.4 Mechanical</td>
<td>78</td>
</tr>
<tr>
<td>9 Financial Estimates</td>
<td>80</td>
</tr>
<tr>
<td>9.1 Marketing Study</td>
<td>80</td>
</tr>
<tr>
<td>9.1.1 Competition</td>
<td>80</td>
</tr>
<tr>
<td>9.1.2 Market Survey</td>
<td>80</td>
</tr>
<tr>
<td>9.2 Business Model</td>
<td>81</td>
</tr>
<tr>
<td>9.3 Cost Estimate</td>
<td>81</td>
</tr>
<tr>
<td>9.3.1 Non Recurring Engineering (NRE)</td>
<td>81</td>
</tr>
<tr>
<td>9.3.2 Production</td>
<td>84</td>
</tr>
<tr>
<td>9.3.3 Extraneous Costs</td>
<td>85</td>
</tr>
<tr>
<td>9.3.4 Kit Cost</td>
<td>85</td>
</tr>
<tr>
<td>9.3.5 Break Even Analysis</td>
<td>86</td>
</tr>
<tr>
<td>10 Project Management</td>
<td>87</td>
</tr>
<tr>
<td>10.1 Work Breakdown Schedule Summary</td>
<td>87</td>
</tr>
<tr>
<td>10.1.1 Work Breakdown Schedule Management</td>
<td>87</td>
</tr>
<tr>
<td>10.1.2 Work Breakdown Schedule Description</td>
<td>87</td>
</tr>
<tr>
<td>10.1.3 Deprecation of KanbanFlow</td>
<td>87</td>
</tr>
<tr>
<td>10.2 Effort Estimation</td>
<td>87</td>
</tr>
<tr>
<td>10.3 Project Milestones</td>
<td>88</td>
</tr>
<tr>
<td>11 Conclusion</td>
<td>89</td>
</tr>
<tr>
<td>11.1 Important Accomplishments</td>
<td>89</td>
</tr>
<tr>
<td>11.2 Shortcomings</td>
<td>89</td>
</tr>
<tr>
<td>11.3 Lessons Learned</td>
<td>89</td>
</tr>
<tr>
<td>11.4 Possibility of Future Work</td>
<td>89</td>
</tr>
<tr>
<td>12 Special Thanks</td>
<td>91</td>
</tr>
<tr>
<td>References</td>
<td>92</td>
</tr>
</tbody>
</table>
Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>SWIM-R System Overview</td>
</tr>
<tr>
<td>1-2</td>
<td>Team SWIM-R pictured left to right: Jon, Mike, Mitch, Jeff</td>
</tr>
<tr>
<td>3-1</td>
<td>Feature Hierarchy</td>
</tr>
<tr>
<td>4-1</td>
<td>SWIM-R System Overview</td>
</tr>
<tr>
<td>4-2</td>
<td>Approximation of Sub location with GPS</td>
</tr>
<tr>
<td>4-3</td>
<td>Improved Sub position based on GPS</td>
</tr>
<tr>
<td>4-4</td>
<td>Comparison between SWIM-R and direct tether system</td>
</tr>
<tr>
<td>4-5</td>
<td>Ethernet wiring scheme. On left, original concept. On right, wiring to take advantage of IP webcam</td>
</tr>
<tr>
<td>5-1</td>
<td>Waypoint Input Window</td>
</tr>
<tr>
<td>5-2</td>
<td>Preliminary GUI Design 1</td>
</tr>
<tr>
<td>5-3</td>
<td>Preliminary GUI Design 2</td>
</tr>
<tr>
<td>5-4</td>
<td>Final GUI Design</td>
</tr>
<tr>
<td>6-1</td>
<td>The Float</td>
</tr>
<tr>
<td>6-2</td>
<td>Inside the Float</td>
</tr>
<tr>
<td>6-3</td>
<td>Outside the Float</td>
</tr>
<tr>
<td>6-4</td>
<td>Float Power Regulator</td>
</tr>
<tr>
<td>6-5</td>
<td>Waterproof Ethernet Connector</td>
</tr>
<tr>
<td>6-6</td>
<td>Float Deployed</td>
</tr>
<tr>
<td>6-7</td>
<td>Computer Side communication logic</td>
</tr>
<tr>
<td>6-8</td>
<td>Communication Logic on the R-PI</td>
</tr>
<tr>
<td>7-1</td>
<td>Sub Electrical Hardware</td>
</tr>
<tr>
<td>7-2</td>
<td>ArduIMU+ V3 sensor board</td>
</tr>
<tr>
<td>7-3</td>
<td>9 DOF - Sensor Stick</td>
</tr>
<tr>
<td>7-4</td>
<td>MultiWii sensor board</td>
</tr>
<tr>
<td>7-5</td>
<td>Relative Axes of the Sub</td>
</tr>
<tr>
<td>7-6</td>
<td>Six-Motor Arrangement</td>
</tr>
<tr>
<td>7-7</td>
<td>Magnetically Coupled Drive Concept</td>
</tr>
<tr>
<td>7-8</td>
<td>Magnetically Coupled Drive Propeller</td>
</tr>
<tr>
<td>7-9</td>
<td>Inventor drawing of magnetically coupled propeller prototype</td>
</tr>
<tr>
<td>7-10</td>
<td>Brushed DC motors used in testing</td>
</tr>
<tr>
<td>7-11</td>
<td>Brushless DC Motor Current Draw vs. Throttle</td>
</tr>
</tbody>
</table>
Figure 7-12: Sub Power Regulation Circuit ................................................................. 64
Figure 7-13: Sub Power Regulation Circuit PCB .................................................................. 65
Figure 7-14: Arduino Software Diagram .............................................................................. 66
Figure 7-15: IMU Parsing Code .......................................................................................... 67
Figure 7-16: Data to User Function .................................................................................... 68
Figure 7-17: Incoming IMU Data Function ........................................................................... 69
Figure 7-18: Control System Diagram for Inertial Navigation ............................................. 70
Figure 7-19 Sub Enclosure Inventor Drawing ........................................................................ 71
Figure 7-20: Sub Enclosure .................................................................................................. 72
Figure 7-21 Pressure Sensor Installation ............................................................................. 72
Figure 8-1: H-Bridge Motor Driver Circuit Schematic ......................................................... 75
Figure 8-2: H-Bridge Circuit for SWIM-R 0.1 ...................................................................... 75
Figure 8-3 SWIM-R 1.0 ....................................................................................................... 79
Figure 9-1: Kit Cost Calculations ........................................................................................ 86
Figure 10-1 Project Milestone Chart ................................................................................... 88
Table of Tables

Table 4-1: Data Bandwidth Estimation ................................................................. 29
Table 6-1: Write Instruction Format ................................................................. 47
Table 6-2: A typical instruction packet telling the Arduino to move the Sub forward ........ 48
Table 6-3: Read Instruction Format ................................................................. 48
Table 7-1: Motor Combinations for Six-Motor Design ........................................... 57
Table 7-2: Test results for DC motor water testing ............................................. 62
Table 9-1: Development Materials Cost List ..................................................... 82
Table 9-2: Adjusted Budget .............................................................................. 83
Table 9-3: Non-Budgeted Items ...................................................................... 83
Table 10-1: Breakdown of Worked Hours in First Semester .............................. 87
Table 10-2: Breakdown of Total Hours Worked throughout Project ....................... 88
1 Project Overview

1.1 Calvin Senior Design
Senior Design Project is a two semester class at Calvin College composed of ENGR 339 and ENGR 340, taken by senior engineering students of every concentration. The purpose of the class is to push students into the “real world” by allowing them to choose a project and take it from conception to a working prototype.

1.2 Problem Statement
Robotics platforms are quickly becoming more available, smarter, and better at unlocking areas of the world that were once inaccessible to the average person. One such example is the increasingly popular quad-copter platform. This device allows the user to explore the skies with ease and is inexpensive. The team seeks to push this age of amateur exploration underwater.

Team 12’s project, SWIM-R, fills the void in accessible, affordable aquatic exploration. Additionally, it has applications in the fields of Biology, Environmental Science, and Geology by enabling the user to take data on aquatic ecosystems. Further, SWIM-R can be used by marinas to inspect the underside and propellers of large boats to minimize the need to bring a boat to drydock.

1.3 Project Proposal
The following details the concept for the project.

1.3.1 Project Overview
SWIM-R provides a window into the sea through the eyes of a submersible marine robot. The major functional areas of the SWIM-R system are depicted in Figure 1-1. The main blocks are the Computer, the Float, and the Sub. The Computer portion consists of software that is run on the user’s personal computer. It provides an interface for steering and a video feed from the Sub that allows the user to see where they are going. The Sub is an example of a Remotely Operated Vehicle (ROV) and is the drivable portion of the system. The Float connects these two systems. It enables conversion from long distance control over the air via Wi-Fi to a wired connection to the Sub via Ethernet.
1.3.2 Differentiation

A typical ROV consists of a robotic unit with sensors, cameras and sometimes a robotic arm, all of which enable the user to accomplish various tasks. Additionally, a typical marine ROV is tethered to a base station computer via a long, thick (between 0.25 and 2 inches in diameter\(^2\)) cable that provides power and data to the submersed robot. This tether typically is both expensive and cumbersome due to its thickness and amount of copper used.

The goal of the SWIM-R is to create an affordable, practical means to explore underwater. An example of a “low-cost” ROV that is commercially available is the Seabotix LBV Submersible\(^3\). The vendor website shows a cost of €7495.00 for each system (roughly $10000). The target cost for the SWIM-R (for full scale production and sale) is between $700 and $1500 in order to fit in the price range of the increasingly popular quad-copter platform. A quad-copter is a highly mobile flying unit that opens the skies for hobbyists and photographers. Team SWIM-R seeks to fill this niche in the aquatic environment similar to how the quad-copter has in airborne robotics.

The SWIM-R system provides a window to the largely unexplored aquatic ecosystem. Team SWIM-R accomplishes this goal by allowing the user to use a personal computer such as a laptop or desktop computer to control SWIM-R. This will reduce cost because the conventional, commercial ROV is hard wired (through the tether) to a custom, dedicated computer. If Team SWIM-R allows the user to control


\(^3\) [http://www.nonstopyacht.com/static/browse/session/Product/View_/NUS&2DSBX&2D150.htm](http://www.nonstopyacht.com/static/browse/session/Product/View_/NUS&2DSBX&2D150.htm)
SWIM-R through a device they already own, the product’s cost will be reduced and the learning curve will be lessened by the familiar device.

Additionally, the cost of the SWIM-R system is reduced by powering the Sub using on-board batteries. The alternative to powering SWIM-R with batteries is powering it from a base-station on land. Commercial ROVs are powered with 120-240V AC power which is then converted to DC when it reaches the ROV. This power is achieved either through an AC generator or batteries and an inverter. Powering the ROV with onboard batteries eliminates the need to have an AC generator on site and makes setup and operation of SWIM-R simpler for the user.

1.3.3 Target Customers
The focus of the SWIM-R project is developing an easy to use ROV that hobbyists and small research institutions or colleges could use. The SWIM-R system seeks to make underwater ecosystems more accessible. It will appeal to aquatic research centers or universities because it will allow a smaller institution to conduct underwater exploration without expensive SCUBA equipment/training. The SWIM-R system allows researchers easy access to an otherwise difficult to reach ecosystem.

The SWIM-R system may also appeal to the physically challenged, or those who may lack the ability to snorkel or SCUBA dive. SWIM-R may prove a means to experience the aquatic world first-hand. Pursuing this customer would require additional emphasis on ease of use (e.g. allowing the user to pre-plan a dive) to aid the user.

Additionally, large marinas and boatyards frequently house large yachts that require routine inspection of their hulls. Normally this requires the marina to take the boat out of the water, do the inspection, and put the boat back in (often requiring the use of a large lift or crane). SWIM-R would allow the marina to leave their boats in the water during the inspection, reducing maintenance time and the potential for damaging boats during removal. Pursuing this market would require greater attention to the SWIM-R’s visual capability. For example, in murky water beneath a large boat, the SWIM-R may require lights to illuminate the boat. Additionally, the camera might need to pivot to look up at the boat hull from below. SWIM-R will not be able to conduct repairs under water; however, with the SWIM-R system, a marina will know when a boat needs to be repaired, reducing unnecessary dry-docks.

1.4 Tailoring to Sponsor Goals
DornerWorks has generously agreed to be an industry sponsor for this project, and as such Team SWIM-R would like to tailor the project to a technology that DornerWorks wants to develop. One such technology is the Human Machine Interface (HMI).

Additionally, Team SWIM-R looked for input from DornerWorks for any customers that they have who may be interested in this technology. If there was such a customer, Team SWIM-R planned to adjust the project goals to meet the desires of DornerWorks and their customer.

1.5 Team Organization
Team SWIM-R consists of four senior Engineering students with concentrations in Electrical & Computer Engineering: Mike Capozzoli, Jon Nieboer, Mitch Fynaardt, and Jeff Kloosterman.

1.5.1 Team Member Bios
Figure 1-2 depicts the members of Team SWIM-R
1.5.1.1 Mike Capozzoli
Mike grew up in Oak Brook, IL where he attended Timothy Christian High School. His interests in Electrical and Computer Engineering stem from his interest in building custom PCs as a hobby. During his time at Calvin he has found interest in microcontrollers, programming, and small electronics projects (he recently prototyped a remote starter because he felt his life wasn’t complete without the ability to start his car from his phone). In his free time Mike enjoys computer games and playing drum set with any bands that will have him.

1.5.1.2 Jon Nieboer
Jon was born and raised in the small West Michigan suburb of Cutlerville, Michigan, where he joined the ranks of graduates from South Christian High School. Jon enjoys disassembling electronics to see how they work and is fascinated at how such small components on a circuit board work together in a system. During his time at Calvin he found great interest in software coding and how it is realized by the electrical hardware. In his spare time he enjoys bowling, rock climbing, and hanging out with his fiancé.

1.5.1.3 Mitch Fynaardt
Mitch is originally from Pella, IA where he attended Pella Christian High School. Mitch’s interest in Electrical and Computer Engineering are derived from his natural curiosity of the complexities of things he interacts with daily such as electronic equipment. In his spare time Mitch enjoys playing racquetball, riding his motorcycle, playing video games, and doing Do-It-Yourself home improvement projects such as building a custom sound system with Jeff Kloosterman during their junior year electronic hardware class.

1.5.1.4 Jeff Kloosterman
Jeff hails from Kalamazoo, MI where he attended Kalamazoo Christian High School. Jeff considers his education as an Electrical and Computer Engineer to be both a career and a hobby. As such he enjoys creating small electronics projects (usually involving LEDs) and he is easily tempted by a good deal on a microcontroller (all deals are good deals). Beside technical and hobby electrical work, Jeff enjoys playing video games, drawing when he gets the chance, and cooking up salsa.
1.5.2 Team Member Responsibilities

Each team member is responsible for a distinct part of the project. Tasks are assigned by Jeff based on team members’ strengths and interests. Task assignments are reevaluated on a regular basis to ensure the work load among team members is balanced.

1.5.2.1 Mike Capozzoli

Mike has been responsible for the communication interface between the Graphical User Interface (GUI) application and the rest of SWIM-R. This entails receiving and processing instruction packets to and from the sub. It also includes transmitting a video stream from the Sub to the Computer through the Float. Mike was also responsible for the design of the power regulation circuits in both the Sub and Float.

Mike was assigned this module because he has experience with using wireless protocols in latency and throughput critical applications. He is interested in RF communication and telemetry.

1.5.2.2 Jon Nieboer

Jon was responsible for creating the GUI software application which the user uses to pilot the robot. This task included creating a user friendly control scheme and an aesthetically pleasing, professional user interface. Jon was responsible for receiving video and sensor data sent from the SWIM-R and displaying them to the user.

Jon Nieboer was the best candidate for these tasks due to his prior experience using Microsoft development tools. Jon is the most experienced team member in programming with the Microsoft .NET framework and has experience creating GUIs at his internship.

1.5.2.3 Mitch Fynaardt

Mitch was responsible for the design of the Sub control system as well as reading the Sub’s sensors. This portion of the project included interfacing an Arduino microcontroller with the Electronic Speed Controllers (ESC) that drive the Sub’s motors. Mitch also interfaced the Arduino board with the chosen sensor board and temperature sensors. Finally, Mitch worked with Mike to develop the serial communication protocol between the two processors in the Sub.

Mitch was assigned these sections due to his interest in the Arduino platform. Additionally, Mitch enjoys programming that has direct tangible outputs such as motor driving and sensor reading. Mitch also assisted Jeff in the construction of the Sub enclosure.

1.5.2.4 Jeff Kloosterman (Project Manager)

Jeff was primarily responsible for designing the Sub’s enclosure. This was done using Autodesk Inventor to give the team an idea of what the system would look like. Jeff also acted as system integrator and thus was required to frequently interact with the other three members of the team to ensure the pieces fit together as expected.

Jeff was chosen for this task because of his experience with 3D CAD software such as Autodesk Inventor. Additionally, Jeff has experience with building physical systems such as his personal quad-copter and his long-board. Jeff acted as system integrator in extension of his role as project manager.

1.5.3 Team Leadership and Management

Team leadership and management is an integral part of this project. Team SWIM-R is supported by a group of mentors, advisors, and other personnel.
1.5.3.1 **Project Manager**
Jeff acted as the Project Manager for Team SWIM-R. Jeff had the responsibility of coordinating the efforts of the team, ensuring that each person was kept productive and no one became overwhelmed. He accomplished this through weekly team meetings (both formal and informal) where each member will discuss his progress, challenges, and concerns. Additionally, he was responsible for keeping the team on schedule. The tool Jeff used for this is KanbanFlow, a web-based task management tool. Jeff was also the team’s liaison to the industry sponsor, DornerWorks, and handled communication with them.

1.5.3.2 **Webmaster**
The second position is the team Webmaster. As part of the Calvin College course, the team was required to publish a website detailing their progress and posting documents and other information. While anyone on the team can make updates to the team website, Jon volunteered to make it primarily his responsibility. Jon enjoys “tinkering” in HTML code and has a knack for website development.

1.5.3.3 **Budget Manager**
Calvin College allocates a budget to each design team according to a team’s projected needs. Mitch was tasked as the primary Budget Manager for Team SWIM-R. Mitch’s responsibilities as Budget Manager included determining the overall project budget, as well as the projected production budget should SWIM-R be mass produced.

1.5.3.4 **Faculty Advisor**
As part of the Senior Design course at Calvin College, each team is assigned a faculty advisor from one of the four concentrations. Professor Steve VanderLeest served as Team SWIM-R’s faculty advisor for this project. Steve’s responsibilities for the project included providing feedback on Team SWIM-R’s project in the form of criticisms, encouragements, enthusiasm, and pushing the team to succeed. Steve also has a wealth of information and experience in the Computer Engineering field.

1.5.3.5 **Mechanical Faculty Consultant**
Ned Nielsen is a Mechanical Engineering professor at Calvin College and was one of the co-professors for the Senior Design course. Ned agreed to meet with Team SWIM-R to address the mechanical considerations of the project.

1.5.3.6 **Industry Consultant**
Tim Theriault is an Engineering Director at GE aviation who graciously took the time to meet with Team SWIM-R to discuss the electrical design process. Tim provided the team with insights from his experience as both a manager and an engineer which proved valuable throughout the project.

1.5.3.7 **Industry Sponsor**
DornerWorks graciously offered to sponsor Team SWIM-R with a donation of $1000 toward the project. David Dorner, President of DornerWorks, is a Calvin College alumnus. Team SWIM-R is very grateful to DornerWorks for providing the financial means necessary to achieve its goals.

1.5.3.8 **Industry Mentor**
DornerWorks also tasked Theo Voss to be a mentor to Team SWIM-R. Theo is a software developer at DornerWorks and a recent graduate from the Engineering program at Calvin College. Theo provided support and suggestions to the team based on his experience as a software developer at DornerWorks and as student at Calvin.
1.6 Report Structure

This report details Team SWIM-R’s development of the SWIM-R system. The document is organized by system components with sections on the Computer (4), Float (5), and Sub (6) as well as the overall system (4). The report also includes a Financial Estimates section (7) detailing the costs foreseen in the production of SWIM-R.
2 Technical Description

The SWIM-R system is designed to make aquatic exploration and inspection easier for the average person. SWIM-R is comprised of three main parts: the Computer, the Float, and the Sub.

2.1 Computer

The Computer portion of the SWIM-R system consists of a software program the user runs on their computer. Commands are sent from the Computer to the Sub via the communications Float that is attached to the Sub with a tether. The tether consists of a Cat5 Ethernet cable which ensures the connection to the Float is secure both physically and electronically. The tether carries commands to the Sub’s processor and relays video feed and sensor data to the user on the Computer.

SWIM-R’s Computer program is written in IronPython. IronPython allows the team to leverage the Microsoft .NET framework while still staying rooted in python. A custom control program is needed because of the uniqueness of the SWIM-R system. Programming in IronPython allows for a custom desktop application that is robust, design specific, and platform agnostic. The program features will include a live video stream displayed on the Computer, a heads up display of the Sub’s sensor data and wireless signal strength, and input readings for controlling the Sub.

2.2 Float

The communications Float consists of a flotation device containing a Wi-Fi router and its battery and regulation circuit. The Float relays messages from the Computer to the Sub and vice versa. The Float is connected to the Sub via a tether.

