Winter 2007

Intelligent airlift system for submersible cage aquaculture

Darren Landino

*University of New Hampshire, Durham*

Follow this and additional works at: [https://scholars.unh.edu/thesis](https://scholars.unh.edu/thesis)

**Recommended Citation**
https://scholars.unh.edu/thesis/333

This Thesis is brought to you for free and open access by the Student Scholarship at University of New Hampshire Scholars' Repository. It has been accepted for inclusion in Master's Theses and Capstones by an authorized administrator of University of New Hampshire Scholars' Repository. For more information, please contact [nicole.hentz@unh.edu](mailto:nicole.hentz@unh.edu).
INTELLIGENT AIRLIFT SYSTEM
FOR SUBMERSIBLE CAGE AQUACULTURE

BY

DARREN LANDINO
B.S.E.E. Clarkson University, 1999

THESIS
Submitted to the University of New Hampshire
In Partial Fulfillment of
the Requirements for the Degree of

Master of Science

In

Electrical Engineering

December, 2007
This thesis has been examined and approved.

L. Gordon Kraft
Thesis Director, L. Gordon Kraft
Professor of Electrical and Computer Engineering

Barbaros Celikkol
Professor of Mechanical and Ocean Engineering

Kenneth C. Baldwin
Professor of Mechanical and Ocean Engineering

12/4/07
Date
DEDICATION

To:
My parents
My family
My friends
ACKNOWLEDGEMENTS

I would like to thank Professors Barbaros Celikkol and Kenneth Baldwin, for supporting me as a graduate student during the course of this thesis. I would like to thank the Electrical and Computer Engineering Department for providing tuition support for the 2006-2007 academic year. I would like to thank Professor Barbaros Celikkol for providing thesis project support.

I would like to thank Professors Barbaros Celikkol and Kenneth Baldwin for their expertise during the course of this thesis. I would like to thank Professor Gordon Kraft, for his advice on all things both thesis and life-related.

Thanks to Judson DeCew, who guided me through countless marine and mechanical engineering challenges. To Paul Lavoie, Jim Irish, Stan Boduch, Andy McLeod, and Glenn McGillicuddy for their expertise and advice on electronic and mechanical design. Finally, thanks to Gopala Muluikutla, Ashley Risso, and the rest of the Ocean Engineering students for their continuous support and encouragement.

This thesis is part of the ongoing development of Open Ocean Aquaculture at the University of New Hampshire, and I would like to thank those whose prior work led to this thesis project.
# TABLE OF CONTENTS

| DEDICATION | iii |
| ACKNOWLEDGEMENTS | iv |
| LIST OF TABLES | viii |
| LIST OF FIGURES | ix |
| ABSTRACT | xii |

<table>
<thead>
<tr>
<th>CHAPTER</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. OVERVIEW</td>
<td>1</td>
</tr>
<tr>
<td>Previous Lifting System</td>
<td>2</td>
</tr>
<tr>
<td>Goals and Objectives of the Intelligent System</td>
<td>6</td>
</tr>
<tr>
<td>Approach</td>
<td>7</td>
</tr>
<tr>
<td>2. LIFTING SYSTEMS</td>
<td>8</td>
</tr>
<tr>
<td>Airlift systems</td>
<td>8</td>
</tr>
<tr>
<td>Winch systems</td>
<td>12</td>
</tr>
<tr>
<td>3. DESIGN PROCESS OF INTELLIGENT AIRLIFT CONTROLLER</td>
<td>20</td>
</tr>
<tr>
<td>Design Process: Requirements</td>
<td>20</td>
</tr>
<tr>
<td>Design Process: Communication</td>
<td>21</td>
</tr>
<tr>
<td>Design Process: Embedded Processor</td>
<td>23</td>
</tr>
<tr>
<td>Design Process: Computer Controller</td>
<td>25</td>
</tr>
<tr>
<td>Design Process: Air movement system</td>
<td>26</td>
</tr>
<tr>
<td>Design Process: Depth/Altitude Sensor</td>
<td>29</td>
</tr>
<tr>
<td>Design Process: Tank level sensor</td>
<td>31</td>
</tr>
<tr>
<td>Design Process: Power requirements</td>
<td>33</td>
</tr>
</tbody>
</table>
**LIST OF TABLES**

Table 2-1  Calculation of number of scuba tanks required at expected air tank depths .......... 11
Table 2-2  Power requirements for various winch systems ...................................................... 18
Table 2-3  Power (heat) loss of various winch systems at 1600W .............................................. 18
Table 3-1  Comparative costs of acoustic modems ........................................................................ 22
Table 3-2  Comparative Costs and Specifications of Model Solenoid Valves ............................... 29
Table 3-3  Cost of proportional ball valves ................................................................................... 29
Table 3-4  Estimate of energy use from all components with boat power .................................... 34
Table 3-5  Estimate of energy use from all components without boat providing power ............. 34
Table 3-6  Comparison of rechargeable battery types to requirements for voltage, current, energy, and cost ........................................................................................................ 35
Table 4-1  Description of A/D channel voltages ........................................................................... 37
Table 4-2  Inputs to the user interface .......................................................................................... 39
Table 5-1  The time needed for the cage to sink with conditions of a) small output valve and b) restricted air flow due to a ball valve .............................................................................. 51
Table 5-2  The time needed for the cage to rise with conditions of a) small output valve and b) restricted air flow from a ball valve ............................................................. 52
Table 5-3  Delays in the control system .......................................................................................... 56
Table 6-1  Known parameters from the SBIR deployment ............................................................. 58
Table 6-2  Summary of effect of delays on the on-off controller .................................................. 67
Table 6-3  Summary of transients of On-Off vs. both lead controllers ...................................... 70
Table 6-4  Summary of effects of changes in delay and airflow on SBIR system, using lead compensator #2 .............................................................................................................. 74
Table C-1  Voltage across output resistor R1 (VR1) of LV20P .................................................... 109
Table E-1  Data for the HAL subsystem block ............................................................................. 119
LIST OF FIGURES

Figure 1-1 Line drawing of the UNH SBIR cage showing the relative position of the cage, airlift, and ballast...........................................................................................................................3
Figure 1-2 Cross-section of JPS air tank showing exit valves and hoses. ........................................4
Figure 1-3 Photograph of JPS airlift in JACOEL engineering tank. .............................................5
Figure 1-4 Cross-section of air tank showing decreasing water levels............................................6
Figure 2-1 Sea Station Ocean Spar cage .......................................................................................8
Figure 2-2 Free body diagram of air tank Airlift system ................................................................. 9
Figure 2-3 Diagram of air tank with scuba tanks attached...............................................................10
Figure 2-4 Free body diagram of a winch motor system .................................................................13
Figure 2-5 Diagram showing the sinking of a motor, winch, and mooring weight over time........14
Figure 2-6 Diagram illustrating how placement of a winch on the ocean floor in deeper water requires longer ballast line and electrical cable.............................................................15
Figure 2-7 Diagram illustrating how placement of the winch motor at the base of the cage does not increase the amount of ballast line or electrical cable............................................16
Figure 2-8 Diagram illustrating a block system located at the top of ballast chain at different depths. ..............................................................................................................................17
Figure 2-9 Diagram of motor winch in housing with bearing, bearing mounts, and coupling..... 19
Figure 3-1 Diagram of a hydrophone system................................................................................22
Figure 3-2 Persistor Instruments CF2, R212, and U4S boards......................................................24
Figure 3-3 B & B Electronics Model 4WSD9TB RC-232 to RS422/485 converter .......................25
Figure 3-4 Omega SV3500 series solenoid valves..........................................................................28
Figure 3-5 Airmarr Smart Sensor DT 800 and Brass Thru-Hull housing.......................................31
Figure 3-6 Cross-section of acoustic sensor................................................................................32
Figure 3-7 Omega LVU 1502 Level Sensor ..................................................................................33
Figure 4-1 Flow Chart of the CF2's Master Control Program .........................................................38
Figure 4-2 LabVIEW user interface panel ....................................................................................40
Figure 4-3 LabVIEW Program Control Flowchart. .......................................................................42
Figure 5-1 Internal Diagram and Photograph of the SPAR airlift system scale model.................43
Figure 5-2. Dimensions of the SPAR airlift system scale model..................................................44
Figure 5-3 Response of scale model. Initial depth of -1.4m to a commanded depth of -4m...........45
Figure 5-4 Response of scale model. Initial depth of -4m to commanded depth of -0.5m..........46
Figure 5-5 Response of scale model. Initial depth of -1.3m to commanded depth of -5m..........46
Figure 5-6 Response of scale model. Initial depth of 4.9m to commanded depth of 3m.............47
Figure 5-7 Response of scale model. Initial depth of -1.4m to commanded depth of -4m...........48
Figure 5-8 Response of scale model. Initial depth of 4.3m to a commanded depth of -2m. ... 48
Figure 5-9 Response of scale model. Initial depth of -1.5m to a commanded depth of -6m. ... 49
Figure 5-10 Response of scale model. Initial depth of -1.5m to commanded depth of -4m. ... 50
Figure 5-11 Response of scale model. Initial depth of -4.4m to a commanded depth of -2m. ... 50
Figure 5-12 Response of scale model. Initial depth of -1.6m to a commanded depth of -2.5m. 51
Figure 5-13 Response of scale model. Initial depth of -3.8m to a commanded depth of -5m. ... 52
Figure 5-14 Response of scale model. Initial depth of -4.3m to a commanded depth of -3.3m. 53
Figure 5-15 Response of scale model. Initial depth of -3.8m to a commanded depth of -2m. ... 53
Figure 5-16 Response of scale model. Initial depth of -1.6m to commanded depth of -2.5m. ... 54
Figure 5-17 Response of scale model. Initial depth of -1.6m to a commanded depth of -2m. ... 55
Figure 6-1 Diagram of airlift. ................................................................................................................. 57
Figure 6-2 Simulink SBIR block diagram model. .............................................................................. 60
Figure 6-3 Response of Simulation, open loop, with no mooring weight attached. ....................... 61
Figure 6-4 Open loop responses of cage from initial conditions of -0.21 m, -2m, -4m, -6m, -8m and -9.14m........................................................................................................................ 62
Figure 6-5 Closed Loop On-Off controller response of SBIR simulation from initial condition of -2m to intended final conditions of -2m, -5m and -8m ......................................................... 63
Figure 6-6 Closed Loop On-Off controller response of SBIR simulation. Initial condition of -5m to intended final conditions of -2m, -5m and -8m. ......................................................... 64
Figure 6-7 Closed Loop On-Off controller response of SBIR simulation. Initial condition of -8m to intended final conditions of -2m, -5m and -8m. ......................................................... 65
Figure 6-8 Effect of a 3 second delay on the response of the on-off controller. Initial condition of -2m to intended final conditions of -2m, -5m, and -8m. ........................................... 66
Figure 6-9 Effect of a three-second delay on the response of the on-off controller. Initial condition of -5m to intended final conditions of -2m, -5m, and -8m........................................... 66
Figure 6-10 Effect of a 3 second delay on the response of the on-off controller. Initial condition of -8m to intended final conditions of -2m, -5m, and -8m........................................... 67
Figure 6-11 On-Off vs. both lead compensated controllers at an initial condition of -2m to an intended final condition of -2m................................................................. 68
Figure 6-12 On-Off vs. both lead compensated controllers at an initial condition of -2m to an intended final condition of -5m................................................................. 69
Figure 6-13 On-Off vs. both lead compensated controllers at an initial condition of -2m to an intended final condition of -8m................................................................. 69
Figure 6-14 Effects of system delays on response of system, at nominal airflow rates. initial condition of -5m to an intended final condition of -3m........................................... 71
Figure 6-15 Effects of system delays on response of system, at nominal airflow rates. Initial condition of -5m to an intended final condition of -8m. .................................................. 71

Figure 6-16 Effects of system delays on response of system, at half of nominal airflow rates. Initial condition of -5m to an intended final condition of -3m. ........................................ 72

Figure 6-17 Effects of system delays on response of system, at half of nominal airflow rates. Initial condition of -5m to an intended final condition of -8m. ........................................ 72

Figure 6-18 Effects of system delays on response of system, at one quarter of nominal airflow rates. Initial condition of -5m to an intended final condition of -3m. ....................... 73

Figure 6-19 Effects of system delays on response of system, at half of nominal airflow rates. Initial condition of -5m to an intended final condition of -8m. ....................... 73

Figure C-1 Solenoid driver circuit. ..................................................................................................... 106

Figure C-2 DT-800 and LVU-1502 turn on circuit. ........................................................................... 107

Figure C-3 LEM to A/D circuit schematic. ....................................................................................... 110

Figure E-1 Circular segment. ............................................................................................................ 118

Figure E-2 Graphed data for the HAL subsystem block. .................................................................. 119
ABSTRACT

INTELLIGENT AIRLIFT SYSTEM
FOR SUBMERSIBLE CAGE AQUACULTURE
by
Darren Landino
University of New Hampshire, December, 2007

The University of New Hampshire's Open Ocean Aquaculture Project operates offshore fish cages in the exposed ocean near the Isle of Shoals. A need exists for improved vertical control of their cages that would permit continuous depth control, communication with operators, and logging of data. This thesis investigates the feasibility and practicality of applying a computer controlled lifting system to an experimental fish cage.

The submersible fish cage investigated has a diameter of 15.54 meters, and an internal volume of approximately 1500 cubic meters. The cage is moved through the water column via an airlift located beneath the cage. Air is supplied from the surface. A series of valves controls the inlet and outlet of air. The project included investigation of lifting systems, design and selection of components, software design, and computer simulation. The computer control system was implemented on a physical scale model for verification of the system's operation. The analysis, simulation, and physical testing showed the system operated and can be applied to a full scale cage.
CHAPTER 1

OVERVIEW

The rising demand for fish, due to world population increase, has led to the depletion of natural fish stocks. To combat depletion, restrictions have been placed on the wild harvest of many commercial species. The reduced supply, along with increasing demand, has led to the growth of aquaculture, or commercial fish farming.

Ocean based aquaculture has primarily taken place inshore, in protected bays. Large inshore aquaculture sites, and those inshore sites looking to expand in size, meet resistance from other users of the protected bays, including lobstermen, and recreational boaters. This has led to the development of open ocean aquaculture (OOA), where there are fewer space constraints.

Open ocean aquaculture presents many challenges that inshore fish farming does not face. One challenge is farming fish species that live in deeper waters. Many fish, such as Atlantic cod, *gadus morhua*, have inflatable swim bladders. The cage that contains the fish needs to be raised and lowered periodically for servicing, harvesting, and other functions. Raising the cage too quickly causes the air bladders in these fish to expand, potentially killing the fish. A critical need exists to develop a system that enables the cages to be raised and lowered in a controlled manner, maintaining the health of the fish.
Previous Lifting System

The University of New Hampshire's Open Ocean Aquaculture Group, operating out of the Jere A. Chase Ocean Engineering Center, (JACOEL) has been involved in open ocean aquaculture research and development since 1995. The group operates several cages in its offshore fish farming site near the Isle of Shoals, about 10km from Portsmouth, NH.

