DEVELOPMENT OF A SUBMERSIBLE ESCAPE CAPSULE FOR OFFSHORE STRUCTURES

SUPANGPEN RUANGWECH

DEVELOPMENT OF A SUBMERSIBLE ESCAPE CAPSULE FOR OFFSHORE STRUCTURES

BY

© SUPANGPEN RUANGWECH, B.ENG.

A thesis submitted to the
School of Graduate Studies
in partial fulfillment of the
requirements for the degree of
Master of Engineering

Faculty of Engineering And Applied Science
Memorial University of Newfoundland
December 2005

St.John's Newfoundland Canada

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-19392-1
Our file Notre référence
ISBN: 978-0-494-19392-1

Abstract

During the past 20 years, many accidents in offshore oil and gas industry have taken away the lives of many workers. A new type of evacuation system is introduced in this research. Whereas, all other evacuation systems developed were purposely designed to remove an entire crew from a structure during emergencies, this research describes the development of a high speed escape capsule that could remove a single person from the structure. Capsules would be placed at strategic locations on the deck of the structure. During an emergency each capsule would be catapulted from the deck to the ocean surface. To avoid the effect of strong wind and waves, after impact with the ocean surface, the capsule would sink into the ocean down to a level where water motions are insignificant and wait until the storm subsides. This research focuses only on the post impact trajectory of the capsule beneath the ocean surface. A physical model of the capsule was designed. Two front thrusters were used to generate surge and yaw motions. Heave and pitch motions were controlled by another thruster located at the back. A pressure transducer was used to measure depth, and a digital compass was used to determine direction. A Peripheral Interface Controller (PIC) was used to control the motion of the capsule using an I^2C bus to send PWM signals to 20 Amp drivers. A SIMULINK model was developed to study control strategies and the influence of various vehicle parameters. The model capsule was tested in the Deep Water Tank in Faculty of Engineering and Applied Science at Memorial University of Newfoundland and in the Offshore Engineering Basin at the Institute for Ocean Technology. The simulations and tests showed that the capsule can move smoothly along preset trajectories.

Acknowledgements

The author wishes to thank Dr. Michael James Hinchey for handing the author an opportunity to conduct the research under his excellent supervision. Dr. Hinchey provided financial support, assisted with the project, supported and spent numerous hours with the author throughout the entire duration of the research. His kindness, support and understanding must at this moment in time be acknowledged.

Personal thanks goes to the author's parents, Montrawong and Sujin Ruangwech, for financial support, their encouragement, respect and understanding during the entire time the author has spent in Canada. Thanks goes to the technicians and staff of the Faculty of Engineering and Applied Science, Memorial University of Newfoundland, namely Tom Pike for his brilliant advice on the Peripheral Interface Controller (PIC); Bernard Healy for his great help with the fabrication of the model capsule; Craig Mitchell and Jim Gosse for accessing the facilities in Thermal Laboratory and Fluid Laboratory and their assistance with the experiment of the model capsule; Richard Newman, Ryan McCarthy, Dennis Johnson, for their advice, support and assistance throughout the project. The author also wishes to thank Technical Services staff at Memorial University of Newfoundland, namely Leo Spurrell for his kindness; Jason Barret of the machine shop, as well as Mac Butler of the welding shop. Furthermore, the author wishes to thank staff at the Institute for Ocean Technologies (IOT), namely António Simões Ré for his great advice on evacuation systems; OEB staff for their assistance with the experiment of the model capsule at the IOT. Special thanks go to the author's brother, sister and all of author's friends for their support and assistance throughout the project.

Table of Contents

List of Figures

List of Tables

List of Symbols

W	weight
Δ	buoyancy
C_G	the center of gravity
C_B	the center of buoyancy
$\overline{RM}$	the righting moment
$\overline{GB}$	the metacentric height for submerged conditions
β	either the heel angle or the trim angle
θ	the heel angle
α	the trim angle
$\overline{GZ}$	the righting arm
F	total force in x or y direction
U	relative velocity of submersible in x, y or z direction
N	total moment about an axis through the center of gravity of submersible and parallel to z-axis
Ω	angular velocity about x, y, or z axes
m	mass of submersible
I	masses/mass moment of inertia of submersible
ψ	angular displacement about x, y or z axes
Q	command signal
e	error (command - response)
K_P	proportional gain
K_I	integral gain

K_D	derivative gain
Δt	the sampling or loop rate
φ	linear or angular displacement of the submersible
L_C	control load from the submersible's thrust-producing devices
L_D	disturbance load
L_H	hydrodynamic load
L_S	hydrostatic load
M	mass of each component
R	distance of each component from axis of rotation
A	dynamic coefficient
B	static coefficient
L_C	control load from the submersible's thrust-producing devices
Q	control signal
C_D	drag coefficient of the body
ρ	fluid density
A	projected area of the underwater portion of the body
C_M	inertia coefficient of the body
V	volume of the body
$\dot{U}$	relative acceleration of the body
W_n	weight of each components
d_n	distance from each component to the reference line and the center of buoyancy can be determined from

V_n	volumetric displacement of each components
d_n	distance from each component to the reference line
F_{Drag}	form drag
K	drag factor
s	distance required for the capsule to travel
v	velocity of the capsule
t	time taken for the capsule to travel
g	gravitational acceleration
ω	angular velocity
K_{CR}	the critical gain
P_{CR}	the critical period
H Bridge Gain	H-Bridge gain uses in the 3D-motion simulation
V_{output}	output voltage from H-Bridge module
V_{input}	input voltage to H-Bridge module

List of Abbreviations

AIF	Atlantic Innovation Fund
AUV	Autonomous Underwater Vehicle
IOT	Institute for Ocean Technology
IRT	Ice Reinforced Totally Enclosed Motorized Personnel
I^2C	Inter-Integrate-Circuit
MUN	Memorial University of Newfoundland
NE	Neutral Equilibrium
OEB	Offshore Engineering Basin
PC	Personal Computer
PIC	Peripheral Interface Controller
PID	Proportional Integral Derivative Controller
PNA	Principles of Naval Architecture
PRAC	Petroleum Research Atlantic Canada
PrOD	Preferred Orientation and Displacement
PWM	Pulse Width Modulation
TEMPSC	Totally Enclosed Motor Propelled Survival Craft
SAR/NIF	Search & Rescue New Initiatives Fund
NECEC	Newfoundland Electrical and Computer Engineering Conference
SE	Stable Equilibrium
USE	Unstable Equilibrium
SPI	Serial Peripheral Interface

Chapter 1

Introduction

Most evacuation systems currently used by offshore structures are designed to remove an entire crew from the structure during emergencies. In this research, a high speed escape capsule for offshore structures, namely the Submersible Escape Capsule, was designed. Its purpose would be to remove a single person who remained on board during emergencies. The goal would be to evacuate this person from the structure at the last minute. Capsules would be placed at strategic positions and each would be catapulted from the deck of the structure. When a capsule reaches the ocean surface, it would sink below the surface and move away from the danger zone near the structure. In shallow water, it would land on the seabed. The motion of the capsule would be controlled automatically. However, it would have a manual override capability. It would also have fail safe features.

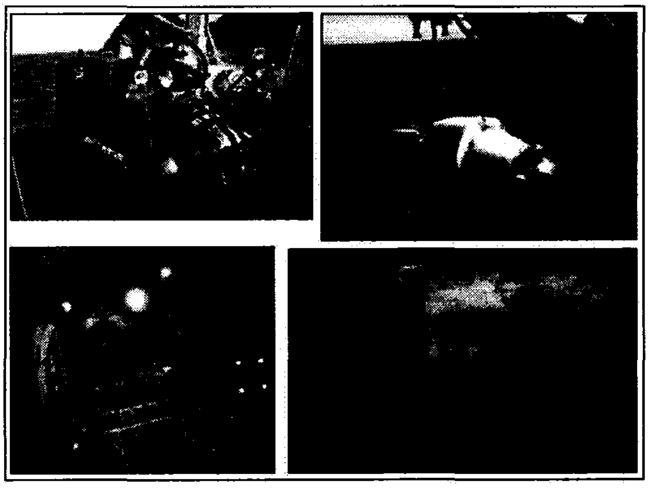


Figure 1.1 Manned Underwater Vehicles

There are many manned underwater vehicles in existence. Some typical vehicles are shown in Figure 1.1. The submersible on the top right is ALVIN which was developed at Woods Hole Oceangraphic Institution. The one below is called PISCES. The one on the top left is DEEP EXPLORER and the one below is called JSL. All of these vehicles are slow moving and designed mainly for scientific missions. They would not withstand the impact with the ocean surface. So a special design is needed.

The capsule was designed to operate in waters off the East Coast of Canada to deal with emergencies such as a severe storms or explosions on oil rigs or platforms. Some structures which have operated in this area are shown in Figure 1.2.

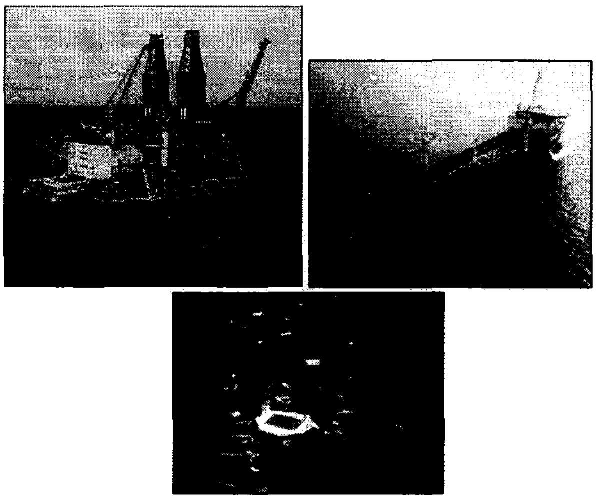


Figure 1.2 Offshore Structures around Newfoundland

The structure in the top left photo is the Hibernia GBS. The structure in the top right photo is the FPSO Vessel for the Terra Nova and White Rose fields. The structure in the bottom photo is the Ocean Ranger Oil Rig. Its sinking in 1982 with the loss of 84 lives provided the motivation for the present work.

A schematic of a prototype capsule is shown in Figure 1.3. Its shape is such that it can hold a single person in a normal seated position. The shape is one that can also impact the ocean surface with low deceleration levels. This is the rationale for the shape. The capsule uses two variable speed reversible propellers to control surge and yaw motions and one variable speed reversible propeller to control heave and pitch motions. The variable speed reversible propellers used in this research have very high power. The capsule would be moved very fast along any trajectory. Thus, it would be able to get away from the structure very fast. A ballast tank is not used because it would move the capsule much slower than propellers and it is mechanically much more complicated.

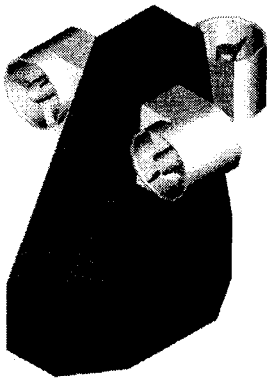


Figure 1.3 The Prototype of Submersible Escape Capsule Schematic

There are many factors that need to be considered for the design of this submersible. Examples include: (1) the catapult launch mechanism (2) the impact with the water surface (3) the pre impact in air trajectory (4) the post impact in water trajectory (5) the capsule structure (6) the capsule components (7) the life support system. This work focused on the 3D trajectory path of the capsule beneath the ocean surface. It examined the possibility of using three high speed motors for the propulsion and maneuvering system. A control system for the capsule was designed for controlling its depth and direction. A model was designed and constructed, and it was tested in the Deep Water Tank at Memorial University or Newfoundland (MUN), and in the Offshore Engineering Basin (OEB) at the Institute for Ocean Technology (IOT). A SIMULINK simulation of the capsule was developed to study its underwater motions.

The thesis starts in Chapter 2 with a review of evacuation problems and evacuation systems currently available. Chapter 3 provides the theoretical background needed to design the Submersible Escape Capsule. In Chapter 4, a 3D path simulation is developed and the derivation of block diagrams and their simulation results are discussed. The model capsule was constructed to confirm this concept. Its details are described in Chapter 5, while Chapter 6 discusses the details of experiments and their results. Concluding remarks and recommendations for future work are contained in Chapter 7.

Chapter 2

Review of Evacuation System

2.1 Preamble

During the past 20 years, many accidents have occurred in the offshore oil and gas industry. According to the historical accident records, most fatalities occur during the escape and evacuation phases. For example, during the time period from 1970 to 1991, offshore accidents in the North Sea caused a total number of 318 fatalities and more than 90% of these fatalities happened during evacuation [1]. One of the worst offshore disasters of all time was the capsizing of Alexander Kielland during March 1980: structural failure of one leg caused it to capsize. A report shows that "a total of 89 of the platform's 212 personnel were rescued" [1]. Fatalities in this accident were caused mainly by the disruption of lifeboat and life raft due to rough weather and heeling of rig [1]. Capsize of Jack Up - Ocean Express in 1976 is another offshore accident which occurred in the North Sea. As a result of this accident, "[a]ll 13 crew were found dead in an overturned rescue capsule" [1]. Another major accident was the capsizing of the Ocean Ranger on February 15th, 1982. The Ocean Ranger was equipped with 2 Harding Lifeboats with 50 person capacity, 2 Watercraft Lifeboats also with 50 person capacity, and 10 inflatable life rafts, each with 20 person capacity. "... [T]here were no survivors from the 84 crew" [1]. After the accident, both Harding Lifeboats were found damaged. They were assumed to have impacted with the rig during launch. One out of two of the Watercraft Lifeboats was missing. No one knows whether it was used or not. The other systems were found unused [1]. Furthermore, the reports made by the Worldwide Offshore Accident Data Bank and other sources reveal that, there were 123 offshore rigs and installations destroyed between the years of 1970 to 1990 and caused the loss of over 1300 lives [2].

2.2 Evacuation System Categories

A variety of evacuation systems have been continually developed. This is to provide safer means of evacuation to the offshore worker. As stated in [3], "[t]he evacuation systems are identified and categorized as dry, semi-wet, and wet systems". The details of each system are described below.

2.2.1 Dry Systems

As described in the Summary Report - Offshore Evacuation Systems prepared for Transportation Development Centre Safety and Security Transportation Canada [4], "[d]ry evacuation systems are systems that involve the emergency evacuation of personnel directly from the offshore installation to a rescue craft, standby vessel, or shore base, which is independent from the offshore installation itself." The most commonly used dry evacuation system is the helicopter [3]. The other examples of this system include gangbridge, personnel transfer basket, and cable transfer systems [4]. The reliability of such systems depends mostly on the time provided for evacuation process and vulnerability to accident, especially when the evacuation process has to take place in severe weather condition [5].

2.2.2 Semi-wet Systems

As described in the Summary Report - Offshore Evacuation Systems prepared for Transportation Development Centre Safety and Security Transportation Canada [4], "[s]emi-wet evacuation systems are systems that involve the emergency transfer of personnel by evacuation vehicle ... " These vehicles are generally installed on the offshore structure, and they are boarded by crew before launching to the ocean surface. A good example for this type of system is the conventional lifeboat; its main purpose is to evacuate large numbers of crew away from the hazard area around the structure. This system typically incorporates two discrete components: the evacuation vehicle and the launch system [4].

2.2.3 Wet Systems

Wet evacuation system includes items such as immersion suits and lifejackets. This system aims to get crew, who are unable to reach either the primary or secondary evacuation systems, away from the area of immediate hazard or for recovery of personnel [1], [5].