The tether is constructed to provide structural integrity and also to allow the Sub and the Float to communicate. A length of Ethernet cord will provide the communications link in addition to providing structural integrity. The Ethernet connection provides a connection between the Float and the Sub that the Wi-Fi link with the Computer cannot accomplish through water.

2.3 Sub

The Sub contains three distinct components: electrical hardware, software, and structural equipment.

2.3.1 Electrical Hardware

Electrical hardware includes processors, motors, sensors, batteries, programming ports, and battery charging equipment. A Raspberry Pi running a Linux based operating system was selected as the main processor for the Sub. The Raspberry Pi is responsible for handling all of the communications into and out of the Sub. It was also originally responsible for processing the video feed to the Computer but due to high latency this feature was converted to an IP webcam as described in Section Error! Reference source not found.. An Arduino was chosen to be the motor control processor because it has better hardware defined Pulse Wave Modulation (PWM) capabilities than the Raspberry Pi. The Arduino will receive its commands from the Raspberry Pi over USB connection and drive the motors accordingly.

The Arduino will also be responsible for handling the inputs from the Sub’s control and digital sensors. The Arduino possesses analog-in pins which are used for reading the analog temperature sensor as well as the battery voltage. A waterproof Ethernet port is added to the hull of the Sub to allow the processors to be reprogrammed via remote login without disassembling the Sub. The team selected Lithium Polymer (LiPo) batteries for the Sub to provide at least one hour of usable life. Additionally the batteries must be

able to withstand the increased pressure of underwater operation. The team had planned to add passive charging to the Sub to further prolong battery life. The team planned to accomplish this by incorporating a thermocouple charger that uses the heat generated from the motors and motor drivers to generate a current to charge the batteries. However this was complicated by the selection of LiPo batteries which must have their cells balanced while charging. The team determined that the time demands of this feature caused it to fall outside the scope of the project.

2.3.2 Software
The software of the Sub includes sensor reading, motor control, communication, data processing, and video streaming. The Raspberry Pi does most of the heavy lifting when it comes to software. The software running on the Raspberry Pi is written in Python running on the Linux OS. The Raspberry Pi handles the communication with the Float. The remaining software was written for the Arduino platform. Its purpose is to execute the movement commands and read and relay the sensor data.

2.3.3 Mechanical
The mechanical portion of the project will be primarily focused on four things: making the Sub waterproof, providing proper mounting for the motors, building SWIM-R to be neutrally buoyant, and making the hull of the Sub pressure-proof. The team has elected to construct a custom enclosure for the Sub using the Calvin College metal shop. This allowed the team flexibility to tailor the enclosure to our specifications. The enclosure is constructed of clear acrylic and polycarbonate pieces. These materials were chosen because they are lightweight, available, and allow the user and observers to see the electronics inside.
3 Requirements

Team 12 has compiled a list of key features and requirements in order to clarify the goals of SWIM-R. They are divided into nine main categories which are then sorted according to priority of completion. The nine categories are the major divisions of this chapter. Section 3.2 details the requirements that are important for making the system user-friendly such as manageable size and affordable price.

Section 3.3 focuses on mechanical considerations of the system. This deals mainly with the robustness of the Sub’s hull to environmental stress. Section 3.4 lays out what the control system software capabilities shall be. Section 3.5 specifies how the user will interact with the system.

Section 3.6 details how the system components will communicate with each other, how they will check that those communications are correct, and how they will behave if errors are detected. Section 3.7 outlines the constraints on the system’s power distribution network such as battery life and configurability.

Section 3.8 lays out the accuracy standards of the Sub’s environmental sensors. Section 3.9 contains requirements for ensuring that SWIM-R does not negatively affect the environment in which it works. Lastly, section 3.10 discusses the specifics of the deliverables that will be expected from Team 12 upon project completion.

3.1 System Features

Figure 3-1 shows the list of possible features Team SWIM-R compiled at the beginning of the project. The team started with the Essential Features and worked toward the Stretch Features. These categories serve as benchmarks of success in the project and also help to determine what things must be accomplished first in order to allow for further development of the system.
Essential Features are items that if not accomplished will yield SWIM-R non-functional. These items include a waterproof case, the ability to control when the Sub motors turn on and off using simple commands, and a power system of any kind. The absence of any of these features result in project failure.

Core Features slightly expand on the Essential Features and provide the lowest level of a complete system. Examples of these features would be a control scheme that allows the Sub to travel in any direction and live video feed streaming to the user.

Baseline Features make up a majority of the hierarchy. Examples are data archiving, data verification protocols, and basic environmental sensors.

Auxiliary Features add a new level of difficulty to the features added at the Baseline level. Two examples of these are autonomous maneuvering based on the PID system and adding sound to the user interface.

The last category is Stretch Features. These are features that add luxury to the SWIM-R system. Examples of these features include a mobile device version of the control software and a robotic arm for taking environment samples.

3.2 Customer Requirements

Customer requirements detail what is needed for SWIM-R to be user friendly.

3.2.1 Weight Limit

REQ- 1: The SWIM-R Sub shall weigh no more than 50 lbs.

This requirement ensures that the end user shall be able to easily transfer the system in and out of its working environment.

3.2.2 Size Limit

The Sub Length, Width, and Height requirements are in place to define maximum allowable values for each dimension. Team SWIM-R has determined that these maximums shall be suitable for working with in development while still maintaining a size that will keep the Sub from becoming too bulky or awkward for the end-user to handle.

3.2.2.1 Length

REQ- 2: The SWIM-R Sub shall be no longer than 18 inches.

3.2.2.2 Width

REQ- 3: The SWIM-R Sub shall be no wider than 12 inches.

3.2.2.3 Height

REQ- 4: The SWIM-R Sub shall be no taller than 6 inches.

3.2.3 Price

REQ- 5: The production version of the SWIM-R system shall cost no more than $1500.

This requirement refers to the retail price for the end-user. Setting a price cap ensures that design decisions are made in accordance with keeping the SWIM-R system affordable for its target customers. This price was determined based on estimated production costs of each system.
3.2.4 Water Type

REQ-6: The SWIM-R system shall be capable of operating in both salt water and fresh water environments. Due to the wide spectrum of applications of the SWIM-R system it must be designed to maintain its structural integrity in both salt and fresh water environments in order to be considered a legitimate solution for the problem definition.

3.2.5 Water Proof

REQ-7: The Sub and Float shall both be waterproof according to the specification in Requirement 3.3.1.1. In order to provide a complete system it must be able to operate in an underwater environment without compromising its hull integrity to be a satisfactory product.

3.2.6 Pressure Proof

REQ-8: The Sub shall be pressure proof to the depth specified in Requirement 3.3.2. Similar to Requirement 3.2.5, the Sub must be capable of operating within the specified environmental stress limits in order to provide a reliable product to the user.

3.2.7 Removable Ethernet Cable

REQ-9: The Ethernet cable used to connect the Sub and Float shall be removable from both the Float and the Sub. This requirement allows the length of cable used with SWIM-R to be adjustable and shall also facilitate ease of transport since the three components of the system can be disconnected and packed away separately instead of as one unit.

3.3 Mechanical Requirements

This section spells out the constraints on the physical shape and performance of SWIM-R.

3.3.1 Water Proof

The Sub and the Float are both to be waterproof in order to protect the electrical hardware housed inside them.

3.3.1.1 Water Proof

REQ-10: The Float shall be watertight with no need for a precise depth it needs to be able to obtain. The Sub shall be waterproof to a depth greater than the specification detailed in Requirement 3.3.2. The Float is only required to be watertight with no depth specification because it will always be positioned at the surface of the water. The Float’s water tightness will be tested by anchoring it completely submerged in water for twenty-four hours and after that time period it will be inspected for leaks. This is an initial test. If possible it will be tested in a pressure chamber as well. The Sub’s waterproofing will not be able to be physically tested at the specified depth. Instead it shall be designed to meet the specification in Requirement 3.3.2 and it will be tested in a similar manner as the Float at a depth of 14.5 feet because that is the maximum depth of the testing facility: Calvin College’s pool.

3.3.2 Pressure Proof

REQ-11: The Sub shall be pressure proof to a depth of 100 feet. The maximum depth at which SWIM-R is to be operated was determined to be 100 feet because that is the limit for amateur SCUBA diving which is what SWIM-R is attempting to emulate. This limit will not
be physically testable due to the 14.5 foot depth of the testing facility. Like in Requirement 3.3.1.1 the Sub shall be designed and constructed to meet the specification and testing will be done at a depth of 14.5 feet. The test for procedure for pressure proofing is to anchor the Sub at the maximum test facility depth and check for any structural damage at the end of the twenty-four hour period. If possible a pressure chamber test will be performed to better simulate the pressure that would be experienced by the Sub at the maximum operational depth.

3.3.3 Access
REQ- 12: The circuitry and logic systems shall be placed in a compartment that can be opened and closed during maintenance to allow modifications and prototyping.
This compartment’s hatch shall have a watertight seal that will maintain the specifications mentioned in Requirement 3.3.1.1

3.3.3.1 Access
REQ- 13: The Sub’s batteries shall be placed in a compartment that is accessible via a watertight hatch in the Sub’s hull to allow for fast replacement or access for charging.
This hatch shall also be water tight based on the specifications of Requirement 3.3.1.1.

3.3.4 Passive Ballast
REQ- 14: The Sub shall have a ballast compartment which shall be accessible in order to add weight to make the Sub neutrally buoyant.
The ballast compartment shall be an internal volume of the Sub hull specially reserved for this purpose. This ballast compartment shall be filled or emptied by the user as deemed necessary for the target depth of each particular dive and the environment SWIM-R will be operating in, either salt or fresh water. The actual dimensions of this compartment shall be determined once the final Sub hull dimensions have been finalized to make it an appropriate size. These numbers shall be determined and implemented in the design of SWIM-R 0.3 which is scheduled to be completed by April 5, 2013.

3.3.4.1 Automated Ballast Filling System
REQ- 15: The internal volume mentioned in Requirement 3.3.4 shall be filled remotely to adjust buoyancy.
To fill the ballast compartment, a solenoid valve shall be remotely opened by sending an electrical signal to the solenoid which shall allow water to fill the compartment.

3.3.4.2 Automated Ballast Emptying System
REQ- 16: The internal volume mentioned in Requirement 3.3.4 shall be emptied remotely to adjust buoyancy.
To empty the ballast compartment a pump shall be remotely activated by an electrical signal.

3.3.5 Sampling Containers
REQ- 17: The Sub shall have two sample compartments for taking samples of the water.
The exact dimensions of these containers shall be determined based on the final hull dimensions. This is scheduled to be completed at the same time as the ballast compartments discussed in Requirement 3.3.4.

3.3.6 Robotic Arm
REQ- 18: The Sub shall have a static robotic arm mounted at its front.
The robotic appendage on the Sub shall be statically mounted. It shall have a two-finger claw with which to grab samples from the environment. The maximum reach of the arm shall be determined after the hull dimensions are finalized to better calculate the rotational forces that would be exerted by lifting a load in front of the center of mass of the Sub. This shall also be taken into consideration when determining the maximum lifting power of the arm. These calculations are to ensure that operating this arm shall not over-tax the battery in the Sub and shorten the battery life below the limit established in Requirement 3.7.1.

### 3.4 Control Requirements

The control requirements section lays out how the sub shall navigate in its working environment and also how it will be kept stable while operating.

#### 3.4.1 Basic Motor Control

**REQ-19:** The user shall be able to manually control the Sub’s on-board motors.

The specifications of these controls are covered in greater depth in Requirement 3.5.1.1.

#### 3.4.2 6-Degrees of Motion

**REQ-20:** The Sub shall be capable of 3-D independent motion.

This requirement means the Sub shall be able to move up, down, left, right, forward, and backward independently. Also it shall be able to adjust its pitch, roll, and yaw independently.

#### 3.4.3 PID Control

**REQ-21:** The Sub shall have the ability to adjust its attitude to the nearest degree.

This specification is based on Requirement 3.8.1.

#### 3.4.4 Ambient Water Temperature

**REQ-22:** The Sub shall be able to measure the temperature of the water around it to the nearest tenth of a degree.

The temperature sensor shall have to calibrated based on where it is mounted on the Sub. Since it shall be outside of the main hull, heat from the Sub should not affect it. Measurement to the nearest tenth of a degree is believed to be an attainable precision. The actual value shall be dependent on the particular sensor that is purchased for the Sub. This sensor is being ordered this semester so that preliminary testing of its reliability may be done during interim.

#### 3.4.5 Compartment Humidity

**REQ-23:** The Sub shall be able to measure the humidity levels in the interior compartments where electrical components are exposed. This sensor’s accuracy is specified in Requirement 3.8.3.

The electrical components inside the Sub are sensitive to moisture and may become damaged if they become wet. This sensor shall be put in place to ensure that the inside of the Sub does not contain too high a concentration of moisture for the components to function.

#### 3.4.6 Return to Surface

The Sub will be able to automatically return to the surface in several situations. This means that given the conditions of a trigger situation are true, the Sub shall execute a set of commands to bring itself back to the surface.
3.4.6.1 Battery Charge Is Critically Low.

REQ- 24: The Sub shall execute the auto-surface sequence if the battery has been depleted to a level that threatens its ability to be recharged or to provide the Sub with enough power to make the return trip to the surface.

3.4.6.2 Loss of Communication with the User for the Predefined Interval

REQ- 25: The Sub shall execute the auto-surface sequence if it loses communication connection with the user for a predetermined amount of time. It shall also begin to slowly navigate its way back toward the user’s last known location based upon Requirement 3.4.6.2.

Currently the system has a set communications timeout of three seconds after which the Sub halts its motors and searches for the connection again. If the connection is reestablished, the system continues as normal. The return to surface function, while present, is not activated for the safety of the team as they handle the Sub during testing.

3.4.6.3 Sub’s Environment Poses Threat to the Sub

REQ- 26: The Sub shall execute a modified version of the auto-surface sequence if the environment poses a threat to the integrity of the Sub.

If the Sub’s sensors detect that the pressure, temperature, or internal humidity are threatening to the Sub’s integrity it shall begin to auto-surface until the condition is no longer outside of operational levels at which point control shall be restored to the user.

3.4.7 Dive Planning

REQ- 27: The SWIM-R system shall be capable of following a set of pre-programmed waypoints in succession.

3.4.7.1 Dive Plan Interruptions

Dive plans shall be able to be interrupted at any time by sending an override command to the Sub. This shall cancel further execution of the current plan and allow the user to take manual control of the Sub. The plan may be called again and it will resume execution from the point of interruption.

3.5 User Interface Requirements

User interface requirements point out the features of SWIM-R that will directly interact with the user in order to control the operation of SWIM-R

3.5.1 Inputs

The Computer is the user input device. There shall be two methods available for the Computer to collect inputs. Requirements 3.5.1.1 and 3.5.1.2 contain the details for each of these methods.

3.5.1.1 Basic Inputs

REQ- 28: The Computer shall receive input commands from the user via the keyboard in order to send basic motor speed commands to the motor controller via the Float and Sub control unit.

The Computer will be receiving keyboard inputs which it will translate into direction of motor spin and speed values and hand off to the motor controller. The motor controller shall receive this information and determine which motors to activate at which speeds based on the specifications in Requirement 3.8.1. These inputs are basic because the keyboard presses shall indicate a desired direction of travel but will not be capable of indicating speed of travel since keys are either pressed or not with no intermediate values.
The keyboard control system was refined to include pseudo-analog input from the keyboard by incrementing the value based on the length of time a button is held or the number of times a button is pressed.

3.5.1.2 Advanced Inputs

REQ-29: The Computer shall receive input commands from the user via a USB controller in order to send speed and direction commands to the motor controller like in Requirement 3.5.1.1.

The Computer shall receive inputs from a game controller including the direction a control stick is pushed as well as the angle at which it is pushed. It shall interpret these inputs into direction of travel and speed of travel and send them to the motor controller. The motor controller shall take these values and combine them with the values produced by the inertial measurement unit (IMU) specified in 3.8.1 to produce speed and direction values that will drive the motors.

3.5.2 First Person View

REQ-30: The Computer shall display the Sub camera image when requested by the user.

The first person view shall allow the user to see what is in front of the Sub in order to steer it safely in its working environment.

3.5.3 Video Archiving

REQ-31: The Computer shall start storing the incoming Sub video feed when requested by the user.

REQ-32: The Computer shall stop storing the incoming Sub video feed when requested by the user.

REQ-33: The Computer shall store video to its hard drive or an external device specified by the user.

The purpose of this requirement is to provide the user with an optional second means of archiving the video feed captured during system operation.

3.5.3.1 Computer Video Archiving

REQ-34: The storage format of the video archived by the Computer shall be MPEG-4. Video definition shall be 720p at a minimum rate of 30 frames per second.

MPEG-4 was chosen as the storage format because it has a high compression ratio of 50:1\(^5\) which shall allow high quality video to be stored efficiently on the memory media. The definition of 720p and 30 frames/second shall provide a high quality picture without over using precious processing power in the system.

3.5.3.2 Sub Video Archiving

REQ-35: The Sub shall be able to store its video feed to an SD card in 480p at a rate of 15 frames/second in MPEG-4 format.

The Sub shall be constantly archiving video when operating. This will ensure that in the event of a link-loss there will be a video record of what happened to the Sub while the video feed could not be seen by the user at the Computer. The Sub’s archive shall be at a lower definition than the Computer’s because it serves only as a backup.

\(^5\) http://www.kanecomputing.co.uk/pdfs/compression_ratio_rules_of_thumb.pdf
3.5.4 Data Archiving
REQ- 36: The Computer shall be able to store the data from the environment sensors mounted on the Sub to its hard drive.
Sensor data shall be stored in Comma Separated Value (CSV) format at a user specified sample rate. Maximum rate shall be 1 sample/second and minimum shall be no sampling.

3.5.5 Heads-Up Display
REQ- 37: The Computer shall display the current readings from the Sub’s sensors including depth, temperature, pressure, hull internal humidity, and battery life.
Heads-Up Display means that the image from the Sub’s camera shall be the main feature of the screen and the sensor data readings shall be displayed around the edge of the screen to not block the user’s view. This will attempt to mimic a car’s dashboard in performance.

3.5.6 Sound
REQ- 38: The Computer shall be able to reproduce a basic version of the audio produced by the environment around the Sub.
The frequency range that will be reproducible by the Computer will be 10Hz – 20 kHz so that it will cover the hearing range of the average user. The sampling rate will be 44 kHz because this will provide an accurate sample. There will be one microphone attached to SWIM-R. This means that the sound reproduced will be mono rather than stereo.

3.5.7 Mobile Device Control
REQ- 39: The user shall be able to send commands and view the Heads-Up Display of the SWIM-R control interface using an app on either an iPhone or Android device.
The purpose of this mobile app is to make the Computer’s interface more portable for the user.

3.6 Communication Requirements
This section of requirements makes clear what elements of SWIM-R must be capable of communicating with each other and how these communications will be structured.

3.6.1 Basic Commands
REQ- 40: The communication protocol shall reserve a portion of each message for Sub commands.

3.6.2 Live Video Feed
REQ- 41: The Sub shall be able to send live video feed to the Computer.
The live video feed will allow the user to drive the Sub more effectively by seeing where the Sub is heading.

3.6.3 Live Data Streaming
REQ- 42: The Sub shall be able to stream all incoming data from its sensors to the Computer, which will be displayed to the user as specified in Requirement 3.5.5.

3.6.4 Data Verification
REQ- 43: The communications protocol between the Sub and Computer shall have error correction packets and data verification packets which will be used to check the accuracy of the messages being sent between them.
3.6.5 Long Distance Control

**REQ- 44:** The Sub shall allow operation at distances up to 1 mile.

A distance of one mile provides the user with a large area of operation while still ensuring that the wireless components of the system will be capable of communicating back and forth to each other.

3.7 Power Requirements

The power requirements discuss how the electrical infrastructure will be constructed in order to ensure that every sub-system of SWIM-R will be supplied necessary power to operate.

3.7.1 Battery Life

**REQ- 45:** The system shall operate for a minimum of one hour without recharging.

The batteries in SWIM-R (i.e. in Computer, Float, and Sub) must be able to maintain voltages suitable for the systems of each battery.

3.7.2 Battery Monitoring

**REQ- 46:** The system shall display on the Computer the remaining minutes of power to the nearest 5 minutes.

The batteries in SWIM-R will be monitored in order to communicate to the user how much usable life remains in them before recharging is necessary in accordance with Requirement 3.5.5.

**3.7.2.1 Critical Power Operation**

**REQ- 47:** The control systems of SWIM-R shall be capable of recognizing system critical battery levels and ration power accordingly to the most important sub-systems.

3.7.3 Emergency Battery

**REQ- 48:** There shall be a back-up battery for use when the main supply has been depleted to its critical level. This level will be determined once a battery for the Sub is selected before February 25, 2013 which is when the battery and its monitoring system shall be implemented in SWIM-R 0.2.

3.7.4 Trickle Charge

**REQ- 49:** The power system shall support charging from a thermocouple electric generator.

Thermocouple electric generators are supplementary battery charging circuits which use a thermocouple to generate a voltage based on a differential temperature. This voltage forces current to flow through the battery which charges it.^[6]

3.7.5 Solar Charge

**REQ- 50:** The power system shall support charging from a solar panel mounted on the Float.

3.7.6 "Plug and Play" Batteries

**REQ- 51:** The batteries of the Sub will be located within six inches of the waterproof hatch specified in Requirement 3.3.3.1.