In 2006, the group worked with JPS Industries to build an experimental cage, called the SBIR cage, to investigate new engineering ideas and technology. The collaboration was funded through a National Oceanic and Atmospheric Association (NOAA) Small Business Innovative Research (SBIR) grant. The cage was moored into a fixed position. Raising or lowering of the SBIR cage was accomplished with a manual airlift system. The SBIR airlift system consisted of a tank, positioned below the cage and above the ballast (large steamer chain). This tank stored air, the amount of which was increased or decreased to or raise or lower the cage respectively. Figure 1-1 illustrates this system.
Figure 1–1 Line drawing of the UNH SBIR cage showing the relative position of the cage, airlift, and ballast.

Six exit valves were used to allow operators to raise and lower the cage in stages. The valves were located along the length of the pressure vessel. These valves were attached to 300 psi maximum pressure hoses that run bundled along the cage to the surface. The hoses were labeled at the surface by to identify each hose/value component. Figure 1-2 shows a cross-section of the apparatus. Figures 1-2 and 1-3 show the apparatus from other perspectives.
Figure 1–2 Cross-section of JPS air tank showing exit valves and hoses.
The airlift moves the chain up in stages by displacing water (see Figure 1.4). When the cage is submerged, the airlift is flooded with water. To raise the cage approximately 9 feet, air was forced through Hose 1, until water stopped pouring through Hose 2. When water stopped pouring through Hose 3, the cage would have risen another 9 feet. If water stopped pouring from Hose 4, the airlift would be at the surface.

The hoses were all equipped with valves at the water surface and were closed when not in use. The air always entered the tank through Hose 1. The sixth valve, located on the bottom of the airlift, was opened and closed for maintenance, emergency, or other purposes.
The airlift system, worked as intended for the four months the SBIR cage was deployed in the ocean, (July-November 2006). However, there were drawbacks with the system. The system allowed only four depth settings. There was difficulty in manipulating the bundle of five air hoses. Repetitive wave motion against the cage caused the hose bundle to wear at the points of mounting to the cage. The decision was made to create an "intelligent" lifting system

Goals and Objectives of the Intelligent System

The goals of this project were to create an intelligent controller that incorporates the following attributes:

- Continuous depth control rather than discrete depth control.
- Elimination of four or all of the air hoses from the surface.
- Ability to log data, including the depth, temperature, and time.
- Future expandability, both in terms of control inputs, and the data that it can collect.

Modification for use on any cage with multiple methods of control.
Approach

The approach taken in this project was as follows:

• Investigate the feasibility and practicality of applying computer control to the lift system.

• Determine whether a mechanical winch system or an air tank buoyancy system is more practical. Including underwater feasibility in the decision.

• Develop a digital controller to communicate with and command the system and components when at depth.

• Create a mathematical model of the physical parameters of the cage and the control system using MATLAB/Simulink and model the response of the system to various inputs for proof of design.

• Construct of a physical scale model.

• Test and validate the system in the UNH oceanic tanks.
CHAPTER 2

LIFTING SYSTEMS

The lifting system for an aquaculture cage enables the operators to raise and lower the cage to adjust to weather and storm conditions, to perform maintenance on the cage, and to harvest the fish.

Airlift systems

Systems that use a change in buoyancy to raise or lower a cage are referred to as airlift systems. The SBIR cage uses an air tank style airlift. Another type of airlift system used by OOA is the Ocean SPAR. A spar is a round support pole, such as a mast on a sailing ship. The Ocean SPAR cage uses a central spar, which provides structural support to the cage and doubles as the ballast tank.

Figure 2-1 Sea Station Ocean Spar cage.
All airlift systems use displacement of water by compressed air for changes in buoyancy. The net upward buoyant force of an object is equal to the magnitude of the weight of fluid displaced by the object.

In an airlift system, air is pumped into a ballast tank to displace water. The weight of the displaced water equals the buoyant force upward. Likewise, purging air from the ballast tank decreases buoyancy and allows the cage to sink.

\[
F_{\text{NET}} = (F_{\text{bc}} \pm F_{\text{bo}}) - (M \cdot a) - (B \cdot v) - (M_T \cdot g)
\]

Figure 2-2 Free body diagram of air tank Airlift system.

The advantages of an airlift system for either the SPAR or air tank systems are the relative simplicity of the components. All that is needed is a ballast tank to store air and water and a valve system to allow the air to enter and exit the ballast tank. A downside of an airlift system is that as the ballast tank ascends or descends, the air
inside the tank will expand or compress. This makes precise simulation and control of the airlift difficult.

An original concept design for the intelligent airlift was to store reserve air next to the ballast tank using standard Scuba tanks. These tanks would be connected in a series or parallel manner.

![Diagram of air tank with scuba tanks attached.](image)

**Figure 2-3 Diagram of air tank with scuba tanks attached.**

In order for such a system to work, the stored air must provide enough air to lift the nominal weight of the cage and chain, as well as provide reserve buoyancy in the case of bio-fouling.
Assuming 10,000lbs of bio-fouling occurring on a cage, the volume of air required for neutral buoyancy is as follows:

\[ V = \frac{F}{d} \]  

\[ V = \frac{10,000 \text{lbs}}{62.28 \text{lbs/ft}^3} = 160.56 \text{ft}^3 = 4.54 \text{m}^3 \]  

A standard scuba tank contains 2.26 m³ at standard atmospheric pressure \( P_0 = 14.625 \text{psi} \), requiring a minimum of two tanks. At greater depths, the hydrostatic pressure increases, compressing the air as it is released from the scuba tank into the air tank. This pressure requirement increases the number of tanks to as many as eight.

Table 2–1 Calculation of number of scuba tanks required at expected air tank depths.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth of Cage Rim</td>
<td>-30 feet</td>
</tr>
<tr>
<td>Depth of Air ballast tank</td>
<td>-125 feet</td>
</tr>
<tr>
<td>Hydrostatic Pressure in air tank at 125 feet</td>
<td>56.81 psig</td>
</tr>
<tr>
<td>Volume of air in scuba tank when released into ballast tank at this pressure. ( V_2 = \frac{P_1 V_1}{P_2} )</td>
<td>20 ft³ = 0.566 m³</td>
</tr>
<tr>
<td>Scuba tanks required at depth</td>
<td>( \frac{4.54 \text{m}^3}{0.566 \text{m}^3} = 8.02 )</td>
</tr>
</tbody>
</table>

Placing multiple scuba tanks at depth introduces the possibility of corrosion in the connectors, and even a small leak would then result in a loss of all air in the scuba tanks. To prevent this, all of the interconnections would have to be dry, housed in a pressure cylinder or case. This would dramatically increase the cost and complexity of the system. To have divers carry tanks down would be time consuming and reduce the simplicity and automation desired. Furthermore, charging to standard scuba tank
pressure of 3000 psi would require an air compressor beyond the resources of most aquaculture operations.

The decision was made to have the air supply based at the surface because of the logistical problems locating the air underwater. Having the air supply at the surface required a connecting air line. It followed that the control system could also be based at the surface. Further, power and communication could be handled by a physical medium, such as protected cable.

This approach has many advantages. Since a boat has fewer power constraints, a personal computer can be used for the graphical user interface (GUI) the attendant uses to operate the system. Power can also be supplied by the boat for the remote activated valves, meaning only the data sensors and embedded processor would need to run off of battery power.

**Winch systems**

A method for lifting the cage mechanically is a winch-controlled lifting mechanism. This mechanism uses an electric motor connected to a drum, which winds a cable connected to the cage for raising and lowering. These systems are simpler to simulate and control than an airlift system. They can make use of many methods of motor control and can operate with great precision.
The barriers for a winch system are winch placement and electrical power consumption. To determine whether a winch system was more feasible than an airlift system, several options for winch placement, and their mechanical and electrical requirements were examined. These include mounting on the sea floor, on a buoy, or directly beneath the cage.

Mounting on the sea floor would allow the use of any size motor, without regard for the mass of the motor affecting buoyancy of the cage. However, this is dependent on several factors. The consistency of the sea floor should be solid rock. At the UNH Isle of Shoals site, the sea floor is loose sand and gravel, and existing moorings have buried themselves in the sand. If the winch mechanism were to become buried, service or repair becomes impossible.
The electrical cable powering the motor would also have to be wound or stored as the cage moved, to prevent the cable from floating with no slack in the water.

The length of ballast line and electrical cable would be dependant on the depth of the sea floor. At the UNH Isle of Shoals site, this is 160 feet, but at other sites, this length could be greater. Maintenance would become more difficult as the depth increased.
Another option would be to have the winch on a buoy. However, having the winch mounted on a buoy incurs the extra expense and time of designing and maintaining such a buoy, and also requires that the buoy be placed away from the cage, using two sea floor mounted blocks. These blocks would have the same issues as the sea floor mounted winch. Further, the winch would now be exposed to the weather.

The third option is to have the winch mounted directly beneath the cage, in the same position as the air tank as described in the background section. In this way, the length of ballast line and electrical cable is kept constant, regardless of the sea floor depth, and there is no concern about the consistency of the sea floor. Using SPECTRA line, which is neutrally buoyant, does not significantly affect the weight of the system.
Figure 2-7 Diagram illustrating how placement of the winch motor at the base of the cage does not increase the amount of ballast line or electrical cable.

To find electrical power needed by the winch, the torque required by the winch motor must be calculated. This torque was calculated for a cage weight of 10,000 lbs, using standard 1" steel line, and ½" SPECTRA line, using a drum length of 12", and a drum shaft diameter of 3". The torques that were required were 3333 ft-lbs for 1" steel, and 2292 ft-lbs for ½" SPECTRA. An additional calculation was carried out to determine if using a block system would help. A block, or pulley, would halve the weight the motor had to turn, but would double the amount of line to be wound.
Using a block system the torques required were 2917 foot pounds for 1 inch steel and 1771 foot pounds for \( \frac{1}{2} \) SPECTRA. The motor should pull at a speed of 5 feet per minute.

The circumference of the drum is

\[ \pi d = 3.14 \times 3\text{in} = 0.785 \text{ft} \]  

(3)

for a pulling speed of 5 feet in one minute, this requires

\[ \frac{5\text{ft}}{0.785\text{ft/rotation}} = 6.37\text{rotations} \]  

(4)

Therefore the RPM is 6.37. The equation for horsepower, \( HP \), in foot-pounds per minute, from torque (\( T \)), in foot-pounds, is:

\[ HP = \frac{T \times RPM}{5252} \]  

(5)
Table 2-2 Power requirements for various winch systems.

<table>
<thead>
<tr>
<th>Case</th>
<th>Max torque Required (Foot Pounds)</th>
<th>HP required</th>
<th>Power Required (kW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steel Line, No block</td>
<td>3333</td>
<td>4.04</td>
<td>3.013</td>
</tr>
<tr>
<td>Spectra Line, No Block</td>
<td>2292</td>
<td>2.8</td>
<td>2.088</td>
</tr>
<tr>
<td>Steel Line, with Block</td>
<td>2917</td>
<td>3.53</td>
<td>2.633</td>
</tr>
<tr>
<td>Spectra Line, With Block</td>
<td>1771</td>
<td>2.14</td>
<td>1.6</td>
</tr>
</tbody>
</table>

The minimum power required is 1600W. A portable generator can generate this kind of power, however, a DC generator would be harder and more costly to obtain. This power also has to be generated at higher voltages, such as 120 or 240 Volts to avoid large electrical currents, which generate large power losses.

Table 2-3 Power (heat) loss of various winch systems at 1600W.

<table>
<thead>
<tr>
<th>Voltage (V)</th>
<th>Current (A)</th>
<th>AWG required</th>
<th>ohms(R)</th>
<th>ohms(R),100 ft</th>
<th>P = I^2 * R</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>133.3</td>
<td>4</td>
<td>0.0002485</td>
<td>0.025</td>
<td>441W</td>
</tr>
<tr>
<td>24</td>
<td>66.7</td>
<td>9</td>
<td>0.0004982</td>
<td>0.05</td>
<td>222W</td>
</tr>
<tr>
<td>36</td>
<td>44.4</td>
<td>11</td>
<td>0.00126</td>
<td>0.12</td>
<td>236W</td>
</tr>
<tr>
<td>120</td>
<td>13.3</td>
<td>19</td>
<td>0.008</td>
<td>0.8</td>
<td>141.5W</td>
</tr>
<tr>
<td>240</td>
<td>6.66</td>
<td>22</td>
<td>0.01614</td>
<td>1.6</td>
<td>71W</td>
</tr>
</tbody>
</table>

Components that are needed for a motor and winch installation are more complex than those for an airlift system. The motor must be housed in a pressure vessel. This pressure vessel must have a seal for the motor shaft to prevent water entry. The winch drum would be made out of stainless steel or other corrosion resistant material and mate to the motor shaft with a coupling. It would be supported on a bearing at the pass-through point. In this way, the winch drum is supported by the housing and not by the motor. A gearbox or motor drive is required to reduce the speed, as most AC motors' rotational speeds are greater than 6.37 RPM.
A further requirement is to have a line leveling device to properly spool the line onto the drum. This prevents the line from tangling or overlapping. A method is also needed to prevent bio-fouling, specifically by mussels attached to the line. These would interfere with or possibly prevent operation of the winch. In addition, the machining and construction of all associated motor housing components, including bearings, mountings, couplings, and seals, make a winch system for this thesis unfeasible.
CHAPTER 3

DESIGN PROCESS OF INTELLIGENT AIRLIFT CONTROLLER

Design Process: Requirements

The goal of this project was to design, develop, simulate, fabricate, and evaluate a prototype control with the following requirements:

- Provide continuous control and monitoring of the depth of the cage
- Develop a means to transfer air in and out of the ballast tank to move the cage.
- Communicate with operators

This prototype was evaluated in the JACOEL Engineering Tank with a depth of 20 feet. The components were selected for use at the UNH OOA site.

The control system of the prototype was a digital, software-based approach. The advantages to using software-based digital control are especially applicable to this project. Reconfiguring a digital control system can be done with software. Ease of reconfiguration is essential, since the system will not be easily accessible. Furthermore, a digital control system allows expandability. Examples of expandability could include additional temperature, GPS, or other sensor data to be part of the control system. In addition, for the simple reason that the airlift system will log data from multiple sensors, which require an A/D and logging computer, it made sense to use computer control.
The digital command and control consisted of the following components and systems.

- A method of communication to allow operators at the surface to operate the components at depth.
- An embedded processor for remotely activating the air valves, communicating with surface based control system, and recording and processing of data.
- A computer control system, operated at the surface to allow the operator to communicate with the embedded processor using a graphical interface.
- A remote operated valve system to allow air into and out of the ballast tank.
- An altitude or depth sensor, which measures the distance from the sea floor.
- A level sensor that determines the level of water (and hence, air) in the ballast tank.
- A temperature sensor.

**Design Process: Communication**

There were minimal design constraints for communication data rates; this allowed a wider range of communication devices. The original design called for the use of acoustic modems. However, the price range of acoustic modems was excessive. LinkQuest and Teledyne Benthos, two recommended manufacturers of these modems were contacted, and gave informal quotes. These prices were an order of magnitude above the total desired cost for this project, so the communication with acoustic modems was abandoned. Refer to the following table for pricing details.
Table 3–1 Comparative costs of acoustic modems.

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Model</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinkQuest</td>
<td>UVM1000</td>
<td>$13,000</td>
</tr>
<tr>
<td>Benthos</td>
<td>ATM-885/891</td>
<td>~$20,000</td>
</tr>
</tbody>
</table>

A second method of communication was envisioned where acoustic hydrophones would be used. The hydrophone would be located at depth, next to the air tank. The hydrophone would listen for a specific signal from the operator, at which point the embedded processor would awaken from its sleep mode and begin operation.