2.2.4 Comparison of Systems

A survey of evacuation systems conducted by Scott Newbury Marine Consulting suggests that "[t]he dry evacuation of personnel from a safe heaven on the offshore platform to a safe heaven on a standby vessel or shore base is an ideal method of evacuation. However, due to their limitations, they may not be considered as a viable evacuation system in an emergency" [5]. Because of this limitation, "... the survey leads to the conclusion that semi-wet systems will play a role in evacuation plans for the foreseeable future" [5].

2.3 Review of Evacuation and Launching Systems

All currently approved systems for the Canadian Offshore employ a Totally Enclosed Motor Propelled Survival Craft (TEMPSC) [5]. Its purpose is to transport people safely from the deck of a structure to the sea and then to a place of safety, during an emergency. Being totally enclosed with an independent oxygen supply, the occupants inside are protected from foreseeable hazards, such as explosions and toxic gases in the air, especially in case of fire, earthquake or collision, in which severe structural damage may be experienced by the structure [5]. During an emergency event, crews board the TEMPSC before it is launched to the ocean surface. A variety of launching systems have been developed for TEMPSC and other types of lifeboat and life raft. Launch from Conventional Davits is the most common launching method. This consists of a cantilevered davit and boarding platform which allows the lifeboat to descend directly into the sea on wire cables. A major problem with this system is the release mechanism, which often releases the lifeboat at the wrong time. Another disadvantage of this system is there are high risks associated with launching in extreme weather conditions. A pendulum like motion of the lifeboat on the launching cables, caused by high winds, can cause the lifeboat to impact with the offshore structure [1], [3], [5].

To eliminate this damage problem, free fall lifeboats have been developed and have been adopted in the North Sea. As the name implies, they free fall from the deck of the structure to the ocean surface: when they hit the ocean surface, they often have momentum pointing them away from the structure. For obvious reasons, they are not suitable for use in Arctic areas, when ice covers the water surface. In addition, "[o]ne of the major concerns of the free fall system is the effect of the deceleration upon entry into the water" [5]. Thus, the system must be designed so that it "... must not expose the occupants of the craft to excessive forces that could cause injury ..." [5] to the crew members.

Another system designed to eliminate the damage problem is Preferred Orientation and Displacement (PrOD) lifeboat launching system. It was originally developed at Watercraft International in the United Kingdom. It was designed to launch the lifeboat away from the structure. In addition, it provides a force on the lifeboat to give it some momentum away from the structure. Although, the potential remain for problems caused by failure of the release mechanisms, this system overcomes the problem of escaping the vicinity of the platform. Recently, the PrOD launch system has undergone successful full scale trials and has been installed on several structures [1], [5].

Another system designed to avoid the impact problem is known as SEASCAPE. It uses a boom instead of cables to launch the lifeboat. The lifeboat sits in a cradle at the top of the boom. During launch, the boom swings slowly out from the structure. The lifeboat enters the water well away from the structure and also pointing away from it. The disadvantage of this system is it is heavier than other launching methods thus it is not suitable for floating platforms [1].

The other types of semi-wet evacuation system include the Inflatable Life Raft and the Marine Escape System. These two systems are sometimes combined with each other. The Marine Escape System is an escape slide or a vertical drop escape chute linking the deck of the offshore installation to an inflatable boarding platform. The evacuees then use this platform to enter the life raft. These systems are intended to provide a secondary means of evacuation as when the evacuation by the primary means fails or is not available [5].

Several special evacuation systems have been developed for Arctic areas. One of these is known as the Ice Reinforced Totally Enclosed Motorized Personnel Safety Craft (IRT). "The IRT hull form is designed ... to ride up out of the water when pressured by iced" [6] which makes the IRT hull suitable for operating in both ice and open water. Davit Launched TEMPSCs have also been used in Arctic areas. They have been modified "... to accommodate structural geometry, which may include icebreaking appurtenances at the water line, and other unique structural characteristics ..." [6] of platforms in Arctic areas. Special superstructures have been designed to house the TEMPSC to prevent icing and other polar debilitating effects that may occur on its launch mechanism [6].

2.4 Previous Works on the Submersible Escape Capsule

The idea of using a submersible escape capsule for offshore structures originated with Hinchey, a Professor of Mechanical Engineering at Memorial University of Newfoundland. The first prototype capsule was designed to have a capacity of 2 persons; it was approximately 4 feet in diameter and 8 feet high. It had a cylindrical center body with approximately hemispherical end caps. View ports were located on the cylinder wall and escape hatches were located in both end caps [3]. A finite element method analysis of the capsule hull was conducted by King (1990). Detailed designs of a door and a port were also conducted by Lunt (1991). The details of these analyses are not discussed here since they are beyond the scope of this research.

The impact of free-fall submersible capsules onto the ocean surface was studied using various basic capsule shapes. The study showed that "[n]early all decelerations were below 5g and had scaled durations less than 0.25 seconds" [3] and "the capsule with the cylindrical bottom feels the greatest deceleration" [3]. The upper limit on deceleration for a human is 10g if it lasts not more than 0.25 seconds; thus it appears that impact deceleration would not be a serious problem for people inside the capsule [3].

The post impact trajectory of the capsule beneath a water surface was studied using an idealized geometry [7]. A 0.25 meter diameter spherical model was constructed with a central buoyancy chamber closed on top but open at the bottom. Two control strategies were chosen for study. The objective of each strategy was to make the spherical model float at 2 meters below the water surface in the MUN deep tank. In the first strategy, airflow was sent into the model by the controller when the depth was below 2 meters, which forced water out of the chamber. This made the buoyancy of the capsule greater than its weight and caused it to ascend toward the control depth. When the depth was above 2 meters, air was vented from the chamber. This allowed water to enter the chamber, increasing the weight of the capsule which caused it to descend toward the control depth. In the second strategy, the direction of motion within a band centered on the command depth was taken into account. Outside the range, air was sent into the model when depth was below the range and it was allowed to vent to atmosphere when the depth was above the band, as in the first strategy. Within the band, air was sent to the model when it was moving downward and air was vented to atmosphere when the model was moving upward. The derivative nature of this control counteracted integration in system dynamics and made the system dynamically stable. Good qualitative agreement was obtained between theory and experiment [7].

High speed launch of a capsule by a gas bottle rocket was studied both theoretically and experimentally by a group of work term students [8]. This work showed that a single diver bottle with a 0.5 inches rocket nozzle at its exit could throw a 125 kilograms capsule out 25 meters from a structure while it is falling 25 meters to the ocean surface. Several work term students and one student design group worked on the rocket launch project.

2.5 Summary

Many accidents have occurred in the offshore oil and gas industry and they have taken away the lives of many workers. Many types of evacuation system have been developed to try to prevent such loss. These evacuation systems can be categorized as dry, semi-wet, and wet systems. None of them are adequate. A new type of evacuation system called a Submersible Escape Capsule is being developed at Memorial University of Newfoundland. This thesis describes a study aimed at examining the ability of the capsule to move smoothly along a 3D trajectory underwater.

Chapter 3

Design of Underwater Vehicles

3.1 Preamble

The Submersible Escape Capsule is basically an autonomous underwater vehicle (AUV). Normally, an AUV is designed to be independent and able to automatically perform underwater tasks. Its main purpose is to eliminate the presence of humans in an unsafe environment. In this research, the Submersible Escape Capsule is designed to automatically evacuate a human from the hazard area around offshore structures when there are severe storms or when there are explosions or fire onboard.

This chapter is concerned with the basic design of submersible vehicle systems. Section 3.2 provides a review of Hydromechanical principles - Hydrostatics and Hydrodynamics. These principles are directly applicable to the design of manned submersible vehicle including their stability and their motion underwater while performing their mission. Section 3.3 provides an overview of control theories normally applied on AUVs. The control strategy selected for the submersible escape capsule is discussed later in section 3.4.

3.2 Hydromechanical Principles

The Submersible Escape Capsule is basically an autonomous underwater vehicle with human inside it. The design of this submersible will follow the guidelines for such vehicles described in the book Submersible Vehicle Systems Design by Allmendinger [9] and in Principles of Naval Architecture [10]. These guidelines are based on hydromechanical principles.

Hydromechanical principles deal with the behavior of a submersible when forces/moments generated by the following sources: weight, buoyancy, lift and drag, thrust and torque, contact forces, and/or inertia are under consideration [9]. The forces/moments created by these sources are shown in Figure 3.1 and their details are described later in Chapter 4.

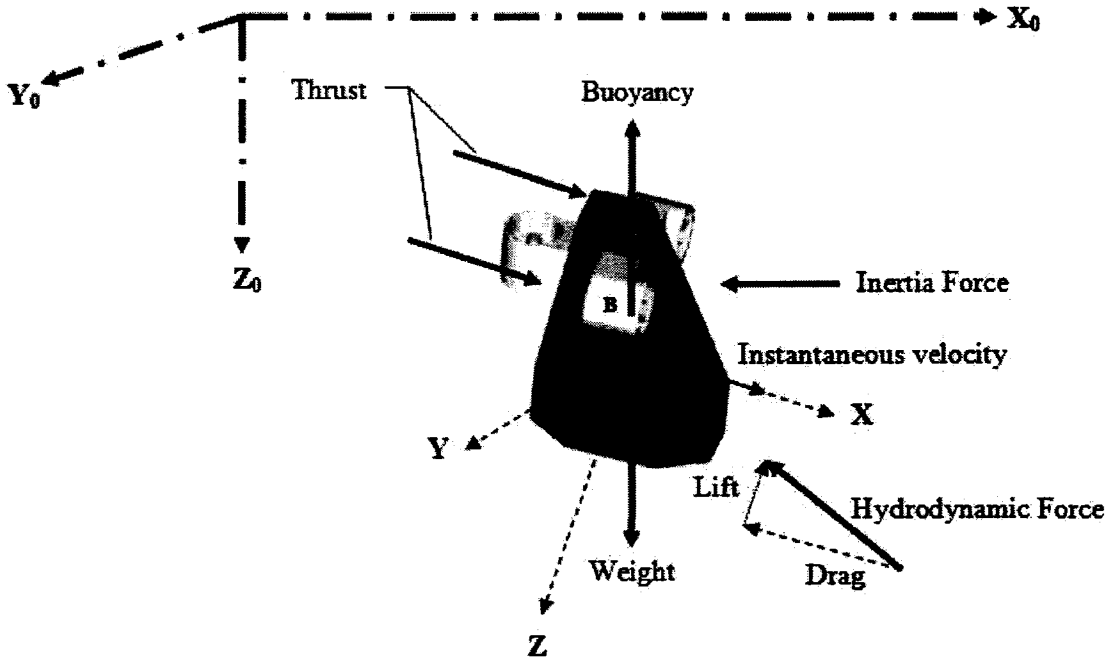


Figure 3.1 Forces and moments acting on a neutral-buoyancy submersible (modified from [9])

The following sections review the principle of hydrostatics and hydrodynamics as they are directly applicable to the design of neutral-buoyancy submersibles.

3.2.1 Hydrostatics

Hydrostatics deals with the case where there is no relative motion between the submersible under study and the surrounding water particles [9]. When applied to the design of submersible, this principle is mostly concerned about: weight-buoyancy relationship of the submersible, its static equilibrium and stability fundamentals [9]. According to the book Principles of Naval Architecture [10], the interaction of weight (W) and buoyancy (Δ) of a submersible is a major consideration for determining equilibrium. Consider a submersible with no external forces acting on it. The submersible will settle when the two of following conditions are satisfied: (1) its buoyancy (Δ) is equal to its weight (W) and (2) its center of gravity (C_G) and its center of buoyancy (C_B) are collinear in the same vertical line; any slight rotation from this position will cause the forces of weight (W) and buoyancy (Δ) to create a moment. This moment tends to return to the submersible to its original equilibrium position or it causes the submersible to depart further seeking for its new position of equilibrium.

Two equilibrium fundamentals must be considered when designing a submersible. First, consider the submersible when it is acted on by two equal but opposite collinear forces. These two forces, as shown in Figure 2.2, consist of the submersible's total weight (W) and its buoyancy (Δ). For equilibrium to exist the resultants of all forces/moments acting on a submersible must be zero [9]. Thus, the submersible is either at rest, with no motion relative to the earth axes origin, or in steady-state motion with respect to these axes [9].

$$\begin{array}{ll} X = 0 & K = 0 \\ Y = 0 & M = 0 \\ Z = 0 & N = 0 \end{array} \tag{3.1}$$

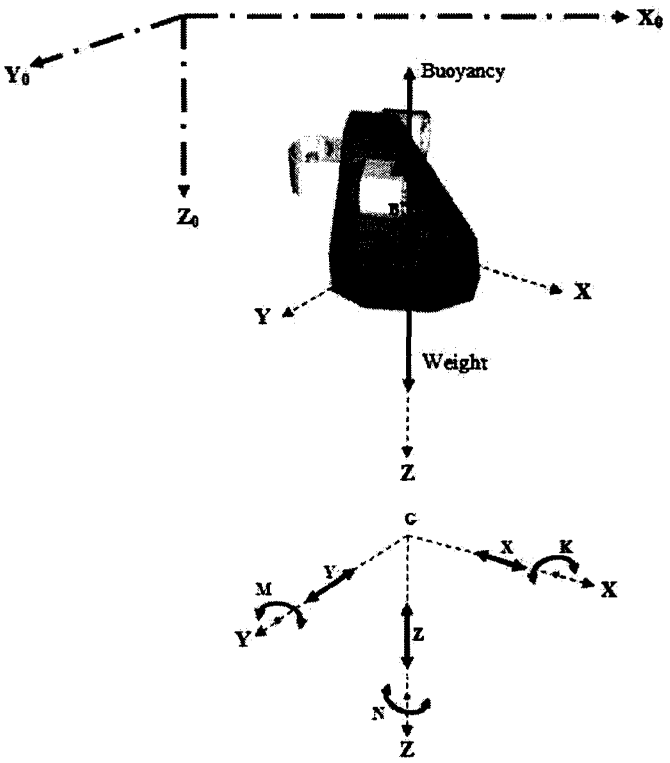


Figure 3.2 Example of static forces

The second fundamental is concerned with equilibrium state of a submersible. There are three different types of equilibrium: stable, unstable, and neutral [9]. The following text is summarized from the book Submersible Vehicle Systems Design by Allmendinger [9]. It describes a submersible when it is in different states of equilibrium. Consider the three neutral-buoyancy submersibles A, B and C in the design condition as shown in Figure 3.3. Suppose momentary, external forces are applied to each of them, long enough to slightly incline each submersible, as illustrated in Figure 3.3 [9].