3.7.7 Multiple Battery System

**REQ- 52:** The power system of the Sub shall have more than one battery on board.

---

3.7.7.1 Single Battery Use

REQ- 53: The Sub shall draw power from only one battery at a time.
This extra battery will serve a two-fold purpose. First it will serve as a backup in the event that the main battery fails. Second, if no system failures occur, it will allow for longer operational time between charges or battery exchanges.

3.8 Sensor Requirements

Requirements in this section point out the specific environmental properties that SWIM-R will track while operating. Also discussed are the tolerances for each property and how that affects the system.

3.8.1 Positioning
REQ- 54: The Sub shall be able to sense its attitude accurately.

An accurate reading of the Sub’s attitude is required to ensure that the PID system of Requirement 3.4.3 will be useful in controlling the system. The actual required accuracy depends on the IMU sensors that will be incorporated into SWIM-R. These sensors will be purchased during January of 2012 and the accuracy will be determined at that point.

3.8.2 Depth
REQ- 55: The Sub shall be able to sense its depth to an accuracy of 5 feet.

The accuracy requirement of the depth sensor is dependent on the particular sensor that will be included in the system.

3.8.3 Case Humidity
REQ- 56: The Sub shall be able to sense the internal humidity of its enclosure to the nearest percent.

3.8.4 Ambient Water Temperature
REQ- 57: The Sub shall be able to measure the temperature of the water around it to the nearest tenth of a degree.

3.8.5 Compartment Temperature
REQ- 58: The Sub shall be able to measure the temperature of the interior compartments where electrical components are housed to the nearest tenth of a degree.

3.8.6 Proximity
REQ- 59: The Sub shall be able to sense if any objects are within 5 feet that are larger than one cubic inch.

3.8.7 pH
REQ- 60: The Sub shall be able to sense the pH level of the water it is operating in.

3.8.8 Modular Sensors
REQ- 61: The Sub shall possess eight standardized 4-pin I/O ports in order to accommodate modular environment sensors.
3.8.9 Pressure Sensor

3.8.9.1 Sensor Accuracy
REQ- 62: The pressure sensor shall be able to detect the water pressure accurately
This requirement is fully dependent on which sensor is chosen for the system.

3.8.9.2 Safe Pressure Operation

3.8.9.2.1 Warning the User
REQ- 63: The Sub shall detect if it is within one tolerance factor of its pressure sensor from the maximum allowable depth and provide a warning to the user via the controller interface.

3.8.9.2.2 Self-Preservation
REQ- 64: If the pressure sensor detects that the Sub has reached its maximum allowable depth, the Sub shall not allow the user to drive it deeper in order to prevent hull damage.
The prevention of the user control under this circumstance will be implemented in the control software.

3.9 Environmental Requirements
This section contains requirements that pertain to the responsible use and protection of the environment and how that can be especially applied to the SWIM-R project.

3.9.1 Efficient Power Use
REQ- 65: The batteries in SWIM-R (i.e. in Computer, Float, and Sub) shall last no less than one hour under maximum utilization.

3.9.2 RoHS Compliance
REQ- 66: The Sub shall be RoHS compliant.

3.9.3 Low Vibrations
REQ- 67: The Sub shall produce small amounts of vibrations.
Vibration amounts will be kept to a minimum in order to cause as little disturbance to the ecosystem as possible while studying it.

3.9.4 Wildlife Friendly
REQ- 68: The Sub shall not leave behind any trace of substances that will harm the native wildlife in its working environments or cause undue stress to wildlife while being operated.

3.9.5 Recyclable Materials
REQ- 69: The Sub hull shall be constructed out of recyclable materials so that when it has reached the end of its lifecycle it can be safely disposed of in an environmentally friendly manner.

3.9.6 Parts Are Fair Trade
REQ- 70: The Sub shall be made only with parts that are produced in fair trade manufacturing facilities.
3.9.7 Off-Grid Charging
REQ- 71: The Sub shall be able to generate its own power through the means described in Requirements 3.7.4 and 3.7.5. This will allow the Sub to be power grid independent.

3.10 Deliverables
This section of requirements details all of the items that Team SWIM-R will have produced during the course of the project to track progress and report findings. These documents, prototypes, and schematics will be turned in at the end of the year for assessment purposes.

3.10.1 PPFS
REQ- 72: Team SWIM-R shall submit a complete project proposal and feasibility study detailing the problem our project seeks to solve, our proposed solution, and the obstacles that our solution will have to overcome to be successful.

3.10.2 Final Report
REQ- 73: Team SWIM-R shall submit a final report, this document, upon project completion that will explain in detail the problem our project seeks to solve, the research conducted in order to draft a proposed solution, our proposed solution, the obstacles our project will have to overcome, the prototyping done in order to build up to a final solution, the testing of the prototypes, the results of our testing, the conclusions drawn from the testing, the adjustments made to the design, and the final design.

3.10.3 Working Prototype
REQ- 74: Team SWIM-R shall demonstrate a final working prototype satisfying the above requirements.

3.10.4 Design Notebooks
REQ- 75: Each member of Team SWIM-R shall submit to the faculty advisor their personal Dropbox folders and Google Drive documents which contain their individual contributions to the project.

3.10.5 Team Website
REQ- 76: Team SWIM-R shall publish a website that will be maintained as a means of allowing interested persons to keep updated on project status.

The team website can be found at: http://www.calvin.edu/academic/engineering/2012-13-team12/

REQ- 77: Team SWIM-R shall publish the final version of the PPFS and Final Design Report on the website, making each available for download as a single file in PDF format.

3.10.6 System Control Software
REQ- 78: Team SWIM-R will submit their SWIM-R system control software with their final report.

3.10.7 Circuit Schematics
REQ- 79: Team SWIM-R will submit any custom circuit schematics that they design and implement in the SWIM-R system.

3.10.8 Mechanical Schematics
REQ- 80: Team SWIM-R will submit their Sub and Float design schematics.
3.10.9 Financial Calculations

REQ- 81: Team SWIM-R will submit all pertinent financial calculations derived in formulating their estimation for production costs of the SWIM-R system.
4 System
This chapter provides a top level view of the SWIM-R project. The chapter includes the Design Norms driving the design process as well as an overview of the various systems in SWIM-R as well as how they interconnect.

4.1 Design Norms
There are several broad goals which Team SWIM-R hoped to achieve with this project. These goals are known as Design Norms and they have driven the decision making process. For this project, Team SWIM-R has selected the Design Norms of Transparency, Stewardship, Integrity, and Justice.

4.1.1 Justice
The Design Norm of Justice states that a design ought to respect the rights and abilities of all people. This Design Norm is important because of the diverse user base all of whom have varying capabilities. SWIM-R allows for a wider range of users to have access to aquatic ecosystems instead of only those who have the ability to SCUBA dive.

4.1.2 Transparency
A Transparent design is one which has a clear purpose to the user and is intuitive for the user. This Design Norm is important to Team SWIM-R because the target user base of SWIM-R is broad. SWIM-R is designed to be accessible to persons of all ability, not just marine biologists or computer programmers. Additionally, Team SWIM-R has published all of its design documents and code to the team website. The team hopes that this open communication of the design will foster more innovation in the development of aquatic robotics.

4.1.3 Stewardship
Engineers are called to Stewardship by making responsible use of the earth’s resources and developing products which are environmentally conscious. Team SWIM-R seeks to produce a product that both uses environmentally friendly materials and does not disturb the ecosystems it is designed to work in. Additionally, the team hopes that by providing better access to aquatic ecosystems, users will consider their own impact on nature.

4.1.4 Integrity
The Design Norm of Integrity requires a complete, reliable system. Team SWIM-R desired to provide a product that can be used with confidence. The team has produced a complete prototype as a part of the Senior Design course.

4.2 Overview
SWIM-R is a part of a class of robots known as Remotely Operated Vehicles (ROVs). The term ROV is general, but is most commonly applied to aquatic systems. As an ROV, SWIM-R requires a human pilot, referred to as the user, to control the system. The SWIM-R system is composed of three parts: the Computer, the Float, and the Sub. The user interacts with the Computer, which communicates with the Float, which in turn communicates with the Sub. The user’s input on the Computer controls the movements of the Sub; the Float is not independently navigable, but rather can be pulled by the Sub via the tethered connection between the two devices. In this report, the tether will be considered as part of the Float and will be discussed in the Float chapter.
Each of the components of SWIM-R will accomplish a specific set of tasks. Figure 4-1 shows the breakdown of the three main components of SWIM-R.

![Diagram of SWIM-R System Overview](image)

Figure 4-1: SWIM-R System Overview

### 4.3 Computer
The Computer is the Human-Machine Interface (HMI) for the system. The Computer is an application that allows the user to view the video feed from the Sub and control its movement. The Computer will interface with the Float via a Wi-Fi link, making use of the Computer’s built-in wireless card.

Team SWIM-R has developed the computer application to run on the Computer, but will not be building a custom computer for it to run on. Instead, the team specified a set of minimum requirements for the user’s computer. The specifications for the Computer are provided in the Computer chapter.

### 4.4 Float
The Float is a major differentiation from the typical remotely operated vehicle (ROV). A typical ROV is tethered directly to its control computer. Team SWIM-R made the decision to tether the Sub to a mobile floating wireless station to increase range without sacrificing on bandwidth. The Float eliminates the need to plug any wires into the Computer, which leaves the user free to move or relocate the Computer.
The Float is a mobile station and therefore needed to be small enough to be pulled by the Sub. The Float must also be waterproof, and positively buoyant. The Float must be able to keep itself afloat and support any negative buoyancy from the Sub and tether.

4.5 Sub

The Sub is the unit that the user controls. The Sub has a logic system which handles all the components in the Sub as well as the communication with the Float. The Sub has motors and motor drivers for navigation as well as sensors to provide feedback for the accurate control. The camera on the Sub provides live video to the user to aid in navigation.

4.6 Navigation

Team SWIM-R anticipated designing a navigation system as part of the control system for the Sub. This navigation system would consist of inertial reckoning based on gyro, accelerometer, and magnetometer sensors. The team also looked into GPS navigation to assist in long-term navigation and dive-planning. The addition of a GPS sensor would be necessary for long term navigation because inertial measurement is unreliable over long term because it relies on integration of sensor data.

4.6.1 Inertial Navigation

Team SWIM-R purchased an Inertial Measurement Unit (IMU) board along with a depth sensor for use in the inertial navigation unit. The IMU includes a three-axis accelerometer, a three-axis gyro, and a three axis magnetometer. These sensors allow the team to determine the Sub’s attitude (pitch and roll) in the water as well as its heading. The depth sensor is used to determine the depth of the Sub below the surface of the water.

4.6.2 GPS Navigation

GPS Navigation is not trivial due to the inability of the GPS signal to propagate underwater. Therefore, a GPS unit cannot be placed on the Sub to determine its position. In order to accomplish this, the team planned to put a GPS sensor in the Float and track its position. This approach allows the team to approximate the position of the Sub based on the position of the Float and the length of the tether as shown in Figure 4-2.

\[\text{http://appft1.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PG01&p=1\&u=%2Fnetuhtml%2FPTO%2Fsrchnum.html\&r=1\&f=G\&l=50\&s1=%2220070025185%22.PGNR.\&OS=DN/20070025185\&RS=DN/20070025185]\]
In Figure 4-2 the light blue semi-circle represents the possible envelope the Sub can be in. The envelope’s diameter (A) is twice the length of the tether. Measuring the depth of the Sub (D) shrinks the diameter of the envelope to the length B. The approximated envelope is shown on the diagram as the shaded yellow region. This approximation will get better as the Sub goes deeper.

If the Float is moving a constant speed, the Sub’s position can be pinpointed with greater accuracy. This method is based on the observation that the tether will be taut if the Float is traveling in a consistent direction which matches the Sub’s heading determined from the magnetometer reading. This process is shown in Figure 4-3.
Figure 4-3: Improved Sub position based on GPS

In Figure 4-3, the line labeled T is the length of the tether and D is the depth of the Sub beneath the surface. The lateral distance from the Float to the Sub (B) can be computed from D and T using the Pythagorean Theorem. This results in an estimation (yellow shaded region) of the Sub’s position within a window defined by the accuracy of the GPS and depth sensors.

4.6.3 Combined Navigation

The method depicted in Figure 4-3 depends on the tether bring taut between the Float and Sub, and it breaks down when it is not. Team SWIM-R has determined that the assumption that the tether is taut is valid because GPS navigation will be used to autonomously navigate to a way-point. When the Sub reaches a waypoint, the team assumes the user will retake control. From here the system will depend on the inertial navigation unit to approximate the Sub’s position within the envelope described in Figure 4-2.

The team has determined that pre-planned navigation (both Inertial and GPS based) falls outside the scope of this project; however the information and ideas are retained in this report for reference.
4.7 Communication Overview

Two major communication systems in SWIM-R connect the Computer, Float, and Sub. The team has chosen a wireless connection between the Float and the Computer to increase operating range while limiting the tether length as shown in Figure 4-4. Team SWIM-R has defined operating range as the linear distance between the user and the Sub.

The increase in range with the addition of the Float can be expressed as the ratio of the user to Sub distances as shown in Equation 4-1.

**Equation 4-1: Range Increase**

\[
RangeIncrease = \frac{\sqrt{x^2 + 100^2}}{100}
\]

\(x = \text{wireless range}\)

If Team SWIM-R achieved its wireless range goal of 1 mile, the range increase will be 53. Even with a Wi-Fi range of 100 feet, the range of SWIM-R will increase by a factor of 1.41.

The team has accomplished wireless connection from the Computer to the Float of about 150 feet. This number could be further improved with the addition of a high gain antenna and a more powerful router.

The team has chosen a wired connection between the Float and the Sub. The tethered connection offers higher data rate than underwater wireless, and is within SWIM-R’s power and cost requirements. The tether will be further discussed in the Float chapter.

The data transferred over both connections will include video, sensor data, and input commands from the user. Team SWIM-R has estimated the bandwidth needed for all communication by summing the individual data rates. Estimates for the data rates are shown in Table 4-1.
Table 4-1: Data Bandwidth Estimation

<table>
<thead>
<tr>
<th>Message</th>
<th>Rate</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Motor Control</td>
<td>0.1152</td>
<td>Mbps</td>
</tr>
<tr>
<td>Video</td>
<td>65.3184</td>
<td>Mbps</td>
</tr>
<tr>
<td>Return Data</td>
<td>0.1152</td>
<td>Mbps</td>
</tr>
<tr>
<td>Sensor Data</td>
<td>0.1152</td>
<td>Mbps</td>
</tr>
<tr>
<td>Total</td>
<td>65.664</td>
<td>Mbps</td>
</tr>
</tbody>
</table>

The video stream rate is estimated assuming 720p video (1280 x 720 pixels) transferring at 30 frames per second with 8-bit, 3 channel color and 10:1 JPEG compression. The data rate of the remaining messages is estimated assuming a serial connection communicating at 115200 baud.

4.7.1 Computer-Float Communication

The Computer-Float communication link is Wi-Fi with line of sight over water. The team has chosen User Defined Protocol (UDP) over Internet Protocol (IP). The team expects that over water there are fewer obstructions than on land, which can limit the wireless signal. This allows for a greater Wi-Fi range over water than over land. Team SWIM-R tested Wi-Fi through the Calvin College Engineering Building to simulate the conditions over obstructed water.

The communication link is achieved using a commercially available off the shelf (COTS) wireless router that offers a data rate greater than the estimation provided in Section 6. The team performed range testing with a COTS router and found that the router was able to communicate reliably at a distance of 100 feet through the Engineering Building.

4.7.2 Float-Sub Communication

The tether provides a connection to the Sub by which Team SWIM-R can recover the Sub in the event of power loss or a system error. Additionally, The Float-Sub communication link is a wired tether. The tether provides an UDP over IP Ethernet interface. Team SWIM-R decided to use a tether for this connection due to the difficulties of wireless communication underwater.

4.8 Video Streaming

The team had developed the video streaming system in order to better determine the communication bandwidth needed. Jeff and Mike have written a Python script which uses the OpenCV module to capture video from a webcam.

For initial testing and development, Jeff used the built in webcam on his laptop and programmed in Ubuntu 12.04. Jeff wrote code to compress a raw image to JPEG format and convert the image data into a String to be sent over Ethernet. Mike wrote code to send and receive the data string using the socket and SocketServer Python modules. Jeff also wrote code to decompress and display the image once it was received by the client.

Further testing with the Raspberry Pi board revealed that the onboard CPU could not keep up with the demands of reading a webcam and resulted in a frame rate of 3-4 frames per second with a 2-3 second delay. The team determined that this was unacceptable for piloting the Sub and has switched to an IP webcam that can connect directly to the router. In order to accomplish this, the team is taking advantage of the 10Base-T Ethernet protocol which only utilizes four of the eight wires in an Ethernet cord. The team was able to use four of the wires for connecting to the Raspberry Pi and the other four for the
webcam. This puts the computational requirements on the Computer for reading the video feed over the network directly from the camera. The original plan for a USB webcam and the new IP webcam is shown in Figure 4-5.

Figure 4-5: Ethernet wiring scheme. On left, original concept. On right, wiring to take advantage of IP webcam.
5 Computer

The Computer portion of SWIM-R interfaces the user to the Float and Sub. The Computer allows the user to control the Sub, see the live video stream of the Sub’s camera, see the Sub’s sensor data, and save data. The saved data includes the sensor data along with a time stamp. The main considerations in designing the Computer were: ease of use, customizability, and aesthetics.

5.1 Hardware Specification

For the Computer, Team SWIM-R wrote a program that can be run from the user’s laptop. The following specifications are to ensure that the program runs smoothly. Team SWIM-R did not design the Computer hardware but is simply stating the preferred specifications of a laptop that can be used for the Computer.

5.1.1 Computer Hardware Minimum Specs

The Computer software is supported by any laptop with the following specifications, but Team SWIM-R suggests using a ruggedized laptop if the situation calls for it. Team SWIM-R recommends the laptop be capable of displaying a 720p video stream. The Computer needs an IEEE 802.11n wireless adaptor for optimum performance. An IEEE 802.11g or IEEE802.11a wireless adaptor could be used with less than optimum performance.

5.1.2 Computer Peripherals Supported

As a baseline for peripherals, Team SWIM-R chose to support a laptop keyboard, the laptop screen and the use of internal and external memory devices. This is because reading inputs from the keyboard, displaying to the screen, and saving to internal or external memory devices are supported by standard libraries. The Computer software supports using an XBOX controller to make driving the Sub more user friendly than a keyboard. The Computer should have at least one USB port if the user wishes to take advantage of this capability.

5.1.3 Computer Operating Systems

The Computer works well with both Windows 7 and Vista. Team SWIM-R had considered using the Mono Project to make the Computer be able to support both Linux and OS/X. Windows was chosen as the de facto OS because Jon was familiar with GUI development using the .NET framework and choosing Windows allows for fast prototyping.

5.2 Software Design

The software design of the Computer incorporated functionality, aesthetics, and user friendliness. The team made the program work reliably while being easy to learn and customizable.

5.2.1 Programming Language

Team SWIM-R wrote the software for the Computer in IronPython using the Eclipse IDE with the help of the PyDev environment plug-in. Some other possible programming languages that Team SWIM-R could have chosen are C, C++, Ruby, Java, etc. using Eclipse or Microsoft Visual Studio. Team SWIM-R made its decision based off of the desire for fast prototyping and continuity between the UI software and the Raspberry PI software, written in Python. Team SWIM-R chose to write in IronPython because Jon had experience in GUI design with extensive .NET framework. Team SWIM-R did early development in C# using Visual Studio but switched to IronPython in Eclipse because the communication software was written in Python using Eclipse, and Team SWIM-R wanted more continuity between systems.
5.2.2 Software Overview
The software consists of four main functional blocks: control interface, data archiving, display, and communication. Each functional block runs on its own thread in order to ensure that each process gets done when it is supposed to, and they don’t have to wait for one another to finish.

5.2.3 Communication
The communication block is in charge of connecting to the Float and the data transfer therein. The communication of basic commands is an essential feature in order to maintain control of the Sub. The receiving of video from the Float was a baseline feature so the users can see where they are driving the Sub. To ensure that the Sub is being controlled by the correct Computer, the Wi-Fi signal produced by the Float shall be password protected. All of the communication between the Computer and the Float are done with the help of the custom Python module ClientInterface which inherits from the Python library, Threading, so is its own thread. ClientInterface is described more in Section 6.

5.2.3.1 Establishing Connection
The connection to the Float is established while the main form is loading. As it loads it initializes an instance of ClientInterface which establishes the Computer as a UDP socket client. The connection state is monitored by a Boolean value representing the isConnected(). This method was chosen because the connection with ClientInterface is how all values are passed between the Sub and the Computer so it is important to establish as soon as possible.

5.2.3.2 Receiving Data
The Computer receives values from the Sub through ClientInterface. ClientInterface receives and stores the value until the main form is ready to ask ClientInterface for updated values. The main form continuously loops reading values from ClientInterface with getters such as getYaw(), getPitch(), etc. This method was chosen to make sure that the User Interface (UI) thread is not cluttered with having to store keep track of these values.