![Diagram of a hydrophone system](image-url)

Figure 3–1 Diagram of a hydrophone system.

This method also had drawbacks. Since the communication is one way, there is no way of knowing if the cage has responded to any commands or is operating correctly. The hydrophone would have to operate continuously, since the operation of the cage is not on a fixed schedule. The communication itself would have to be designed so that
the embedded processor could distinguish transmitted data from external noise sources, both natural and mechanical. The solution to the communication problem presented itself after the analysis of the mechanical engineering issues. Since a cable would now be used, communication will be handled by RS-232, RS-422 or RS-485 communication protocols.

**Design Process: Embedded Processor**

A computer located at depth has electrical power as its main limiting factor. A personal computer could not be used for this reason. One of the design goals of this system is the ability to operate a month before batteries needed to be replaced. A previous underwater data collection device that used a PC based computer, was examined, and found to consume 30 Watts on average. At 12 Volts, this amounts to

\[
I = \frac{30W}{12V} = 2.5A
\]  

(6)

Using a single 225Ah deep cycle battery,

\[
\frac{225A-h}{2.5A} = 90\text{hours} \quad \frac{90\text{hours}}{24\text{hours/day}} = 3.75\text{Days}
\]  

(7)

A 24V system would last 7.5 days, and 36V system 11.25 days. The low temperatures at depth would not affect the battery at this low of a discharge rate. Note that a typical 225A-H battery can measure 20"x11"x10" and weigh 161lbs. A battery this size would not fit inside a standard pressure vessel. The smaller 12V 7-ah batteries that JACOEL normally uses would fit in the space, but the battery life is less than one day.

A PC-based computer could clearly not operate long enough using existing battery technology. The alternative is an embedded controller. There are many embedded processors available. The Persistor Instruments model CF2 was chosen for this application. It is compact, and consumes very low amounts of power, approximately 100-200mA when operating, and only 20uA when in its lowest power mode. It is
programmed with C, allowing powerful programs that do not require large overhead (RAM) to operate. The CF2 has output pins that can be used to turn on and off electrical devices, such as the sensors, and air movement valves. Each of these boards has ready made starter code. In addition, the CF2 has existing peripheral components, including a serial driver board, and an A/D converter. The CF2 has its own command line operating system, called PicoDos, and a built in RS-232 port for data transmission. The CF2 has been widely used by JACOEL and Woods Hole Oceanographic Institute personnel, who have used the CF2 before, with much success.

![Figure 3-2 Persistor Instruments CF2, R212, and U4S boards.](image)

While the RS-232 port can be used for communication, a limitation exists for RS-232. The Electronic Industries alliance (EIA) RS-232C standard imposes a cable length limit of 15 meters because RS-232 signal quality degrades at longer distances. This is
due to capacitive effects of the cable length. While the cable length can be extended using lower capacitance cables, a more robust protocol would serve better.

Protocols RS-422 and RS-485 can function properly at distances to 1000 feet, and RS-232 to RS-422 or RS-485 converters were inexpensive, easy to install, and widely available. The converter chosen for this application was the B+B electronics 4WSD9TB. This model allowed for RS232 to RS422 or RS 485 communication, with terminal block style connectors. It also is port-powered, which means it does not require an external power source but can use one if needed.

![Figure 3-3 B & B Electronics Model 4WSD9TB RC-232 to RS422/485 converter.](image)

**Design Process: Computer Controller**

The PC GUI system was designed in National Instruments' LabVIEW. LabVIEW is a graphical interface software that was widely used to perform computer based instrumentation, measurement, and control. The LabVIEW hierarchy was based off of Virtual Instruments (VI). These VIs are analogous to functions in ANSI C. This gave the operators the ability to visually see and operate the system as if they were at a control terminal. LabVIEW can communicate using serial communication, and has powerful mathematical and computational tools.
Design Process: Air movement system

The air movement system controls the flow of air into and out of the ballast tank using remote activated valves. Solenoid valves are a simple and inexpensive way to accomplish remote activation. A solenoid valve is a valve that is opened or closed by an electromagnet. This action is achieved by the movement of a magnetic plunger to seal off or open a port when voltage is applied. The solenoid is a coil of wire wrapped around the magnetic plunger. When voltage is applied, the current flowing through the coil creates a magnetic field which moves the plunger.

The orifice size on the SBIR/JPS airlift system was \( \frac{3}{4} \)" at both intake and exhaust. For this application, two valves at both inlet and outlet would be used. Using two valves gives the advantage of redundancy if one of the valves fails. An additional advantage is a more controlled rise and or descent, by opening one, or both valves. Omega Engineering's solenoid valve selection process was employed to select valves with the proper flow characteristics.

This method uses the Cv, or flow factor of the valve, as the primary design criteria. To calculate Cv, the pressure drop across the valve must be found. The pressure drop across the valve is the difference between the output pressure of the compressor and the hydrostatic pressure at depth.
The compressor supplied six Standard Cubic Feet per Minute (SCFM) at 90 to 150 psi. The hydrostatic pressure at 60 and 100 feet was found from the following equation: 

\[ P = \frac{\tau \ G \ H}{e} \]

\( P_{c1} = 90 \text{ psi} \)  
Nominal Pressure output of compressor

\[ P_{h1} = 62.28 \left( \frac{\text{lbm}}{\text{ft}^2} \right) 32 \left( \frac{\text{ft}}{\text{sec}^2} \right) 60 \left( \text{ft} \right) = 119600 \frac{\text{lbm}}{\text{ft} \ \text{s}^2} \]  
(8)

\[ 1 \text{ psi} = 4633 \frac{\text{lbm}}{\text{ft} \ \text{s}^2} \]  
(9)

\[ \frac{119600 \frac{\text{lbm}}{\text{ft} \ \text{s}^2}}{4633 \frac{\text{lbm}}{\text{ft} \ \text{s}^2}} = 25.81 \]  
Hydrostatic Pressure at 60 ft.

\[ P_{c2} = 150 \text{ psi} \]  
Maximum pressure output of compressor

\[ P_{h2} = 62.28 \left( \frac{\text{lbm}}{\text{ft}^3} \right) 32 \left( \frac{\text{ft}}{\text{sec}^2} \right) 100 \left( \text{ft} \right) = 199300 \frac{\text{lbm}}{\text{ft} \ \text{s}^2} \]  
(11)

\[ \frac{199300 \frac{\text{lbm}}{\text{ft} \ \text{s}^2}}{4633 \frac{\text{lbm}}{\text{ft} \ \text{s}^2}} = 43.07 \text{ psi} \]  
Hydrostatic Pressure at 100 ft.

(12)

The equation for \( C_v \) is dependant on the difference in pressure across the valve.

If \( P_c < 0.53 \times P_h \), then

\[ C_v = \frac{Q}{1349} \sqrt{\frac{460 + t}{DP \ P_h}} \]  
(13)

If \( P_c \neq 0.53 \times P_h \), then

\[ C_v = \frac{Q \sqrt{460 + t}}{704 \ P_h} \]  
(14)

\[ P_{c1} = 0.53 = 47.7 \text{ psi} \]  
(15)
\[ P_{c1} \ 0.53 = 79.5 \text{ psi} \]  

\[ P_{c1} \text{ and } P_{c2} \text{ will always be greater than } P_{h1} \text{ or } P_{h2} \]

For \( P_{c1} = 90 \text{ psi} \)

\[ Cv = \frac{360 \sqrt{460 + 15}}{704 \ 90} = 0.124 \]  

(16)

for \( P_{c1} = 150 \text{ psi} \)

\[ Cv = \frac{360 \sqrt{460 + 15}}{704 \ 150} = 0.074 \]  

(17)

The maximum pressure differential (MOPD) would be

\[ 150 \text{ psi} - 25.81 \text{ psi} = 124 \text{ psi} \]  

(19)

The valves selected for this purpose were the Omega SV3500 series solenoid valves using 12Watt, 24VDC coils.

Figure 3–4 Omega SV3500 series solenoid valves.
Table 3–2 Comparative Costs and Specifications of Model Solenoid Valves

<table>
<thead>
<tr>
<th>Model Solenoid Valve</th>
<th>Price</th>
<th>Pipe Size</th>
<th>Orifice</th>
<th>Cv (flow factor)</th>
<th>MOPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>SV3502</td>
<td>$114</td>
<td>3/8&quot;</td>
<td>7/16&quot;</td>
<td>1.4</td>
<td>200</td>
</tr>
<tr>
<td>SV3504</td>
<td>$117</td>
<td>3/4&quot;</td>
<td>5/8&quot;</td>
<td>2.8</td>
<td>160</td>
</tr>
<tr>
<td>12W, 24VDC coil</td>
<td>$23</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

The total cost for this project is $554.

These orifice sizes were chosen because their cross sectional area is equal to that of a single ¾" orifice.

Proportional valves were also investigated. However, these had a much higher cost than the solenoid valves chosen as illustrated in the following table:

The total cost for this project would have been $2950.

Table 3–3 Cost of proportional ball valves.

<table>
<thead>
<tr>
<th>Model Proportional valve</th>
<th>Price</th>
<th>Orifice</th>
<th>Cv (Flow factor)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PV34B</td>
<td>$1425</td>
<td>¾&quot;</td>
<td>6.5</td>
</tr>
</tbody>
</table>

Additionally, conversations with the manufacturer indicated that these were meant to be used for a system in continuous use, not a system that would operate intermittently, or be down for an extended duration. Since 24VDC was required for the tank level sensor, solenoid valve coils were also 24VDC. Using a higher voltage will result in lower current draw, creating less voltage and power drop across the cable.

Design Process: Depth/Altitude Sensor

The control system will operate based on data from the depth or altitude sensor.

There are two types of depth sensors widely available: acoustic based altitude sensors and pressure based depth sensors. Acoustic depth sensors beam a high frequency sound wave, and record the time it takes for that wave to reflect back to the emitter. These measure altitude above the sea floor. Pressure based depth sensors
continuously measure water pressure on their casing. These types measure the depth below the surface.

A drawback of measuring from the surface is the depth reading can be influenced by normal tidal action. As the tides move in and out, the depths recorded by a pressure based depth sensor varied. For this reason, an acoustic altitude sensor is preferred. Acoustic depth sensors most commonly fall into two categories: those used at very shallow depths such as recreational fish finders, and those used at very deep depths, such as bathymeters. Recreational fish finders are generally inexpensive, around $100-$400, provide altitude readings to 200 feet, and output National Marine Electronics Association (NMEA) data strings that can be read by the CF2. Additionally, some include temperature measurements. One was used in a similar application by RIT.

The disadvantages of these are the power consumption and the physical size. The power requirements of most of this style of fish finder are in the 15-20 watt range, and require their LCD display in order to transmit data. Examples of this type are the Garmin Fish Finder 340C and 160C.

Bathymeters have all the advantages of the recreational style altimeters, plus are rated for great depths. The main disadvantage is the high cost, which is in the $2000-5000 range. Examples in this category are the Benthos PSA 900 and 916.

Since the final testing would be in the oceanic tank, a compromise solution was the Airmarr Smart sensor DT-800. This is an acoustic transducer with a NMEA string output that does not require a display screen. It operates on low power (40mA), outputs temperature, and costs $250. It can measure altitude to 200 feet with an accuracy of approximately 1.2 feet at a 235Khz sounding frequency, at a beam angle of 14 degrees.
The limitation of the DT800 was that it was only pressure rated to 10 feet. This was not a concern in the oceanic tank. However, a different sensor must be employed at the UNH OOA site.

**Design Process: Tank level sensor**

The water level in the tank was measured using an acoustic level sensor. These operated in a manner that was similar to the acoustic altitude sensor, but are used for measuring height above a liquid surface. They beam an acoustic signal and calculate the time it takes for that signal to be reflected off the surface of the measured medium. The level sensor was encased in a pipe that will allow more accurate measurement.

Acoustic level sensors have been designed to be positioned at the top of a tank of liquid and transmit a narrow beam signal. The reflected signal at the air-water surface was recorded.
The pressure in the tank at maximum depth can be found using the following equation:

\[ P = \rho \cdot g \cdot H, \quad (20) \]

At a maximum depth of 30.48m,

\[ P = \rho \cdot g \cdot H = 62.28 \frac{\text{lbs}}{\text{ft}^3} \cdot 32 \frac{\text{ft}}{\text{sec}^2} \cdot 30.48 \text{m} = 43.25 \text{ psi} \quad (21) \]

The only model found that exceeded this rating was the Omega LVU-1502, with a rating of 100psi.
Figure 3-7 Omega LVU 1502 Level Sensor.

This model provides an isolated 4-20mA output. The device can measure from six inches to ten feet, and has a user selectable range.

Design Process: Power requirements

The electrical power requirements of the airlift system were calculated for two scenarios. One scenario was to have the operations boat provide power for all devices when raising or lowering the cage, and the batteries only needed to provide power for recording data when the cage was stationary. This was the expected operation. The second scenario is to have the batteries provide all the power. This was a worst case scenario.

The following tables indicate that an estimate of 1.05 A-h/day was required to power the devices. The recording interval was 15-minutes once per hour per day.
Table 3-4 Estimate of energy use from all components with boat power.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Quantity</th>
<th>Current: mA</th>
<th>Total Current</th>
<th>Time (hrs)</th>
<th>Amp hours needed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistor ON</td>
<td>1</td>
<td>75</td>
<td>75</td>
<td>6</td>
<td>0.45</td>
</tr>
<tr>
<td>Persistor OFF</td>
<td>1</td>
<td>0.02</td>
<td>0.02</td>
<td>18</td>
<td>0.00036</td>
</tr>
<tr>
<td>DT-800 On</td>
<td>1</td>
<td>40</td>
<td>40</td>
<td>6</td>
<td>0.24</td>
</tr>
<tr>
<td>Depth On</td>
<td>1</td>
<td>20</td>
<td>20</td>
<td>6</td>
<td>0.12</td>
</tr>
<tr>
<td>Tank Level On</td>
<td>1</td>
<td>40</td>
<td>40</td>
<td>6</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>1.05036</strong></td>
</tr>
</tbody>
</table>

The next table shows the estimate of energy use from all components, when the power is not supplied by the boat, and assumes one hour to lift the cage and 15 minutes to lower cage.

Table 3-5 Estimate of energy use from all components without boat providing power.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Quantity</th>
<th>Current: mA</th>
<th>Total Current</th>
<th>Time (hrs/day)</th>
<th>Amp hours needed/day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persistor ON</td>
<td>1</td>
<td>75</td>
<td>75</td>
<td>7.25</td>
<td>0.536</td>
</tr>
<tr>
<td>Persistor OFF</td>
<td>1</td>
<td>0.02</td>
<td>0.02</td>
<td>16.75</td>
<td>0.000335</td>
</tr>
<tr>
<td>DT-800 On</td>
<td>1</td>
<td>40</td>
<td>40</td>
<td>7.25</td>
<td>0.29</td>
</tr>
<tr>
<td>Depth On</td>
<td>1</td>
<td>20</td>
<td>20</td>
<td>7.25</td>
<td>0.145</td>
</tr>
<tr>
<td>Tank Level On</td>
<td>1</td>
<td>40</td>
<td>40</td>
<td>7.25</td>
<td>0.29</td>
</tr>
<tr>
<td>Solenoid Valves On</td>
<td>2</td>
<td>660</td>
<td>1320</td>
<td>1.25</td>
<td>1.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td><strong>Total</strong></td>
<td><strong>2.911</strong></td>
</tr>
</tbody>
</table>
Design Process: Batteries

The 12V, 12A-h lead was the best compromise between size, cost, and energy, as it provides the needed current and requires only two in series for 24V. These batteries also have lower source impedance than alkaline batteries. In addition, WHOI personnel have used this battery type in their buoys. These batteries fitted inside their standard pressure vessel tube and will give enough runtime for 11.4 days. While this is less than the 30 days initially desired, they are rechargeable.