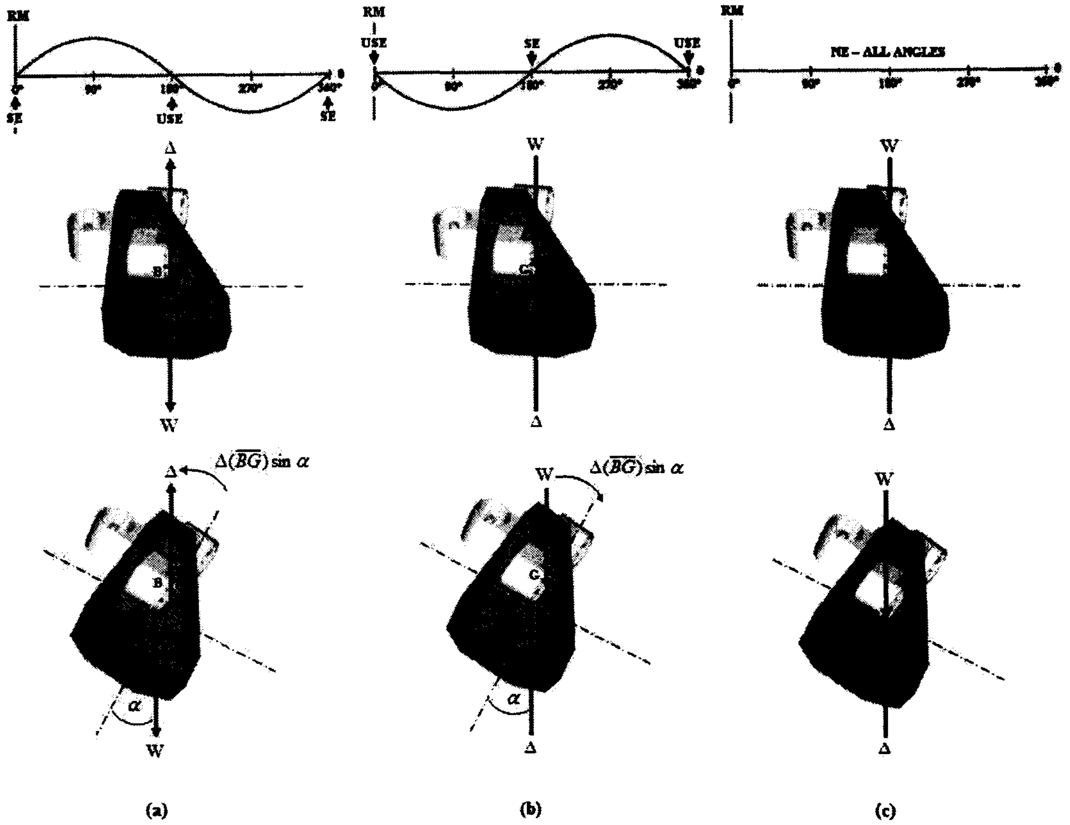


Figure 3.3 Equilibrium and stability (modified from [9])

Submersible A, as shown in figure 3.3(a), has the center of gravity (C_G) below the center of buoyancy (C_B). At 0 or 360 degrees, the weight-buoyancy couple acts to return the submersible to its initial position. This restoring couple is called a positive righting moment; submersible A's initial state is described as stable equilibrium (SE). As the submersible is inclined beyond 180 degrees, negative righting moments are developed. As a result of this moment, the submersible departs further from the 180 degrees and seeks a new position of equilibrium. The submersible state at 180 degrees is described as unstable equilibrium (USE). The new position of equilibrium for this submersible is at 360 or 0 degrees [9]. As concluded by Allmendinger [9], "… submersible A will "insist" on returning to its upright position when inclined to any angle once the inclining influence is removed."

Submersible B, as shown in figure 3.3(b), has the center of gravity (C_G) above the center of buoyancy (C_B). At 0 or 360 degrees, the weight-buoyancy couple now acts to incline the submersible further away from its initial position until it reaches a new position of equilibrium. This new position of equilibrium is at 180 degrees. If the submersible B is inclined beyond 180 degrees, positive righting moments are developed. These positive righting moments then cause the submersible to return back to the 180 degree position [9]. As concluded by Allmendinger [9], "... submersible B will "insist" on floating in a "keel-up" position once it inclined slightly away from the upright position".

Submersible C, as shown in figure 3.3(c), has the center of gravity (C_G) and the center of buoyancy (C_B) at a coincident point. Thus, there is no weight-buoyancy couple created at any angle of inclination from 0 to 360 degrees [9]. As concluded by Allmendinger [9], "... the submersible neither returns to nor departs further from its incline position ..." once it has been inclined away from the upright position and the momentary, external force is removed. Submersible C's initial state is determined as neutral equilibrium (NE) [9].

The righting moment ($\overline{RM}$) for all three cases can be calculated from the following equation:

$$\pm \overline{RM} = \Delta(\pm\overline{GB})\sin\beta \qquad \textbf{(3.2)}$$

where Δ = displacement of a submersible

$\overline{GB}$ = metacentric height for submerged conditions

β = either the heel angle (θ) or the trim angle (α)

When considering the stability of a submerged vehicle, as opposed to a surfaced submersible, the center of buoyancy (C_B) of the submerged vehicle does not move with respect to the vehicle as the vehicle is inclined. Therefore, despite the presence of effects causing a virtual rise of G, the center of buoyancy (C_B) and the metacenter[1] are coincident [9]. Thus, "[s]ubmerged stability is a function of metacentric height ($\overline{GB}$) throughout the entire range of transverse and longitudinal inclinations" [9]. As described briefly by Allmendinger [9], maneuvering requirements are used when determining the upper limit of$\overline{GB}$. Anyhow, the lower limit can also be determined but it must be such that to provide enough positive stability when a virtual rise of G can be presented.

The righting moment ($\overline{RM}$) in equation (2) can also be replaced by the righting arm ($\overline{GZ}$), where $\overline{GZ}$ is the distance between the parallel line-of-action of weight (W) and buoyancy (Δ) when the submerged vehicle is inclined. Thus, the righting-arm curve for a submerged vehicle is equal to the metacentric height ($\overline{GB}$) multiplied by the sine of the angle of inclination and can be restated as

$$\overline{GZ} = \overline{GB}\sin\beta \qquad \textbf{(3.3)}$$

When a submersible is designed to have the center of gravity (C_G) below the center of buoyancy (C_B), the theoretical range of positive stability is from 0 to 180 degrees [10].

[1] The *metacenter* is the point where a vertical line through the center of buoyancy (C_B), when the surfaced submersible is inclined, intersects the original vertical through the center of buoyancy (C_B).

This means capsizing of an intact submerged vehicle[2] is impossible, if the metacentric height of the submersible has at least a small positive value. Even through this is theoretically true, in a practical sense the factor such as crew comfort, maximum angle of safety, and effective operation equipment must be considered when determining the operating range of inclination of a submersible [9].

3.2.2 Hydrodynamics

Hydrodynamics deal with the case where there is a relative motion between the submersible under study and the surrounding water particles. This motion generates hydrodynamics forces/moments on the submersible; and this "[m]otion can be furthered characterized in terms of degrees of freedom" [9].

As in the case of an untethered-neutral-buoyancy submersible, like the Submersible Escape Capsule, the submersible is free to move in three dimensions. Thus, the submersible may generate up to the total of six degrees of freedom. These six degrees of freedom include translation in x, y, and z directions and in rotation about these axis [9]. The examples of forces/moments applied to generate motion of the submersible are: lift and drag; thrust and torque; contact forces; and inertia. According to Allmendinger [9], "The forces acting on a submersible can be equated to time rate of change of momentum of the submersible and the entrained water moving with the submersible" ; and "[t]he moments are equal to the time rate of change of angular momentum" [9]:

[2] Its hull can maintain a complete watertight integrity.

$$F = \frac{d(Momentum)}{dt} = \frac{d(mU)}{dt} \qquad \textbf{(3.4)}$$

$$N = \frac{d(Angular\ Momentum)}{dt} = \frac{d(I\Omega)}{dt} \qquad \textbf{(3.5)}$$

where F = total force in x,y or z direction

U = velocity of submersible in x, y or z direction

N = total moment about an axis through the center of gravity (C_G) of submersible and parallel to z-axis

Ω = Angular velocity about x, y, or z axes

When the mass of the submersible is constant, the equations of motion of a submersible reduce to the familiar Newton's Law equation of

$$F = m\frac{dU}{dt} \qquad \textbf{(3.6)}$$

$$N = I\frac{d^2\psi}{dt^2} \qquad \textbf{(3.7)}$$

where F = total force in x or y direction

m = mass of submersible

U = velocity of submersible in x, y or z direction

N = total moment about an axis through the center of gravity (C_G) of submersible and parallel to z-axis

I = mass moment of inertia of submersible about an axis through the center of gravity (C_G) of submersible and parallel to z-axis

ψ = angular displacement about x, y or z axes

The above differential equations are called the governing equations of motion. They are used to analyze the motion of the submersible; the path generated by the submersible can be predicted using these equations [9]. Since an untethered-neutral-buoyancy submersible may possess up to six degrees of freedom, a complex set of six coupled ordinary differential equations are required to predict its motions and its path. These six differential equations are derived for: (1) Surge, (2) Sway, (3) Heave, (4) Roll, (5) Pitch, and (6) Yaw. The details on the governing equations of motion used in the design of the Submersible Escape Capsule and the details of each control loop for these motions will be discussed later in Chapter 4.

3.3 Submersible Control System

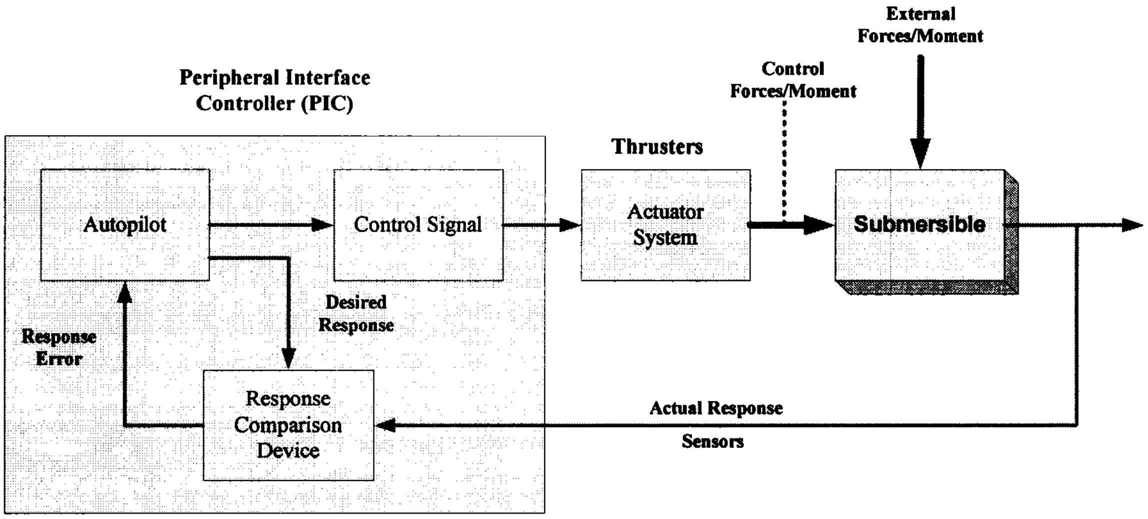


Figure 3.4 An Overall Control System for Submersible Escape Capsule (modified from [9])

The main function of control systems of submersibles is "... to enable them to change safely and effectively from one operating condition to another or to retain a particular operating condition in the presence of forces/moments attempting to change it" [9]. In the case of the Submersible Escape Capsule, the main function of its control system is to hold the submersible at minimum error to the command path. An automatic control would be properly designed to detect the error between the actual path and the desired path. This automatic control would also be used to generate a corrective action to drive the submersible to its command depth and point it in the command direction. This corrective action is generated by mean of the control forces/moments generated by the thrust producing units. The major components and overall block diagram for the control system of the Submersible Escape Capsule is shown in Figure 3.4.

3.3.1 An Overview of Control Strategies

There are many different types of control strategies that could be used for motion control of a submersible. They can be divided into two categories, namely, conventional control strategies and supervisory control strategies [11]. Conventional control strategies generate control signals based on the difference between the desired response and the actual response. These types of control can also be called error-driven control or feedback control even though some of them make use of the control system governing equations. Examples for these types of control include Proportional Integral Derivative or PID control, Switching control, Computed Load control, and Sliding Mode control. Supervisory control strategies are designed to replicate a human operator [12]. Their command response is generated based on knowledge provided by experts. Action rules, such as IF-THEN-ELSE, weighting system or a combination of both are used for decision making. Examples for these types of control system include Artificial Intelligence control, Neural Networks control, and Fuzzy Logic control [11].

A popular control strategy for exploration submersibles is the control strategy known as Subsumption. Mostly, this type of control strategy has been used in the application of mobile robots which can operate independently in a closed environment.

Subsumption was developed by Rodney Brook and his colleagues at the Massachusetts Institute of Technology (MIT). It was designed to mimic the human thought process by breaking it down into a number of layers. Each layer represents a behavior of the system. These layers normally operate independently of each other. However, the top layers have higher priority. It will take over or subsume control from lower layers when conditions permit [12], [13]. Subsumption gives robots insect like intelligence.

Finally, control strategies currently in use on submersibles are sometimes a combination of both categories. For example, supervisory control such as Neural Networks or Fuzzy Logic can be used to feed information, such as set point or appropriate gains, to the conventional controls as their input. As a result, their efficiency in performing underwater tasks is increased [12].

3.3.2 Control Strategy Selected for Submersible Escape Capsule

The control strategy selected for motion control of the Submersible Escape Capsule is error-driven control. This type of control can also be called feedback control. Figure 3.4 shows a typical feedback control system. There are two types of error-driven control: Switching control and Proportional Integral Derivative (PID) control [14].

There are many types of switching control. They often have trouble with overshooting. Basic relay switching is the simplest example. During operation, the switching control would try to make the propellers rotate at a constant speed. The direction of rotation would depend on the sign of depth error. Relay with deadband can be used to allow the submersible to drift, once it gets inside a band surrounding the set point. The propulsion device would be shut down and drag load would cause the submersible to slow down. Relay with hysteresis would reverse the direction of control before the submersible gets to the set point. In this case, the propulsion device would act as a brake. A bias signal could be added to relay signals to counteract disturbance.

The PID control strategy is a combination between Proportional, integral and derivative control. The control signal for this type of control is based on the difference between the actual response and the desired response of the submersible's position. The controller then computes the derivative and integral of this error signal. The control signal of the PID controller is then calculated from

$$Q = K_P e + K_I \int edt + K_D \frac{de}{dt} \tag{3.8}$$

where Q = command signal

e = error or difference between the actual response and the desired response

K_P = proportional gain

K_I = integral gain

K_D = derivative gain

In this research, the Submersible Escape Capsule is controlled by a small onboard computing unit. Thus, the control signal is generated within a digital control loop. Thus, the equation for the lowest level PID scheme, equation (3.8) is now modified to be

$$Q = K_P e + K_I \sum e\Delta t + K_D \frac{\Delta e}{\Delta t} \tag{3.9}$$

where Δt = the sampling or loop rate

$\sum e\Delta t$ is approximately the integral of the error

$\frac{\Delta e}{\Delta t}$ is approximately the derivative of the error

In this case, the sampling or loop period must be much smaller than the basic period of submersible motion. Otherwise, severe overshooting could be developed. If the submersible was controlled remotely by a computer onboard a ship, the time taken for the signal to travel between the submersible and the ship could cause overshooting. This is because the submersible would be responding to past error, not present error.

By applying proportional alone to the submersible, its control signal would cause the propellers to generate forces/moments in such a way that the submersible moves toward its desired position. The amount of forces/moments would be proportional to depth error. When the submersible reaches its desired position, the control signal would be zero and no thrust would be produced by the submersible's thruster units. However, large gain would generate a very large control signal when the actual position is far away from the desired position. In some case, this can burn out equipments under control. To prevent this, a limit is usually put on the magnitude of the control signal. Even though the proportional controller permits good stability criterion, its disadvantage is that it gives an offset to the response when there is a disturbance from external sources [14].