5.2.3.3 Sending Commands
The Computer sends commands to the Sub through ClientInterface. ClientInterface sends values whenever the main form gives it a new value to send. When the main form has a command to send, it gives it to ClientInterface with the help of setters such as setArm(), setPitch(), etc. This method was implemented in order to keep the main UI thread free from having to do the sending of the values.

5.2.4 Control Interface
The control interface block is in charge of receiving inputs from the user for Sub control. Handling user inputs was an essential feature because it is the means by which to control the Sub. Key_Map is a custom class that was written in order to store and manage user inputs controls. Key_Map is essentially a dictionary of custom object class names Key_Entry for each direction of movement capable by the Sub (two for each degree of motion). A Key_Entry object has six different attributes that are used in the control interface.

The six different attributes are Code, Key Value, Speed, Counterpart, Is Pressed, and Kill. Code is a string that is used to indicate which direction of which degree of motion the Key_Entry represents. Key Value is a string that represents a button on the keyboard that corresponds to that the Key_Entry. Speed is an integer that is used to tell to the Sub how fast to go in that direction. Counterpart is an integer value that gives that location of the Key_Entry in Key_Map that is responsible for opposite direction. IsPressed is a Boolean value that indicates whether the user is pressing the button that for the Key_Entry. Kill is a
Boolean value that indicates whether or not the threads sending speed values for the Key_Entry need to be stopped.

5.2.4.1 Standard Steering Inputs

The Computer has a dropdown menu strip along the top where one of the items is “Arm”. By clicking this, the user arms the ESCs, which is necessary before any user steering can happen. Also in the menu strip is a “Settings” form where the user can specify which key on the keyboard they want to control which direction. In the “Settings” menu the user can also choose from two steering command methods. Both steering settings allow the user have more of analog steering feel while working off of the digital inputs of the keyboard.

The first steering method allows the user to hold down a button on the keyboard which fires a KeyDown() event that after checking if it is a steering key and the counterpart is not currently pressed, it begins a new thread by initializing an instance of a custom class called KeyDown_Custom(). KeyDown_Custom loops while the IsPressed value is true and increments the speed value stored in Key_Map until it reaches max speed or the user releases the button and fires a KeyUp() event to stop the incrementing and return the speed value back to neutral. While every time through the loop the values stored in ClientInterface are updated and sent to the Sub.

The second steering method allows the user to tap a key on the keyboard to increment the speed value in Key_Map by means of the KeyDown() event. The speed then stays constant until the key is pressed again to increment it or the key for the opposite direction is pressed to decrement it. The keyboard keys that map to the five directions of freedom that the Sub can go are customizable in the “Settings” menu.

5.2.4.2 Waypoint Autonomous Navigation

Team SWIM-R considered allowing the user to input multiple waypoints (i.e. longitude and latitude coordinates and depth) that would have been used to simulate individual user inputs that are transmitted according to section 6.4.2.1. This was deemed out of the scope of this project because it would have involved buying a GPS and doing extensive calculations.

![Set Waypoint](Figure 5-1 Waypoint Input Window)

5.2.4.3 Mechanical Feature Control

The ballast, sampling mechanism, and robotic arm were considered to be functions controllable by the user, but these were considered to be outside the scope of this project.
5.2.5 Data Archiving

The data archiving block is responsible for saving the sensor data, pictures and videos taken during the excursion. These were all baseline goals because it is important for researchers and marinas to have proof of what the video captured and not have to go back down every time they want to see it. The locations where the items are saved can be individually specified in the “Settings” dropdown menu. Team SWIM-R chose this method so that the user knows exactly where the saved items are and not have a hard time finding them. The predetermined location also ensures that the user will not have to stop steering the Sub while they specify locations in a SaveDialog. The default locations for data archiving is the directory in which the program resides.

5.2.5.1 Sensor Data Archiving

Storing of sensor data was going to be done in a similar manner to a program that Jon wrote during the summer of 2012 that used BinaryFormatter or Serializable properties utilizing either the System.IO or the System.Runtime.Serialization.Formatters.Binary libraries. Instead the means by which sensor data is stored uses standard IronPython functions. The methods used are open() and write() which makes it not necessary to import more libraries. The sensor data is archived every minute with the default location being “DefaultData.csv”.

5.2.5.2 Video and Picture Archiving

Team SWIM-R thought of video and picture archiving as an optional feature that would have been complete if time had allowed. Picture archiving would have been done by extracting an individual frame from the video feed and storing it at the location specified in the Settings menu. The default name for the extracted image will be the date and time that it was saved. Team SWIM-R did not implement video archiving or picture archiving because it was low on the priorities list as the team was not able to mount the video camera as desired.

5.2.6 Display

The display block is responsible for updating the video feed and the heads-up-display (HUD) and for being aesthetically pleasing to the user. Having a video stream display on the Computer is a core feature because without it the Sub would be hard to control and our project could not be considered investigative. Having a HUD is a baseline feature because it makes SWIM-R easier to use because the user doesn’t have to rely on the saved sensor data.

5.2.6.1 Displaying Video Stream

In order for the video stream to be displayed, the program has a looping thread that uses the System.Net library for receiving the video from an IP camera. This approach updates a PictureBox value from a video stream established using an HttpWebRequest.

5.2.6.2 Updating Heads-Up Display

The data on the heads up display is updated on a looping thread that pulls values from the ClientInterface class which receives them from the Sub via the Float when the Sub sends values. The communication between systems is further described in Section 6. The values received from ClientInterface are converted to a float. The yaw value is used to update a compass needle image with a custom method DoRotate(). The roll value will use the same custom method to graphically display the roll angle of the Sub. The depth and pitch graphical indicators will be updated each time through the loop with a method called DoVerticalMove(). This method has not been written yet, but it will involve setting the location of the indicator based off of the value received form ClientInterface. All of the value received are then converted into a string and set as the text for the appropriate label.
5.2.6.3 Aesthetics

Aesthetics was important because Team SWIM-R wanted the user to enjoy the SWIM-R experience. Team SWIM-R did some preliminary Graphical User Interface (GUI) design work that are shown in Figure 5-2 and Figure 5-3. The design in Figure 5-2 received more positive feedback so Team SWIM-R chose to use the same layout with updated graphics which is shown in Figure 5-4. Team SWIM-R chose to use graphical indicators for roll, pitch, yaw, and depth. These were chosen to be displayed with images because it is easier to understand and is similar to flight systems on airplanes.

All of the displayed data was not chosen to be represented in image form because it would have cluttered up the HUD and made it busy with too many moving parts. The GUI is broken down into six different images so that the design scales appropriately for the screen size, meaning that the roll, pitch, and yaw indicators will always be perfect circles and will not be oversized on a small screen or too small on a large screen.

The GUI has a full screen option for people who wish to get the most out of every frame of the video which increases customizability. Customizability was important so the team can cater to a variety of people which fits with Team SWIM-R’s design norm of Justice.

Figure 5-2: Preliminary GUI Design 1

Figure 5-2 represents one possibility for the GUI. In this design, the visual representations of the data are displayed along the bottom of the screen. Displaying the data like this allows the user to feel more like they are driving in a cockpit. This is the format of the GUI that was chosen as a result of an in class survey.
Figure 5-3: Preliminary GUI Design 2

Figure 5-3 is similar in how it shows the data for everything except the battery life and signal strength. The main difference in this design is showing the data on the sides of the screen. This method allows the user to view the full image instead of having it cut off or distorted like in the previous design. This design was less supported in Team SWIM-R’s surveys.

Figure 5-4: Final GUI Design
Figure 5-4 is the final GUI design that was chosen using the same layout as Figure 5-2 with updated graphics.

5.2.7 Preliminary Testing Designs
Jon developed a number of preliminary designs to experiment with user interaction.

5.2.7.1 Version One
The first version of the software was a simple Windows Form that consisted of six labels each with an initial value of “0” that represented the possibility of six motors. The form used a “KeyDown” event to recognize when the user hit a button on the keyboard and used a “KeyUp” event to recognize when the user was no longer holding down the button. The software would check, by means of a case statement, if one of twelve buttons on the keyboard was pressed. The twelve buttons that it recognized were ‘Q’, ‘W’, ‘E’, ‘R’, ‘T’, ‘Y’, ‘U’, ‘I’, ‘O’, ‘P’, ‘[‘, ‘]’, and ‘\’. The first six buttons caused the corresponding label to change to a one while it was pressed and turn back to a zero when the button was released. The second six buttons caused the labels to change to a negative one while pressed and turn back to a zero when the button was released. This was used a way to recognize when the user wanted to turn on a specific motor.

5.2.7.2 Version Two
The second version of the software displayed a live video stream of the feed from a webcam that was attached to the computer. Before the program could display the image to the screen, however, the image was first copied to the clipboard. This version of the software was based largely off sample code8.

5.2.7.3 Version Three
The third installment of the software design included the ability to continually send a string consisting of six characters to another windows form. The main window of the program consists of a button that when pressed started a “System.ComponentModel.BackgroundWorker.” The background worker would send the second form.

5.2.7.4 Version Four
This version of the software is similar to version three above but instead of sending the strings to another windows form, the string was converted into a byte array and sent to an UDP server written in Python that send a string back.

---

6 Float and Communication System

Team SWIM-R has chosen to implement a hybrid wired and wireless communication link between the Computer and the Sub. The Float serves as the transition between the wireless connection from the Computer and the wired connection down to the Sub. The wireless leg of the connection allows for control at a distance of 100 to 300 ft. and the wired leg bypasses the complications of sending wireless signals underwater. Using wireless communication underwater only allows for data rates on the order of kbps and requires special signal processing hardware. The wired leg of the connection is referred to as the tether.

The Float contains a wireless router, voltage regulation circuitry, and LiPo battery. The main design considerations were waterproofing, buoyancy, connection type, and power delivery. Figure 6-1 highlights all of the Float components as well as the connections between them. Figure 6-2 and Figure 6-3 show the proof of concept implementation of the Float.

In the subsequent sections, the electrical hardware design, mechanical design, and the communication system design are described.

---

Figure 6-1: The Float

---

Figure 6-2 Inside the Float
6.1 Electrical Hardware Design

The electrical hardware design consists of the communication connection between the Computer and Sub and power supply design. The communication connection is the system that allows communication between the Computer and the Sub. The Float’s power supply provides the 5V needed to power the wireless router. In addition, the Float’s power supply illuminates an external warning LED when the battery voltage falls to 10.8V. Originally, the Float was also going to contain a GPS receiver and microcontroller so that the Float’s coordinates could be sent to the Computer to calculate the position of the Sub with some trigonometry. However, due to time constrains, a GPS receiver in the Float was not implemented.

6.1.1 Communication Connection Alternatives

Team SWIM-R considered several connection schemes from the Computer to the Sub.

6.1.1.1 All wireless communication (no Float)

Team SWIM-R has rejected using only wireless communication from the Computer to the Sub. One example of completely wireless communication is low frequency acoustic signals used by submarines for audio communication. Unlike electromagnetic waves underwater, acoustic signals can propagate over long distances. However, this type of communication is subject to frequency dependent propagation loss,
signal multipath and low speed of sound propagation. All of these combined factors make for an overall low data rate on the order of kbps and the need for specialty hardware such as the WHOI Micro-Modem from Acoustic Communications. This modem has a maximum bit rate of 5400 bps and has maximum power consumption of 50W when transmitting. Both of these specifications are incompatible with the battery life and live video requirements for SWIM-R. As stated in Section 4.7, 10:1 JPEG video streaming at 720p requires up to a 65Mbps connection. Additionally the Micro-Modem’s high power consumption would require a very large battery and regulation circuit. All of these factors make completely wireless communication via acoustic modem a poor choice for SWIM-R.

6.1.1.2 All Tethered Communication (no Float)
Team SWIM-R has rejected completely tethered communication in order to allow SWIM-R to be controlled from a long distance away. This type of connection limits the mobility of the User and the Sub. A very long cable is also very heavy. This creates extra strain on the Sub motors as they try to compensate for the additional load and causes the batteries to be drained faster because they must use more energy to move the tether. This also creates a situation where the user is forced to manage the tether and keep the Sub from dragging their Computer into the water.

6.1.1.3 Bluetooth Communication from the Computer to the Float
Bluetooth has been rejected as the wireless communication protocol. While many computers are compatible with Bluetooth, it does not have a wired counterpart akin to Wi-Fi and Ethernet. Using Bluetooth would thus require additional conversion hardware on the Float to convert a Bluetooth signal to a wired signal like Fiber or Ethernet. Additionally, Bluetooth 2.0 has a nominal data rate of 3.0 Mbps. While this is faster than underwater acoustic communication, it is not sufficient for streaming video at 720p.

6.1.2 Chosen Communication Connection

6.1.2.1 Wi-Fi and Ethernet Hybrid
Team SWIM-R has decided to use Wi-Fi to connect the Computer to the Float and Ethernet from the Float down to the Sub. Wi-Fi provides an adequate data rate for streaming live video from the Sub. Team SWIM-R used a D-Link DIR-615 wireless router for the SWIM-R 1.0 prototype. A wireless router will also be included in the production design. Team SWIM-R chose a wireless router because it is inexpensive, readily available from several major companies (e.g. D-Link, Cicso, Netgear), low power (approximately 5W), and allows for a high data transfer rate. Also, Wi-Fi cards are typically a standard component of an off the shelf computer; this makes SWIM-R more likely to be compatible with a computer that the user already owns. The DIR-615 provides a maximum of 300Mbps wireless connection and a 100 Mbps Ethernet connection. This data rate is sufficient to accommodate the 100 Mbps Ethernet connection built into the Raspberry Pi. Off-the-shelf wireless routers also provide user configurable firmware that is configured in GUI menus allowing the user to specify a Wi-Fi password for SWIM-R. The production design will include a router with a static IP address assigned to Sub as well as instructions for the user to configure their own Wi-Fi password.

While the DIR-615 provides a convenient and inexpensive platform to test wireless communication it does not meet the range requirement of 1 mile but instead has a nominal range of roughly 300ft outdoors.

---

11 http://acomms.whoi.edu/umodem/index.html
The power supply in the Float provides 5V and a maximum of 1A. Another router compatible with these power requirements with a longer range could be a drop-in replacement for the DIR-615 but was not necessary for initial proof of concept.

6.1.3 Power Supply
There are several alternatives for powering the electronics on the Float: solar cells, Power over Ethernet (PoE) from the Sub to the Float, and direct battery. Team SWIM-R has chosen to power the float directly with a LiPo battery.

6.1.3.1 Regulation Circuit
The wireless router is powered from an 11.1V LiPo battery, a voltage regulator, and several decoupling capacitors. In order to provide battery monitoring the voltage regulation circuit has a warning LED that turns on when the battery voltage falls below 10.8V. The warning LED can be seen while the Float is closed. The power regulation circuit is shown in Figure 6-4. The 10K potentiometer is used to tune at what voltage the warning LED turns on. The LED after the voltage regulator is simply to indicate if the circuit is powered. For the proof of concept Float, a 5Ah LiPo battery was used providing an estimated 7 hours of operation for the Float. Any LiPo battery terminated with a female Dean’s connector supplying between 7.5 – 15V can be used to power the Float.

[Image of the float power regulator circuit]

Figure 6-4 Float Power Regulator

6.1.3.2 Power Alternative: Solar Cells
Team SWIM-R has rejected solar cells as the primary power source for the Float because of the variability of solar energy availability. SWIM-R should not be usable only during sunny days. Small solar cells (less than 1 ft²) are available from SparkFun electronics[^13] that can supply up to 10W of power. However, this power solution would still have required a supplemental battery to provide continuous power. Due to readily available LiPo batteries that provide ample operating life beyond the required 1 hour, solar cells were deemed superfluous for proof of concept.

[^13]: https://www.sparkfun.com/products/9759
6.1.3.3 Power Alternative: Power over Ethernet (PoE)

Team SWIM-R also considered power over Ethernet to power the Float from the power supply in the Sub. The benefits are that the Float and Sub could be powered from one power supply. However, this requires extra hardware to inject power into the tether. The hardware that PoE would eliminate is replaced by the hardware needed to deliver power over the tether. Additionally, the extra wires in Ethernet cables that are typically used for PoE were used to accommodate the IP camera in the Sub.

6.2 Mechanical Hardware Design

The mechanical hardware of the Float includes the enclosure, the waterproof connector, and the tether down to the Sub. The goal was to create a waterproof enclosure with minimal design work. This resulted in buying a dry box from a local store. The Float enclosure must not interfere with wireless communication and it must be buoyant so that it cannot be pulled under water by the Sub. The Tether must be strong enough to allow the Sub to drag the Float around without becoming disconnected.

6.2.1 Float Enclosure

For the initial proof of concept build, Team SWIM-R purchased a waterproof recreational dry box often used to keep valuables dry when hunting or fishing. The box was inexpensive and large enough to accommodate a wireless router, battery, and the necessary circuitry. It is pictured in Figure 6-3. Additionally, an inner tube was added to keep the dry box upright when it sits on top of the water.

6.2.2 Tether and Waterproof Connectors

The tether is made up of a single 50ft Ethernet cable. On either end, waterproof connectors were installed that attach to ports on the outside of the Float and Sub. The connectors, one of which is shown in Figure 6-5, are IP68 rated and have proven to be reliable. A longer Ethernet cable would also have worked, but given that the deepest available testing area was 15ft, team SWIM-R decided to use a donated 50ft Ethernet cable for the initial build.
6.3 Float Testing

The Float was tested for structural integrity, battery life, and tether strength. During many pool tests, the Float was deployed and provided a reliable signal to the laptop on shore. When the Sub traveled across the pool the Float gently followed it (Figure 6-6). After each pool test, the inside of the Float was inspected for leaks and none were found. Given that the tether is so long compared to the testing pool, the Float was not ever in a situation where the Sub was pulling it down. After 1 hour of driving, the Float had consumed roughly 11% of its battery’s charge.
6.4 Communication System

The Communication System is software that delivers data from the Sub up to the User Interface (UI) software and delivers commands from the UI software down to the Sub. For this to work the R-PI, Arduino, and UI software all needed to communicate efficiently.

In this section the interface that allows the UI software to send and receive data from the R-PI is described. Then the communication between the R-PI and the Arduino is described. Lastly, the implementation of disconnect handling is described.

6.4.1 Communication from the Computer down to the R-PI

The UI software retrieves and writes data to/from the R-PI via the methods specified in the python module `ClientInterface.py`. This module encapsulates all Ethernet I/O on the Computer to control SWIM-R. The `ClientInterface.py` module hides all of the details of sending and receiving data over Ethernet and only allows access to data of interest.

6.4.1.1 Data that is read

The following methods in `ClientInterface.py` return various data of interest to the user interface software.

getConnectionStatus()

Returns True if the Computer is connected to the Sub and False otherwise. The implementation is described in the subsequent section entitled Disconnect Handling.
**getCaseTemperature()**

Returns a float containing the temperature inside of the Sub in degrees Celsius.

**getWaterTemperature()**

Returns a float containing the water temperature outside of the Sub in degrees Celsius.

**getBatteryLife()**

Returns a float containing the current battery voltage in Volts.

**getError()**

Returns True if the Arduino is properly communicating with the R-PI and False otherwise.

**getDepth()**

Returns a float containing the current depth of the Sub in meters.

**getRoll()**

Returns a float containing the current roll of the Sub in degrees.

**getPitch()**

Returns a float containing the current pitch of the Sub in degrees.

**getYaw()**

Returns a float containing the current yaw of the Sub in degrees.

---

6.4.2  Data that is written

6.4.2.1  Set X,Y,Z, Yaw, Pitch, Roll, Arm (7 methods)

The UI software interprets user input from keystrokes or a game controller. When new user input is generated, the appropriate movement methods are called and the new values are immediately sent down to the Sub. The argument for each method is an integer between 0-255 with 127 being neutral and 0 and 255 representing minimum and maximum respectively. Whenever one of these 6 methods is called, the most recent values of all 6 dimensions are sent as one structure. However, the methods for setting the roll, pitch, yaw, X, Y, Z of the sub do nothing until setARM(True) is called by the UI software. When the UI software arms or disarms the Sub via this method, all of the movement dimensions are set to neutral.
For example if the current movement state is:

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>True</td>
<td>ROLL</td>
<td>132</td>
<td>PITCH</td>
<td>111</td>
</tr>
<tr>
<td>YAW</td>
<td>164</td>
<td>X</td>
<td>56</td>
<td>Y</td>
<td>98</td>
</tr>
<tr>
<td>Z</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

and `ClientInterface.setX(254)` is called, a structure containing the following data is sent to the R-PI

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ARM</td>
<td>True</td>
<td>ROLL</td>
<td>132</td>
<td>PITCH</td>
<td>111</td>
<td></td>
</tr>
<tr>
<td>YAW</td>
<td>164</td>
<td>X</td>
<td>254</td>
<td>Y</td>
<td>98</td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>101</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notice that each dimension is kept the same until it is updated. When any dimension is updated, the whole packet is sent to the R-PI.

6.4.3 Communication from the R-PI to the Arduino

The R-PI and Arduino communicate via byte arrays over a serial connection. Because only the numeric values are written and read, an instruction format was agreed upon so that the Arduino could attribute the correct data values to their corresponding dimensions.

6.4.3.1 R-PI Writing data to the Arduino

When new data arrives from the Computer over Ethernet, its values are formatted into a fixed width byte array and then written to the Arduino. Each field is encoded as 1 byte resulting in 9 byte packets that are written to the Arduino. The order of the fields is depicted below. Note that the first byte is the number of subsequent fields. For the instruction pictured below, the size field would hold a value of 8.