Table 3–6  Comparison of rechargeable battery types to requirements for voltage, current, energy, and cost.

<table>
<thead>
<tr>
<th>Battery type</th>
<th>Volts</th>
<th>A-h</th>
<th>Number needed in parallel for 1.05 A-H</th>
<th>Number needed in series for 24V</th>
<th>Days can run at 1.05 A-h per day</th>
<th>Cost per</th>
<th>Total cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>AA</td>
<td>1.2</td>
<td>2.7</td>
<td>1</td>
<td>20</td>
<td>2.57</td>
<td>2</td>
<td>40</td>
</tr>
<tr>
<td>C</td>
<td>1.2</td>
<td>5</td>
<td>1</td>
<td>20</td>
<td>4.76</td>
<td>7</td>
<td>140</td>
</tr>
<tr>
<td>D</td>
<td>1.2</td>
<td>11</td>
<td>1</td>
<td>20</td>
<td>10.47</td>
<td>12</td>
<td>240</td>
</tr>
<tr>
<td>9V</td>
<td>9</td>
<td>0.23</td>
<td>4.5</td>
<td>3</td>
<td>0.99</td>
<td>9</td>
<td>27</td>
</tr>
<tr>
<td>12V gel LA</td>
<td>12</td>
<td>12</td>
<td>1</td>
<td>2</td>
<td>11.42</td>
<td>33</td>
<td>66</td>
</tr>
<tr>
<td>12V gel LA</td>
<td>12</td>
<td>18</td>
<td>1</td>
<td>2</td>
<td>17.14</td>
<td>39</td>
<td>78</td>
</tr>
<tr>
<td>12V gel LA</td>
<td>12</td>
<td>150</td>
<td>1</td>
<td>2</td>
<td>142.81</td>
<td>40</td>
<td>80</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER 4

PROGRAMMING LABVIEW AND PERSISTOR

Embedded Processor Program Design

The CF2 has two main functions:

- Active mode: communicates with the LabVIEW controller to acquire data and operate the solenoid valves
- Passive mode: acquires data for a 15-minute interval every hour, writes it to file, and then enters a low power mode until the next hour.

Program flow

Upon activation of the program, the CF2 will first output identifying information about the CF2. It then attempts to initialize the U4S Serial communications card. If the U4S dos not initialize successfully, the program exits to PicoDos. If successful, the program enters the main loop and waits for another input from either of the following.

Q command: Quits the master control program and returns the CF2 to PicoDos
X command: Samples each channel of the R212 A/D board and the NMEA data from the depth sensor, and then displays the information on the screen. Refer to the following table for details about the displayed A/D data.
Table 4-1 Description of A/D channel voltages.

<table>
<thead>
<tr>
<th>A/D channel</th>
<th>Data source</th>
<th>Data Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
<td>Wired to ground</td>
</tr>
<tr>
<td>1</td>
<td>Level Sensor</td>
<td>From LEM LV20-P output</td>
</tr>
<tr>
<td>2</td>
<td>24V battery voltage</td>
<td>1/13.33 ratio voltage divider</td>
</tr>
<tr>
<td>3</td>
<td>12V battery voltage</td>
<td>1/6.66 ratio voltage divider</td>
</tr>
<tr>
<td>4</td>
<td>Large input valve voltage</td>
<td>1/13 voltage divider</td>
</tr>
<tr>
<td>5</td>
<td>Small input valve voltage</td>
<td>1/13 voltage divider</td>
</tr>
<tr>
<td>6</td>
<td>Large output valve voltage</td>
<td>1/13 voltage divider</td>
</tr>
<tr>
<td>7</td>
<td>Small output valve voltage</td>
<td>1/13 voltage divider</td>
</tr>
</tbody>
</table>

After the X command was executed, the CF2 waited for another input.

R command: Upon entering the data recording mode, the program first turned off the DT-800 and LVU1502. It then checked the present time, and compared it to the predetermined recording start time. If the time matched, the program turned on the DT 800 and LVU-1502 and acquired data for a 15-minute continuous interval. If the time did not match, the program entered a low power mode for one minute, exited, and then checked the time again. This continued indefinitely. In this way, the unit recorded one 15-minute interval per hour and repeated the process each hour. The data recorded by the CF2 in this state were written to a .dat file in the c:\data directory. The flow chart of this program can be seen in Figure 4-1.
Figure 4-1 Flow Chart of the CF2's Master Control Program.

**Inputs**

The inputs controlled the turn-on and turn-off of the solenoid valves. They were intended to be used by the LabVIEW control program but were available for manual control, if needed. After execution of a valve command, the CF2 waits for another input. The inputs are summarized in Table 4.2.
Table 4-2 Inputs to the user interface.

<table>
<thead>
<tr>
<th>Primary</th>
<th></th>
<th>Secondary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q</td>
<td>Quit Program</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>Acquires data from the A/D and each of the sensors once</td>
<td>1</td>
</tr>
<tr>
<td>R</td>
<td>Enters the record and sleep mode</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>A</td>
<td>Turn off large output valve</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>Turn off small output valve</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>Turn off both output valves</td>
<td>C</td>
</tr>
</tbody>
</table>

**Outputs**

There was one primary output. This occurred when the command X was executed. The data acquired was printed to the screen for reading by the user and by LabVIEW. The other outputs were diagnostic messages that were generated when a subroutine was opened or closed.

**LabVIEW Controller Program Design (Inputs, Outputs, Flow)**

The governing physical equation for the airlift system was as follows:

\[
F_{\text{net}} = (F_{bc} - F_{bo}) - (M_T \cdot a) - (B \cdot v) - (M_T \cdot g) \tag{22}
\]

However, the variables in this equation were subject to several continuously changing and unknown environmental factors. An example was the increase in mass of the system (\(M_T\)) with bio-fouling. The coefficient of viscous friction (\(B\)) was affected by bio-
fouling. Nonlinear effects on buoyant force, such as the decrease in air flow rate through an air hose as depth increases, were also present.

The control method employed a read and react system, where the present altitude of the cage, along with the cage velocity, was used to determine whether the cage should move up and down. The controller continuously monitored the altitude of the cage, and compared it with the user’s desired altitude, (position error) and the present velocity of the cage. The program used the position error and velocity of the cage to determine the amount of air to let into or out of the air tank.

The front panel was constructed to allow the user full control, while it maintained an uncluttered view in an easy-to-operate, easy-to-understand format.

Figure 4–2 LabVIEW user interface panel.
The code was designed to have the user input critical of the following values before selecting the start arrow in the LabVIEW toolbar: file path, VISA Resource Name and associated data (Baud rate, data bits, parity, and stop bits), and the desired depth setting. When the start arrow was selected, the program entered a repeating loop that performed the following six main steps.

**Step 1:** LabVIEW sent the Airlift Command to the CF2. The response from the CF2 was read and displayed on the Read Buffer.

**Step 2:** The A/D and sensor data from the CF2 were parsed into separate variables for use in indicators and were written to a file.

**Step 3:** Data were displayed on various indicators. The data included Tank Level, Water temperature, Present Depth, 24V battery, 12V battery, current date and time. The Boolean indicators Big Input valve, Small Input Valve, Big Output Valve, and Small Output Valve were used to alert the user which solenoid valves were activated.

**Step 4:** The present depth was subtracted from desired depth, which resulted in depth error. The velocity of the cage was computed. This was accomplished by subtracting the previous loop iteration's value of time and depth, from the present loop iteration's value, resulting in the following deltas:

\[
D_d \quad \text{and} \quad \frac{D_t}{D_t}
\]

Then, the following computation resulted in the velocity of the system:

\[
\frac{D_d}{D_t}
\]

If the depth error was less than 0.5 meters, LabVIEW commanded all valves off because the cage was assumed to be near enough to the desired position. If the depth error was greater than 0.5, then the position error and velocity were used by a lookup table to determine what amount of air to let into or out of the air tank. The lookup table accomplished this by correlating position error and velocity to the valve commands used.
by the CF2. Because it was required that the lookup table must be numbers, A, B, and C were not used in the lookup table, but were replaced by 10, 11 and 12, respectively.

**Step 5:** All data was written to file. The valve commands were selected by a case structure, which sent a string corresponding to the numeric command from the lookup table.

**Step 6:** The valve command was written to the CF2, and the loop returned to step one.

![LabVIEW Program Control Flowchart](image)

**Figure 4–3 LabVIEW Program Control Flowchart.**

Refer to Appendix B for a complete description of all front panel controls and indicators.
CHAPTER V

SCALE MODEL FABRICATION AND TESTING

Physical Scale Model

A physical scale model was developed and tested to verify the mathematical simulation and operation of the computer programs. Testing was performed in the JACOEL ocean engineering tank; this tank provided ideal conditions to evaluate the airlift without concern for weather, visibility and other environmental factors. The scale model was based on an Ocean Spar design, utilizing a 62" long, 18" OD HDPE tube as the SPAR. The scale model plan was fabricated as shown in Figure 5-1. Three plates divided the inside of the tube into two compartments, a watertight section for the LVU-1502 level sensor, and the main ballast section.

Figure 5-1 Internal Diagram and Photograph of the SPAR airlift system scale model.
Figure 5-2. Dimensions of the SPAR airlift system scale model.

In this airlift system, ½ inch air hoses were mounted at 29.5" from the bottom of the tube, this was the maximum height allowed due to the 10" sensor length and the 12" "dead zone" that was required by the LVU-1502. The DT-800 depth sensor was mounted in a watertight pressure vessel that was mounted to the side of the tube. During testing, the unit sporadically returned erroneous readings. The assumption was made that the acoustic beam was striking the side of the tank, and/or the chain beneath the tube.

To prevent this, the minimum distance from the tube and the wall was calculated. Since the acoustic beam was 14 degrees wide, the minimum horizontal distance was 2.45 feet from the center of the tube. Electrical connections were made using waterproof connectors and cable.
First Round of Testing

Testing of the scale model was performed in the JACOEL Ocean tank. The dimensions of the tank are 40' wide x60' long x20' deep, which provided enough range to demonstrate the system. The first round of testing used a simple lookup table. Four tests were conducted to investigate the expected performance of the scale. The response was defined best by a depth vs. time plot, as shown in Figure 5-3.

![Response of Scale Model from -1.4m to -4m](image)

Figure 5–3 Response of scale model. Initial depth of -1.4m to a commanded depth of -4m.
There were erroneous readings prior to 5 seconds (figure 5.4), and the scale model only rose to \( \sim -1.4 \text{m} \).

Figure 5–5 Response of scale model. Initial depth of -1.3m to commanded depth of -5m.
Figures 5.2 and 5.4 show the scale model did not move to the commanded depth, having stopped at approximately 1.5m. The data from Figures 5.1 and 5.3 were inconclusive, as they indicated the scale model moved where commanded, however there was a +/- 0.5m range of error. Further investigation revealed several problems in operation. One was that the resolution of the depth error was much too low. The low resolution resulted from the depth error computed as a non floating point number. This meant that, for example, that 4.9 - 4 would equal 0, not 0.9. Subsequently, the controller operated as if the depth error was 0, and was at steady state.

Another problem was that the lookup table used by the scale model operated on nearest values, any depth error less than 1.52m was read as zero.
Second round of testing

This round of testing introduced a more accurate lookup table, and allowed the use of floating point numbers for improved accuracy. The results were again plotted as depth versus time.

Figure 5-7 Response of scale model. Initial depth of -1.4m to commanded depth of -4m.

Figure 5-8 Response of scale model. Initial depth of 4.3m to a commanded depth of -2m.
Figure 5–9 Response of scale model. Initial depth of -1.5m to a commanded depth of -6m.

Note that the device in the previous figure does not reach the floor of the tank. It reaches a depth similar to that in the testing that is shown in figures 5-3 and 5-5.

The data presented in figures 5-7, 5-8, and 5-9 were inconclusive. The final depth reached by the scale model in figure 5-9 is similar to that reached in figures 5-3 and 5-5. Hence, it could not be determined whether the system was responding as desired, or whether the system is simply reaching its maximum or minimum depth. It is likely that these systems would oscillate if the 0.5m error range were not included.

In addition, these tests revealed limitations of the control system. In order to properly read the depth sensor data from the CF2, LabVIEW required an artificial delay of 750ms. This resulted in a total loop time of ~2.5sec.

When the scale model started accelerating or decelerating, it simply moved too rapidly for the control system to respond in time. To compensate for this, ball valves were placed at the inlet and outlet of the air system, and opened to their minimum
aperture. Additionally, the compressor was lowered to its minimum output pressure, 20psi. Figures 5-10, 5-11, and 5-12 illustrate the results of the tests using the lowered compressor pressure and ball valve restriction.

**Figure 5-10** Response of scale model. Initial depth of -1.5m to commanded depth of -4m.

**Figure 5-11** Response of scale model. Initial depth of -4.4m to a commanded depth of -2m.
Figure 5-12 Response of scale model. Initial depth of -1.6m to a commanded depth of -2.5m.

Figure 5-12 shows the scale model tried to go to a medium distance, but was oscillatory. Figures 5-10 and 5-11 are inconclusive, and Figure 5-12 conclusively shows that the control did not function as desired, which resulted in an oscillatory response. Further, it seemed certain that the maximum depth the scale model could reach was ~ -4.5m and the minimum depth was approximately -1.5m. The speed required to operate the control system for a desired response was determined from measurements of the cage speeds while it was rising and sinking.

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Final state</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum amount of air</td>
<td>Top of Tube just below surface</td>
<td>2:21</td>
</tr>
<tr>
<td>Top of Tube just below surface</td>
<td>Maximum Depth</td>
<td>29:15 sec</td>
</tr>
</tbody>
</table>
Table 5-2 The time needed for the cage to rise with conditions of a) small output valve and b) restricted air flow from a ball valve.

<table>
<thead>
<tr>
<th>Initial state</th>
<th>Final state</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zero volume of air in tank, at max depth</td>
<td>Top of Tube Breaches surface of water</td>
<td>53 sec</td>
</tr>
<tr>
<td>Top of Tube Breaches surface of water</td>
<td>Tank full of air</td>
<td>1:47</td>
</tr>
</tbody>
</table>

A burst method was developed to overcome the time delay of the control system. The LabVIEW controller was programmed to pulse air in or out for one second after every 10 iterations of the control loop. This was approximately every 25 seconds. The response of the physical system would be slowed, as the pulse method would deliver air at a slower rate than previous trials. The results of the burst method testing are illustrated in Figures 5-13, 5-14, 5-15, and 5-16.