The integral controller is the controller that has an ability to reject disturbances. It gives zero offset. However, it cannot be used alone because it often overshoots the desired response. This can create an oscillatory response and lead to instability. By its nature, the integral controller would cause the propellers to spin in such a way that the submersible would move toward the desired position. As the submersible moves closer to the desired position, the control signal generated by integral would get bigger and bigger. As a result, the propellers would spin faster and faster causing the submersible to overshoot the desired position. One other thing about the integral controller is what is known as integral windup. Integral windup happens when the submersible is held well away from its command position: integration of error causes a very large control signal to be generated. Thus the integral controller should be activated only within a limited band surrounding the desired position to avoid integral windup [14].

The derivative controller, by its nature, can predict the error and produce corrective action before the error becomes too large. It can be used to help motions settle down. However, the derivative controller cannot be used alone as it responds to the rate of change of error, not the error itself. Consequently, it must be used in combination with proportional, integral or both [14].

A combination of all three components of PID controller allows the best characteristic of each controller to be employed. Therefore, the submersible would get to the desired position faster with minimal overshooting [14].

The goal of this research is to pick a controller that can make the Submersible Escape Capsule hold itself, with minimum error, to the command path. In order to accomplish this goal, the combination of these two types of error-driven control is selected as control strategies for the Submersible Escape Capsule. The error signals are separately computed for vertical and horizontal planes. In this case, distance from the water surface, or heave distance, is the response measured in vertical plane; and the angle clockwise (positive) or counter-clockwise (negative) from the north direction, or yaw angle, is the response measured in horizontal plane. In order to hold the submersible at its command path, the autopilot is programmed in such a way that the submersible operating range is divided into three stages. The outside band is used to protect the integral windup, which may caused by the integral controller. Outside this band, the switching control is selected to control the submersible motion. The propellers of the submersible are programmed to rotate at a constant speed. The direction of rotation would depend on the sign of the error. In between the outside and the inside band, the PID controller is selected to control the submersible motion. Thus, inside this range, the submersible will smoothly make its way to the desired position. Once it get to the inside band, in which can also be called the deadband, the propulsion device would be turned off. By doing so, drag load would cause the submersible to slow down and drift itself to the desired position. The Submersible Escape Capsule is designed to be neutrally buoyant. Thus, it can maintain itself at the desired position as long as there is no external force action on it. Once there is the external force act on the submersible and the submersible is forced to be outside the deadband, the PID controller would be turned on and would control the submersible to go back to its desired position.

3.4 Summary

The design of the Submersible Escape Capsule is based on Hydromechanical principles. These principles can be described as Hydrostatic and Hydrodynamics. They give us the equations governing the motion and stability of the capsule. There are many different types of control strategies that could be used for motion control of a submersible. The control strategy selected for the Submersible Escape Capsule is called error-driven control or sometime called feedback control. There are two types of error-driven control: Switching control and PID control. The combination of these two types of error driven controls is used in the Submersible Escape Capsule.

Chapter 4

Capsule Simulation

4.1 Preamble

In order to assist the design of a model capsule for the Submersible Escape Capsule, a 3D motion simulation was developed. The purpose of this simulation is to analyze the motion of the model capsule when it is traveling under water. The governing equations of motion of the model capsule were used to create this simulation and design controllers.

4.2 Governing Equations of Motion and Parameters Determination

As stated in Chapter 3, motion of a submersible can be characterized in terms of degrees of freedom [9]. Normally, a neutral-buoyancy submersible with 3D mobility is free to move in translation in x, y, z directions and in rotation about these axes. Thus, it processes a total of six degrees of freedom. Consequently, six equations of motion are required for an analytical procedure [9].

In the case of the Submersible Escape Capsule, motion caused by forces/moments generated by the submersible's thrust producing devices is of major interest. Thrust producing devices used in this submersible consist of two variable speed reversible propellers, which are used for controlling surge and yaw motions; and one variable speed reversible propeller, which is used for controlling pitch and heave motions. Therefore, the submersible will process four degrees of freedom. Thus, four equations of motion are required to generate the path of the submersible's body axis origin with reference to the earth axes origin. These four equations of motion are for: (1) Surge, (2) Heave, (3) Pitch, and (4) Yaw. Note that the submersible will be commanded to descend to a level in which water motions are insignificant and will maintain itself at this level by its neutral buoyancy characteristic. As a result, the effect of wind and waves can be considered very small and thus be neglected.

These equations contain many parameters; most of which can only be obtained from sophisticated experiments. Usually, there is inertia coupling of degrees of freedom. There can also be coupling due to hydrodynamics loads such as: pressure and inertia loads, wall drag loads and wake drag loads. The Submersible Escape Capsule design is such that there is very little coupling. Consequently, the governing equation for each motion can simplify to Newton's Second Law as shown in equation (3.6) and (3.7) for linear acceleration and angular acceleration, respectively [9]. This greatly simplifies dynamics and ensures that the capsule can be designed to be statically and dynamically stable. As described by Hinchey [14], the forces/moments acting on a submersible can be derived from

$$I\frac{d^2\varphi}{dt^2} = L_C + L_D + L_H + L_S \qquad \textbf{(4.1)}$$

where I = masses/mass moments of inertia of the submersible

φ = linear or angular displacement of the submersible

L_C = control load from the submersible's thrust-producing devices

L_D = disturbance load

L_H = hydrodynamic load

$L_S =$ hydrostatic load

The details on these parameters are discussed below:

4.2.1 Inertia (I)

In the equations of motion for surge and heave, I is just the mass of the submersible while in the equations of motion for yaw and pitch I is a mass moment of inertia. The mass moment of inertia can be calculated separately for yaw and pitch using the following equation:

$$I = \sum MR^2 \tag{4.3}$$

where $M =$ mass of each component

$R =$ distance of each component from axis of rotation

Three sources contribute to the mass and mass moment of inertia. These three sources are: (1) the submersible masses, (2) the water entrained in the submersible's envelope and its appendages, and (3) the water adjacent to the submersible which is accelerated positively or negatively by its motion [9]. Masses and mass moments of inertia from source (1) and (2) are constant for each particular submersible. Note that masses and mass moments of inertia from source (2) are not included in the equations of motion used in this study since there is no water entrained inside the submersible's envelope. The contribution from source (3) or what is called added mass or added mass moment of inertia can vary with type of motion and submersible's envelope. The reason for this is because different quantities of water move in translation with the submersible as it moves along x, y, z axes and in angular motion about these axes [9]. These parameters are only roughly estimated in this research.

4.2.2 Control Load from Thrust-producing Devices (L_C)

A first order model is used to determine the force from thrust producing units of the model capsule [14]. This equation is shown below

$$A\frac{dL_C}{dt} + BL_C = Q_V \tag{4.3}$$

where A = dynamic coefficient

B = static coefficient

L_C = control load from the submersible's thrust-producing devices

Q = control signal

The Proportional Integral Derivative (PID) control strategy is used to generate control signal, Q. A and B are normally determined by experiment. In the case of the Submersible Escape Capsule, these two parameters are approximated since a facility for finding them exactly was not available.

4.2.3 Disturbance Load (L_D)

Disturbance loads (L_D) could be due to wave and winds. These sources normally act on the submersible only near the water surface. Thus, they do not play a major role in this research since the Submersible Escape Capsule will spend most of its time underwater at the level where these forces can be considered insignificant.

4.2.4 Hydrodynamic Load - Lift and Drag (L_H)

Lift and Drag loads (L_H) are created when there is relative motion between a submersible's body, its appendages and the surrounding water particles [9]. Lift force is normal to the direction of fluid flow and usually exists when there is a non-zero angle of attack between the submersible's body axis and the direction of fluid flow. It usually occurs when a body has asymmetry with respect to an axis or when a symmetric body is inclined to the flow direction or both. Lift forces are normally applied to control the submersible motion when it is traveling at high forward speed [9]. Drag force or resistance (L_H) is defined as the force that opposes forward motion through the fluid and is parallel to the direction of fluid flow. It exists when there is either zero or nears-zero angle of attack [9]. There are two different types of drag force: (1) *Form Drag* caused by the fluid that flowing over the body. Separation from the body creates turbulence and results in pockets of low and high pressure that leaves a wake behind the body, (2) *Skin Friction Drag or Wall Drag* caused by the actual contact of the fluid particles against the surface of the body [15].

For moving structure likes a submersible, the relative velocities and accelerations must be used to get forces. Morrison's Equation gives these forces as:

$$L_H = \frac{1}{2} C_D \rho A U \mid U \mid + C_M \rho V \dot{U} \qquad \textbf{(4.4)}$$

where $L_H =$ hydrodynamic load

$C_D =$ drag coefficient of the body

ρ = fluid density

A = projected area of the underwater portion of the body

U = relative velocity of the body

C_M = inertia coefficient of the body

V = volume of the body

$\dot{U}$ = relative acceleration of the body

C_D and C_M can be determined experimentally. Details can be found in Appendix B.

4.2.5 Hydrostatic Load - Buoyancy and Weight Load (L_S)

When there is static inclination of a submersible away from its equilibrium position, the force of weight (W) and buoyancy (Δ) will then create moments to resist inclinations. These moments will try to move the submersible back to its original equilibrium position. This restoring couple is called ***positive righting moment*** or here called *metacentric stability*. This positive righting moment or metacentric stability can be calculated by multiplying weight (W) or buoyancy (Δ) of the submersible with the normal distance between the parallel lines-of-action of weight (W) and buoyancy (Δ) forces [9].

4.3 A 3D Motion Simulation

An overall block diagram for the capsule is shown in Figure 4.1. It describes the overall operation of the capsule. A simple simulation was developed to avoid using many parameters that would be extremely difficult or impossible to calculate. Newton's Second law is used to derive equation for each degree of freedom. As shown in section 4.2, a first order model is used for each thrust producing unit. PID is selected as the control strategy.

A saturation block is added to each sub-block for motors. Its upper and lower limit represents the maximum and minimum speed of the thruster, respectively. This is to protect the thrusters from burn out and to provide enough power to the thrusters in order to make them spin underwater. The effects of Switching control and the deadband are not accounted in this simulation. Zero-Order Hold is used in this simulation since the command signal is generated within a digital control loop. The sampling time is set to 0.2 since the model capsule is programmed to generate its command signal at approximately every 0.2 seconds. Transport delay is also added to each sub-block for motors. This is to account for the communication delay between the controller and the actuators: this would be important only if the submersible was controlled by a computer on board a host ship. Specified depth and yaw angle are set before the capsule starts its mission. The sub block labeled VERTICAL_CONTROLLER takes in the specified depth and compares it with the actual depth. Output from this sub block is a command force from the back motor. This signal then goes into the HEAVE sub block to give heave distance. This signal also goes to the PITCH sub block to give pitch angle. The vertical component of the forward motion of the capsule is then calculated and integrated to give the actual heave or z location of the capsule. Sub blocks named LEFT_MOTOR and RIGHT_MOTOR takes in an error signal between the specified yaw angle and the actual yaw angle of the capsule. The two front motors are controlled by these sub blocks and the comparators are used to determine which motor will be working during the operation. Output from the comparators then goes into the YAW sub block to give yaw angle. It is also sent to the SURGE sub block to give surge velocity. The horizontal component of the forward motion on the XY-plane is then calculated. This motion is then divided into x and y components. Integration then gives the capsule location on the XY-plane. The WORKSPACE sub block sends the XYZ coordinates out to MATLAB where the location of the capsule on 3D plane are plotted using the "*plot3*" function.

4.3.1 Heave and Pitch Block Diagram

Heave along the z-direction and pitch about the y-axis are generated by a variable speed reversible motor located at the backside of the model capsule. While operating, this motor will generate force either in upward or downward direction. The direction of this force depends on the turning direction of the propeller. This force then moves the capsule along the z-direction. Since this motor is located far to the back of the capsule, torque due to moment arm from the center of gravity of the capsule and the motor itself will be created. This results in pitching moment about the y-axis. Figure 4.4 shows details of the HEAVE sub block diagram. Figure 4.5 shows detail of the PITCH sub block diagram. Experiments were done to get drag constant for heave. Other parameters and drag constant for pitch were determined by approximation. Details of this can be found in Appendix B.

4.3.2 Yaw and Surge Block Diagram

Surge along the x-direction and yaw about z-axis are generated by two variable speed reversible motors. These two motors are installed in such a way that they can counteract the effect of each other. As shown in Figure 4.3(a) and 4.3(b), comparators are used to determine which motor is turned on during operation. An error signal is calculated from the desired yaw angle minus the actual yaw angle. This signal is then fed into the comparator. This error signal is then compared with zero. If it is less than zero, the comparator will pass a control signal through the controller and to the drive for the right motor. This will turn the right motor on. The capsule will be turned counter clock wise toward the specified direction. If the error signal is greater than zero, the comparator will pass a control signal through the controller and to the drive for the left motor. This will turn the left motor on. The capsule will be turned clock wise toward the specified direction. As soon as the error is equal to zero, both comparators will pass a control signal to both motors. This will turn both motors on. Torques will be created from both motors to counteract each other; and forces will be generated to move the capsule forward along the desired direction. Figure 4.6 shows details of the YAW sub block diagram. Figure 4.7 shows detail of the SURGE sub block diagram. Experiments were done to get drag constant for surge and yaw. Other parameters were determined by approximation. Details of this can be found in Appendix B.

4.3.3 Z-Component and XY-Component Block Diagram

Figure 4.8 shows the z-component sub block. It takes in heave velocity and pitch angle created by the back motor. Integration of this product gives the capsule location caused by the back motor. It also takes in location of the capsule that resulted from surge and pitch. These signals are later combined together. Its output is used to obtain the vertical motion of the capsule along z-direction.

Figure 4.9 shows the xy-component sub block. It takes in surge velocity and pitch angle. These are used to obtain the horizontal motion of the capsule. Integration of this product gives the location of the submersible in the xy-plane. Yaw angle is then used to break up this signal into x and y components.

4.3.4 Simulation Results

Some simulation results are shown in Figure 4.10 and Figure 4.11. The 3D simulation result shown in Figure 4.10 strongly confirms that the capsule is capable of moving along a 3D trajectory. Figure 4.11 shows the simulation results obtained when the heave motion and yaw motion are considered separately. In the figures, the oscillation shown during the capsule is ascending to the surface in Figure 4.10 can also be seen in Figure 4.11(a) for heave motion. In order to simulate the model capsule when its envelope is enclosed, the mass of the model capsule was increased 5 times its original mass. The results from this simulation show that the increase of mass does not have much effect on the trajectory path but it does slow the system don.

4.4 Summary

A 3D motion simulation and controller strategies for the model capsule were developed. The simulation block diagram consists of three main groups of sub-blocks. The first group consists of three sub-blocks for motors. Each sub-block represents a motor. The second group consists of sub-blocks for heave, pitch, yaw and surge. These sub-blocks are used to generate control forces for each motion. The last group consists of sub-blocks that use for generating position of the model capsule on a 3D-plane. The simulation showed that the capsule is capable of moving smoothly along 3D trajectories.

The simulation showed that the capsule is capable of moving smoothly along 3D trajectories.