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>Error</td>
<td>Arm</td>
<td>Roll</td>
<td>Pitch</td>
<td>Yaw</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Y</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Z</td>
</tr>
</tbody>
</table>

The error field is used to designate to the Arduino if there are connectivity issues between the R-PI and the Computer. The Arduino will only spin motors based on user input if the Arm field is set to True and there are not Ethernet connectivity issues (Error field contains False).
Table 6-2: A typical instruction packet telling the Arduino to move the Sub forward

| 8 | 0 | 1 | 127 | 127 | 127 | 170 | 127 | 127 |

6.4.4  R-PI Reading data from the Arduino
Periodically the Arduino sends sensor data to the R-PI. First, three dollar signs ($$$) are sent to the R-PI to indicate new data is about to arrive. Because the data fields can take on a wide range of values (positive, negative, many decimal places) the byte widths are not constant. Accordingly, commas are used as delimiters between data fields. The end of a data packet is recognized by the R-PI when it receives a pound sign (#).

Table 6-3: Read Instruction Format

| Roll | Pitch | Yaw | Case Temperature | Case Humidity | Water Temperature | Depth | Battery Voltage |

6.4.5  Connection Handling
In general the communication connections of SWIM-R are maintained by a system of heartbeats and timeouts. The two communication connections are the serial connection from the Arduino to R-PI and the Ethernet + Wi-Fi connection from the R-PI to the Computer. The methods for establishing connections and monitoring them are described in the following sections.

6.4.5.1 The Ethernet + Wi-Fi Connection from the Computer to the R-PI
This connection is implemented using a UDP/IP protocol. UDP was selected because it is lower latency than its counterpart TCP/IP. Because UDP/IP is packet based instead of connection based, a system of timeouts in the application layer is used to ensure that the connection has not been interrupted. Whenever there is no new user input or new sensor data from the Sub, Ping messages are sent and received between the R-PI and the Computer. On both ends, if a message is not received in a 3 second interval, the communication software declares that the connection has been lost and automatically attempts to reconnect. When this happens, the R-PI alerts the Arduino that the connection has been lost and the UI software alerts the User that the connection has been lost. Figure 6-7 shows the communication logic running on the Computer.
Figure 6-7 Computer Side communication logic
6.4.5.2 The Serial Connection between the R-PI and the Arduino

From initial testing, this connection has shown to be much less volatile than the Wi-Fi connection. However, timeouts have been put in place to monitor the connection. When there is not new user input to be written to the Arduino, the R-PI simply writes the old packet to the Arduino. If data cannot be successfully written to the serial port in 15 seconds, the UI software is alerted via the `getError()` method. Figure 6-8 shows the communication logic running on the R-PI.

![Figure 6-8 Communication Logic on the R-PI](image)
7 Sub

The Sub portion of SWIM-R includes the electronics and mechanical structure necessary to pilot in the water. Additionally it possesses sensors to collect desired data and the software needed to make use of the electronics and communicate with the Computer. The Sub also houses a video camera whose data is streamed to the Float. The Sub has three areas of design: Hardware, Software, and Mechanical addressed in Section 7.1, Section 7.2, and Section 7.3 respectively.

7.1 Hardware Design

The inputs to the Sub include a video stream from a forward facing camera, data from several onboard sensors, and control input from the user through the tether. The outputs from the Sub include data and video streamed over Ethernet to the Float and physical movement of the Sub through the water. Figure 7-1 shows the Sub’s hardware and the connections between the pieces.

Figure 7-1: Sub Electrical Hardware
7.1.1 Development Boards

Team SWIM-R chose two development platforms for prototyping: a Raspberry Pi and an Arduino. The team chose to use two development platforms to facilitate parallel development. Using two dev boards allowed two team members to work directly on the hardware at once.

7.1.1.1 Raspberry Pi

The Raspberry Pi was used as the main computing unit of the Sub. The Raspberry Pi is an embedded Linux computer that costs $35. The board includes 26 general purpose input/output (GPIO) pins which can be used for reading digital sensors, processor clocked at 700MHz, and 512MB of RAM.

The BeagleBoard is an alternative to the Raspberry Pi. The BeagleBoard offers a vast online community similar to the Raspberry Pi, and is more powerful than the Raspberry Pi. However, the BeagleBoard is significantly more expensive than the Raspberry Pi at $199.95\(^\text{14}\)\(^\text{15}\).

The BeagleBone is a variant of the BeagleBoard mentioned above. The BeagleBone offers less RAM than the Raspberry Pi at 256MB, but has more GPIO pins. The BeagleBone is priced between the Raspberry Pi and BeagleBoard at $89\(^\text{15}\).

The team chose to use the Raspberry Pi because it is inexpensive yet powerful enough to run Linux. The Raspberry Pi also has a vast online community which the team was able to tap into for advice in utilizing features of the board. The team was wary of using the Raspberry Pi due to its limited availability, but Jeff volunteered his personal Raspberry Pi board for early prototyping while the team waited for one to arrive.

The Raspberry Pi comes in two models: A and B. Team SWIM-R decided to use the Model B version because it includes a 10/100 wired Ethernet connection and a second USB 2.0 port which are not present in Model A. The Ethernet port was required for the tethered connection to the Float and the second USB port was originally needed for the webcam to provide the video stream. This requirement was modified near the completion of SWIM-R 0.3. Details of this change may be found in section 7.1.4. The first USB port was designated to be the means of communication between the Arduino and the Raspberry Pi. The Raspberry Pi’s System on Chip (SoC) is a Broadcom BCM2835, which contains an ARM11 with floating point. The processor is run at 700MHz, but can be over-clocked to 800MHz. Additionally, the SoC includes a Videocore 4 GPU\(^\text{16}\). This allows the Sub to capture, compress, and send a video stream.

7.1.1.2 Arduino

Arduino is a family of popular microcontrollers that are open hardware and open software.\(^\text{17}\) Team SWIM-R chose the Arduino due to the team’s familiarity with the device and the number of open source software libraries that are available with it. Additionally, the Arduino allows quick prototyping with its easy to use IDE, availability, low cost, and vast online community. The team was able to begin immediate development with the Arduino using Jeff, Mike, and Mitch’s personal Arduino boards while the team decided which type of Arduino would be best suited for the project.

Early development was done using an Arduino Uno. This board is the most common of the Arduino family and provides a wide range of interfaces. The major features of the board that made it attractive to the team were the 6 PWM pins that could be used to drive motors, an Inter-Integrated Circuit (I2C) bus

\(^{14}\)https://www.sparkfun.com/products/9978
\(^{15}\)https://www.adafruit.com/products/513
\(^{16}\)http://www.raspberrypi.org/faqs
\(^{17}\)http://www.arduino.cc/
for digital sensors, 6 analog input pins for additional sensors, a built-in USB-to-serial driver chip for communicating with the Raspberry Pi, and its small board footprint to save space inside the Sub.

The Arduino was required to communicate with the Inertial Measurement Unit (IMU) discussed in section 7.1.2.1 in order to control the Sub and track its position accurately. Because the Uno only possesses one hardware serial port, this channel of communication was going to be established by utilizing the SoftwareSerial library which is built into the Arduino cores. This was successfully accomplished in February, but when used in conjunction with the Servo library it was discovered that a hardware conflict existed. Both libraries utilized the same hardware timer which would cause the servo motors to twitch unexpectedly. Because this could not be controlled or remedied without making drastic changes to the internal workings of the board, the team made the decision to instead use an Arduino Mega2560. This board features four hardware serial ports. This allowed the team to use one of the additional serial ports for communications between the IMU and the Arduino. Serial0 on the Arduino Mega is built into its USB B port which the team connected directly to one of the Raspberry Pi’s USB A ports. The three additional serial ports consist of pairs of Transmit (Tx) and Receive (Rx) pins. Serial1’s Rx pin was connected to the IMU’s Tx pin and vice versa according to the convention of serial communication.

The Arduino acts as the main motor driver unit in the Sub as well as reading the environment sensors. Additionally, there are versions of the Arduino available which incorporate sensors onto the board and are discussed in Section 7.1.2.1.

7.1.2 Sensors
The sensors contained in the Sub can be broken down into three categories: control, safety, and data logging. The following sections discuss them in detail.

7.1.2.1 Control Sensors
The control sensors are necessary for the Sub to maintain accurate position and form a portion of the feedback loop to the motor control system. The control sensors include a three-axis accelerometer, a three-axis gyro, and a three-axis magnetometer. These sensors combine to form an IMU with 9 Degrees of Freedom (DOF). The team only needs 6 DOF to compute the Sub’s attitude (orientation) in the water and its translational motion; however the accelerometer and gyro together only provide 5 DOF. This is because the z-axis gyro has no reference point for long term accuracy of the Sub’s yaw rotation and left unchecked, the reading from the gyro alone will drift over time. The three axis magnetometer provides the earth’s magnetic pole as a reference point. The team needs this yaw lock to approximate the Sub’s position using inertial navigation.

The team decided to purchase an IMU board which combined all three sensors on a single board and provided an I2C interface. This saved on cost compared to purchasing the sensors separately. The team also decided to use sensors with an I2C interface wherever possible in order to have multiple sensors on the same bus, using only two I/O pins. Additionally Jeff had previous experience using the I2C bus on an Arduino board. The team identified three alternatives for an IMU board. These alternatives are described in detail in the following paragraphs.

The DIYDrones ArduIMU+ V3 shown in Figure 7-2 is intended for use in a quad-copter platform known as the ArduCopter and as such has an Atmega328 microcontroller on board. The board is programmable using the Arduino IDE and programming language. Team SWIM-R believed that a board designed for use in a flying platform would work well for the Sub as both require 6 DOF with yaw lock. The Atmega328 also has several GPIO pins which are broken out on the sensor board. This was an attractive feature to the team because this would allow for the IMU to be used as a sensor unit as well as a motor driver. The hardware and software for the board is open source.
The 9 Degrees of Freedom – Sensor Stick shown in Figure 7-3 is the smallest of the three IMU boards the team has identified. The board does not have a microcontroller built in meaning the Sub would still require a separate Arduino for motor control. This board is also the most expensive at $99.95 from SparkFun.

Figure 7-2: ArduIMU+ V3 sensor board

Figure 7-3: 9 DOF - Sensor Stick

---

18 https://www.sparkfun.com/products/11055?
19 https://www.sparkfun.com/products/10724?
The MultiWii board shown in Figure 7-4 is similar to the ArduIMU+ V3 as it too is intended for use in a quad-copter platform and has an Atmega328 microcontroller on board. The board is programmable using the Arduino IDE and programming language. This board utilizes the sensors commonly found in the Nintendo Wii Remote and Nunchuck controllers\(^{20}\). The software for the MultiWii board is open source.

![MultiWii sensor board](image)

**Figure 7-4: MultiWii sensor board**

The team determined that the ArduIMU+ V3 was the most suitable IMU board for the project. This was because it was also based on an ATMEGA microprocessor so would therefore communicate nicely with the Arduino in the Sub and because it possessed the additional GPIO pins that could be utilized as motor driver pins without having to first communicate with the Arduino and then have the Arduino drive the motors. Another reason the team chose this IMU is that it came fully equipped with software to calculate the Sub’s orientation that could be slightly modified to suit the needs of SWIM-R.

Soon after purchasing the ArduIMU it was discovered that it would not be easy to utilize the GPIO pins for driving the Sub’s motors. The control software that is provided for use with the ArduIMU is very complex and making slight changes to it caused many major errors. The team decided that using the IMU to drive the motors was not worth the risk of completely disabling the IMU’s ability to compute and communicate the Sub’s position. Also, once the IMU arrived it was found to not have enough PWM pins to drive all 6 motors so the team decided to port all of them over to the Arduino for consistency as well as convenience.

### 7.1.2.2 Safety Sensors

In order to ensure the Sub stays in a safe operation zone underwater, it has sensors to monitor its depth under water, the level of humidity in the Sub enclosure, and the status of the Sub’s onboard batteries.

The sub’s depth must be monitored both to provide feedback to the depth control system and to ensure that the Sub does not descend below the depth to which the enclosure is rated. The team purchased an MPX4250A series Manifold Absolute Pressure sensor. This sensor was chosen because it uses a 5V supply and produces analog output between 0.2-4.9V which are both ideal for use with the Arduino.

The internal humidity of the Sub must be monitored because many electronics do not operate correctly, or may fail completely in humid air. Also, the internal temperature of the Sub enclosure must be monitored in order to warn the user of possible component failure. The RHT03 Humidity and Temperature sensor was purchased from SparkFun in order to accommodate both of these needs. This sensor is a unique one-

\(^{20}\) [http://www.multiwii.com/](http://www.multiwii.com/)
wire device that requires only one digital input pin on the Arduino to report the percent humidity and temperature. The humidity is accurate to within +/−2% and the temperature is accurate to within +/−0.5 degrees Celsius.

The status of the Sub’s onboard batteries is monitored to determine the remaining operation time before new or recharged batteries are needed. As a battery is discharged, the voltage decreases. At first the team considered buying a separate Analog-to-Digital converter to sense the battery’s voltage and signal the user when to change the battery. This was deemed unnecessary because the Arduino Mega possesses many Analog-Input pins which read a signal between 0 and 5 Volts. This signal can be stored in a variable and interpreted appropriately using mathematical manipulation. Since the Sub’s battery is approximately 12 volts when fully charged and 10 volts when discharged a voltage divider was required to scale down the voltage before being read by the Arduino. The battery’s voltage was divided into thirds using resistors and then read by the Arduino after the second resistor. This proved to be a very accurate way of measuring the battery’s voltage that also allowed the team to save some money that had been allocated for an additional sensor. When the battery reaches a level considered to be low, a Boolean is switched to true in the Arduino software which sends a message to the user to alert them that the battery should be charged.

7.1.2.3 Data Logging Sensors

The team was originally intending to incorporate many sensors into the Sub design for data collection. These sensors were to include water temperature, pH, dissolved oxygen, and possibly chemical detection sensors. These sensors were to be included as permanent features of the Sub or as add-ons to attach to provided ports depending on customer specifications. The team decided to include only a water temperature sensor in the final design as proof-of-concept for data logging because this feature of the Sub was not necessary for project success and deemed to be beyond the scope of the project. This was due to a large time investment that would be required to interface the Arduino and Raspberry Pi with each type of sensor.

7.1.3 Motors

The Sub’s motors provide the thrust necessary to move and rotate the Sub along the x, y, and z axes. The axes of the Sub are shown in Figure 7-5.

Figure 7-5: Relative Axes of the Sub

The Sub must be able to translate along the x, y, and z axes, and be able to rotate around the z axis (yaw). Pitch (rotation around the y axis) and Roll (rotation around the x axis) do not need to be controlled
actively with motors; rather, pitch and roll can stabilized by constructing the top of the Sub out of positively buoyant material and the bottom of negatively buoyant material.

7.1.3.1 Motor Arrangement

Control of x, y, and z translation and yaw can be achieved using four statically mounted motors. However, the team planned to use six statically mounted motors to achieve better control over pitch and roll. The six-motor arrangement is shown in Figure 7-6. Table 7-1 shows the combination of motors as well as their directions to achieve six degrees of control.

![Figure 7-6: Six-Motor Arrangement](image)

<table>
<thead>
<tr>
<th>Movement</th>
<th>Motor Combination</th>
</tr>
</thead>
<tbody>
<tr>
<td>X-translation</td>
<td>1 (F), 2 (F)</td>
</tr>
<tr>
<td>Y-translation</td>
<td>3</td>
</tr>
<tr>
<td>Z-translation</td>
<td>4 (F), 5 (F), 6 (F)</td>
</tr>
<tr>
<td>Pitch</td>
<td>4 (F), 5 (F), 6 (R)</td>
</tr>
<tr>
<td>Roll</td>
<td>4 (F), 5 (R)</td>
</tr>
<tr>
<td>Yaw</td>
<td>1 (F), 2 (R)</td>
</tr>
</tbody>
</table>

Motor 3 was not included in the final design of the Sub. This was due to a re-design of the Sub enclosure. The team’s original design for the Sub was a rectangular shape with a void in the center that would allow
water to pass through and provide translation along the y axis. Jeff was able to obtain a cylindrical tube which was instead used as the main body of the Sub. This was unable to accommodate the void. Having a horizontal motor mounted on the top or the bottom of the cylinder would not have provided lateral translation but instead caused the Sub to roll in place. In place of direct translation along the y axis, SWIM-R rotates around the z axis and then translates along the x axis.

7.1.3.2 Motor Types
Team SWIM-R identified three alternatives for motors: brushed DC, brushless DC, and modified bilge pumps. Brushed DC motors are the least expensive of the three solutions, but are likely to degrade the fastest under water due to the increased load on the physical contacts of the motor brushes from the water. Brushless DC motors use electromagnets in place of brushes to drive a magnetic field and rotate the motor. As a result, a brushless motor can be made to withstand the effects of water by encasing the electromagnet windings in an epoxy. Brushless DC motors do however draw more current than their brushed counterparts.

A bilge pump is a DC motor that is designed to be run in and move water. Bilge pumps operate in the same manner as brushed DC motors with the exception that bilge pumps are designed to withstand the wear caused by prolonged use in water and would therefore degrade at a slower rate compared to brushed DC motors. The potential drawback of using bilge pumps is that some are not designed to run in air. If the user decided to test these motors in air the motors could be destroyed. If bilge pumps were to be used this drawback could be avoided by purchasing bilge pumps that are capable of dry testing to prevent damage but these pumps would also be more expensive.

The team decided to use brushless DC motors for the drive system in SWIM-R because they cannot be damaged by driving them in open air and because a method was found that allowed the team to coat the windings of the motors in silicone which would prevent them from degrading when used in water. A can of WD-40 silicone lubricant was purchased for this purpose. The motors can be thoroughly coated with this substance before each use to ensure that water degradation is minimized.

7.1.3.3 Motor Drivers
There were two types of motor driver circuits Team SWIM-R considered. The two options were an H-Bridge circuit and an Electronic Speed Controller (ESC). Both alternatives allow for variable speed and bidirectional control.

H-Bridge circuits are often used for bidirectional control of brushed DC motors. The H-Bridge requires two wires from a microcontroller to achieve bidirectional, variable speed. Using the six-motor arrangement described in Section 7.1.3.1, this would require 12 connections to the Arduino.

An ESC is necessary for driving a brushless DC motor with three-phase power. An ESC has the benefit of only requiring one wire per motor from the controller for bidirectional, variable speed. Using the six-motor arrangement described in Section 7.1.3.1, this would require 6 connections to the Arduino.

The team chose to use ESCs in the final design because they chose to use brushless DC motors for the drive system. This was the only option for motor driving electronics due to the brushless DC motors. The ESCs that were chosen are rated at 18 Amps max current draw and are designed to be used in model cars and boats. They are capable of driving the motors in both forward and reverse and have many other user adjustable settings such as braking and adjustable start mode or “punch” which is how fast do the ESCs react to an incoming signal. Each motor is rated at a 12 Amp max current draw so these ESCs possess a safety factor of 1.5 to prevent damage to them. The product ID number for the ESCs is 18A-SL. They were purchased from HobbyKing.
7.1.3.4 Magnetically Coupled Drive

Under normal conditions brushed and brushless DC motors cannot be completely waterproofed when direct-driving a propeller. An alternative design to the direct-drive system is a magnetically coupled drive\textsuperscript{21}. In a magnetically coupled drive the motor spins freely in a sealed tube with an array of magnets attached to the rotor. Outside of the tube is a propeller which has an array of opposite magnets. Figure 7-7 and Figure 7-8 show the system in greater detail.

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{magnetically_coupled_drive_concept.png}
\caption{Magnetically Coupled Drive Concept}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{magnetically_coupled_drive_propeller.png}
\caption{Magnetically Coupled Drive Propeller}
\end{figure}

\textsuperscript{21} http://openrov.com/forum/topics/magnetically-coupled-drive?commentId=6365107%3AComment%3A21661
This system improves long-term reliability, as the motors are no longer exposed to water, but reduces the torque output of each motor. Additionally, the magnetically coupled drive system can be used with either brushed or brushless DC motors.

The team did some research into fabricating these magnetically coupled propellers early second semester, including constructing a prototype using the Calvin College 3D printer. Jeff used Autodesk Inventor to design the inner rotor and the outer propeller units to fit around a length of PVC pipe as shown in Figure 7-9.

![Figure 7-9: Inventor drawing of magnetically coupled propeller prototype](image)

The rotor and propeller each have slots for four neodymium magnets, which were borrowed from Calvin College’s Physics department for the test. During the test, one of the magnets in the rotor cracked the brittle acrylic plastic and the rotor shattered. Due to the lack of more durable materials for the 3D printer, as well as the time to redesign and cost to print more prototypes, the team decided to purchase and modify commercially available propellers.

The team also determined that the selected brushless motor could be adequately waterproofed using silicone spray. This takes advantage of the lack of physical contacts inside a brushless motor. The electromagnet windings are already coated in insulation and the team took an added precaution of drying the motors with compressed air and reapplying the silicone spray after each time in the water.

7.1.4 Camera

The team required that SWIM-R contain a video camera that is able to stream video to the user. A video feed is necessary to allow the user to successfully drive SWIM-R in an aquatic environment which may not be visible from the surface. Reasons that SWIM-R may not be visible from the surface include being too far underwater for the human eye to see and being beneath a structure that obscures vision from the surface.