![Response of Scale Model from -3.8m to -5m](image)

Figure 5-13 Response of scale model. Initial depth of -3.8m to a commanded depth of -5m.
Figure 5–14 Response of scale model. Initial depth of -4.3m to a commanded depth of -3.3m.

Figure 5–15 Response of scale model. Initial depth of -3.8m to a commanded depth of -2m.
The burst method appeared to have better success, but to verify, the margin of error was reduced from 0.5m to 0.15m, to see how closely the system could place the scale model. Figures 5-14 and 5-16 show erroneous depth measurements.

Figure 5-16 Response of scale model. Initial depth of -1.6m to commanded depth of -2.5m.
The operation using the burst method was oscillatory, demonstrating that earlier successes using the burst method were due to large error range. The acceleration of the scale model from the compression and expansion of air was still too fast for the control system in its present form.

Several methods were tried to overcome the slowness of the control system. One attempt was to reduce time by bypassing the embedded controller when reading the depth sensor. This introduced the problem of requiring two com ports, and actually slowed the system to 4-6 seconds, without artificially introduced delays.

A second attempt was to read data only from the A/D converter. This reduced the time to 1.35 seconds, but this was still too slow for this system. Further, the minimum loop time was 1 second, due to 1Hz update rate of the DT-800 sensor. Additionally, the DT-800 sensor still gave occasional erroneous readings.
Scale model results analysis

Overall, the scale model testing was inconclusive. Testing revealed fundamental problems with the control system as constructed. The speed of the physical system, when accelerating, required a very fast response time from the control system, and there were many time delays preventing this. These included the communication delays, as well as mechanical delays in opening and closing the solenoid valves. These delays can exceed six or seven seconds, in a worst case scenario.

Table 5–3 Delays in the control system.

<table>
<thead>
<tr>
<th>Component</th>
<th>Minimum delay</th>
<th>Maximum delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>LabVIEW communication with CF2</td>
<td>1.35 seconds</td>
<td>2.5 seconds</td>
</tr>
<tr>
<td>Solenoid valve opening</td>
<td>0.1 seconds</td>
<td>1 second</td>
</tr>
<tr>
<td>Solenoid valve closing</td>
<td>0.7 seconds</td>
<td>4 seconds</td>
</tr>
<tr>
<td>Totals</td>
<td>2.05 seconds</td>
<td>7.5 seconds</td>
</tr>
</tbody>
</table>

Note that even with a continuously updating depth sensor, the communication time from LabVIEW to extract data from the CF2 was 1.35 seconds. This was too slow for the physical system. Further, as seen in the above data, if the system had even a small perturbation, the resulting motion was oscillatory, and if there was a larger perturbation, the system moved to either the maximum or minimum physical limit, indicating an unstable system.
CHAPTER VI

SBIR SYSTEM SIMULATION

A mathematical model of the SBIR cage airlift system was created to determine if an air control system was feasible on a full scale SBIR type cage. This model was based on the free body diagram of the complete physical system and the known parameters from the SBIR deployment from July-November of 2006.

![Free Body Diagram of Airlift](image)

**Figure 6-1** Diagram of airlift.

\[
F_{\text{NET}} = (F_{\text{NE}} - F_{\text{BE}}) - (M_r \cdot a) - (B \cdot v) - (M_r \cdot g)
\]
The force the acted on the cage in the upward direction was the buoyancy of the cage. Downward forces were the weight of the cage and chain (mass x gravity). Forces that opposed the direction of motion are drag force (B), and inertia (Mt x a). The net force of the air and water in the air tank (F_{ba}) also acted in either direction.

**Table 6-1 Known parameters from the SBIR deployment.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description of parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_c</td>
<td>Mass of cage</td>
</tr>
<tr>
<td>M_a</td>
<td>Mass of air tank</td>
</tr>
<tr>
<td>M_{ch}</td>
<td>Mass of ballast chain</td>
</tr>
<tr>
<td>M_t</td>
<td>Total mass of system</td>
</tr>
<tr>
<td>X</td>
<td>Direction of motion</td>
</tr>
<tr>
<td>a</td>
<td>Acceleration of system</td>
</tr>
<tr>
<td>v</td>
<td>Velocity of system</td>
</tr>
<tr>
<td>g</td>
<td>Acceleration of gravity</td>
</tr>
<tr>
<td>B</td>
<td>Coefficient of viscous friction (drag force)</td>
</tr>
<tr>
<td>F_{bc}</td>
<td>Buoyant force of cage</td>
</tr>
<tr>
<td>F_{ba}</td>
<td>Buoyant force of air in air tank</td>
</tr>
<tr>
<td>F_{net}</td>
<td>Sum of forces acting on system</td>
</tr>
</tbody>
</table>

**Mathematical Model**

The mathematical model of the cage system was constructed in MATLAB/Simulink. The model was based on known parameters from the deployed SBIR/JPS cage. The time responses for this model were much faster than those experienced at the site. Several assumptions were made for the construction of this model. It did not consider real-world factors such as pressure drop and rate of airflow drop in the air line, due to hydrostatic pressure, or friction in the air line. The mooring chain at the OOA site was generally embedded in mud; this caused extra drag on the system. The mass of the cage was constant as added mass due to bio-fouling was not considered. All of these factors would increase the time taken for the cage to rise to the surface. The amount each of these factors contributed to the time taken to raise the cage was unknown.
There were no recorded data on actual flow rates of air or water, amount of biomass, or precise times for the cage to rise and sink. The block diagram for the model is shown in Figure 6–2.
Simulations

The first simulation was of the open loop response, without any mooring weight. This, in essence, was just the cage pulled down 30 feet, with no air in the tank, and released. This represented an unstable system, as the buoyant force exceeded the gravitational force, and the cage rose to the surface. The test was performed to determine if the drag force that was calculated for this was reasonable. Refer to Appendix E for details.

![Graph of open loop response with no mooring weight attached](image)

**Figure 6–3** Response of Simulation, open loop, with no mooring weight attached.

The time to rise to the surface was similar to estimates given by cage design engineers. After confirming the drag force, the open loop response of the cage with the mooring weight was simulated at depths of -0.21m, -2m, -4m, -6m, -8m, and -9.14m, to see if the simulated cage would maintain position.
A simple toggle style on-off controller, represented by the signum function in Simulink, was employed to see if the cage could be controlled using such a controller.

Figure 6–4 Open loop responses of cage from initial conditions of -0.21m, -2m, -4m, -6m, -8m and -9.14m
Figure 6–5 Closed Loop On-Off controller response of SBIR simulation from initial condition of -2m to intended final conditions of -2m, -5m and -8m.
Figure 6-6 Closed Loop On-Off controller response of SBIR simulation. Initial condition of -5m to intended final conditions of -2m, -5m and -8m.
Figure 6–7 Closed Loop On-Off controller response of SBIR simulation. Initial condition of -8m to intended final conditions of -2m, -5m and -8m.

The oscillatory behavior was not desirable for a response. At -2m and -5m, the oscillations were approximately 1m. This simulation was performed with no delays. If a 3-second delay was introduced, similar to the delays seen in the scale model testing, (Figure 5.3) the response was affected, resulting in higher magnitude transients.
Figure 6-8 Effect of a 3 second delay on the response of the on-off controller. Initial condition of -2m to intended final conditions of -2m, -5m, and -8m.

Figure 6-9 Effect of a three-second delay on the response of the on-off controller. Initial condition of -5m to intended final conditions of -2m, -5m, and -8m.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 6–10  Effect of a 3 second delay on the response of the on-off controller. Initial condition of -8m to intended final conditions of -2m, -5m, and -8m.

Table 6–2 Summary of effect of delays on the on-off controller

<table>
<thead>
<tr>
<th>IC</th>
<th>FC</th>
<th>Oscillation, 0s delay</th>
<th>Oscillation, 3s delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>2m</td>
<td>2m</td>
<td>~0.87m</td>
<td>~2.5m</td>
</tr>
<tr>
<td>2m</td>
<td>5m</td>
<td>~0.7m</td>
<td>~2.2m</td>
</tr>
<tr>
<td>2m</td>
<td>8m</td>
<td>~0.4m</td>
<td>~1.1m</td>
</tr>
<tr>
<td>5m</td>
<td>2m</td>
<td>~0.87m</td>
<td>~2.2m</td>
</tr>
<tr>
<td>5m</td>
<td>5m</td>
<td>~0.7m</td>
<td>~2.5m</td>
</tr>
<tr>
<td>5m</td>
<td>8m</td>
<td>~0.4m</td>
<td>~1.1m</td>
</tr>
<tr>
<td>8m</td>
<td>2m</td>
<td>~0.87m</td>
<td>~2.5m</td>
</tr>
<tr>
<td>8m</td>
<td>5m</td>
<td>~0.7m</td>
<td>~2.2m</td>
</tr>
<tr>
<td>8m</td>
<td>8m</td>
<td>~0.4m</td>
<td>~1.1m</td>
</tr>
</tbody>
</table>

Lead compensated control was developed to eliminate the oscillatory behavior.

Lead compensators reduce the transient (oscillatory) response of a system. A lead compensator is a passive network with an additional pole and zero. The lead
The compensator's pole and zero were determined using the frequency of oscillation of the on-off controller with no delay.

\[ 5 \times \frac{\text{s} + 0.6}{\text{s} + 3} \quad \text{Lead Compensator #1} \quad (25) \]

Lead controller #1 will provide approximately 40 degrees of phase margin. The DC gain of the compensator should be unity, so the transfer function is multiplied by a gain of five. A second compensator,

\[ 20 \times \frac{\text{s} + 0.3}{\text{s} + 6} \quad \text{Lead compensator #2} \quad (26) \]

with a phase margin of approximately 60 degrees, was tried and compared.

Figure 6–11 On-Off vs. both lead compensated controllers at an initial condition of -2m to an intended final condition of -2m.
**Figure 6–12** On-Off vs. both lead compensated controllers at an initial condition of -2m to an intended final condition of -5m.

**Figure 6–13** On-Off vs. both lead compensated controllers at an initial condition of -2m to an intended final condition of -8m.
Table 6-3 Summary of transients of On-Off vs. both lead controllers.

<table>
<thead>
<tr>
<th>IC</th>
<th>FC</th>
<th>Oscillation, On-Off Controller</th>
<th>Oscillation, Lead controller #1</th>
<th>Oscillation, Lead Controller #2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2m</td>
<td>2m</td>
<td>0.87m</td>
<td>0.32m</td>
<td>0.12m</td>
</tr>
<tr>
<td>2m</td>
<td>5m</td>
<td>0.7m</td>
<td>0.55</td>
<td>0.1m</td>
</tr>
<tr>
<td>2m</td>
<td>8m</td>
<td>0.4m</td>
<td>0.25m</td>
<td>0.1m</td>
</tr>
</tbody>
</table>

The lead compensators were very effective at reducing the transient response of the system. Lead Compensator #2 is more effective, owing to the higher phase margin. The effect of time delays on the lead compensated system was conducted with artificial time delays, using lead compensator #2. The simulation was then repeated using different airflows. The effects of time delays and different airflows are illustrated in Figures 6-14, 6-15, 6-16, and 6-17.
Figure 6–14 Effects of system delays on response of system, at nominal airflow rates. Initial condition of -5m to an intended final condition of -3m.

Figure 6–15 Effects of system delays on response of system, at nominal airflow rates. Initial condition of -5m to an intended final condition of -8m.
Figure 6–16  Effects of system delays on response of system, at half of nominal airflow rates. Initial condition of -5m to an intended final condition of -3m.

Figure 6–17  Effects of system delays on response of system, at half of nominal airflow rates. Initial condition of -5m to an intended final condition of -8m.
Figure 6–18 Effects of system delays on response of system, at one quarter of nominal airflow rates. Initial condition of -5m to an intended final condition of -3m.

Figure 6–19 Effects of system delays on response of system, at half of nominal airflow rates. Initial condition of -5m to an intended final condition of -8m.
Table 6-4  Summary of effects of changes in delay and airflow on SBIR system, using lead compensator #2.

<table>
<thead>
<tr>
<th>Airflow</th>
<th>IC</th>
<th>FC</th>
<th>Oscillation, 0s delay</th>
<th>Oscillation, 1s delay</th>
<th>Oscillation, 2.5s delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nom</td>
<td>5m</td>
<td>3m</td>
<td>~0.0625m</td>
<td>~0.56m</td>
<td>~1.375m</td>
</tr>
<tr>
<td>Nom</td>
<td>5m</td>
<td>8m</td>
<td>~0.0625m</td>
<td>~0.375m</td>
<td>~0.625m</td>
</tr>
<tr>
<td>Half</td>
<td>5m</td>
<td>3m</td>
<td>~0.1875m</td>
<td>~0.43m</td>
<td>~1.875m</td>
</tr>
<tr>
<td>Half</td>
<td>5m</td>
<td>8m</td>
<td>~0.2m</td>
<td>~0.25m</td>
<td>~0.6875m</td>
</tr>
<tr>
<td>Quarter</td>
<td>5m</td>
<td>3m</td>
<td>~0.25m</td>
<td>~0.7m</td>
<td>~2.8m</td>
</tr>
<tr>
<td>Quarter</td>
<td>5m</td>
<td>8m</td>
<td>~0.15m</td>
<td>~0.3m</td>
<td>~0.6m</td>
</tr>
</tbody>
</table>
The control system as tested functioned properly. The system was able to read depth and use the data to effect movement on a scale model. The supporting electronics, embedded processor, and sensors worked as expected. The LabVIEW controller communicated with the CF2, and was able to remotely active the air solenoid valves. The primary shortcoming of the system was the effect of delays, both electrical and mechanical, on the system. There are several ways the system could be improved.

**Equipment and software upgrades**

A determination should be made to see if the communication delay between the LabVIEW controller and the CF2 can be eliminated. An alternate method to increase speed of the control system response is to eliminate the LabVIEW graphical user interface. Instead, the control would be handled by the CF2, with a PC running only a text based emulator program. This makes the system simpler, and any time delays to the PC at the surface would not affect the control system. The CF2 can sample at up to 10 kHz, and its processor operates at 16 MHz. The C code for the CF2 would have to be updated to incorporate these changes.

The depth sensor chosen should be a continuously updating model that outputs a non isolated voltage or current that can be read by the A/D. The depth sensor data could be sampled at up to 10 KHz, rather than the 1 Hz from the DT-800. The air movement valves, whether they are solenoid valves or variable ball valves, should have a much smaller opening and closing delay time than the valves chosen for this project.
An alternate method of measuring the level of water in the tank is to use pressure based sensors. Pressure sensors have faster update rate, are less expensive, and could be used in non SPAR based cage designs, such as the SBIR, and OCAT cages.

Methodology

An alternate method of lift control is to have the ballast tank act as a closed system. The system would pump water, rather than air, into and out of the ballast tank. The compression of air inside the tank would not be subject to changing hydrostatic pressure. This would require pumps, and pump housings. Accurate data from the next SBIR cage deployment can be used to update the existing Simulink model to make it more accurate.
REFERENCES

1) "JPS Industries SBIR Cage Details", JPS industries, 236 Ragged Mountain Highway, Bristol, NH 03222, University of New Hampshire Ocean Engineering Department.

2) "TECH 797, Acoustics Measurement Buoy" Laurel Gaudet, Kevin Jerram, Ashley Risso, Dr. Kenneth C. Baldwin, UNH Ocean Engineering.