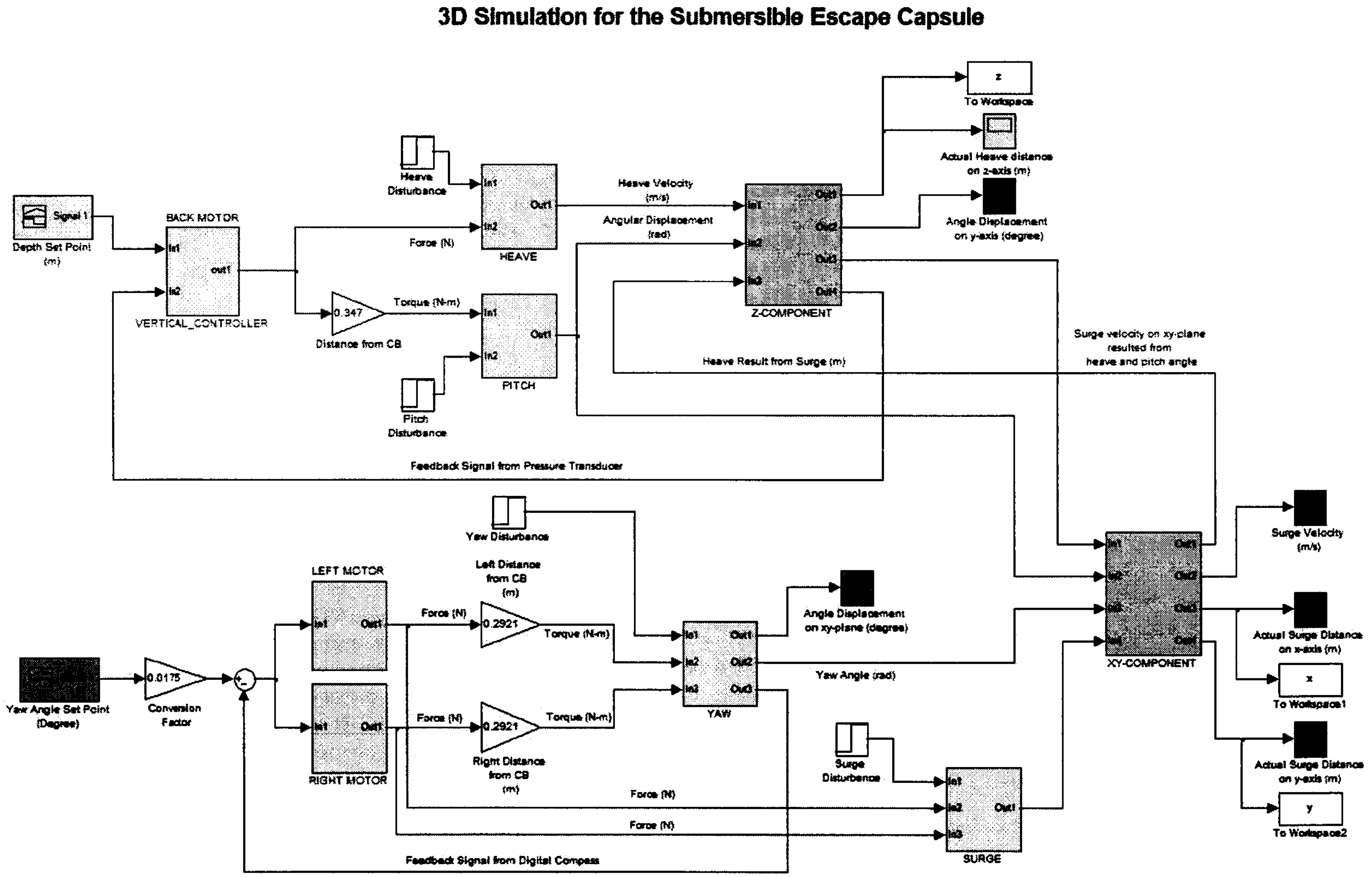


Figure 4.1 3D Motion Simulation for the Submersible Escape Capsule

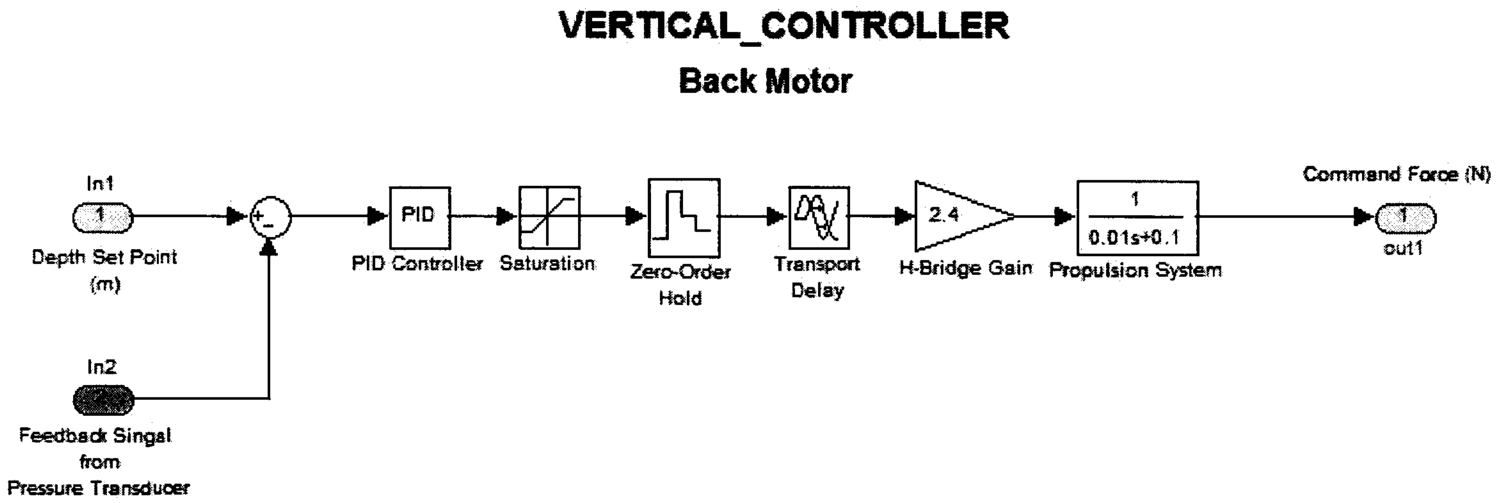
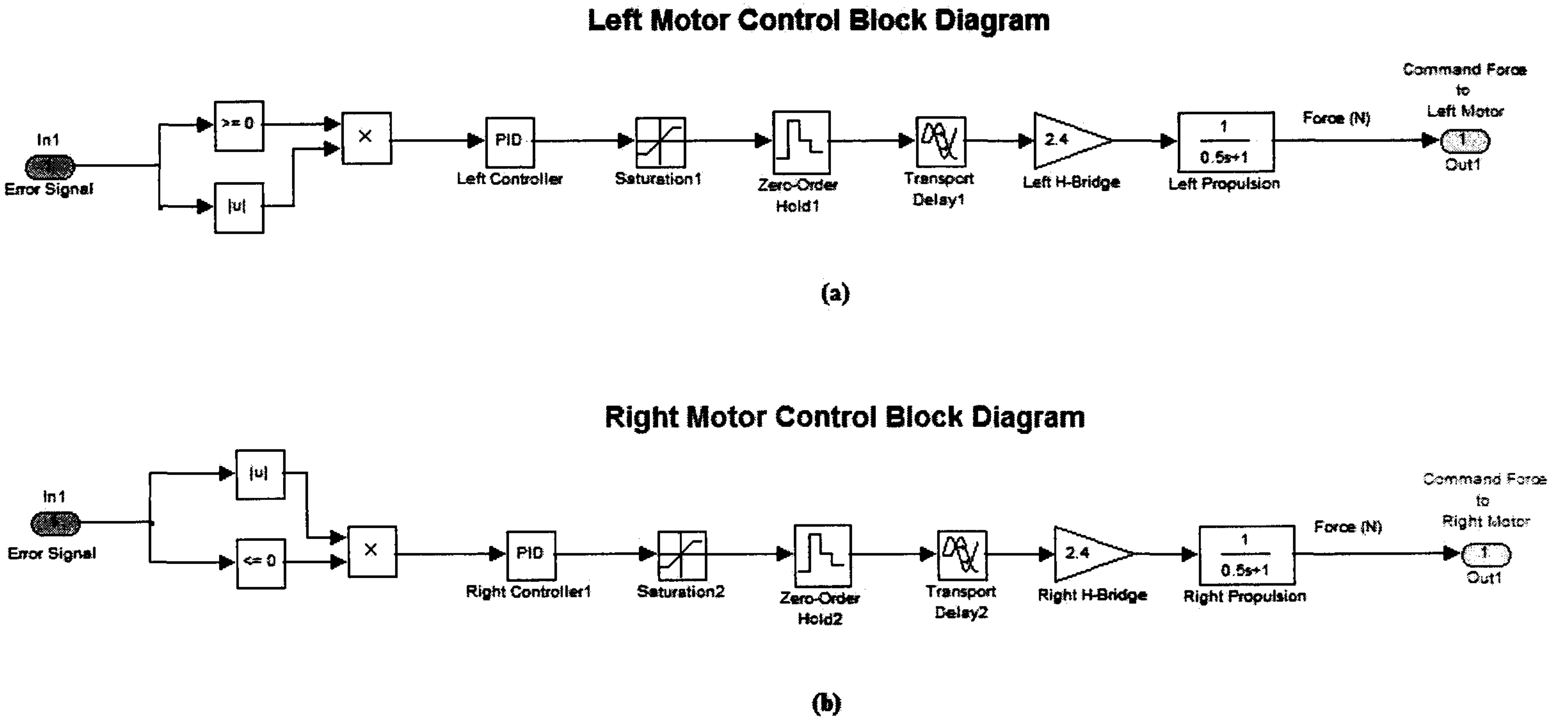


Figure 4.2 Vertical Motion Controller or Back Motor Block Diagram


Left Motor Control Block Diagram

(a)

Right Motor Control Block Diagram

(b)

Figure 4.3 Horizontal Controller or Side Motor Block Diagram (a) Left Motor (b) Right Motor