A camera that is able to capture and send still images periodically will function similar to a video camera that can stream its video. A still image camera can be made to look like a video camera by decreasing the amount of time between sent images. The drawback of a still image camera is the image stream will appear like low frame-rate video to the user. While this drawback should not affect the performance of the system (movement is typically slow underwater) it will affect the user experience.
A video camera that streams video as it captures it is the only alternative that provides usable output to inform the user of the position of SWIM-R and uphold the quality of the user experience. The camera should have the ability to choose a video resolution in order to fit the data rate of the communication link from the sub to the user.

The Raspberry Pi is compatible with a variety of USB webcams\textsuperscript{22}. The team experimented with a Logitech C120 Webcam and the OpenCV python package for reading the webcam. The system worked well between two laptops, but when ported to the Raspberry Pi the team was unable to achieve a frame rate above four frames per second. The team identified two solutions to still achieve a suitable camera feed; the first was purchasing an IP webcam and bypassing the Raspberry Pi, and the second was over-clocking the Raspberry Pi.

The team decided to pursue purchasing an IP webcam and splitting the Ethernet tether as shown in Figure 4-5. This decision was made because the team did not want to risk damaging the Raspberry Pi or reducing its lifetime by over-clocking it.

7.1.5 Testing and Prototyping
Team SWIM-R decided to do early prototyping with brushed DC motors. The low cost of the brushed motors was the deciding factor. The team purchased four each of a 7 Volt DC motor (FF-050S-11170-R) and 12 Volt DC motor (MD5-1885-R) along with three H-Bridge integrated circuits (L298N) for use in SWIM-R 0.1.

The team conducted tests using a variety of brushed DC motors shown in Figure 7-10 to determine the effect of running a motor in water. The motors were powered directly from a benchtop powersupply.

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{motor_image.png}
\caption{Brushed DC motors used in testing}
\end{figure}

Each motor was run first in air to determine a baseline current draw. The team then fixed the rotor in place to determine the motor stall current. Each motor was then submerged in water and run for about 5 minutes, ensuring that no air bubbles remained in the motor, and the current draw was recorded. The results are recorded in Table 7-2.

\textsuperscript{22} http://elinux.org/RPi_VerifiedPeripherals
Table 7-2: Test results for DC motor water testing

<table>
<thead>
<tr>
<th>Motor</th>
<th>In Air</th>
<th>In Water</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Voltage (V)</td>
<td>Current (mA)</td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>20.1</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>157</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>15</td>
</tr>
</tbody>
</table>

The test results in Table 7-2 show that running a DC motor with no waterproofing in water increases the current draw, but the current draw does not approach the motor stall current. This means that it is possible to run a DC motor underwater.

Once the team acquired their brushless DC motors, similar testing was conducted to determine how much current draw could be expected from each motor. A single motor was mounted to a plastic bracket and then inserted into a bucket of water. The motor was then spun at different speeds and the current drawn was recorded in a spreadsheet. Figure 7-11 shows the results of this test.

![Figure 7-11: Brushless DC Motor Current Draw vs. Throttle](image)

From Figure 7-11 it can be seen that when the motors are driven at 50% throttle they draw approximately 7.5 amps. Due to the exponential shape of the curve the team did not think it wise to drive the motors above 50% throttle because if the motors were to be driven at a greater rate there would be risk of drawing more than the 12 amps the motors are rated to handle. This test allowed Mitch to establish a benchmark for programming the motor driving software to ensure that this 50% throttle would not be exceeded by any motor driving operations in the Arduino software.

7.1.6 Power System

SWIM-R is powered from internal batteries. Alternative ways to power the sub included an on board fuel cell, renewable (e.g. solar) power, and high voltage AC through a tether. Design criteria for the power system included portability, reliability, and light weight.
7.1.6.1 Fuel Cells

Fuel cells were quickly dismissed because there are many drawbacks to using fuel cells. One major issue is temperature requirements. Most fuel cells require a very stable operating temperature to function properly without experiencing thermal loading and rapid failure. The actual operating temperatures vary between types of cells but for all cells the temperature must be stable. Keeping this temperature stable would be a challenge because the reaction that generates the electricity is highly exothermic and generates excessive heat inside the cell.\(^{23}\) SWIM-R will be exposed to a wide variety of temperatures which means that it would be necessary to equip SWIM-R with a thermally insulated chamber to house the cells. This would add unwanted weight and cost to the unit.

Another problem with using fuel cells is that the cells require a very precisely controlled flow of chemical ingredients in order to operate. The particular fuel is brought into one side of the cell and air containing hydrogen is brought into the other and they are reacted across the electrolyte via an anode and cathode. If one ingredient begins to flow too fast or slow the fuel cell will rapidly break down\(^ {24}\). Precise ingredient control requires a costly control system to regulate the flow of each chemical into the cell.

7.1.6.2 Renewable Power

Renewable power offers many advantages. Powering the entire unit using only renewable sources of power (e.g. solar, thermoelectric, wind) offers the benefit of limitless operating time and minimizes the environmental impact of using batteries or fuel. However, these methods only work in the correct conditions; the power output from a wind turbine decreases when there is no wind.

Power generation using solar or wind must be done on the surface and transported to the Sub. Much of the small amounts of power generated using these methods is lost over the long copper tether. As an example, the 10 Watt solar panel available from SparkFun outputs 10W at 8V and 1.25 A.\(^ {25}\) 24 AWG has a resistance of 26.2 Ω per 1000 feet.\(^ {26}\) Over a 100 foot length of 24 AWG copper wire connected to the 10W solar panel, the voltage drop due to the resistance of the wire alone is calculated by Equation 7-1.

**Equation 7-1: Voltage Drop across 100 feet of 24 AWG copper wire**

\[
V_{\text{drop}} = 1.25 \, A \times \frac{26.2 \, \Omega}{1000' \times 100'} = 3.275 \, V
\]

The voltage drop of 3.275V from 8V results in a voltage at the end of the wire of 4.725V, too low to power a 5V device.

Due to the variable nature of renewable power, Team SWIM-R rejected it as an option for the Sub’s main power source. Thermoelectric power generation was under consideration for a means of trickle charging the Sub’s batteries. Heat generated by the Sub’s motors and motor drivers must be dissipated and this heat could be used in a heat exchanger to generate electrical power. The team came to the conclusion that incorporating this process in the Sub design was beyond the scope of the project because the LiPo batteries incorporated in the design must have their cells balanced while charging and the time demands of perfecting this feature would have caused the project to be delayed beyond the deadline.


\(^{25}\) https://www.sparkfun.com/products/9759

\(^{26}\) http://www.engineeringtoolbox.com/copper-wire-d_1429.html
7.1.6.3 Power over Tether

Many conventional ROVs are powered over a copper tether using AC due to the losses that occur in DC over the long copper wires. In order for this method to be effective in the field, a separate AC power source (i.e., generator, DC inverter) must be used to supply power where a wall outlet is not available. Additionally, this requires the sub to have an onboard AC-DC converter to transform the high voltage AC into usable DC voltage. Both of these issues violate the desires to be small, portable, and inexpensive. Additionally, the Computer-Float-Sub design would require the AC generator be placed on the Float. An AC generator unit on the Float would greatly increase its size and weight, resulting in a less mobile system.

7.1.6.4 Battery Power

Battery power offers mobility and extended operating time with the use of spare batteries. Additionally, the batteries used by the Sub will be rechargeable. The alternative to rechargeable batteries is non-rechargeable batteries. Non-rechargeable batteries are not feasible because they violate the desires to be inexpensive and eco-friendly. Rechargeable batteries greatly reduce the amount of waste SWIM-R will produce falling in line with the requirement to be environmentally friendly. While the cost of a single rechargeable battery is greater than the cost of a single non-rechargeable battery, the number of charge cycles that a rechargeable battery can go through more than make up the cost difference.

Batteries come in many capacities and chemistries. Common battery chemistries used in remote control electronics include Lithium Polymer (LiPo) and Nickel-Metal Hydride (NiMH). Team SWIM-R chose to use LiPo batteries because they have a high charge density and a greater ability to source high current than NiMH batteries. Also, LiPo batteries do not have any kind of charge memory so charging the batteries when the battery has not been fully discharged does not affect future discharge cycle duration.

The electronic hardware in the Sub requires two voltage levels. One voltage level is used to power the Sub’s motors. This voltage level is between 10V and 12V depending on the charge state of the battery. The second voltage level powers the Sub’s electronics including the Raspberry Pi, Arduino, and sensors. This voltage level is 5V. Team SWIM-R achieved these two voltages using a DC battery in the operating range of the motors and by designing a voltage regulation circuit for powering the 5V devices. An alternative to this approach was to use two separate batteries at different voltages, but this would have created a more complicated system for the user. A schematic of the circuit used in the Sub is shown in Figure 7-12 and the PCB layout is shown in Figure 7-13.

Figure 7-12: Sub Power Regulation Circuit
The team determined that the current draw from the Sub’s motors was the largest component of the total current draw. The team purchased 3 cell, 5 amp-hour batteries. These batteries allowed the SWIM-R system to be driven for at least one hour without having to change the batteries under normal operating conditions.

7.2 Software Design

This section provides details on the software design of the Sub. Team SWIM-R developed custom software to run on both the Raspberry Pi and the Arduino board.

7.2.1 Raspberry Pi Software

The Raspberry Pi is an embedded computer that runs the Debian-based operating system (OS) known as Raspbian “wheezy”. Team SWIM-R made the decision to use an OS in order to handle the Sub’s multiple I/O streams. Raspbian is the supported OS for use in Raspberry Pi development. Team SWIM-R selected Python 2.7 as the programming language for development on the Raspberry Pi. Python is included in the standard distribution of Raspbian. Upon startup, the Raspberry Pi initializes the Arduino and begins looking for the Computer. Once all connections have been established, the Raspberry Pi continuously sends new data to the computer and writes new user commands to the Arduino. This communication software is detailed in section 38.


27
7.2.2 Arduino Software

The code for an Arduino microcontroller must include the functions setup and loop as an artifact of the Arduino programming language. This language also provides a serial interrupt handler function: SerialEvent. Figure 7-14 shows the processes in each of these three functions.

Figure 7-14: Arduino Software Diagram

The setup function is run at startup and contains code to initialize the serial connections between the Arduino and Raspberry Pi and between the Arduino and IMU. The setup function also contains code which attaches the Servo objects that drive the motors to PWM output pins on the Arduino.

After setup is complete the Arduino enters its main loop which runs continuously as long as the Arduino is powered on. This function contains code for reading control and environment sensors, reading and interpreting user commands, reading IMU data, calculating the proper speed of each motor, driving the motors with those speeds, packaging and sending information to the user, and returning the Sub to the surface in the event of a system error.
7.2.2.1 SerialEvent

The SerialEvent function is utilized in both the serial communication channels to the Arduino. SerialEvent is a built-in function of the Arduino cores that executes when data becomes available on the serial port. On the Arduino Mega which has four serial ports, the SerialEvent functions are named SerialEvent, SerialEvent1, etc. This function raises a flag which signals the Arduino to execute its parsing functions for deciphering the incoming messages from either the Raspberry Pi or the IMU.

7.2.2.2 Data Parsing

The parsing functions that are used to interpret the user commands and the IMU data operate in a similar fashion. They use a set of delimiter characters to mark the beginning and end of each packet as well as divide the important pieces of the packet so that they may be split up into different values. Figure 7-15 shows a small piece of the code that deciphers the incoming IMU data.

```java
// Next 6 statements are used to mark indices of the delimiter characters
int colon_1 = IMU_input_string.indexOf(' :');
int comma_1 = IMU_input_string.indexOf(',', 1);
int colon_2 = IMU_input_string.indexOf(' :', colon_1 + 1);
int comma_2 = IMU_input_string.indexOf(',', comma_1 + 1);
int colon_3 = IMU_input_string.indexOf(' :', colon_2 + 1);
int star = IMU_input_string.indexOf(' *', colon_3 + 1);

String roll_string = IMU_input_string.substring(colon_1 + 1, comma_1);
String pitch_string = IMU_input_string.substring(colon_2 + 1, comma_2);
String yaw_string = IMU_input_string.substring(colon_3 + 1, star);

// Conversion from strings to ints
parsed_IMU_data[1] = roll_string.toInt();
parsed_IMU_data[2] = pitch_string.toInt();
parsed_IMU_data[3] = yaw_string.toInt();
```

Figure 7-15: IMU Parsing Code

The IMU sends the delimiter characters as part of its message and the Arduino marks the indices of those characters and then creates strings based on those character indices. These strings are then cast as int type variables which can be used in calculating motor speeds.

7.2.2.3 Motor Speed Calculation

After the Arduino has read the latest set of user commands and incoming IMU data it computes what speeds to drive each motor. This is done using a complex algorithm. One thing to note is that the user commands which come to the Arduino are within the range of 0 to 255 with 127 being neutral and the two extremes being full speed in either direction. These inputs are mapped across the range -250 to 250 which is the maximum increment for each motor’s throttle. This throttle increment was determined based on the data of the brushless motor test discussed in section 7.1.5.

Next the IMU values are mapped across the usable throttle range as well. The IMU values come to the Arduino as angles ranging from -180 to 180 depending on how the Sub is oriented. Once all of the values have been mapped, the Arduino runs through a series of conditional statements to determine whether the IMU throttle adjustments should be used to re-stabilize the Sub or if the user has sent commands to override the adjustments. During this process the appropriate throttle adjustment is added to the neutral throttle value for each motor.
After the throttle for each motor has been incremented or decremented the throttle values are constrained within the usable range again in case there was overlap between IMU adjustment and user command adjustment. Once constrained the throttle values are written to the Servo objects.

### 7.2.2.4 Motor Control

The Arduino controls the speed of the Sub’s motors using Pulse Width Modulation (PWM). PWM works by changing the width of an electrical signal in a pulse train. With an inductive load such as an electric motor, PWM can be used to simulate an analog signal to drive the motor at variable speeds. The width of the pulse is called its duty cycle. In the Sub’s software system a 1500 microsecond duty cycle is defined as neutral. Setting the duty cycle to 2000 or 1000 will drive the motor at full speed forward or backward respectively. In section 7.2.2.3 it was stated that the maximum motor increment in the forward direction is 250 microseconds due to current draw restrictions. If the user were to drive the Sub in the forward direction at full speed both the axial motor ESCs would receive a 1750 microsecond PWM signal. This signal is interpreted by the ESC into the correct three phase power signal which is used to create the physical forward movement of the Sub.

### 7.2.2.5 Data to User

The function which packages and sends data to the user interface is quite simple. Figure 7-16 shows the code which performs this operation.

```c
void send_data()
{
    float data_to_user[8];
    data_to_user[0] = paused_IMU_data[0];
    data_to_user[1] = paused_IMU_data[1];
    data_to_user[2] = paused_IMU_data[2];
    data_to_user[3] = 0.0;
    data_to_user[4] = internal_temp;
    data_to_user[5] = internal_humidity;
    data_to_user[6] = 0.0;
    data_to_user[7] = total_voltage;

    Serial.print("$$");
    Serial.print(data_to_user[0]);
    Serial.print(',");
    Serial.print(data_to_user[1]);
    Serial.print(',");
    Serial.print(data_to_user[2]);
    Serial.print(',");
    Serial.print(data_to_user[3]);
    Serial.print(',");
    Serial.print(data_to_user[4]);
    Serial.print(',");
    Serial.print(data_to_user[5]);
    Serial.print(',");
    Serial.print(data_to_user[6]);
    Serial.print(',");
    Serial.print(data_to_user[7]);
    Serial.print('$$");
}
```

**Figure 7-16: Data to User Function**

Each data packet sent to the user begins with three dollar signs to alert the Raspberry Pi that a new packet has arrived. Then each value is separated from the others by a comma and the packet is terminated by a pound sign. Each time the send_data() function is called in the Arduino loop the array elements are re-written with current values and sent to the user via the Raspberry Pi.
7.2.2.6  Inertial Measurement Unit

In the original Sub design the team considered buying separate control sensors to construct an IMU and then program the Arduino to interface with these sensors to calculate the Sub’s attitude in three dimensions. Team SWIM-R identified several algorithms for this process including the Kalman Filter, the Complementary Filter, and the Direction Cosine Matrix. The team was favoring using the Kalman Filter method because Jeff had some prior experience using this in the construction of a quad-copter. The sensors the team was considering using for the IMU would have been integrated into the system using the Arduino Wiring library that interfaces with the Arduino I2C port. This would have allowed the team to quickly develop a means of reading the digital sensors.

As described in Section 7.1.2.1 the team opted for purchasing a board that had all three sensors incorporated because of cost considerations. This allowed the team to segregate the position calculating algorithms onto the IMU board processor to avoid timing issues and software conflicts.

The code that came with the IMU worked extremely well for calculating the attitude of the Sub. The team did run into some issues in getting the Arduino to communicate reliably with the IMU. This was solved by adding delimiters to the outgoing stream of the IMU and using the function shown in Figure 7-17 to read the incoming stream from the IMU.

```c
void new_IMU_data_received()
{
  if( new_IMU_data_present)
  
    new_IMU_data_present = false;
    String valid_chars = "-,;\n01234567890";
    if( IMU_input_string.length() == 0)
    {
        char in_char;

        do
        {
            in_char = Serial1.read();
        } while( in_char != '
');

        while( in_char != '*')
        {
            in_char = Serial1.read();

            for( int i = 0; i < valid_chars.length(); i++)
            {
                if( in_char == valid_chars[i])
                {
                    IMU_input_string += in_char;
                    break;
                }
            }

            if( in_char == '!')
            {
                break;
            }
        }
    }
}
```

**Figure 7-17: Incoming IMU Data Function**

When new data arrives from the IMU a flag is raised that triggers this function. This function then reads the stream until an ‘!’ is found. Once an ‘!’ is found the Arduino concatenates each incoming byte to a string until a ‘*’ is found. This signals the Arduino to break from the function and divide the string into its roll, pitch, and yaw components as described in Section 7.2.2.2.
7.2.2.7 **Inertial Navigation**

Team SWIM-R initially planned to implement a navigation system which used a PID controller and the reading from the IMU to determine the needed thrust for each motor. Figure 7-18 shows a diagram of the control system which would have been replicated for each movement direction.

![Control System Diagram for Inertial Navigation](image)

**Figure 7-18: Control System Diagram for Inertial Navigation**

In Figure 7-18 the input from the user is the desired response (R) and the actual response of the system (Y) is the current rate of movement. The output from the PID unit will be scaled, added and then passed as the output speed to the motors.

This portion of the system was not included because of time constraints. The team would have had to purchase a separate PID unit and calibrate it for use with the system and find a way to interface the unit with both the IMU and the Arduino. Instead of incorporating a PID unit the team created a motor control module in the Arduino software which scales the motor speeds according to the angles reported by the IMU.

7.3 **Mechanical Design**

The mechanical design of the Sub must ensure that the electronics inside remain dry, allow access to the electronics for debugging and programming, and include a waterproof Ethernet port for connection to the Float. Additionally, the body of the Sub must include a ballast tank that the user will use to make the Sub neutrally buoyant. The user will manually add or remove weights to the ballast tank based on target depth and water density.

7.3.1 **Sub Hull**

Team SWIM-R originally intended to design the Sub enclosure to be operational down to a depth of 100 feet. The water pressure at 100 feet below water is 2.95 atmospheres, or 43.3 psi. This requirement was not met due to major difficulty in achieving a waterproof enclosure during the design process. The team was able to successfully design an enclosure that could withstand the pressure at a depth of 14.5 feet in

---

the Calvin College pool. The team decided not to test the enclosure at a greater pressure for fear of damaging their only enclosure late in the semester. Further testing of this part of the system could be carried out and is detailed in Section 11.4.

Team SWIM-R considered using a commercial-off-the-shelf waterproof enclosure for the outer hull of the Sub. The team did not expect to find a product that met the needs of SWIM-R exactly, but the team planned to modify a COTS product to suit its needs. An example of a COTS product that Team SWIM-R researched is the OtterBox Drybox 3500, which is waterproof to 100 feet.29

The other option that Team SWIM-R considered was a custom enclosure for the Sub constructed out of PVC or aluminum. This alternative offered the benefits of a custom fit for the Sub with the necessary ports built in that a commercially available product could not.

The team chose to use the custom enclosure approach because finding a product that was large enough to house all of the Sub components and customizable enough to accommodate all of the sensors and motors proved extremely difficult and would have been very expensive. The final design of the Sub includes a piece of 8 inch diameter acrylic tube fitted with polycarbonate end caps that were fabricated in Calvin’s metal shop. Constructing the Sub end caps out of polycarbonate allowed the team to incrementally add each port and test for water-tightness to ensure the reliability of the enclosure.

In addition to constructing the main body of the Sub the team had to come up with a method for mounting the motors to the Sub’s body. This was done by cutting pieces of 4 inch diameter acrylic tube and attaching them to custom polycarbonate brackets which were made by programming the mill in the Calvin College metal shop. These brackets were mounted to the Sub using acrylic cement. The enclosure was drawn in inventor shown in Figure 7-19.