5) "IMAGENEX Sonar Controller", Peter Traykovski, WHOI. Diane Foster, Ohio State. Brad Butman, Marina Martini, USGS.

6) UNH OOA ¼ ton and 1 ton feed buoy controllers.

7) Electronic Industries Association EIA232 standard.

8) Electronic Industries Association EIA standards EIA-422, TIA-422.


15) "Technical principles of valves." Omega Engineering, INC. Stamford, CT
APPENDICES
APPENDIX A

C CODE FOR PERSISTOR CF2
/* ******************************************************************************************************************
  * Master control Program Rev 5.c
  ******************************************************************************************************************/

#include <cfxbios.h>  // Persistor BIOS and I/O Definitions
#include <cfxpico.h>  // Persistor PicoDOS Definitions
#include <cfxad.h>    // Generic SPI A-D QPB Driver for CFx
#include <U4S.h>      // Header file for U4S

/* Place non-PicoDAQ A-D definitions here - before including <ADExamples.h> */

#include <ADExamples.h>  /* Common definitions for the A-D Examples */
#include <assert.h>
#include <ctype.h>
#include <errno.h>
#include <float.h>
#include <limits.h>
#include <locale.h>
#include <math.h>         /* Standard C math functions */
#include <setjmp.h>
#include <signal.h>
#include <stdarg.h>
#include <stddef.h>
#include <stdio.h>       /* Standard C input output files */
#include <stdlib.h>      /* Standard C library functions */
#include <string.h>      /* standard C string functions */
#include <time.h>        /* Standard time library */

#include <dirent.h>     /* PicoDOS POSIX-like Directory Access Defines */
#include <dosdrive.h>   /* PicoDOS DOS Drive and Directory Definitions */
#include <fcntl.h>      /* PicoDOS POSIX-like File Access Definitions */
#include <stat.h>       /* PicoDOS POSIX-like File Status Definitions */

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
#include <termios.h> /* PicoDOS POSIX-like Terminal I/O Definitions */
#include <unistd.h> /* PicoDOS POSIX-like UNIX Function Definitions */

/* Place DEFINE statements here */

#define MAX 16 /* MAX is the size of the datastruct array */
#define chan1 1 /* define channel 1 as 1 */
#define baud 19200 /* Define Baudrate */
#define AIRMARBUFFSIZE 75 /* Buffersize for airmar sting data */
#define TRUE 1
#define FALSE 0
#define AirmarrPin 30
#define LVU1502Pin 31
#define EMIT 34 /* Start minute of recording interval */
#define DURATION 4 /* Duration of recording interval */

typedef struct{
    char date [21]; /* String for storing date */
    char time [21]; /* String for storing time */
    double ADdata [8]; /* Array for A/D volt data */
    char Celcius [15]; /* String for storing temperature */
    char depth [15]; /* String for storing Depth */
} datastruct;

typedef struct{
    char nmeal [25]; /* Strings for storing nmea data before */
    char nmea2 [25]; /* passing into datastruct */
} temporarystruct;

/* Function Definitions go here */

void identify (void);
void digcf2Record (datastruct, temporarystruct[]);
void LabVIEW (temporarystruct[]);
void MakeFileName (char *Filename);
void U4S_init (void);
void GetRS232Data (unsigned char * airmarbuff, temporarystruct[]);
int ParseRS232Data (temporarystruct[]);
bool checktime (void);
void LowPower (void);
void irq4RxISR (void);

/* Main Program block */

int main (void)
{
    char answer1 = 0;
    char answer2 = 0;
    datastruct list;
    temporarystruct list2[MAX];
    int kmax=0;
    int j=0;
    bool istime = FALSE;
    bool inmin = FALSE;
    char d;
    int i=0;
    int z=0;
    /* this function prints the bios revision, etc */
    identify();

    /* This function initializes the U4S */
    U4S_init();

    /* This function configures the U4S */
    U4SConfigure(chan1, baud, 'n', 8, 1);

    /* This if else statement sets an input queue of bytes. */
    If the U4S cannot set an input queue, the user is notified and the buffer is flushed */
if (U4SInitInputQueue(chan1, 16384))
    cprintf("Input queue initialized for Airmar channel \%d\n", chan1);
else{
    cprintf("Unable to initialize input queue!\nFlushing buffer in Channel \%d\n", chan1);
    U4SFlush(chan1);
}
/* Make sure the airmarr and LVU are on */
PIOSet(AirmarrPin);
TPUHostServiceCheckComplete(AirmarrPin, true);
PIOSet(LVU1502Pin);
TPUHostServiceCheckComplete(LVU1502Pin, true);

while (true)
{
    printf("x to read values\nr to enter record mode\nq to quit\n");
    d = cgetc();
    switch (d = toupper(d))
    {
/* Turn All Valves off */
    case 'O':  if (PinRead(26))
        {PIOClear(26);
            TPUHostServiceCheckComplete(TPUChanFromPin(26), true);
        }
    if (PinRead(27))
        {PIOClear(27);
            TPUHostServiceCheckComplete(TPUChanFromPin(27), true);
        }
    if (PinRead(28))
        {PIOClear(28);
            TPUHostServiceCheckComplete(TPUChanFromPin(28), true);
        }
    if (PinRead(29))
        {PIOClear(29);
            TPUHostServiceCheckComplete(TPUChanFromPin(29), true);
        }
}
*/ Turn Input valves on, and output valves off */

 case '1': if (!PinRead(26)) /* if pin 26 is low(off) then turn it on */
      {
         PIOSet(26); /* if it is already on, do nothing */
         TPUHostServiceCheckComplete(TPUChanFromPin(26),true);
      }
      if (PinRead(27)) /* if Pin 27 is on, then turn it off */
      {
         PIOClear(27);
         TPUHostServiceCheckComplete(TPUChanFromPin(27),true);
      }
      if (PinRead(28)) /* if Pin 28 (large out) is on, then turn it off */
      {
         PIOClear(28);
         TPUHostServiceCheckComplete(TPUChanFromPin(28),true);
      }
      if (PinRead(29)) /* if Pin 29 (small out) is on, then turn it off */
      {
         PIOClear(29);
         TPUHostServiceCheckComplete(TPUChanFromPin(29),true);
      }
      printf("PRIMARY INPUT valve ON\n");
      break;

 case '2': if (!PinRead(27)) /* if Pin 27 is low(off) then turn it on */
      {
         PIOSet(27); /* if it is already on, do nothing */
         TPUHostServiceCheckComplete(TPUChanFromPin(27),true);
      }
      if (PinRead(26)) /* if Pin 26 is on, then turn it off */
      {
         PIOClear(26);
         TPUHostServiceCheckComplete(TPUChanFromPin(26),true);
      }
      if (PinRead(28)) /* if Pin 28 (large out) is on, then turn it off */
      {
         PIOClear(28);
         TPUHostServiceCheckComplete(TPUChanFromPin(28),true);
      }
      printf("ALL valves OFF\n");
      break;

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
if (PinRead(29))
    {PIOClear(29); /* If Pin 29 (small out) is on, then turn it off */
        TPUHostServiceCheckComplete(TPUChanFromPin(29),true);
    }
printf("SECONDARY INPUT valve ON\n");
break;
case '3': if (!PinRead(26))
    {PIOSet(26);
        TPUHostServiceCheckComplete(TPUChanFromPin(26),true);
    }
if (!PinRead(27))
    {PIOSet(27);
        TPUHostServiceCheckComplete(TPUChanFromPin(27),true);
    }
if (PinRead(28))
    {PIOClear(28); /* If Pin 28 (large out) is on, then turn it off */
        TPUHostServiceCheckComplete(TPUChanFromPin(28),true);
    }
if (PinRead(29))
    {PIOClear(29); /* If Pin 29 (small out) is on, then turn it off */
        TPUHostServiceCheckComplete(TPUChanFromPin(29),true);
    }
printf("PRIMARY AND SECONDARY INPUT valves ONn");
break;
/* Turn Input Valves off */
case '4': if (PinRead(26))
    {PIOClear(26);
        TPUHostServiceCheckComplete(TPUChanFromPin(26),true);
    }
if (PinRead(28))
    {PIOClear(28); /* If Pin 28 (large out) is on, then turn it off */
        TPUHostServiceCheckComplete(TPUChanFromPin(28),true);
    }
if (PinRead(29))
    {PIOClear(29); /* If Pin 29 (small out) is on, then turn it off */
        TPUHostServiceCheckComplete(TPUChanFromPin(29),true);
    }

85

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
TPUHostServiceCheckComplete(TPUChanFromPin(29),true);

printf("PRIMARY INPUT valve OFF\n");
break;

case '5': if (PinRead(27))
  {
    PIOClear(27);
    TPUHostServiceCheckComplete(TPUChanFromPin(27),true);
  }
if (PinRead(28))
  {
    PIOClear(28); /* If Pin 28 (large out) is on, then turn it off */
    TPUHostServiceCheckComplete(TPUChanFromPin(28),true);
  }
if (PinRead(29))
  {
    PIOClear(29); /* If Pin 29 (small out) is on, then turn it off */
    TPUHostServiceCheckComplete(TPUChanFromPin(29),true);
  }
printf("SECONDARY INPUT valve OFF\n");
break;

case '6': if (PinRead(26))
  {
    PIOClear(26);
    TPUHostServiceCheckComplete(TPUChanFromPin(26),true);
  }
if (PinRead(27))
  {
    PIOClear(27);
    TPUHostServiceCheckComplete(TPUChanFromPin(27),true);
  }
if (PinRead(28))
  {
    PIOClear(28); /* If Pin 28 (large out) is on, then turn it off */
    TPUHostServiceCheckComplete(TPUChanFromPin(28),true);
  }
if (PinRead(29))
  {
    PIOClear(29); /* If Pin 29 (small out) is on, then turn it off */
    TPUHostServiceCheckComplete(TPUChanFromPin(29),true);
  }
printf("PRIMARY AND SECONDARY INPUT valves OFF\n");
/* Turn Output Valves On */
case '7': if (!PinRead(28))
    {
        PIOSet(28);
        TPUHostServiceCheckComplete(TPUChanFromPin(28),true);
    }
    if (PinRead(29))
        {
            PIOClear(29); /* If Pin 29 is on, then turn it off */
            TPUHostServiceCheckComplete(TPUChanFromPin(29),true);
        }
    if (PinRead(26))
        {
            PIOClear(26); /* If Pin 26 (large input) is on, then turn it off */
            TPUHostServiceCheckComplete(TPUChanFromPin(26),true);
        }
    if (PinRead(27))
        {
            PIOClear(27); /* If Pin 27 (small input is on, then turn it off */
            TPUHostServiceCheckComplete(TPUChanFromPin(27),true);
        }
    printf("PRIMARY OUTPUT valve ON
n");
    break;

case '8': if (!PinRead(29))
    {
        PIOSet(29);
        TPUHostServiceCheckComplete(TPUChanFromPin(29),true);
    }
    if (PinRead(28))
        {
            PIOClear(28); /* If Pin 28 is on, then turn it off */
            TPUHostServiceCheckComplete(TPUChanFromPin(28),true);
        }
    if (PinRead(26))
        {
            PIOClear(26); /* If Pin 26 (large input) is on, then turn it off */
            TPUHostServiceCheckComplete(TPUChanFromPin(26),true);
        }
    if (PinRead(27))
        {
            PIOClear(27); /* If Pin 27 (small input is on, then turn it off */
            TPUHostServiceCheckComplete(TPUChanFromPin(27),true);
        }
    break;
printf("SECONDARY OUTPUT valve ON\n")
break;

case '9': if (PinRead(28))
{
    {PIOSet(28);
        TPUHostServiceCheckComplete(TPUChanFromPin(28),true);
    }
    if (PinRead(29))
    {
        {PIOSet(29);
            TPUHostServiceCheckComplete(TPUChanFromPin(29),true);
        }
    }
    if (PinRead(26))
    {
        {PIOClear(26); /* If Pin 26 (large input) is on, then turn it off */
            TPUHostServiceCheckComplete(TPUChanFromPin(26),true);
        }
    }
    if (PinRead(27))
    {
        {PIOClear(27); /* If Pin 27 (small input is on, then turn it off */
            TPUHostServiceCheckComplete(TPUChanFromPin(27),true);
        }
    }
    printf("PRIMARY AND SECONDARY OUTPUT valves ON\n")
    break;
}

/* Turn Output Valves off */

case 'A': if (PinRead(28))
{
    {PIOClear(28);
        TPUHostServiceCheckComplete(TPUChanFromPin(28),true);
    }
    if (PinRead(26))
    {
        {PIOClear(26); /* If Pin 26 (large input) is on, then turn it off */
            TPUHostServiceCheckComplete(TPUChanFromPin(26),true);
        }
    }
    if (PinRead(27))
    {
        {PIOClear(27); /* If Pin 27 (small input is on, then turn it off */
            TPUHostServiceCheckComplete(TPUChanFromPin(27),true);
        }
    }
    printf("PRIMARY AND SECONDARY OUTPUT valves OFF\n")
}
break;

case 'B': if (PinRead(29))
{
    PIOClear(29);
    TPUHostServiceCheckComplete(TPUChanFromPin(29),true);
}
if (PinRead(26))
{
    PIOClear(26); /* If Pin 26 (large input) is on, then turn it off */
    TPUHostServiceCheckComplete(TPUChanFromPin(26),true);
}
if (PinRead(27))
{
    PIOClear(27); /* If Pin 27 (small input is on, then turn it off */
    TPUHostServiceCheckComplete(TPUChanFromPin(27),true);
}
printf("PRIMARY AND SECONDARY OUTPUT valves OFF\n");
break;

case 'C': if (PinRead(28))
{
    PIOClear(28);
    TPUHostServiceCheckComplete(TPUChanFromPin(28),true);
}
if (PinRead(29))
{
    PIOClear(29);
    TPUHostServiceCheckComplete(TPUChanFromPin(29),true);
}
if (PinRead(26))
{
    PIOClear(26); /* If Pin 26 (large input) is on, then turn it off */
    TPUHostServiceCheckComplete(TPUChanFromPin(26),true);
}
if (PinRead(27))
{
    PIOClear(27); /* If Pin 27 (small input is on, then turn it off */
    TPUHostServiceCheckComplete(TPUChanFromPin(27),true);
}
printf("PRIMARY AND SECONDARY OUTPUT valves OFF\n");
break;

/* Record data for labview program */

case 'X': LabVIEW(list2);

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
/* This is the main recording loop. It operates as long as the keyboard is not pressed */

break;

/* Turn off the airmarr and LVU to conserve power */

PIOClear(AirmarrPin);
TPUHostServiceCheckComplete(TPUChanFromPin(AirmarrPin),true);
PIOClear(LVU1502Pin);

TPUHostServiceCheckComplete(TPUChanFromPin(LVU1502Pin),true);

/* Main loop of recording mode */

while(!kbhit())
{
    if ((istime = checktime()) == TRUE)
    {
        if (inin)
        {
            inmin = TRUE;

            /* Turn on pins for sensor devices and let them power up oriented */

            PIOSet(AirmarrPin);
            TPUHostServiceCheckComplete(TPUChanFromPin(AirmarrPin),true);
            PIOSet(LVU1502Pin);
        }
        for (z=0;z<5001;z++)
        { Delay1ms();

            /* Function to record RS232 data, */
            /* and sample and record A/D data */
            digcf2Record(list,list2);