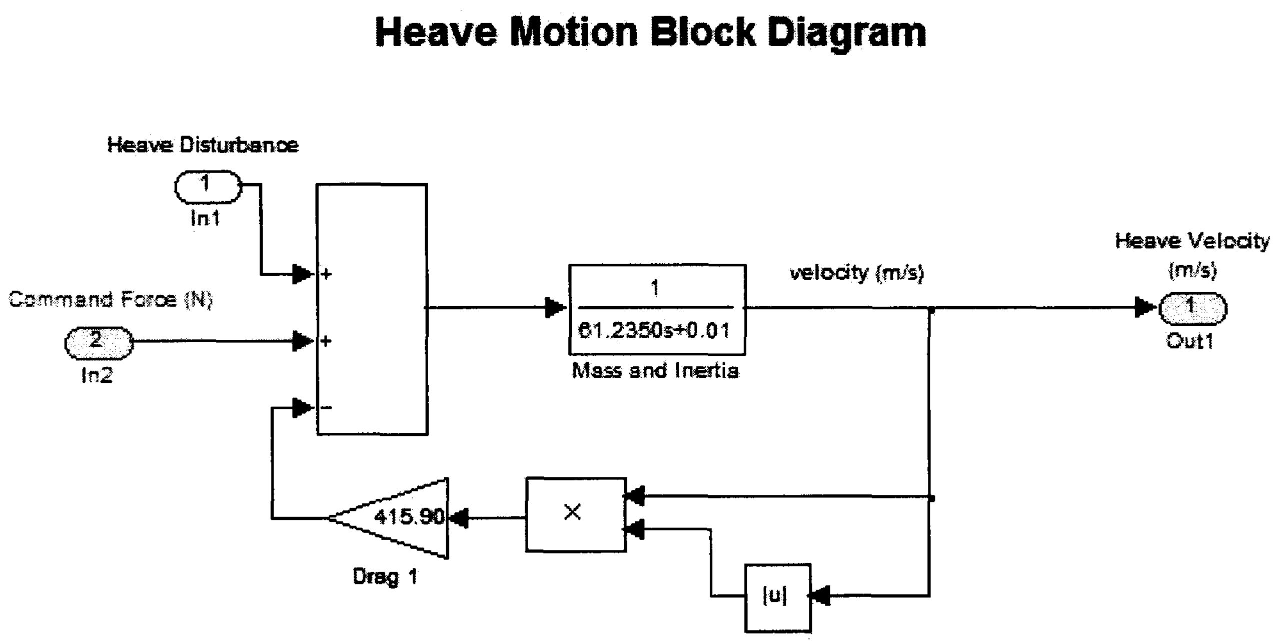


Figure 4.4 Heave Motion Block Diagram

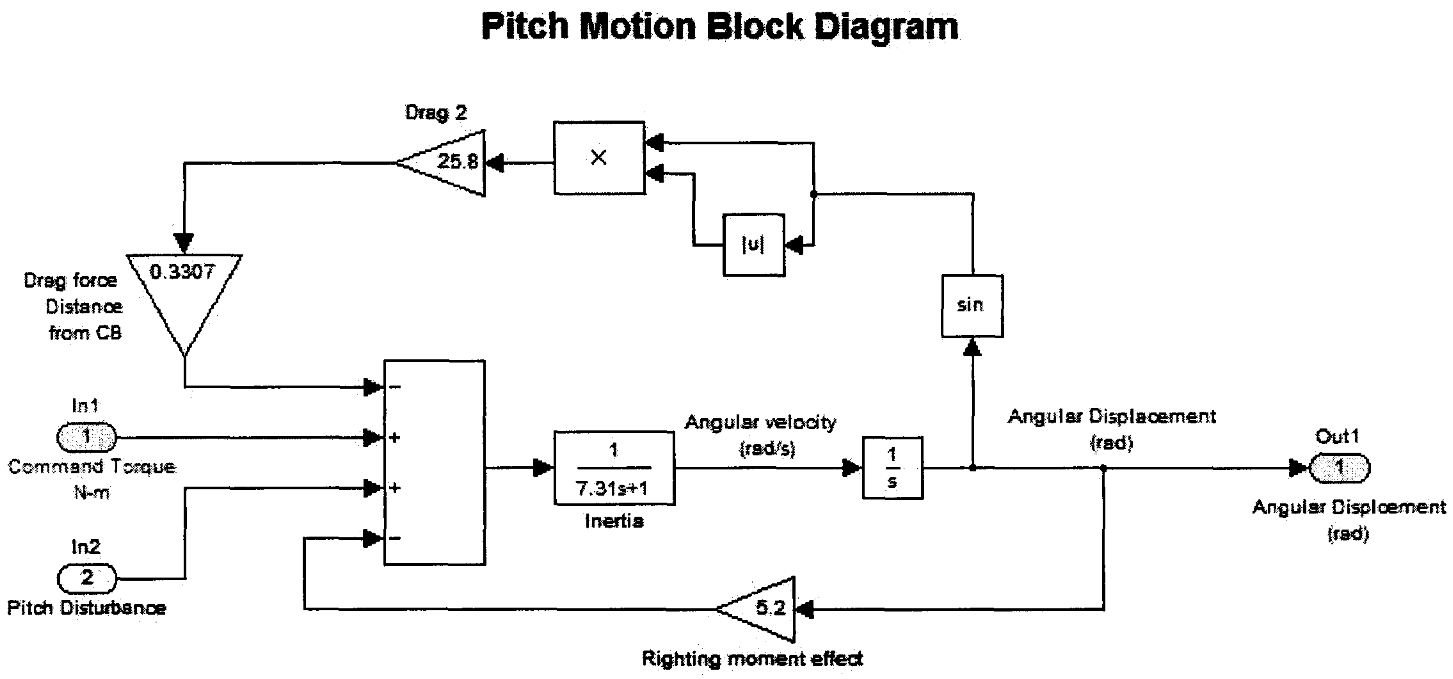


Figure 4.5 Pitch Motion Block Diagram

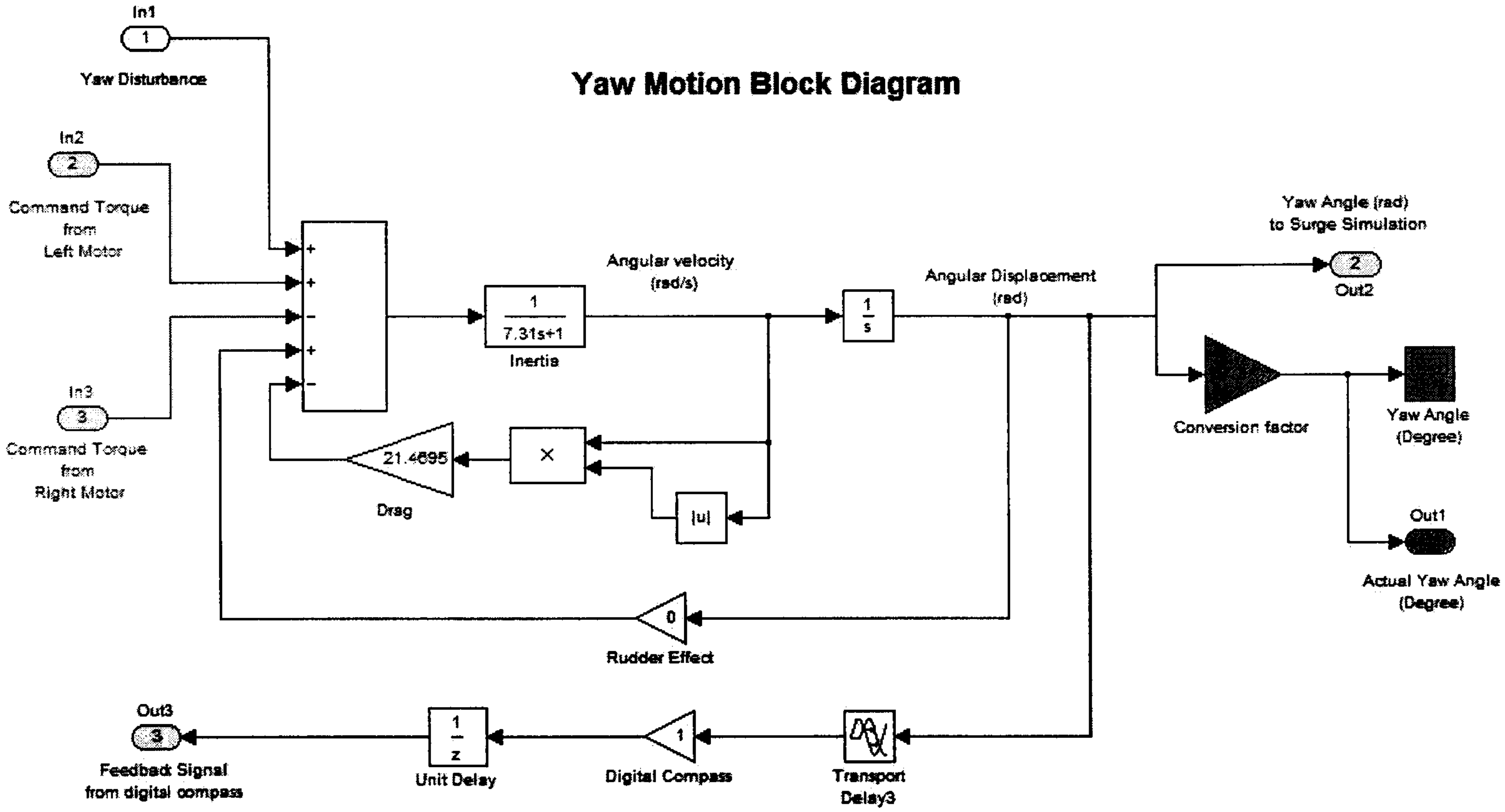


Figure 4.6 Yaw Motion Block Diagram

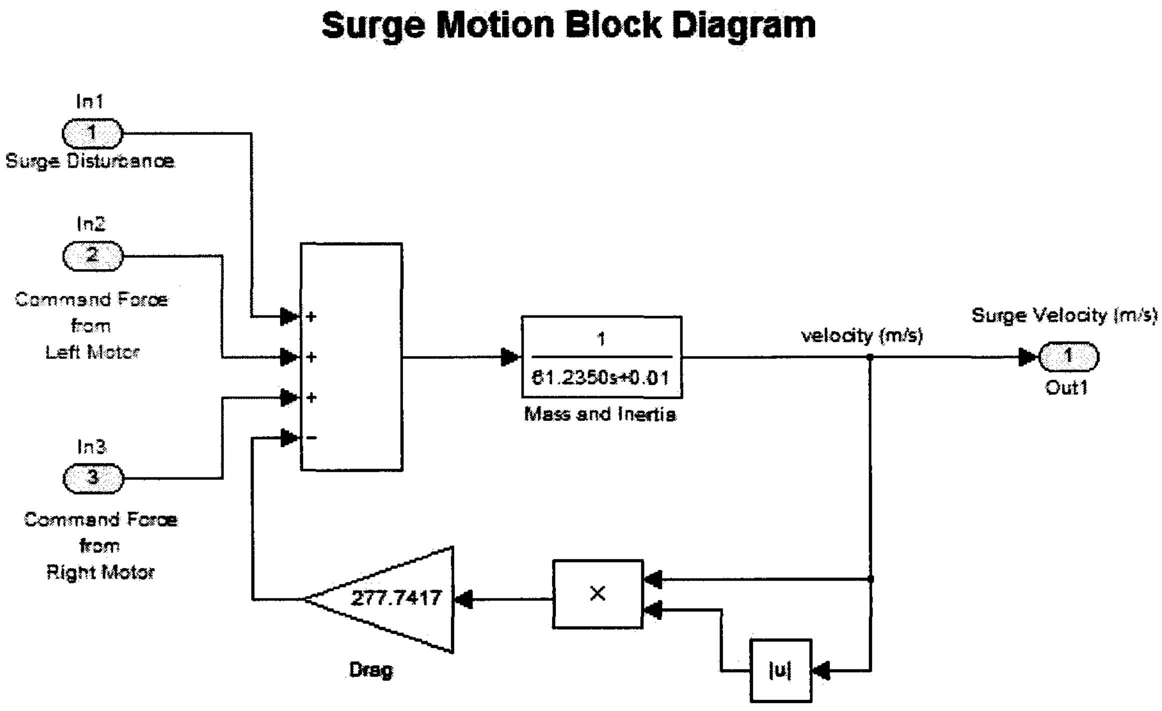


Figure 4.7 Surge Motion Block Diagram

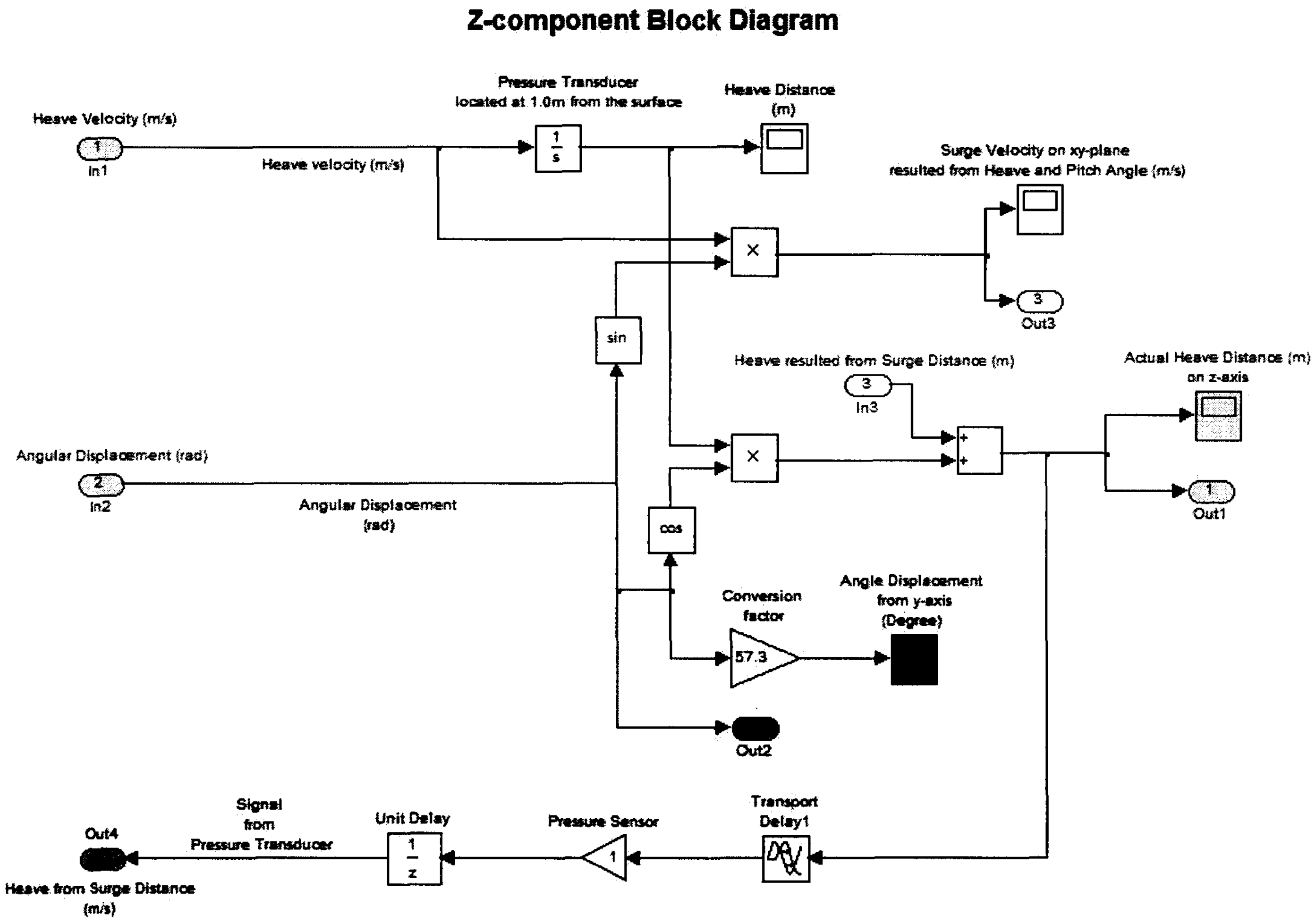


Figure 4.8 Z-Component Block Diagram

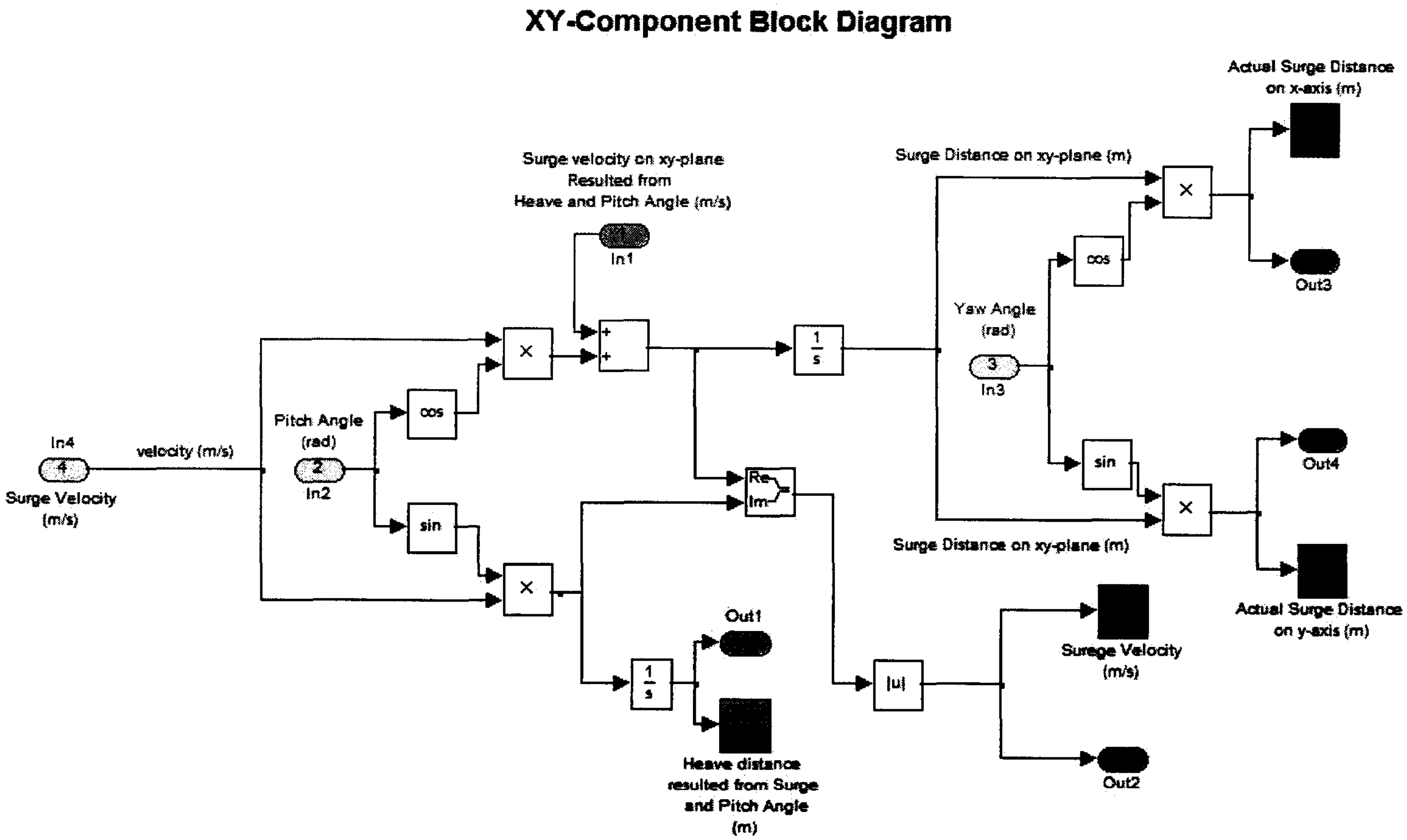

XY-Component Block Diagram
Actual Surge Distance on x-axis (m)
Surge velocity on xy-plane Resulted from Heave and Pitch Angle (m/s)
Surge Distance on xy-plane (m)
1
In1
×
+
+
1/s
cos
Out3
Yaw Angle (rad)
3
In3
×
cos
Pitch Angle (rad)
In4
4
velocity (m/s)
Surge Velocity (m/s)
2
In2
sin
Re
Im
sin
Out4
×
Surge Distance on xy-plane (m)
×
Actual Surge Distance on y-axis (m)
Out1
Surege Velocity (m/s)
1/s
|u|
Heave distance resulted from Surge and Pitch Angle (m)
Out2