![Figure 7-19 Sub Enclosure Inventor Drawing](http://www.otterbox.com/OtterBox-Drybox-3500/OTR3-3500S.default.pd.html?dwvar_OTR3-3500S_color=01&start=2&cgid=otterbox-3000-cases)
Figure 7-20 shows the Sub with the two end caps, the motor mounts, and the waterproof Ethernet port installed. The rear cap also has a port that allows temperature and pressure sensors to be extended outside the Sub to measure the water temperature and total pressure for calculating the Sub’s depth. Figure 7-21 shows the installation of the pressure sensor in the back cap.

Figure 7-20: Sub Enclosure

Figure 7-21: Pressure Sensor Installation
7.3.2 Access Port
Team SWIM-R required the ability to access the electronics of the Sub for programming and development. The Drybox offered a solution as it has a watertight lid that can be opened as needed. Once the team had decided to use a custom enclosure they had to design this access port themselves. The team decided that building an access port into the side of the Sub posed too great a threat to the internal electronics of the Sub. Instead, the rear cap of the Sub was designed to be removable and all of the electronics were placed on a shelf that could be slid in and out of the Sub easily for programming purposes.

7.3.3 Waterproof Ethernet Port
The Sub is connected to the Float by means of an Ethernet cable. Team SWIM-R desired that the Ethernet cable be removable from the Sub to ease transportation and development. For this reason a waterproof Ethernet port was installed in the rear cap of the Sub. This allows the cable to be detached from the Sub at any time. The Ethernet ports that the team purchased for this purpose are a female PX0833 and a male PX0834/B. These connectors are IP68 rated which means they are designed to withstand prolonged submersion in water. The team bought two pairs of these connectors. One for the Sub and one for the Float.

7.3.4 Ballast
A neutrally buoyant Sub demands less power to hold its depth in the water than a non-neutrally buoyant one. The decrease in power consumption increases the operating time of SWIM-R. The Sub required a means of adding or subtracting weight to make the unit neutrally buoyant in water. Team SWIM-R considered two types of ballasting for the Sub: passive and active.

7.3.4.1 Passive Ballast
The passive ballast system consists of a cavity in the body of the Sub used to hold weights or air. The user will be required to adjust the ballasting of the Sub before use to make the Sub neutrally buoyant. The user is required to do this because all bodies of water are different and the density of salt and fresh water is not the same. A production model of SWIM-R would be near neutrally buoyant, minimizing the additional weight to be added by the user. This approach is less costly, requires less hardware, and is easier to prototype than the active ballast system. This approach is also more time consuming and less user friendly.

7.3.4.2 Active Ballast
The active ballast system would automate the buoyancy of the Sub through a system of pumps and water tanks. The sensors on the Sub would be able to approximate its buoyancy based on inertial movement when no motors are providing thrust and pump water in or out of its ballast tanks accordingly. This approach reduces the complexity of operating the Sub and allows for a “just-throw-it-in-the-water” system. This approach also increases the amount of hardware and sensors needed on the Sub and thus increases the cost.

The team decided to use the passive ballast method in the final design of the Sub. This decision was made because the complexity of adding an active ballast system to the Sub was beyond the scope of the project. If the project had more development time this portion of the Sub may have been feasible but with the given time constraint it was deemed unnecessary for project success.

The passive ballast is implemented in the form of a cavity on the underside of the electronics shelf outfitted with brackets to hold additional weights as needed to make the Sub neutrally buoyant.
8 Prototypes

Team SWIM-R adopted an agile development approach that encouraged rapid, iterative prototyping. For this reason, the team was able to construct multiple prototypes over the course of the school year, resulting in a fully functional system in May of 2013. Each successive prototype built on the achievements of the previous ones and reused both components and software. Features were added to the prototypes in a way that basic features present in one prototype were expanded in the successive prototypes.

8.1 SWIM-R 0.1

SWIM-R 0.1 is first in the series of prototypes. The purpose of this prototype was to test motor functionality underwater. Mitch was responsible for writing the basic motor control software in the Arduino IDE which was used to drive the motors during the tests. The target completion date for SWIM-R 0.1 was December 1, 2012. This prototype was actually completed by November 16 which allowed the team to do the demo for the special Engineering Fridays at Calvin.

8.1.1 System

Only the Sub portion of the SWIM-R system was present in SWIM-R 0.1. The Sub consisted of an aluminum frame, motors mounted to the frame, and foam used for floatation. No waterproofing was done to the Sub so all of the electronics besides the motors were kept separate from the Sub. Waterproofing was not done for this prototype because this was not necessary to test the motor functionality and would have pushed the schedule back further into December.

8.1.2 Hardware

The hardware present on SWIM-R 0.1 included motors, H-bridge motor driver integrated circuits (ICs), an Arduino microcontroller, and a computer with the Arduino IDE installed. The Sub had a total of two motors mounted to it for the demo on November 16 and had a third added to it on November 20 to test for vertical position adjustment in the water.

8.1.2.1 Motor Drivers

The team purchased three H-Bridge motor driver ICs. The part number of the IC is L298N and was purchased from Jameco Electronics. The schematic for the circuit that was used to drive the motors is pictured in Figure 8-1.
8.1.3 Software
Mitch developed Arduino code to receive bytes from the serial monitor on a PC to drive a DC motor. The software was structured to receive two char’s from the stream. The first dictated the direction and the second the speed. To make the control scheme transparent the WSAD control structure was used for the
direction with ‘W’ being forward, ‘S’ being reverse, ‘A’ being left, and ‘D’ being right. The speed char
was an integer from 0 to 9 which was mapped across the interval 0 to 255 to match the resolution of the
Arduino PWM library. Linked below is a link to a video showing a visiting student driving SWIM-R 0.1:

http://youtu.be/phkalFbFtj0

8.1.4 Mechanical
SWIM-R 0.1 did not have a waterproof compartment for electronics. Therefore, the mechanical design of
the system was minimal. The team did design the frame, built it out of aluminum found in the Calvin
College metal shop, and mounted the motors to it.

8.1.4.1 Motors
Team SWIM-R purchased four each of two different DC motors for use in SWIM-R 0.1. The motors
purchased were 9 and 12 Volt brushed DC motors from Jameco Electronics. The team decided to use
brushed DC motors for SWIM-R 0.1 due to their low cost and high availability. The motors in SWIM-R
0.1 were powered from a bench-top power supply.

8.2 SWIM-R 0.2
The purpose of SWIR-R 0.2 was to incorporate the Raspberry Pi and a Wi-Fi router into the
communication line. This prototype was to include rudimentary wireless communication to send
commands to the Sub and receive data back. Additionally, this prototype planned to have a rudimentary
user interface which would display sensor information. The target completion date for this prototype was
February 25, 2013; however the team missed this mark and completed SWIM-R 0.2 on March 15, 2013
with a successful water test in Calvin College’s pool.

8.2.1 System
This prototype included versions of the Computer, Float, and Sub. The Computer was able to
communicate wirelessly with the Float, display some information on-screen for the user, and recognize
user input steering commands from the keyboard. The Float consisted of a router powered with a 120V
wall outlet. The Float did not have an enclosure, and thus was on the pool deck, not in the water. The
Sub was waterproofed and powered by batteries, but did not house any sensors.

8.2.2 Hardware
The added hardware for this prototype included a battery for the Sub electronics, the Raspberry Pi
development board, a Wi-Fi router, the Computer, and regulation circuit built on a breadboard to power
the Raspberry Pi and Arduino boards. Additionally, the Sub included two brushless DC motors and ESCs
to allow forward, backward, and turning movements.

8.2.3 Software
The addition of more hardware required new software to be written to make use of it. New software in
SWIM-R 0.2 included Python code to communicate serially with the Arduino and Python code to set up
the Raspberry Pi as a network server using the SocketServer Python module. The team also wrote
Arduino code to control the speed of the brushless motors. Also, the team wrote software to
communicate wirelessly with the Float using a laptop.

http://www.youtube.com/watch?v=YM1MrEZ_zjo
8.2.4 Mechanical
SWIM-R 0.2 required a complete rebuild of the Sub due to the added electronics. The enclosure had to be waterproof at least to the depth of the bottom of the pool (14.5 feet). This was necessary to keep the Sub’s control sensors, development boards, and the battery safe. Additionally, this prototype had a data tether from the Sub to the Float. This consisted of a waterproof Ethernet port on the Sub and a waterproof Ethernet cord attached to the port and the router. The mechanical design was primarily responsible for the delay of this prototype. The team was able to locate an 8 inch diameter piece of clear acrylic tube for the main body of the Sub to design around. Mitch and Jeff were able to machine caps for the tube that can be removed; yet provide a reliable water-tight seal using silicone O-rings.

8.3 SWIM-R 0.3
The purpose of SWIM-R 0.3 was to have proof of concept for every feature of the final SWIM-R prototype present. This prototype prepared the team for the final development stage, SWIM-R 1.0. The target completion date for SWIM-R 0.3 was April 5, 2013. The team completed this prototype on April 12, 2013 with a demonstration to high school students at the Engineering Fridays at Calvin event. This prototype was completed behind schedule, but was not as far behind schedule as SWIM-R 0.2.

8.3.1 System
This prototype had the Computer, Float, and Sub in nearly final form. The Computer was able to display sensor data from the Sub, however there was no camera on the Sub. The team had developed a video feed system, tested it with a Raspberry Pi, and determined that the Raspberry Pi could not keep up with the demands of video capture. The Float was an enclosed unit, powered by battery, and was able to float. The Sub was an enclosed unit, powered by battery, and had ballast rods to adjust the buoyancy to neutrally buoyant. The Sub also incorporated data logging temperature and humidity sensors and streamed the data to the Computer.

8.3.2 Hardware
The hardware added to the system in SWIM-R 0.3 includes a battery and regulation circuit for the Float, data logging sensors, and additional motors. The team designed and constructed three PCBs for this prototype: one each for the power regulation in the Sub and Float, and a third for attaching sensors to the Arduino in the Sub.

8.3.3 Software
New software written for SWIM-R 0.3 included Python code to capture camera data and stream it through the Raspberry Pi’s Ethernet port, Arduino code to read the temperature and humidity sensors, and a higher fidelity motor control system. The motor control system involved reading the user input and the Sub’s IMU to determine whether the Sub had rotated too far off course and correct it. The Arduino software also included a battery monitoring system that reads the voltage on the Sub’s batteries and determines whether they have dropped to an unsafe voltage. The Computer was able to display a bank of sensor readings on screen.

8.3.4 Mechanical
For SWIM-R 0.3 the team designed and built an enclosure for the Float. The enclosure consisted of a COTS dry-box which the team was able to add a port to for the Ethernet tether. The team determined that the SWIM-R 0.2 enclosure for the Sub was satisfactory moving forward and decided not to redesign it. This was influenced by the amount of time and money needed to fabricate or purchase a different enclosure.
8.4 SWIM-R 1.0

SWIM-R 1.0 was the culmination of nine months of hard work by Team SWIM-R. SWIM-R 1.0 was the prototype on display on Calvin Senior Design night on May 4. This prototype resembled SWIM-R 0.3 with improvements made to the code and superfluous features added. This prototype demonstrated Team SWIM-R’s abilities as budding electrical engineers. The team finished this prototype on May 3, 2013 after a full week of pool testing.\(^{31}\)

8.4.1 System

SWIM-R 1.0 included the Computer, Float, and Sub in their final forms. The Computer consisted of a python script that can be run from the command line with a single statement. The Float consisted of an enclosed unit with an internal battery and power system, as well as a waterproof Ethernet port to which the tether will attach. The Sub included two internal batteries wired in parallel and a power regulation system along with a circuit to monitor the battery charge. The Sub also included a video camera and sensors for data logging and control.

8.4.2 Hardware

This prototype added the additional pressure and temperature sensors in the water. The pressure sensor is an absolute pressure sensor which the Arduino uses to calculate depth under water. The Sub also included a fifth motor and a fan for heat distribution inside the Sub.

8.4.3 Software

For SWIM-R 1.0 the team improved the quality, reliability, and reliability of the code. Additionally, the team experimented with the speed of communications in an attempt to improve latency. The Computer application also had the ability to log data and video to be viewed at a later time, although these features are not fully developed or well tested. The Computer supported a USB gamepad as an input device, specifically an XBOX 360 controller.

8.4.4 Mechanical

The mechanical design of SWIM-R 1.0 remained largely unchanged from SWIM-R 0.3 with the exception of fine-tuned ballasting and weight distribution in the Sub and Float. SWIM-R 1.0 is pictured in Figure 8-3.

\(^{31}\) http://www.youtube.com/playlist?list=PLdFfWvWkG89HjXx2ihQ9UKf3vVwdksOfrD
Figure 8-3 SWIM-R 1.0
9 Financial Estimates

SWIM-R’s financial estimates take into account the cost of research, development, and production. They also analyze potential competitors and customers to estimate the monetary value of SWIM-R to the end user.

9.1 Marketing Study

The marketing study is comprised of some examples of competition SWIM-R would have as well as examples of potential customers.

9.1.1 Competition

There are several competitors in the underwater ROV market.

9.1.1.1 OpenROV

This product stems from a community of Do-It-Yourself tele-robotics centered on underwater exploration and education. It can reach depths of up to 100 meters and is going to be offered as a variety of do-it-yourself kits. The robot weighs in at 2.5 kg and has a battery life of approximately 1.5 hours. It has 3 onboard brushless DC motors and is not capable of full 6 degrees of motion. OpenROV released its first kit in early 2013. This kit sells for $849.00 on the OpenROV website. If a customer were to build their own kit from the Bill of Materials published by OpenROV they would have to spend $604.22.

While SWIM-R is similar to the OpenROV project it remains distinct by providing wireless control and full 5 degrees of freedom. SWIM-R also uses an Arduino as a dedicated motor controller, which allows the Raspberry Pie to focus on processing other information.

9.1.1.2 Dawson’s underwater robot

Another competitor is Dawson’s underwater robot for $700. However, this robot serves a slightly different purpose than SWIM-R. Dawson’s underwater robot does provide a view to the user from underwater, but the system is merely a claw attached to a cable with a camera on it. Its primarily purpose is to retrieve items from the bottom of shallow bodies of water. In order to do this the user must be directly above the item of interest. The robot is capable of lifting objects up to 220lbs, uses high quality marine cable, and has an 8 hour battery life. The robot’s camera can be adjusted to be able to inspect a boat’s hull, but cannot be adjusted by computer while under the water. It must be adjusted to the correct angle before lowering it under, making it relatively inconvenient for the duty of inspection.

SWIM-R is a mobile version of the Dawson robot. While they both provide video feed, the SWIM-R does not have the ability to pick up objects, but is inherently more mobile than Dawson’s robot.

9.1.2 Market Survey

There are a variety of uses for an ROV including hobby projects, boat inspections at marinas, and underwater research.

9.1.2.1 Example Customer: Shoals Marine Laboratory

Shoals Marine Laboratory (SML) is a laboratory in Maine that offers undergraduate classes in marine sciences. Cornell University and the University of New Hampshire operate SML. They offer a two-week course.

---

32 http://openrov.com/
In order to take this class a student must acquire SCUBA certification. SCUBA certification is time consuming and costly ranging from about $400-$700. In addition, the students need to provide their own SCUBA equipment. If instead of actual diving, SML gave the alternative of using a SWIM-R to do the research they could offer the class to more students who cannot afford or obtain SCUBA certification. Additionally, SWIM-R kits would be reusable by students from year to year, adding additional value. For example if two students could use one SWIM-R kit instead of paying for SCUBA gear, SWIM-R would be worth approximately $1200. Allowing students access that otherwise couldn’t is an example of SWIM-R conveying justice.

9.1.2.2 Example Customer: The Hobbyist

SWIM-R is also targeted at the DIY Hobbyist. There is an existing community around building and customizing quad-copters. SWIM-R hopes to fill a similar niche but instead in the underwater world. A typical quad-copter kit costs between $250 and $900 dollars. SWIM-R is more sophisticated system and is not intended to compete with but instead complement the established interest in ROVs.

9.2 Business Model

Team SWIM-R has decided that the SWIM-R system will be marketed as a “do-it-yourself “ kit. The kit will contain all the components necessary to build a SWIM-R system. DIY kits allow for increased customer participation in the creative process as well as decreased assembly costs. DIY kits are common in the ROV community and SWIM-R would follow suit. Fully assembled version of SWIM-R will be available for customers to order if they so desire. Team SWIM-R foresees these customers being individuals or organizations that intend to use the system for research purposes and are not interested in the DIY aspect of SWIM-R.

9.3 Cost Estimate

The cost estimate is comprised of non-recurring engineering (NRE) costs and Production costs. The details of what each of these categories include is featured in the following sections.

9.3.1 Non Recurring Engineering (NRE)

The NRE cost for SWIM-R is comprised of an hourly rate for Engineering design time as well as the cost of the materials to test and prototype.

9.3.1.1 Design Time

Between the project start date of September 17, 2012 and end date of May 3, 2013 the team logged a total of 1700 hours. The team estimated in November of 2012 that they would spend approximately 1550 hours on the project. More details on the development time of SWIM-R are located in Section 10.2. At a rate of $100/hour/engineer the total design time cost is $170,000. This is the price that would be charged to a customer by a development firm that takes into account all fringe benefits, salaries, and profit for the firm.

---

34 [http://www.sml.cornell.edu/sml_cc_ur.html](http://www.sml.cornell.edu/sml_cc_ur.html)
36 [http://store.diydrones.com/](http://store.diydrones.com/)
### 9.3.1.2 Prototyping Material Cost

During the first semester of the project the team assembled an initial estimate for the cost of prototyping and development. The table below shows an itemized list of components that the team submitted as its operational budget estimate to the Engineering Department.

Total cost: $1460

<table>
<thead>
<tr>
<th>Part</th>
<th>Quantity</th>
<th>Unit Cost</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi</td>
<td>1</td>
<td>$40.00</td>
<td>$40.00</td>
</tr>
<tr>
<td>Arduino</td>
<td>1</td>
<td>$30.00</td>
<td>$30.00</td>
</tr>
<tr>
<td>DC Motors</td>
<td>4</td>
<td>$3.00</td>
<td>$12.00</td>
</tr>
<tr>
<td>DC Motors</td>
<td>4</td>
<td>$3.00</td>
<td>$12.00</td>
</tr>
<tr>
<td>DC Motors</td>
<td>6</td>
<td>$4.00</td>
<td>$24.00</td>
</tr>
<tr>
<td>H Bridge</td>
<td>4</td>
<td>$3.00</td>
<td>$12.00</td>
</tr>
<tr>
<td>Brushless DC motors</td>
<td>6</td>
<td>$18.00</td>
<td>$108.00</td>
</tr>
<tr>
<td>Speed Controllers</td>
<td>6</td>
<td>$15.00</td>
<td>$90.00</td>
</tr>
<tr>
<td>Propellers</td>
<td>10</td>
<td>$1.00</td>
<td>$10.00</td>
</tr>
<tr>
<td>Wi-Fi-router #1</td>
<td>1</td>
<td>$30.00</td>
<td>$30.00</td>
</tr>
<tr>
<td>Wi-Fi-router #2</td>
<td>1</td>
<td>$70.00</td>
<td>$70.00</td>
</tr>
<tr>
<td>Accelerometer</td>
<td>1</td>
<td>$30.00</td>
<td>$30.00</td>
</tr>
<tr>
<td>Gyro</td>
<td>1</td>
<td>$50.00</td>
<td>$50.00</td>
</tr>
<tr>
<td>Magnetometer</td>
<td>1</td>
<td>$50.00</td>
<td>$50.00</td>
</tr>
<tr>
<td>100’ Ethernet Cable</td>
<td>1</td>
<td>$100.00</td>
<td>$100.00</td>
</tr>
<tr>
<td>Protoboard</td>
<td>1</td>
<td>$30.00</td>
<td>$30.00</td>
</tr>
<tr>
<td>Custom PCB</td>
<td>2</td>
<td>$50.00</td>
<td>$100.00</td>
</tr>
<tr>
<td>USB webcam</td>
<td>2</td>
<td>$60.00</td>
<td>$120.00</td>
</tr>
<tr>
<td>Waterproof Ethernet Ports</td>
<td>2</td>
<td>$20.00</td>
<td>$40.00</td>
</tr>
<tr>
<td>OtterBox</td>
<td>2</td>
<td>$30.00</td>
<td>$60.00</td>
</tr>
<tr>
<td>Data Sensors</td>
<td>4</td>
<td>$50.00</td>
<td>$200.00</td>
</tr>
<tr>
<td>Marine grade silicone</td>
<td>1</td>
<td>$30.00</td>
<td>$30.00</td>
</tr>
<tr>
<td>Batteries</td>
<td>4</td>
<td>$40.00</td>
<td>$160.00</td>
</tr>
<tr>
<td>Analog Circuit elements</td>
<td>1</td>
<td>$28.00</td>
<td>$28.00</td>
</tr>
<tr>
<td>SD card</td>
<td>2</td>
<td>$10.00</td>
<td>$20.00</td>
</tr>
<tr>
<td>USB cable</td>
<td>2</td>
<td>$2.00</td>
<td>$4.00</td>
</tr>
</tbody>
</table>

Throughout the course of the second semester the team began to make adjustments to their spending based on project needs. Some of the items such as the Accelerometer, Gyro, and Magnetometer were not purchased. Instead the all-in-one ArduIMU V3+ mentioned in Section 7.1.2.1 was substituted for these sensors. Some items such as the third set of DC motors were not needed for testing so that money was instead dedicated toward purchasing higher quality propellers for the Sub’s drive system. A full breakdown of the team’s spending is reported below in Table 9-2 and Table 9-3.
Table 9-2: Adjusted Budget