            /* Turn off pins for sensors */
            PIOClear(AirmarrPin);
        }
    }
}

for (z=0;z<5001;z++)
    Delay1ms();
TPUHostServiceCheckComplete(TPUChanFromPin(AirmarrPin),true);

else
{
inmin=FALSE;
/* Here is the lowpower mode */
cprintf("Entering Low power mode for 1 minute\n");
LowPower();
printf("Entered higher power mode\n");
printf("Checking Time\n");
}
} /* END of ELSE */

/* END Of Kbhit while */
P1OSet(AirmarrPin);
TPUHostServiceCheckComplete(AirmarrPin,true);
P1OSet(LVU1502Pin);
TPUHostServiceCheckComplete(LVU1502Pin,true);
printf("Exiting Recording mode (wait 3 seconds)\n");
for (i=0;i<3000;i++)
{ Delay1ms();
break;
case 'Q': printf("program quit as commanded\n");
return 0;
} /* END OF SWITCH */
} /* END OF WHILE */

U4SFlush(chan1);
U4SClose();
cprintf("Program terminated normally");
return 0;
} //END OF MAIN

*****************************************************************************

91
FUNCTION GetRS232Data

Function Purpose: To retrieve data strings from equipment connected to the U4S, and write them to a file

Function Input: An unsigned char

Function Output: NONE

void GetRS232Data(unsigned char * airmarbuff, temporarystruct list2[])
{
    int i=0;
    int imax=0;
    int j=0;
    int jmax;
    int intvar=0;

    if (U4SRxCharsAvail(chan1)>0)
    {
        while (intvar<AIRMARBUFFSIZE)
        {
            airmarbuff[intvar] = U4SRxGetChar(chan1);
            intvar+=1;
        }
    } /* END OF While */
    else
    {
        for (i=0;i<100;i++)
        {
            Delay1ms();
            /* END OF For */
            /* END OF ELSE */
        }
    }

    /* Increase the counter and add a null character to end the string */
    intvar+=1;
    airmarbuff[intvar] = '\0';

    if (strcmp(airmarbuff, "$SDDP", 5)==0)
    {
        strncpy(list2[i].nmea1, &airmarbuff[7],4);
    }
}
```c
list2[i].nmea1[4] = '0';
i++;
imax = i;
}
/* END OF IF */

else if (strcmp(airmarbuff,"$YXM", 4)==0)
{
    strncpy(list2[j].nmea2, &airmarbuff[7],4);
    list2[j].nmea2[4] = '\0';
j++;
jmax = j;
}
/* END OF ELSE IF */

// else if (strcmp(buffer,"N",1)==0)
// { strcpy(list2[0].nmea1,"None");
//   strcpy(list2[0].nmea2,"None");

U4SFlush(chan1); 
}
/*END OF FUNCTION GETDATA*/

************************************************************************
* Function Dlgcf2Record                                           *
* Function Purpose: This is a modified version of Dlgcf2. It records the *
* file name in ddmmyhm format, then records a header line *
* it then enters a loop, where it records the date, time, *
* all 8 A/D channels, then the altitude and depth for a *
* User specified period of time                                    *
* Function Input : Array of typedef struct datastruct             *
* Function output : Int 0                                           *
************************************************************************
void digcf2Record(datastruct list, temporarystruct list2[])
{
    bool uni = true;
    /* true for unipolar, false for bipolar */
    bool sgl = true;
    /* true for single-ended, false for differential */
```
short *samples;
short i = 0; /*counter */
short k = 0; /*counter */
CFxAD adbuf, *ad;
float vref = VREF;
FILE *thefilep;
time_t t;
size_t n;
struct tm *Currenttime;
char daymonthyear[20];
int m;
char hourminsec[20];
char Filename[22];
char filenm[9];
unsigned char airmar_array[AIRMARBUFFSIZE] = "AIRMAR";
int minafter;
ad = CFxADInit(&adbuf, ADSLOT, ADInitFunction);

    t = time(&t);
    Currenttime = localtime(&t);
    m=9;
    minafter = Currenttime->tm_min;
    strftime(filenm,m,"%d%m%y%H",Currenttime);
    strcpy(Filename,"C:\Data\");  
    strcpy(Filename,"C:\Data\");  
    strcat(Filename,filenm);  
    strcat(Filename,".dat");  

    thefilep = fopen(Filename,"w");

    printf("created filename\n");
    /* Put timestamp and date stamp at beginning of file, also print onscreen */
    fprintf(thefilep,"Date Time Ch1 Ch2 Ch3 Ch4 ");
    fprintf(thefilep," Ch5 Ch6 Ch7 Ch8 Depth Temperature");
    fclose(thefilep);
printf("recording data\n");

thefilep = fopen(Filename,"a");

while (minafter<EMIT+DURATION)
{
    t = time(&t);
    Currenttime = localtime(&t);
    n=20;
    /* The following code gets the current date, and writes it to the file */
    strftime(daymonthyear,n,"%x",Currenttime);
    strcpy (list.date, daymonthyear);
    fprintf(thefilep,"\n\n%s " ,list.date);
    /* The following code gets the current time, and writes it to the file */
    strftime(hourminsec,n,"%X",Currenttime);
    strcpy (list.time, hourminsec);
    fprintf(thefilep,"%s ",list.time);
    /* This function samples all 8 A/D channels */
    fflush(stdout);
    DBG( PinClear(25); )
    samples = CFxADSampleBlock(ad, 0, 8, 0, uni, sgl, false);
    DBG( PinSet(25); )
    for (i = 0; i < 8; i++)
    {
        list.ADdata[i] = CFxADRawToVolts(ad, samples[i], vref, uni);
        fprintf(thefilep,"%5.3f ",list.ADdata[i]);
    }
    /* END OF "i" FOR LOOP */
    /* This function checks for and receives data from the U4S */
    /* into a file called Rawnmea.txt */
    GetRS232Data(airmar_array,list2);
    strcpy (list.depth, list2[k].nmea1);
    fprintf(thefilep,"%s ",list.depth);
    strcpy (list.Celcius, list2[k].nmea2);
    fprintf(thefilep,"%s",list.Celcius);
    /* END of Kmax for loop */
U4SFlush(chan1);
minafter = Currenttime->tm_min;
*/ END OF EMIT WHILE LOOP */
fclose(thefilep);

print("Finished recording data\n");
return;
} /* END OF FUNCTION DIGCF2RECORD */

оличество функции LabVIEW

* Function Purpose: This is a modified version of Digcf2record. It sends the *
* same data as digcf2record, but does not record it to file *
* 
* Function Input :  Array of typedef struct datastruct
*  
* Function output :  Int 0
* 
void LabVIEW(temporarystruct list2[])
{
bool uni = true; /* true for unipolar, false for bipolar */
bool sgl = true; /* true for single-ended, false for differential */
short * samples;
short i = 0; /*counter */
short k = 0; /*counter */
CFxAD adbuf, *ad;
float vref = VREF;
time_t t;
size_t n;
struct tm *Currenttime;
char daymonthyear[20];
unsigned char airmar_array[AIRMARBUFFSIZE] = "AIRMAR";

ad = CFxADInit(&adbuf, ADSLOT, ADInitFunction);
flush(stdout);

DBG( PinClear(25); )
samples = CFxADSampleBlock(ad, 0, 8, 0, uni, sgl, false);
DBG( PinSet(25); )

t = time(&t);
Currenttime = localtime(&t);
n=20;
strftime(daymonthyear,n, "%x", Currenttime);
printf("Date: %s\n", daymonthyear);
printf("Time: %d:%d:%d", Currenttime->tm_hour, Currenttime->tm_min, Currenttime->tm_sec);

for (i = 0; i < 8; i++)
{
    printf("\nCh %d = %5.3fV", CFxADRawToVolts(ad, samples[i], vref, uni));
}
/* END OF "i" FOR LOOP */

GetRS232Data(airmar_array,list2);

printf("\nDepth:%sm\n", list2[0].nmea1);
printf("\nTemperature:%s\n", list2[0].nmea2);
/* END of Kmax loop */

/* needed to make channel 7 appear on screen. Who knows why? */
printf("\n");

return;
} /* END OF FUNCTION LabVIEW */

*******************************************************************************
* FUNCTION Checktime *
* Function Purpose: Determine if it is time to get data *
* Function input : None *
*******************************************************************************
Function output: Boolean depending on whether it is time to collect data.

```
bool checktime(void)
{
    int minafter;
time_t t;
    struct tm *Currenttime;

    t = time(&t);
    Currenttime = localtime(&t);
    minafter = Currenttime->tm_min;
    if (minafter == EMIT)
    {
        return (TRUE);
    }
    else
    {
        return (FALSE);
    }
}
```

FUNCTION LOWPOWER

Function Purpose: To enter a low power state
Function Input: NONE
Function Output: NONE

```
void LowPower(void)
{
    int counter = 0;
    int i=0;
    // IEVInsertAsmFunct(lrq4RxISR,level4InterruptAutovector);
    // IEVInsertAsmFunct(lrq4RxISR,spuriousInterrupt);
    SCITxWaitCompletion();
}
```
EIAForceOff(true);
PITSet100usPeriod(PIT40Hz);
//  PinBus(IRQ4RXD);

for (counter=0;counter<40*60;counter++)
{
    LPStopCSE(FullStop);
}

EIAForceOff(false);
PITSet100usPeriod(PITOff);

return;
}"END OF FUNCTION LOWPOWER */

//void rte(void) = {0x4E73};

//void irq4RxdSR(void)
//     {PinlO (IRQ4RXD);
//     }

FUNCTION U4S_init
Function Purpose: This function initializes the U4S, using code taken verbatim ** from the U4S example files.*
Function input :  NONE
Function Output :  NONE

void U4S_init(void)
{
    short numu4s;
    short irqused;

    SCBInit(SCB_DEFAULTS, SCB_DEFAULTS, true);

    SCSDevices = SCSInit();
cprintf("InSCSDevices = %d in", SCSDevices);

} /*END OF FUNCTION LOWPOWER */
/* This revision of U4SFindCards displays a lot of diagnostic data. Future revisions will get a flag to determine whether to show the diagnostics or not. */

numu4s = U4SFindCards(true);
if(numu4s < 1){ /* can't do anything--exit to PicoDos */
cprintf("No U4S supervisor was found! 
");
    BIOSReset(); /* full hardware reset */
}
else{ /* get base address from */
    U4SResetUART(u4devs[0].scdev);
u4sbase = u4devs[0].scbase;
    irqused = 2; //u4devs[0].scirq; /* Original code, changed */
}
    /* END OF ELSE */ /* due to IQR crashing */

/* Setting the irq to 2, as it is set in the hardware jumper This seems to have fixed the constant crashing due to IRQ issues */
    U4SSetInterrupt(irqused);
cprintf("u4sbase from u4Devs array = %lx IRQ= %d \n", (long)u4sbase, irqused);

SCITxWaitCompletion();

Function Identify *

Function Purpose: Prints bios and revision information to the screen
* code taken Verbatim from examples from PII

Function Input : NONE

Function output : NONE

*---------------------------------------------------------------------*/

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
void identify(void)
{

    /*Identify the program and build */
    printf("inixeProgram: %s: %s %s \n", __FILE__, __DATE__, __TIME__); 

    /* Identify the device and its firmware */
    printf("Persistor CF%d SN:%ld BIOS:%d.%02d PicoDOS:%d.%02d/nin", CFx, 
BIOSGVT.CFxSerNum, BIOSGVT.BIOSVersion, BIOSGVT.BIOSRelease, 
BIOSGVT.PICOVersion, BIOSGVT.PICORelease);
    return;
}

/* END OF FUNCTION IDENTIFY */

/* Revision History */

Rev 1: This is the first attempt at combining the recording code and the active control code.
Now, the recording code from active control2.c is a subset of the SWITCH, 'R'. This is
to avoid having to make changes to multiple code files, etc.

Rev 2: Added a function LabVIEW. Because Digcf2 record now has a 900 run loop, I cannot use it for
talking to labview, as I did in active control. The function labview will be basically a 1 or
2 run loop of digcf2record, but without writing to file. Changed digcf2record to record for 15
minutes rather than a loop, this way if the speed of acquisition changes, there will still be
15 minutes of data. Last Nmea data of 1 file becomes 1st nema data in next file, resulting
in a temperature anomaly.

Rev 3: Added a lowpower moe to the code. During the recording phase, the unit checks te time, and
if it is not time to record, the unit enters LPS(fullstop) and turns off the RS232 driver on the
CF2 for 1 minute. Suspend mode does not provide enough benefit for the amount of code required.
Changing the clock speed to decrease power also did not provide any benefit, and caused ASCII
characters to become garbage, and cause communication failure. Code was also added to make sure the Airmarr and
LVU1502 are on when needed for data recording.

Rev 4: The code was updated to include:
1) a '0' command in the switch statement to shut off all valves
2) All Valve commands are modified so that input and output valves
will not operate at the same time

101

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
**Descriptions**

**Airlift command:** This pull down menu has four choices. Each of the four choices will send an ASCII character to the CF2.

- **Quit:** Q command.
- **Raise or lower mode:** X command
- **Record mode:** R command
- **Write buffer:** User typed command, for diagnostic purposes.

**Desired Depth:** The user commanded depth setting, the one the program will try to achieve via air control.

**E-stop:** This Boolean command turns off all valves. The main loop continues to run, however, the 0 command is always sent to the CF2. It is red for visibility, and turns gray, with a green light, when activated. It can also be used as a diagnostic tool.

**File Path:** LabVIEW will write all recorded data to a .csv file specified here.

**Stop Airlift program:** This stops the program after allowing the main loop to finish executing. Using the taskbar stop will stop the program at whatever section of the loop it is currently on, and will not close the serial port properly, or write the last line of data to file.

**VISA Resource Name:** This is the name of the Com port used by LabVIEW to communicate. VISA, (Virtual Instrument Software Architecture) is a series of pre-created VIs for initializing, reading, writing, or closing a communication port. In this case, it is the COM port of the computer used by the operator. The Parity, Baud, stop and data bits are controls on the on the front panel, but are hidden, since these parameters are not needed to be changed very often.
**Big Input valve, Small Input Valve, Big Output Valve, and Small Output Valve:**

These indicators are red when the solenoid valve they represent is off, and will turn green if the valve is on. These Booleans activate if the voltage feedback from the respective solenoid valves is greater than 1.6V, indicating voltage is applied to the valve.

**Current Time:** Current time, in military format

**Current Date:** Current date in format

**Depth Chart:** Graphical display of depth readings

**Depth Error:** Difference between Present Depth and Desired Depth

**Present Depth:** This bar represents the present depth of the cage. It is accompanied by a vertical graph bar.

**Read buffer:** This string indicator displays any ASCII data coming from the CF2.

**Tank level:** This numerical bar graph indicator represents the amount of water/air currently in the air tank. It is accompanied by a vertical graph bar.

**Temperature Chart:** Graphical display of water temperatures

**Velocity:** Present velocity of cage

**Water Temperature:** This represents the current temperature in degrees Celsius. It is accompanied by a vertical graph bar.

**24V battery:** Present Voltage level of 24V battery.

**12V battery:** Present Voltage level of 24V battery.

Note that any variable or indicator used by the LabVIEW block diagram can be made to appear on the front panel for diagnostic purposes.
Solenoid driver circuit

The output pins on the CF2 can generate 3.3V, but they are for logic level only, meaning they cannot supply the electrical current needed to drive the solenoid valves or the sensors. A Field Effect Transistor (FET) can be used; however the TPU also does not drive the FET efficiently. A FET driving circuit must be used.