Figure 4.9 XY-Component Block Diagram

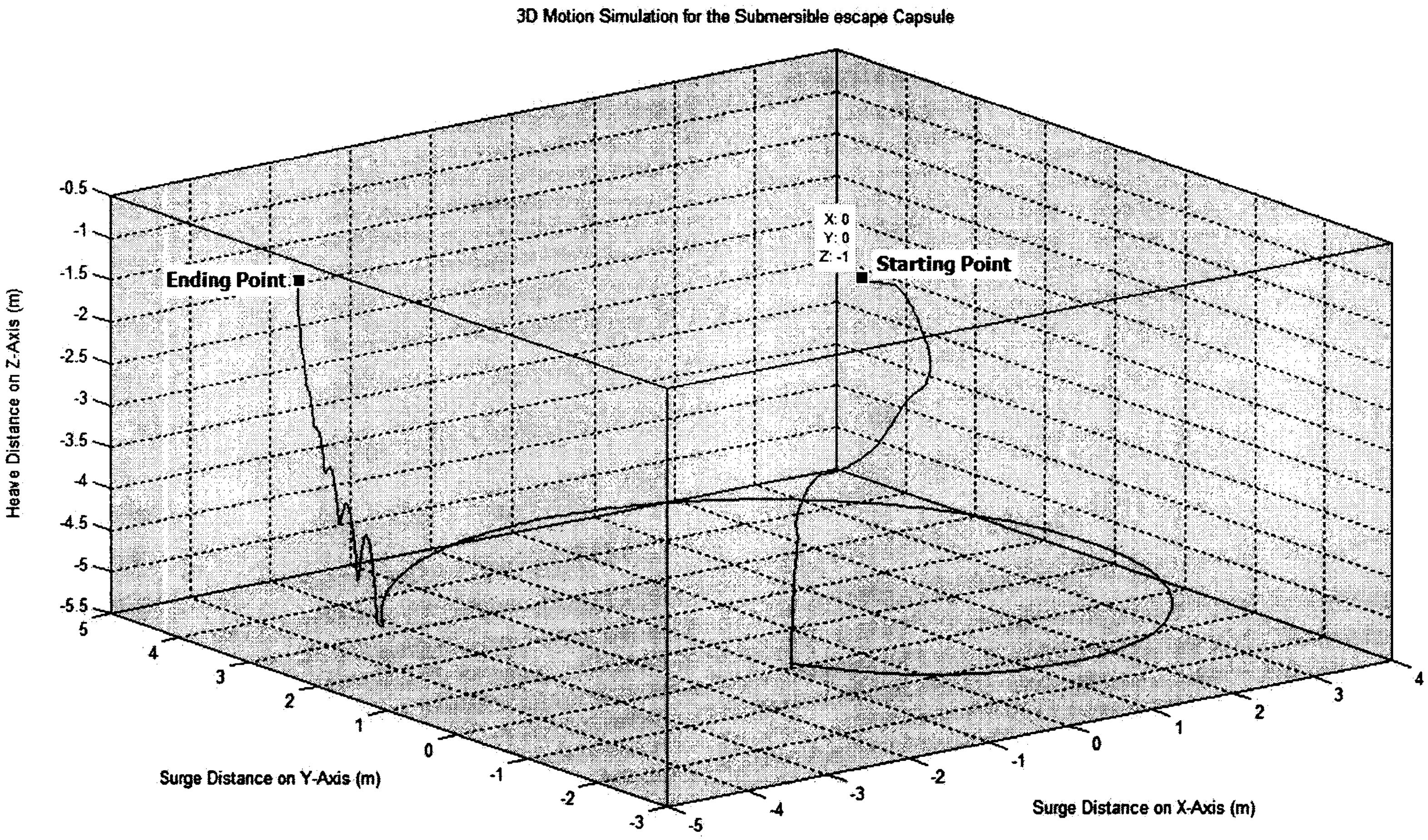
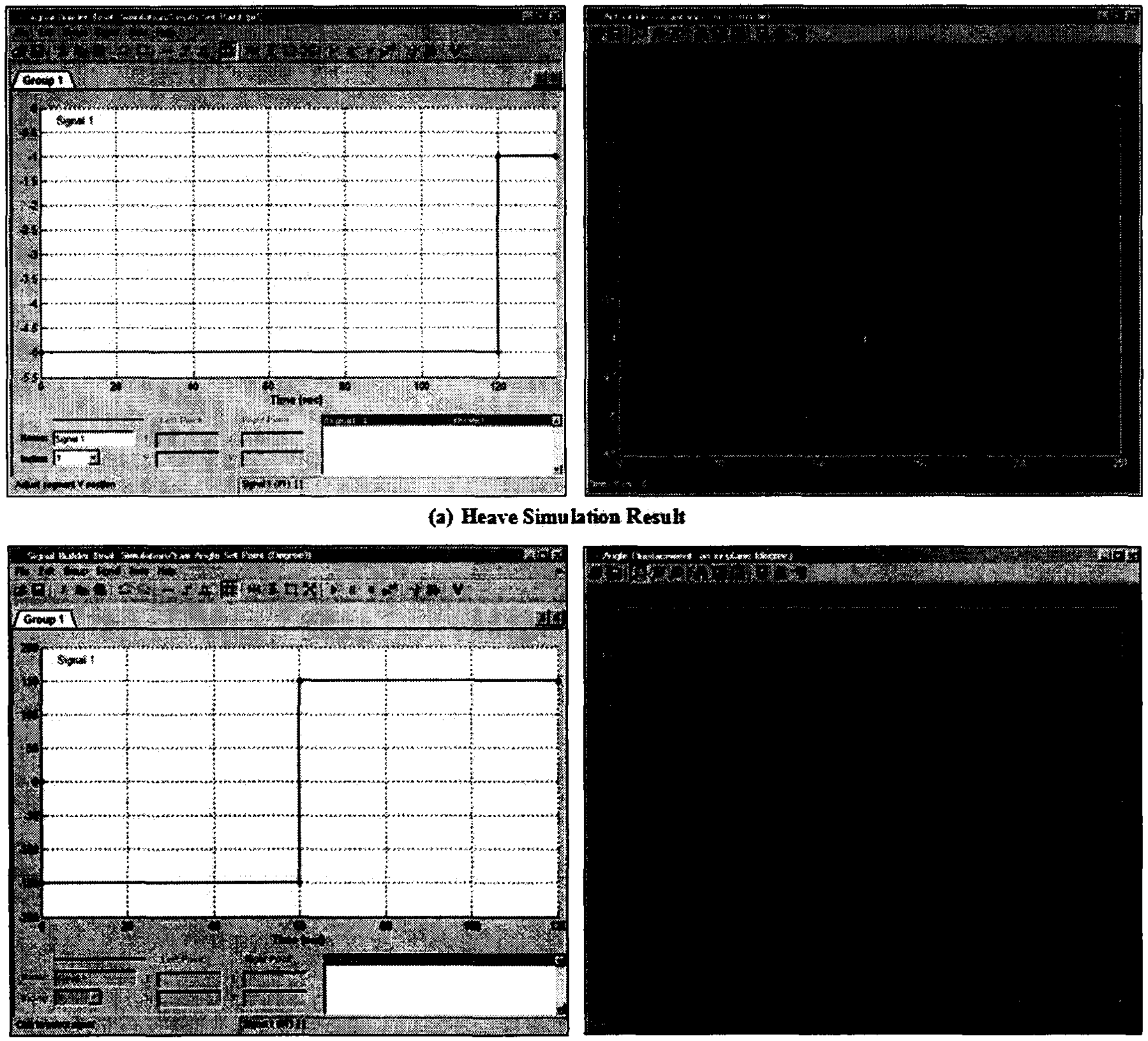


Figure 4.10 3D-Motion Simulation Result for the Submersible Escape Capsule



(a) Heave Simulation Result

(b) Yaw Simulation Result

Figure 4.11 Heave Distance and Yaw Angle Simulation Result

Chapter 5

The Submersible Escape Capsule

5.1 Introduction

In Chapter 4, a 3D motion simulation was developed to study control strategies and the influence of various submersible parameters on the trajectory path underwater of Submersible Escape Capsule. It gives results that strongly support the concept.

The Submersible Escape Capsule Model was designed, constructed and tested. It was designed by applying hydrostatic and hydrodynamic principles. It was designed to be very modular in construction so that parts could easily be replaced. As stated in Chapter 3, its design criteria must include the following:

1. Force of weight (W) equals to buoyancy (Δ) for the submersible to have neutral buoyancy.
2. Force of weight (W) and buoyancy (Δ) are on the same vertical line normal to submersible's xy-plane for the submersible to be in an upright position.
3. The center of gravity (C_G) is beneath the center of buoyancy (C_B) for the submersible to have stable equilibrium.

In addition, the shape of the model capsule must have low drag. The model must be highly maneuverable. Most importantly, it must be dynamically stable.

5.2 Main Components of the Model Capsule

The main components of the model capsule consist of its main structure, three high speed thrusters for propulsion and maneuvering, a water tight electronics control chamber, batteries for power during operations, and waterproof connectors and wires using for signal and power transmission. An overall size of the model capsule is approximately 1.5:1 scaled of the prototype capsule. It has a total mass of 41.38 kg or 91.23 lbs with an overall height of 1.22 meters or 4 feet. An overhead crane must be used for lifting, deployment and recovery. Figure 5.1 shows the complete model capsule. Specification sheets for various components are given in Appendix F.

5.2.1 Structure and Frame

The main structure of the model capsule is made from aluminum, 1 inch box section tube, with 1/8 inches thick wall. It consists of three major parts: main body of the capsule, thruster holders and crane hook holder. Some parts of this are welded together while other parts are bolted together. The overall dimensions of the main structure and detailed drawings are given in Appendix C. The two aluminum bars in the middle part of the model are used to hold a control chamber while the batteries are held on the bottom rail of the model.

5.2.2 Energy Storage System

The energy storage system on the model capsule consists of two separate sets of batteries. The first set is a small 12 VDC battery which is used as a source of power for the electronics control parts. This battery is housed within the control chamber and is shown in Figure 5.2(a). The other set consists of two 12 VDC batteries connected in parallel. This set of battery is used to provide power to the thrusters. They can be seen attached to the bottom rail of the capsule in Figure 5.1 and 5.2(b).

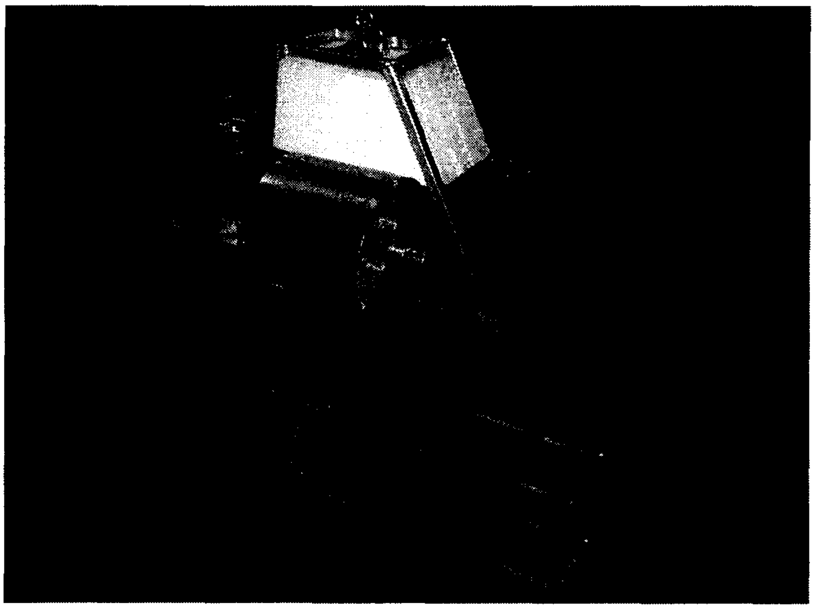
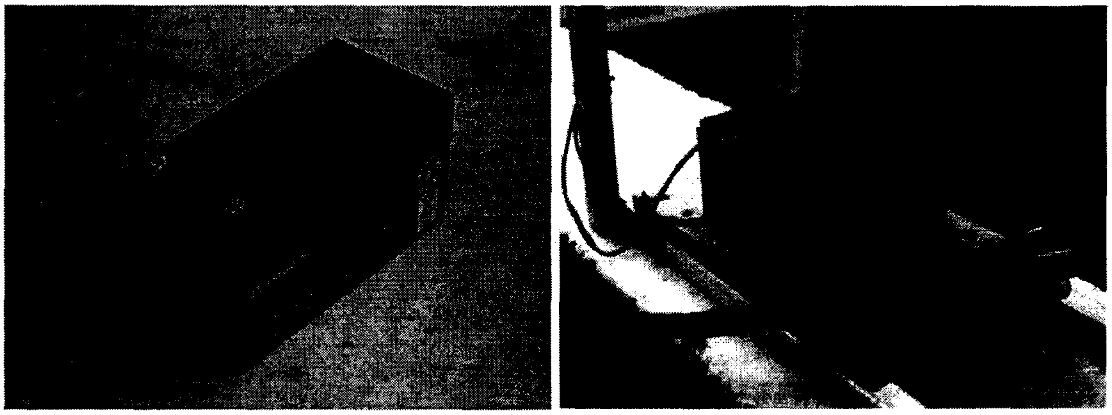


Figure 5.1 The complete model capsule



(a)

(b)

Figure 5.2 (a) A small 12 VDC battery for electronics board (b) Two 12VDC batteries connected in parallel. These are used to power thrust producing units of the model capsule.

5.2.3 Propulsion and Maneuvering System

The propulsion and maneuvering system used in the model capsule consists of three DC trolling motors. These motors are designed to move a sports fishing boat with several people onboard. They provide more than enough power for the capsule. Each can generate maximum thrust of 30 lbs-force. The speed of each motor is set by the DC voltage or PWM signal applied to it. The current for the motors is limited at 20Amps maximum. Each one of them is located inside a protective shroud and it is attached to a holder on the model structure using nuts and bolts. Figure 5.3 shows one of the thrust producing units with its shroud. Unfortunately, the specification sheet for this trolling motor is not available as it is no longer manufactured. Subconn waterproof connectors and wires are used to connect the motors to the control chamber.

5.2.4 Capsule Control System

The control system of the model capsule consists of the box that contains control circuits. The main function of this box is to provide signals for propulsion and maneuvering of the capsule. The box has an outside diameter of 32 millimeters or 12 ¾ inches and a height of 20.32 millimeters or 8 inches. It contains a small Peripheral Interface Controller or PIC circuit board, three 20Amps H-Bridge drives (Daventech Model MD03), a pressure transducer (Omega Model P40), a robot compass module (Daventech Model CMPS03), two 5V voltage regulators, and a 12VDC battery. One of the voltage regulators is generates 5VDC for the PIC and H-bridges. Another one provides 5VDC for the sensors. A photo of control circuit box is given in Figure 5.4. The

PIC circuit board can be seen in the center of the photo. The three H bridges can be seen on the left. The voltage regulators circuit and the compass can be seen on the right. The pressure sensor is mounted on the box cover below the control circuits.

Figure 5.3 One of the thrust producing unit with its shroud

A technique developed in earlier AUV work at MUN was used to make sure that the control box maintained a water tight seal during operation. After all components were installed in the box and the covers were put in place, suction was used to hold the covers tightly to the box. To help maintain a seal, an o-ring covered in lubrication grease was placed in a groove on each side of the box (one on the top and one on the bottom). Air was then removed from the box through a small hole in one of the covers. A vortex blower in reverse operation was used to generate the suction. Duct tape was used to cover the hole after sufficient suction had been generated. Two aluminum hooks were attached to the box to lock the covers in place and hold the box to the capsule.