<table>
<thead>
<tr>
<th>Part</th>
<th>Qty</th>
<th>Unit Cost</th>
<th>Budgeted</th>
<th>Spent</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi</td>
<td>1</td>
<td>40</td>
<td>40</td>
<td>$39.95</td>
<td>-$0.05</td>
</tr>
<tr>
<td>Arduino Uno</td>
<td>1</td>
<td>30</td>
<td>30</td>
<td>$0.00</td>
<td>-$30.00</td>
</tr>
<tr>
<td>DC Motors</td>
<td>4</td>
<td>3</td>
<td>12</td>
<td>$11.80</td>
<td>-$0.20</td>
</tr>
<tr>
<td>DC Motors</td>
<td>4</td>
<td>3</td>
<td>12</td>
<td>$13.00</td>
<td>-$1.00</td>
</tr>
<tr>
<td>DC Motors</td>
<td>6</td>
<td>4</td>
<td>24</td>
<td>$0.00</td>
<td>-$24.00</td>
</tr>
<tr>
<td>H Bridge</td>
<td>4</td>
<td>3</td>
<td>12</td>
<td>$8.25</td>
<td>-$3.75</td>
</tr>
<tr>
<td>Brushless DC motors</td>
<td>6</td>
<td>18</td>
<td>108</td>
<td>$18.00</td>
<td>$0.00</td>
</tr>
<tr>
<td>Speed Controllers</td>
<td>6</td>
<td>15</td>
<td>90</td>
<td>$108.00</td>
<td>$10.20</td>
</tr>
<tr>
<td>Propellers</td>
<td>10</td>
<td>1</td>
<td>10</td>
<td>$67.90</td>
<td>$57.90</td>
</tr>
<tr>
<td>Wifi-router #1</td>
<td>1</td>
<td>30</td>
<td>30</td>
<td>$34.99</td>
<td>$4.99</td>
</tr>
<tr>
<td>Wifi-router #2</td>
<td>1</td>
<td>70</td>
<td>70</td>
<td>$0.00</td>
<td>-$70.00</td>
</tr>
<tr>
<td>Accelerometer</td>
<td>1</td>
<td>30</td>
<td>30</td>
<td>$0.00</td>
<td>-$30.00</td>
</tr>
<tr>
<td>Gyro</td>
<td>1</td>
<td>50</td>
<td>50</td>
<td>$0.00</td>
<td>-$50.00</td>
</tr>
<tr>
<td>Magnetometer</td>
<td>1</td>
<td>50</td>
<td>50</td>
<td>$0.00</td>
<td>-$50.00</td>
</tr>
<tr>
<td>100’ Ethernet cord</td>
<td>1</td>
<td>100</td>
<td>100</td>
<td>$0.00</td>
<td>-$100.00</td>
</tr>
<tr>
<td>Protoboard</td>
<td>1</td>
<td>30</td>
<td>30</td>
<td>$0.00</td>
<td>-$30.00</td>
</tr>
<tr>
<td>Custom PCB</td>
<td>2</td>
<td>50</td>
<td>100</td>
<td>$0.00</td>
<td>-$100.00</td>
</tr>
<tr>
<td>USB webcam</td>
<td>2</td>
<td>60</td>
<td>120</td>
<td>$29.99</td>
<td>-$90.01</td>
</tr>
<tr>
<td>WP Ethernet Ports</td>
<td>2</td>
<td>20</td>
<td>40</td>
<td>$115.30</td>
<td>$75.30</td>
</tr>
<tr>
<td>Otterbox</td>
<td>2</td>
<td>30</td>
<td>60</td>
<td>$34.48</td>
<td>-$25.52</td>
</tr>
<tr>
<td>Data Sensors</td>
<td>4</td>
<td>50</td>
<td>200</td>
<td>$54.45</td>
<td>-$145.55</td>
</tr>
<tr>
<td>Marine grade silicone</td>
<td>1</td>
<td>30</td>
<td>30</td>
<td>$0.00</td>
<td>-$30.00</td>
</tr>
<tr>
<td>Batteries</td>
<td>4</td>
<td>40</td>
<td>160</td>
<td>$159.96</td>
<td>-$0.04</td>
</tr>
<tr>
<td>Analog Circuit elements</td>
<td>1</td>
<td>28</td>
<td>28</td>
<td>$35.78</td>
<td>$7.78</td>
</tr>
<tr>
<td>SD card</td>
<td>2</td>
<td>10</td>
<td>20</td>
<td>$8.99</td>
<td>-$11.01</td>
</tr>
<tr>
<td>USB cable</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>$7.95</td>
<td>$3.95</td>
</tr>
</tbody>
</table>

Table 9-2 compares the projected cost of SWIM-R with the actual cost of the components the team purchased. Items highlighted in green are items that came in under budget and things highlighted in red are items that cost more than estimated. The large gap in the cost of the propellers is due to the first set of propellers not being able to produce enough thrust to move the Sub. This was discovered in the testing of prototype 0.1. The team was able to reallocate funds from other portions of the project to make up this difference.

Table 9-3: Non-Budgeted Items

<table>
<thead>
<tr>
<th>Part</th>
<th>Qty</th>
<th>Unit Cost</th>
<th>Budgeted</th>
<th>Spent</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shipping</td>
<td>1</td>
<td>$51.52</td>
<td>$0.00</td>
<td>$51.52</td>
<td>$51.52</td>
</tr>
<tr>
<td>O-rings</td>
<td>6</td>
<td>$4.25</td>
<td>$0.00</td>
<td>$25.50</td>
<td>$25.50</td>
</tr>
<tr>
<td>IMU Board</td>
<td>1</td>
<td>$49.99</td>
<td>$0.00</td>
<td>$49.99</td>
<td>$49.99</td>
</tr>
<tr>
<td>3-D Printed Parts</td>
<td>1</td>
<td>$11.46</td>
<td>$0.00</td>
<td>$11.46</td>
<td>$11.46</td>
</tr>
<tr>
<td>IP webcam</td>
<td>1</td>
<td>$79.99</td>
<td>$0.00</td>
<td>$79.99</td>
<td>$79.99</td>
</tr>
<tr>
<td>Arduino Mega</td>
<td>1</td>
<td>$58.95</td>
<td>$0.00</td>
<td>$58.95</td>
<td>$58.95</td>
</tr>
<tr>
<td>4mm bullet extensions</td>
<td>5</td>
<td>$2.98</td>
<td>$0.00</td>
<td>$14.90</td>
<td>$14.90</td>
</tr>
<tr>
<td>Acrylic Dome</td>
<td>1</td>
<td>$40.50</td>
<td>$0.00</td>
<td>$40.50</td>
<td>$40.50</td>
</tr>
<tr>
<td>4mm bullet connectors</td>
<td>1</td>
<td>$10.95</td>
<td>$0.00</td>
<td>$10.95</td>
<td>$10.95</td>
</tr>
<tr>
<td>Deans connectors</td>
<td>1</td>
<td>$21.91</td>
<td>$0.00</td>
<td>$25.91</td>
<td>$25.91</td>
</tr>
</tbody>
</table>

83
During the course of the project some items were needed that had not been accounted for in the initial budget. Table 9-3 provides the details of these items. One of these items is the Arduino Mega. The original budget called out an Arduino Uno which is $30 cheaper than a Mega. A Mega was needed because of hardware conflicts described in Section 7. Another large item not in the original budget is the IMU board. This is a substitute item for the Accelerometer, Gyro, and Magnetometer in the original budget and the IMU board is actually cheaper than all three of those sensors. The other expensive item in Table 9-3 is the IP webcam. This item was found to be necessary during the testing of SWIM-R 0.3 as detailed in Section 7.1.4.

All of the items in the table were able to be purchased within budget. As of the project completion date of May 3, 2013 the team had exactly $259.34 remaining in their budget. This money was returned to the Calvin College Senior Engineering Design Budget.

9.3.2 Production
The production costs of SWIM-R are comprised of the labor and manufacturing expenses required to mass produce the system kits.

9.3.2.1 Labor
The team consulted with Phil Jasperse regarding the cost to produce the mechanical parts of SWIM-R including the Sub hull, the motor mounting brackets, and the Float enclosure. It was estimated that to produce 100 SWIM-R kits it would cost approximately $14,000. Included in the $14,000 is the typical contract cost of $80/hour for a technician in a machine shop which does include the overhead setup fees of the tooling. Also included is the material cost of $400 per sheet of polycarbonate along with $20 worth of acrylic tubing for each SWIM-R kit. It was estimated that each kit would take approximately 1 hour to machine and that a machinist could produce 10 sets of end caps and brackets from one polycarbonate sheet. Shown below in Equation 9-1 is the calculation showing how the total cost was computed.

Equation 9-1: SWIM-R Production Cost Calculation

\[
\text{Cost}_{\text{SWIMR}} := \left( \frac{80 \text{ USD}}{\text{hr}} \right) \times 100 \text{hr} + (20 \text{ USD} \times 100) + (400 \text{ USD} \times 10) = 14000 \text{ USD}
\]

9.3.2.1.2 Tooling and Manufacturing Equipment
The first major cost of manufacturing SWIM-R is PCB printing and populating. PCBs would be made that encompass the roles of the Arduino, Raspberry Pi, environment sensors as well as PCBs for the power supply circuitry in the Float and Sub. This process would be outsourced to a PCB printing company such as PCBExpress or AP Circuits. If boards were ordered in lots of 100 from AP Circuits the approximate cost of the boards would be $2,369 per lot.\footnote{http://apcircuits.com/} If this board were ordered from PCBExpress the total cost of one lot would be $1,135 per lot.\footnote{http://www.pcbexpress.com/products/prices.php#2pricing} The population of these circuit boards would be outsourced as well. If the boards were populated using Seeed the total cost would be $2,448 per lot of 100.\footnote{http://www.seeedstudio.com/propagate/index.php?controller=estimation&action=calculate}
9.3.3  Extraneous Costs
If the team were to pursue mass production of SWIM-R there are some other cost items that would have to be considered such as warehousing, distribution, advertising, and insurance. These items are covered in greater detail in the next few sections.

9.3.3.1  Warehousing
Because the team would plan to sell 100 kits throughout the course of each year but would produce all 100 units at one time to save on cost there would be a need for storage of un-purchased units. The team has estimated that they would be able to fit all 100 units inside a 10 foot x 10 foot x 8 foot self-storage container. They have quoted a price of $84/month for this unit which would result in a yearly cost of $1,008.40

9.3.3.2  Distribution
SWIM-R would not establish its own distribution network. Instead the team would utilize the existing network that has been created by UPS. The team has determined that a 2 foot x 1 foot x 1.5 foot box would be required to ship a SWIM-R system. The cost to ship units is completely dependent on where the package is being sent but the team calculated what it would cost to ship the package within the city of Grand Rapids to different locations and called this their minimum possible shipping charge. This was quoted at $24.11.41 The team has decided that if a customer were to place an order from a location that would cost more than $100 to ship the customer would be required to pay any additional shipping beyond the $100. This results in an annual cost ranging from $2,411 to $10,000 in shipping charges.

9.3.3.3  Advertising
The team does not plan to expressly spend any money on advertising. SWIM-R has already been featured on the Raspberry Pi Foundation’s website due to its use of the Raspberry Pi in development. The article posted on the website has generated widespread publicity reaching as far as the Middle East which has been tracked on Facebook.42 The team hopes to be featured in a similar fashion on the DIY Drones website for their use of the ArduIMU V3+ IMU board in the project. These two avenues of advertisement are projected to stimulate enough interest in the project that additional advertising would be unnecessary.

9.3.3.4  Insurance
The team has decided that because the project is open source and almost completely do-it-yourself project they will not be purchasing any insurance. The code and all parts of the kit will be covered in disclaimers that very clearly indicate to the user that they are agreeing to accept full responsibility for any injuries to themselves and others while operating the kit. The team has decided that this is the only way to proceed with the business model of SWIM-R because the applications that the kits could be used for are so widespread that publishing a user guide with specific usage instructions for each application would be impossible. However, a basic user guide instructing customers on the functionalities of the system will be published and sent out with each unit.

9.3.4  Kit Cost
After consolidating all of the cost factors the team estimates that it would cost $937.31 to produce each SWIM-R kit. The end cost to the user would be $1,200 to allow for the team to make a profit that would

40 http://www.extraspace.com
42 https://www.facebook.com/CalvinSWIMR?fref=ts
allow for them to continue developing improvements to the SWIM-R system and provide limited customer support in the form of an online forum. The team feels that their list price is quite comparable to that of OpenROV which is their main competitor because while SWIM-R is a bit more expensive, it has more features to offer its users. Figure 9-1 shows the calculations used to compute the overall cost to produce each SWIM-R kit. These costs only include the components that were used in the final prototype of SWIM-R and not the entire prototype budget.

\[
\text{Cost}_{\text{electronics}} = 687.23 \\
\text{Cost}_{\text{warehousing}} = 1008 \\
\text{Cost}_{\text{distribution\_best}} = 2411 \\
\text{Cost}_{\text{distribution\_worst}} = 10000 \\
\text{Cost}_{\text{production}} = \left( \frac{80 \text{ hour}}{100 \text{ kits}} \right) + (20 \times 100) + (400 \times 10) = 14000 \\
\text{Cost}_{\text{total}} = \text{Cost}_{\text{electronics}} + \frac{\text{Cost}_{\text{warehousing}}}{100} + \frac{\text{Cost}_{\text{distribution\_worst}}}{100} + \frac{\text{Cost}_{\text{production}}}{100} = 937.31
\]

**Figure 9-1: Kit Cost Calculations**

9.3.5 Break Even Analysis

Based on the calculations in Figure 9-1 the team calculated that it will cost them $93,731 up-front to produce 100 SWIM-R kits. That price is based on buying all of the components for the kits in lots of 100 in order to receive lower price breaks and does not include any engineering design time. This means that all of the costs involved are recurring costs and would be incurred with every lot of 100 kits produced i.e. each year. Engineering design time is not included in any cost calculations because this project was developed as an educational project so therefore none of the hours spent on the project are billable. Additionally, because the team is not hiring a firm to perform system development there is no non-recurring engineering cost incurred. The team is developing the system themselves and therefore only receives paybacks when customers purchase the system.
10 Project Management

10.1 Work Breakdown Schedule Summary
A work breakdown schedule (WBS) was used to ensure deadlines were met. Team 12 did their WBS online at KanbanFlow.com that had the deadlines for the project. It can be found on their website (http://www.calvin.edu/academic/engineering/2012-13-team12/documents.html). The WBS was regularly updated to allow for accurate planning throughout the first semester and January.

10.1.1 Work Breakdown Schedule Management
Team 12’s WBS on KanbanFlow.com was largely managed by both Jeff and Mike. KanbanFlow allows everyone on the team to change deadline due dates. This made the whole time responsible for keeping the board up to date. On average it was updated once a week and checked regularly.

10.1.2 Work Breakdown Schedule Description
The WBS on KanbanFlow was arranged into numerous categories such as Sub Software, Sub Hardware, Mechanical, User Interface, Power Requirements, In Progress, Done, etc. Each category has subsections that can be assigned to certain individuals.

10.1.3 Deprecation of KanbanFlow
Unfortunately, use of this tool fell off during the second semester of development. Instead the team relied on the schedule of prototypes found in Section 7 of this report. The prototyping schedule drove the development of features and kept the team on track in place of KanbanFlow.

10.2 Effort Estimation
After the first semester of development, the team had recorded 540 hours with the breakdown shown below in Table 10-1. The team also estimated about 35% project completion at this time. Extrapolating from these numbers, the team estimated that they would need a total of 1550 hours to complete the project. The team determined this was feasible as if assuming a six day work week, each member on the team would have to average about 1.7 hours of work per day or 10.2 hours per week.

<table>
<thead>
<tr>
<th>Name</th>
<th>Total Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeff</td>
<td>181</td>
</tr>
<tr>
<td>Mike</td>
<td>116</td>
</tr>
<tr>
<td>Jon</td>
<td>123</td>
</tr>
<tr>
<td>Mitch</td>
<td>120</td>
</tr>
<tr>
<td>Total</td>
<td>540</td>
</tr>
</tbody>
</table>

Table 10-1: Breakdown of Worked Hours in First Semester

Development ramped up during the second semester and the team completed the prototype on time with 1700 total work hours. The breakdown of hours throughout the project is shown in Table 10-2.
Table 10-2: Breakdown of Total Hours Worked throughout Project

<table>
<thead>
<tr>
<th>Name</th>
<th>Total Hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jeff</td>
<td>456</td>
</tr>
<tr>
<td>Mike</td>
<td>406</td>
</tr>
<tr>
<td>Jon</td>
<td>403</td>
</tr>
<tr>
<td>Mitch</td>
<td>435</td>
</tr>
<tr>
<td>Total</td>
<td>1700</td>
</tr>
</tbody>
</table>

After the first semester, the team was able to estimate the total hours needed for the project to within 10% of the actual hours worked. The team sees this as an accomplishment because time estimation is an important skill in industry.

10.3 Project Milestones

The team used an Agile approach to development for this project which required them to design and build a prototype, learn from it, and redesign for a new prototype. The team developed a total of four prototypes through this project, although the last three use many of the same parts. The major milestones for the project were the completion of these prototypes as detailed in Section 7 of this report. Figure 10-1 was used to track the progress SWIM-R.

![Figure 10-1 Project Milestone Chart](image)
11 Conclusion
This document summarizes nearly a year of work by Team SWIM-R. The project has reached its conclusion and left the team with many lessons learned.

11.1 Important Accomplishments
The team strived all year to have a prototype to be proud of and demonstrate during the Senior Design night open house. The team accomplished this and was able to test the system well before the open house in order to adjust controls. The team attributes this accomplishment to good time management and hard work put in by each member of the team throughout the year.

Additionally, the team was able to operate SWIM-R for the full two hours of the open house, double the specified minimum operation time. The open house was the longest test of the system and confirmed the assumptions the team made about Sub power consumption.

11.2 Shortcomings
A major shortcoming of the project development was neglecting to purchase a dome for the front of the Sub in time. The team ordered an 8 inch acrylic dome with the intention of mounting it on the front of the Sub with a camera inside. Unfortunately the dome did not arrive in time. The team was able to mount a webcam inside the Sub to demonstrate its ability to stream video; however the only available space required that the camera face vertically. The result was a video feed that was not helpful for piloting the Sub.

11.3 Lessons Learned
Throughout this project the team learned several lessons about product development and project management. The first lesson learned was that it is important manage time well and front load a project as much as possible. The team believes that they did a good job loading up work in the first semester of the project developing an early prototype in November.

A lesson learned by Jeff is how much effort goes into coordinating a team of people as a project manager. The task of managing the project was larger than anticipated and as a result Jeff worked to assist the other team members in all areas of the project.

Additionally the team learned that it is important to develop a bill of materials early in the project so that all parts can be ordered in plenty of time for implementation. This lesson is a result of ordering the dome for the front of the Sub too late in the semester for implementation in the prototype at the Senior Design open house.

11.4 Possibility of Future Work
There are several areas that the team sees as potential development. The first is mounting the acrylic dome to the front of the Sub and better utilizing the video feed for operating the system. Additionally, the team could introduce a higher power router to achieve a longer Wi-Fi range. However, these two items require minimal work.

Areas that would require more effort include improving the latency of the Ethernet communication, improving the video quality (frame rate and resolution), and better utilizing the IMU. Currently the IMU is mainly used for attitude reading and only incorporates rough adjustments. In future development, the team would make better use of the sensor by incorporating some of the inertial navigation features discussed in Section 7.2.2.7.
One goal of SWIM-R was to be able to dive to a depth of 100 feet. This feature would require significant redesign of the Sub enclosure in order to withstand the increased pressure without leaking.
12 Special Thanks

There are several people who have supported Team SWIM-R throughout the fall semester. They are listed below.

DornerWorks has chosen to sponsor Team SWIM-R with a donation of $1000. David Dorner, President of DornerWorks, is a Calvin College alumnus. Team SWIM-R is very grateful to DornerWorks for providing the financial means necessary to achieve their goals.

Professor Steve VanderLeest served as the primary instructor for Engineering 339/340 as well as the advisor for Team SWIM-R. The team is grateful to Professor VanderLeest for his patience and wisdom helping to make SWIM-R a success.

Professor Ned Nielsen has graciously volunteered to aid the team as a mechanical engineering mentor. The team is grateful to Professor Nielsen for his insights and experience in mechanical engineering.

Calvin College requires that its engineering students take a Business for Engineers course in which the students plan a business around a senior design project. Team SWIM-R enlisted the help of junior engineering students Kristen Herder and Spencer Olsen to help with the business plan portion of this document.

Phil Jaspers is the Calvin College metal and wood shop supervisor/mechanical genius. The team is enrolled in the metal shop course offered by Phil, who will likely prove indispensable as the team constructs the various mechanical components of SWIM-R.

Tim Theriault is an engineering director at GE Aviation who graciously took time out of his busy schedule to meet with Team SWIM-R to talk about the electronics design process and how to appropriately scope a project. Tim’s insights are greatly appreciated by the team.

DornerWorks has tasked Theo Voss to be a mentor to Team SWIM-R. Theo is a software developer at DornerWorks and a recent graduate from the Engineering program at Calvin College. The team is grateful to Theo for his support and suggestions based on his experience as a software developer at DornerWorks and as student at Calvin.
References