![Solenoid driver circuit diagram](https://example.com/diagram.png)

**Figure C-1** Solenoid driver circuit.

When the 3.3V TPU pin from the CF2 is on, the PNP transistor Q1 is forward biased (turned on) and provides a path for current to flow from the 24VDC source to ground through R5 and R1. This creates 16V from gate to source on the P-channel FET U1, turning it on. Current then flows from the 24VDC source through the FET to the solenoid valve, turning it on. D1 is reverse biased, and is off. The voltage divider formed by R3 and R4 creates a 2V signal to be fed back to the A/D converter on the CF2 to provide indication of the valve is on or off. When the 3.3V TPU is turned off, Q1 turns off, and hence U1. The energy stored in S1 flows through D1 until dissipated.

Each solenoid valve will have one of these drivers, for a total of four.
Depth and Level Sensor Turn on Circuit

The DT-800 altitude sensor and the LVU 1502 tank level sensor do not need to be operating except when the CF2 is actively recording. This occurs when the X command is executed, or when the record mode (R command) is in the 15 minute recording interval. At other times, the devices are shut off to conserve energy. CF2 TPU pins will be used for this purpose, using the same turn on circuit as the solenoid driver circuit. These circuits do not have a resistor divider feedback; if the devices fail, there will be no exporting of data; this can be used to determine failure. Further, the devices could fail in such a way that voltage is present, but data is not exported. The purpose of the regulator, set at 22V, is to set a common level that satisfies the required ranges of the altitude sensor (11.5-25V) and level sensor (18-30V) using the 24V battery (20-27V).

Figure C–2 DT-800 and LVU-1502 turn on circuit.
A drawback to the LVU-1502 is the isolated output. The 4-20mA current output of this level sensor is not referenced to the same ground potential as the sensor's power supply, or the CF2 A/D. Simply connecting the 4-20mA to the CF2 A/D through a resistor will result in erroneous voltage readings. A solution is to use a Hall Effect sensor. A Hall Effect sensor is a transducer that varies its output voltage in response to changes in magnetic field. The model chosen for this application is the LV-20P from LEM. This model has an input range of 0-10mA, and an output range of 0-25mA. Since the LVU outputs 4-20mA, a current divider was placed on the input to the LV-20P. One half of this current divider is the internal resistance of the LV-20P, (220ohms) and the other half is R4, a 220ohm carbon film resistor. This causes 2-10 ma to flow through the LV-20P input, giving an output current through R1 of 5-25mA, resulting in a voltage (VR1) of 0.195 to 0.977 V across R1. Since this voltage will be VR1 + V2 to ground, R3 and R2 form a voltage divider to reduce the voltage below the 2.5V required by the A/D.
Table C–1  Voltage across output resistor R1 (VR1) of LV20P.

<table>
<thead>
<tr>
<th>LVU-1502 output, mA</th>
<th>LV20P input, mA</th>
<th>VR1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>2</td>
<td>0.196</td>
</tr>
<tr>
<td>4.5</td>
<td>2.25</td>
<td>0.220</td>
</tr>
<tr>
<td>5</td>
<td>2.5</td>
<td>0.244</td>
</tr>
<tr>
<td>5.5</td>
<td>2.75</td>
<td>0.269</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>0.293</td>
</tr>
<tr>
<td>6.5</td>
<td>3.25</td>
<td>0.318</td>
</tr>
<tr>
<td>7</td>
<td>3.5</td>
<td>0.342</td>
</tr>
<tr>
<td>7.5</td>
<td>3.75</td>
<td>0.367</td>
</tr>
<tr>
<td>8</td>
<td>4</td>
<td>0.391</td>
</tr>
<tr>
<td>8.5</td>
<td>4.25</td>
<td>0.415</td>
</tr>
<tr>
<td>9</td>
<td>4.5</td>
<td>0.440</td>
</tr>
<tr>
<td>9.5</td>
<td>4.75</td>
<td>0.464</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>0.489</td>
</tr>
<tr>
<td>10.5</td>
<td>5.25</td>
<td>0.513</td>
</tr>
<tr>
<td>11</td>
<td>5.5</td>
<td>0.538</td>
</tr>
<tr>
<td>11.5</td>
<td>5.75</td>
<td>0.562</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>0.587</td>
</tr>
<tr>
<td>12.5</td>
<td>6.25</td>
<td>0.611</td>
</tr>
<tr>
<td>13</td>
<td>6.5</td>
<td>0.635</td>
</tr>
<tr>
<td>13.5</td>
<td>6.75</td>
<td>0.660</td>
</tr>
<tr>
<td>14</td>
<td>7</td>
<td>0.684</td>
</tr>
<tr>
<td>14.5</td>
<td>7.25</td>
<td>0.709</td>
</tr>
<tr>
<td>15</td>
<td>7.5</td>
<td>0.733</td>
</tr>
<tr>
<td>15.5</td>
<td>7.75</td>
<td>0.758</td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>0.782</td>
</tr>
<tr>
<td>16.5</td>
<td>8.25</td>
<td>0.806</td>
</tr>
<tr>
<td>17</td>
<td>8.5</td>
<td>0.831</td>
</tr>
<tr>
<td>17.5</td>
<td>8.75</td>
<td>0.855</td>
</tr>
<tr>
<td>18</td>
<td>9</td>
<td>0.880</td>
</tr>
<tr>
<td>18.5</td>
<td>9.25</td>
<td>0.904</td>
</tr>
<tr>
<td>19</td>
<td>9.5</td>
<td>0.929</td>
</tr>
<tr>
<td>19.5</td>
<td>9.75</td>
<td>0.953</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>0.978</td>
</tr>
</tbody>
</table>
The device is powered from -12 and +12V sources. Each power input has diode protection to prevent damage in case of reverse polarity connection. The output current of the LV-20P flows through output resistor R1, back to the reference of the +12 and -12V sources.

Figure C-3 LEM to A/D circuit schematic.
**Depth command**
This is the depth the cage is commanded to move to. The range is 0 to -9.14m, (0 to -30 feet). This number is the height of the bottom of the cage rim below the ocean surface.

**Start depth**
This is the initial starting position of the cage. It is also based on the height of the cage rim below the surface. The range is the same as the depth command.

**Volume tank limit.**
This sets the upper and lower limits of air inside the tank to 0 and 1 cubic meter of air, respectively.

**Volume to force conversion.**
This gain relates the volume of air in the air tank into its corresponding buoyant force, via the equation $\text{Force}=\text{gravity} \times \rho \times \text{Volume}$

**Downward gravitational force of cage**
This is the total mass of the cage and airlift, multiplied by the acceleration of gravity. It exerts a negative (downward) force at all times.

**Downward gravitational force of cage**
This is the total mass of the chain always suspended. It exerts a negative (downward) force at all times.

**Main systems**

**Upward buoyant force**
This is the buoyant force caused by the displacement of the cage. This excludes the buoyant force of the air tank. It always acts in a positive (upward) direction.

**Variable transport delay, and Time Delay**
These are used to simulate the effects of electrical or mechanical delays.
Integrator
This integrates the flow rate, creating a volume of air inside the tank. The initial condition is specified by Initial air in tank subsystem.

Subsystems

Flow rate decider
If the cage is rising, air is supplied by a compressor, with an output pressure of 90 psi, and a flow rate of 6 standard cubic feet per minute. If the cage is sinking, the flow rate is determined by the pressure differential between atmospheric pressure and the hydrostatic pressure at the present depth of the cage. This block inputs the direction the cage is moving, and the present depth of the cage, and outputs either the input or output flow rate. The output flow rate lookup table was computed with an online calculator, using a hose length of 90 feet, and a hose ID of 0.5".

Physical plant.
The physical plant subsystem consists of parameters associated with the physical system. The inputs are the sum of all forces, and the starting position of the cage. The sum of forces is multiplied by the reciprocal of the mass of the system, giving the acceleration of the system, via $F = m \frac{\Delta F}{m} = a$. The acceleration is then integrated, giving the velocity of the system. The velocity is used internally to the physical plant subsystem in a feedback loop, to calculate drag force, and passed through an integrator, to calculate position. The signal is also an output of the physical plant subsystem, for use in other parts of the simulation model. The second input, start depth, is used as the initial condition of the position integrator, i.e., the starting depth of the cage rim. The position is fed back through a gain loop to calculate the increase in mass due to the lifting of chain when the cage rises, and the decrease in mass when the cage descends, and the chain rests on the ocean floor. The drag force output is fed to...
the **Drag Force Direction** subsystem. The Velocity squared output is used only for diagnostic purposes.

**Drag Force Direction**

The drag force acts opposite the direction of velocity, thus it can exert a positive or negative force. Since the Simulink summing block inputs are either positive or negative, this subsystem inverts the sign (and thus direction) of the drag force to account for direction. It examines the velocity, and compares it to zero. If the cage is moving upwards, (positive velocity, \( v > 0 \)) the drag force is multiplied by 1, hence the drag force will be subtracted at the summing block. If the cage is moving downward, (negative velocity, \( c < 0 \)) the drag force is multiplied by -1, hence the drag force will be added at the summing block. If the cage is not moving, the drag force will be zero, and the sign is irrelevant.

**Limit buoyant force at zero**

The upper rim of the cage is comprised of hollow tubes filled with air, these provide the permanent buoyancy. These are oriented parallel to the surface of the water. When the top of the cage rim starts to breach the surface of the water, (depth = -0.2m) the buoyant force of the cage is reduced, since the air in the tubes are no longer displacing water. This subsystem examines the position of the cage, and determines if the cage rim is at or below -0.2m. If it is, the switch passes through the normal buoyant force of the cage, and the subsystem has no effect. If the cage rim moves above -0.2m, then the HAL 9000 subsystem block calculates the percent decrease in buoyant force of the cage based on the decrease in volume of water displaced as the cage rim moves above the surface. The normal buoyant force is then multiplied by this percentage.
**Hal 9000**

The Hal 9000 block input to is position, in negative meters, which is converted to positive inches. The main body of the subsystem is a representation of the equation

\[ y = 0.003x^5 - 0.0055x^4 + 0.0312x^3 - 0.0252x^2 + 0.0126x - 0.0005 \]

where \( x \) is the vertical distance of the cage rim below the surface of the water, and \( y \) is the percent buoyant force remaining in the cage rim at that distance. Thus, at -0.2m feet, the cage retains all of its buoyancy, but as the cage rim rises to 0 feet, the cage buoyant force reduces to zero. The derivation for the formula is in appendix?

**Air volume expansion in tank**

As the air tank rises or descends, the air inside the tank expands or compresses in volume, due to the change in hydrostatic pressure. Since the mass of the air cannot change, if the air is compressed, the pressure will change, and vice versa. The compression over the range 54 feet to 84 feet was found to be 34%. The block inputs the present depth, and applies a delay to this signal. The delayed signal is subtracted from the original signal, and this delta is multiplied by the average change in volume per meter.

**Air volume expansion into tank**

This subsystem accounts for the for the air expansion into the air tank from the inlet hose. It uses Boyle's law (assuming constant temperature), \( P_1 V_1 = P_2 V_2 \). \( P_1 \) and \( V_1 \) are the pressure and volume rate from the air source (normally an air compressor) and assumed to be constant. \( P_2 \) is the hydrostatic pressure at a certain depth, via \( P = \rho g H \), where \( H \) is the height below the surface. Thus, the new volumetric flow rate can be expressed as

\[ V_2 = \frac{P_1 V_1}{P_2} \]
**Increase in gravity force due to chain lifted.**

As the cage rises, it lifts the chain off the sea floor, which increases the downward gravitational force on the cage. This block models the effect of this extra chain.

**Initial air in tank.**

This represents the air in the tank at specific heights. This is the air needed for the system to have a stable open loop response.
APPENDIX E

CALCULATION OF CIRCULAR SEGMENTS

APPLIED TO THE RIM OF THE SBIR CAGE
Calculations

The equation used in the Hal 9000 subsystem block was derived using the principle of the area of circular segments. Figure X shows a circular segment, shaded yellow.

![Circular segment diagram]

Figure E-1 Circular segment.

If figure X represents a cross section of one of the tubes of the cage rim, and the yellow shaded area represents a section of the cage rim out of the water, then the formula

\[
A = \frac{R^2}{2} (Q - \sin(Q)),
\]

where \(R = d + h\), can be used to find the area of that segment. The formula was used to find the area of successive segments, using 0.5" increases in \(R\) of the cage rim as it moved out of the water. These segment areas were divided by the total area of the pipe.
Table E-1  Data for the HAL subsystem block.

<table>
<thead>
<tr>
<th>R=h+d</th>
<th>Theta</th>
<th>Area of segment</th>
<th>Percent of total area</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00%</td>
</tr>
<tr>
<td>0.5</td>
<td>0.40</td>
<td>0.08</td>
<td>0.17%</td>
</tr>
<tr>
<td>1</td>
<td>0.80</td>
<td>0.63</td>
<td>1.30%</td>
</tr>
<tr>
<td>1.5</td>
<td>1.20</td>
<td>2.06</td>
<td>4.23%</td>
</tr>
<tr>
<td>2</td>
<td>1.59</td>
<td>4.61</td>
<td>9.47%</td>
</tr>
<tr>
<td>2.5</td>
<td>1.99</td>
<td>8.39</td>
<td>17.22%</td>
</tr>
<tr>
<td>3</td>
<td>2.39</td>
<td>13.27</td>
<td>27.24%</td>
</tr>
<tr>
<td>3.5</td>
<td>2.79</td>
<td>18.97</td>
<td>38.96%</td>
</tr>
<tr>
<td>3.9375</td>
<td>3.14</td>
<td>24.33</td>
<td>49.96%</td>
</tr>
<tr>
<td>4</td>
<td>3.19</td>
<td>25.10</td>
<td>51.54%</td>
</tr>
<tr>
<td>4.5</td>
<td>3.59</td>
<td>31.17</td>
<td>64.00%</td>
</tr>
<tr>
<td>5</td>
<td>3.99</td>
<td>36.71</td>
<td>75.38%</td>
</tr>
<tr>
<td>5.5</td>
<td>4.39</td>
<td>41.34</td>
<td>84.89%</td>
</tr>
<tr>
<td>6</td>
<td>4.78</td>
<td>44.82</td>
<td>92.04%</td>
</tr>
<tr>
<td>6.5</td>
<td>5.18</td>
<td>47.09</td>
<td>96.69%</td>
</tr>
<tr>
<td>7</td>
<td>5.58</td>
<td>48.27</td>
<td>99.12%</td>
</tr>
<tr>
<td>7.5</td>
<td>5.98</td>
<td>48.67</td>
<td>99.94%</td>
</tr>
<tr>
<td>7.875</td>
<td>6.28</td>
<td>48.71</td>
<td>100.01%</td>
</tr>
</tbody>
</table>

This chart was then graphed, and an equation was found to relate the percent decrease in area to the value of R above the surface of the water.

\[ y = 0.0003x^5 - 0.0055x^4 + 0.0312x^3 - 0.0252x^2 + 0.0126x - 0.0005 \]

Figure E-2  Graphed data for the HAL subsystem block.