The Microchip PIC16F873 microcontroller was used as the main controller and its pin layout is shown in Figure 5.5. The controller runs at an operating speed of 4 MHz.

It has 4 input/output ports. Port A of the controller can be used as analog input channels. One of these channels is used to receive analog signal from the pressure transducer (Omega Model P40). The controller also has the Master Synchronous Serial Port or MSSP module which can be used as a serial interface for connecting with other peripheral or microcontroller devices. This serial port module can operate either in Serial Peripheral Interface (SPI) mode or Inter-Integrated-Circuit (I^2C) mode. For the model capsule, the I^2C mode is used for communication between the PIC and the H-Bridges and the robot compass module. A small circuit board containing a MAXX RS-232 chip and a USB Serial Converter was used to download programs to the PIC. This circuit board is shown in Figure 5.6. The circuit diagram for the main circuit board of the capsule can be found in Appendix D.

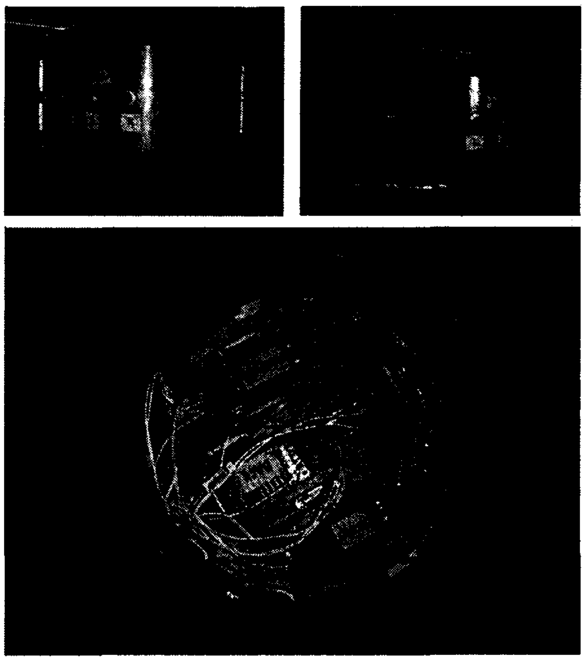


Figure 5.4 The control circuit box and its components

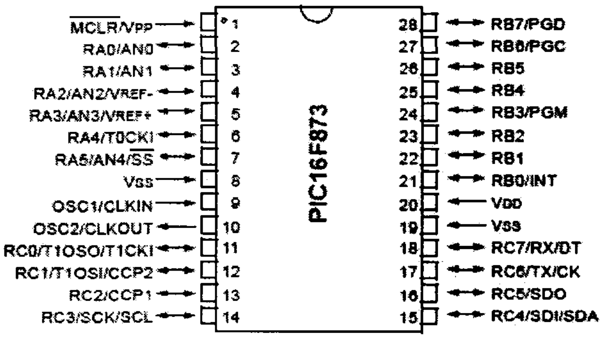


Figure 5.5 PIC16F873 pin diagram

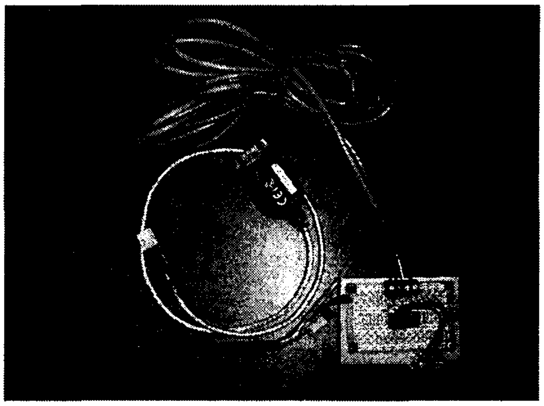


Figure 5.6 MAXX RS-232 and USB Serial Converter

The 20Amps H-Bridge motor drives (Daventech Model MD03) were chosen for the model because they are compatible power wise with thrusters. The H-Bridges themselves require 5VDC for logic supply and they can supply 5V to 50V to each thruster. An advantage of these H-Bridge Modules is that they can be controlled by any of the following options: (1) I^2C bus, up to 8 modules, switch selectable address, (2) 0V-2.5V-5V analog input, 0V for full reverse, 2.5V center stop and 5V for full forward, (3) 0V to 5V analog input with separate direction control, (4) RC mode, controlled directly from the RC receiver output, and (5) PWM which is converted by a simple onboard filter to analog. The first method was selected since three MD03 modules must be controlled at the same time and the communication between PIC and MD03 via the I^2C serial interface is quite straightforward. The speed, direction and acceleration rate of each drive can be easily controlled with a few lines of instruction code.

Two sensors were used in the model capsule to help control depth and aid navigation. A 15 psi pressure transducer (Omega Model P40) was used to sense depth. It has a calibration factor of 0.4 V per meter depth. This is sufficient for the deep tank test. A robot compass module (Davantech Model CMPS03) was chosen as an aid to navigation. It detects the magnetic field of the Earth and produces a unique number to represent the direction the capsule is facing. The compass module requires a 5VDC power supply at a nominal 15 mA. It was calibrated by the manufacturer for latitude 67°N which is fairly close to the test location (70°N). The bearing of the capsule can be obtained as a PWM signal or via an I^2C interface. In the model capsule, the I^2C interface was used with a resolution of about 0.1 degrees. The output was read in 16-bits.

5.3 Center of Gravity (C_G) and Center of Buoyancy (C_B) Calculation

In order to include the design condition criterions to the model capsule, its weight (W) and buoyancy (Δ) as well as the location of the center of gravity (C_G) and the center of buoyancy (C_B) must be carefully computed.

Once all components had been constructed, the weight of each component was measured. The volumetric displacement of each component was roughly estimated from their dimension using "*massprop*" command in AutoCAD program. The total weight and buoyancy force were then computed. According to the calculation, the capsule has insufficient buoyancy. A certain volume of Styrofoam had to be added to the model to increase its buoyancy. The shape of this Styrofoam was determined by where it had to be positioned on the capsule. Moment calculations were then used to position the batteries and motors to get zero trim. The details of this work are given in Appendix A.

5.4 Summary

The Submersible Escape Capsule Model was designed and constructed. The major components of the capsule consist of (1) Structure and Frame, (2) Energy Storage System, (3) Propulsion and Maneuvering System, and (4) Capsule Control System. All components of the capsule were designed to be very modular in construction. As a result, all parts could easily be removed or replaced if any maintenance would be required.

Moreover, the model capsule weight (W) and buoyancy (Δ) as well as the location of the center of gravity (C_G) and the center of buoyancy (C_B) were carefully computed since they determine the stability of the capsule.

Chapter 6

Experimentations

6.1 Preamble

The SIMULINK simulation was first used to help design the model capsule control system. Once the model capsule had been constructed, calculations and experiments were used to get more accurate estimates of the parameters in the model, such as inertias and drag factors. These works are described in Appendix B.

The experiment for the model capsule was divided into 4 main steps. The first step was to ensure that the model capsule was neutrally buoyant, and the capsule was capable of maintaining its water tight seal. In the next step, the ability to perform certain tasks was tested. The mission assigned to the model capsule can be viewed as two different missions: To hover at the control depth and to travel along the specified direction. Two separate experimentations were then set up for each mission. These experiments were completed in the Deep Water Tank at Memorial University of Newfoundland. Finally, the last step was to combine the two missions together and this test took place in the Offshore Engineering Basin (OEB) at the Institute for Ocean Technologies (IOT), Newfoundland and Labrador. The details of each test are given in the following sections.

6.2 Neutral buoyancy and water tight check

In order to adjust the capsule buoyancy, all components were attached to the main structure of the capsule. Once the capsule was ready, it was lifted by an overhead crane and deployed into the Deep Water Tank at MUN. The hook was then released and the model capsule floated by its own buoyancy in the tank. This test demonstrated that the model capsule was not quite neutrally buoyant and it had insufficient weight. The reason for this was the volumetric displacement of the main frame and the propellers were not accounted for in the calculation shown in Appendix A. Some lead weight was then added to the model to make it neutrally buoyant. These procedures took about 30 minutes to complete. After the model capsule was adjusted to be neutrally buoyant, the lift crane was again used to recover the model capsule from the tank. The control chamber was then disassembled from the main structure and was opened to check for leaks. No sign of leakage was found in the box. This confirmed that the suction created in the control chamber using the vortex blower was adequate to prevent leaks.

6.3 Depth Control test

The depth control test was performed to ensure that the model capsule was capable of hovering itself at a specified depth. This test was also carried out in the Deep Water Tank at MUN, which has the maximum depth of 3.5 meters. The code was first loaded into the controller. A small plastic tube was connected to the pressure transducer (Omega Model P40) and the model capsule was then turned on outside the tank. By blowing into the small plastic tube, we could simulate the capsule heave control. This was to ensure that motors and electronic board were operating. During the test, the speed and the turning direction of the back propeller was changed by blowing air into this tube. All systems were operating effectively. In the next step, all components were installed back to the main structure and the model capsule was prepared for testing in the tank.

Before the test was performed, an additional 4 wires were connected to the Subconn connector. This connector was then connected to the control chamber. These 4 wires were used to transfer power to the RS232 board as well as for communication purpose between PIC and the host computer (PC) via a serial port connector.

The objective of this test was to get the model to descend to 1.2 meters below the water surface and keep itself at that level for approximately 40 seconds and then ascend to the surface. The PIC was programmed so that it read depth signals from the pressure transducer (Omega Model P40) and sent the data back to the computer every 0.2 seconds. The HyperTerminal program that comes with Windows Operating System was used to establish the connection between PIC and PC. Several tests were required to obtain satisfactory hovering performances and debug program errors. After several changes of the code, the mission was quite successful. Data taken by sending back the signal from PIC for this test is presented in Figure 6.1. An oscilloscope was also used to record data by connecting two extra wires to the pressure transducer and its result is shown in Figure 6.2. The model capsule started its operation at rest at the water surface and then descended to the specified depth. It then tried to hover itself at that level for approximately 40 seconds before ascending back to the surface and floating. Results shown in Figure 6.1 and Figure 6.2 confirm that the model capsule is capable of keeping itself at the specified depth. The computer code is given in Appendix E.

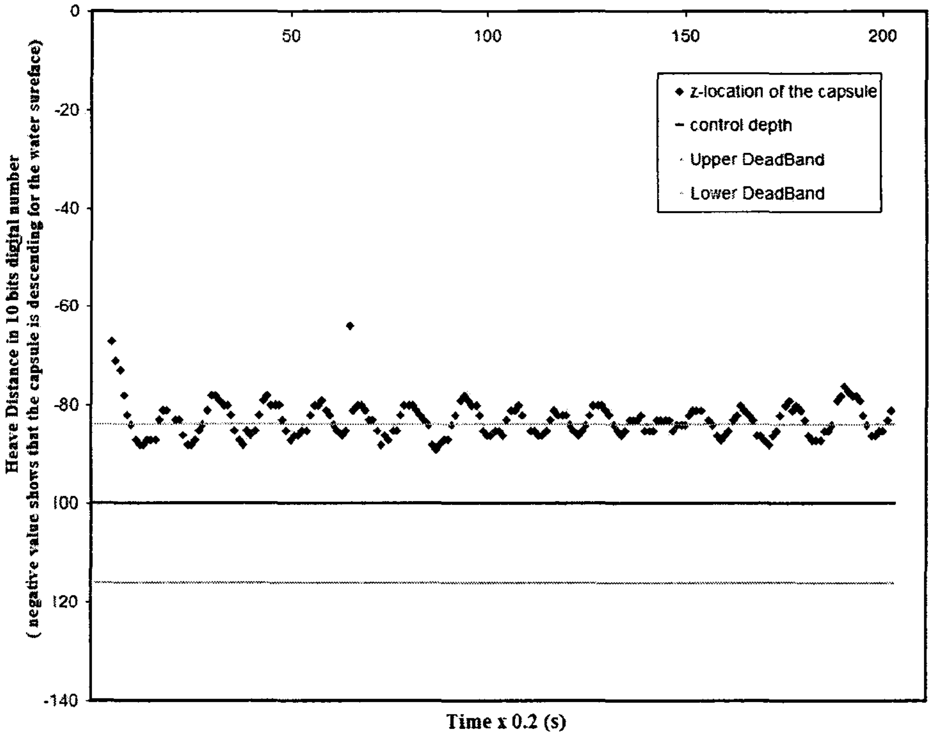


Figure 6.1 Result for Depth Control Experiment

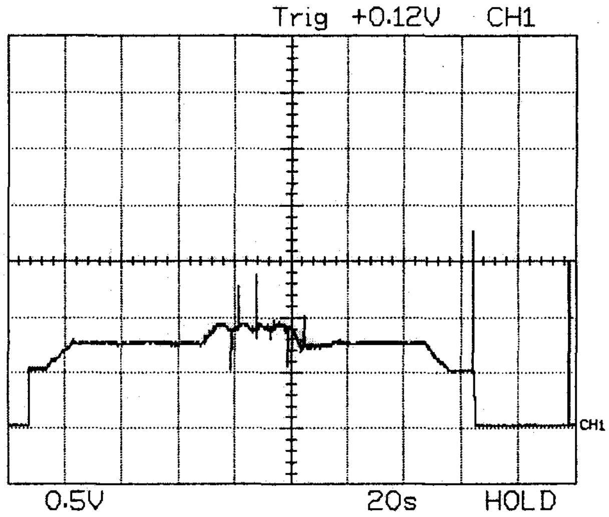


Figure 6.2 Oscilloscope Result for Depth Control Experiment

6.4 Direction Control Test

The direction control test was also done in the Deep Water Tank. It was done in a similar way to the depth control test. A new code was written so that the model capsule would try to turn itself facing outward from the edge of the tank. It would keep facing in this direction for approximately 40 seconds before turning itself toward the opposite direction and try to come back to where it was released. An overall operation would take about 80 seconds. To prepare for the test, all components were installed back to the main structure. In order to measure the bearing of the model capsule, the digital compass (Davantech Model CMDS03) had to be oriented in a proper position. Then the box was closed and sealed in the usual way.

First, the model capsule was tested outside the tank. The overhead crane was used to lift the model capsule off the floor. Once the operation had been started, one of the side motors would be turned on, as the capsule was supposed to turn itself toward the specified direction. Which motor would be turned on depended on the starting position of the model capsule. The capsule was then rotated by hand. As it approached the specified direction, the motor slowed down. Once the capsule was in a band surrounding its specified direction, both motors would be turned on. The result from the outside tank test showed that all systems were operating effectively.

Next, the model capsule was deployed into the Deep Water Tank, again by using the overhead crane. The code was changed to stop both motors when the capsule was facing in the specified direction, and the crane hook was left attached to the model. This was to prevent the capsule from crashing into the wall. The operation began by the capsule trying to turn itself until it faced outward from the wall. Both motors were then turned off and the capsule would try to keep itself in this direction for approximately 40 seconds before turning itself toward the opposite direction. Several tests were required to get satisfactory performance and debug program errors. After several changes of the code, the mission was quite successful. Data taken by sending back the signals from the PIC for this test is presented in Figure 6.3.

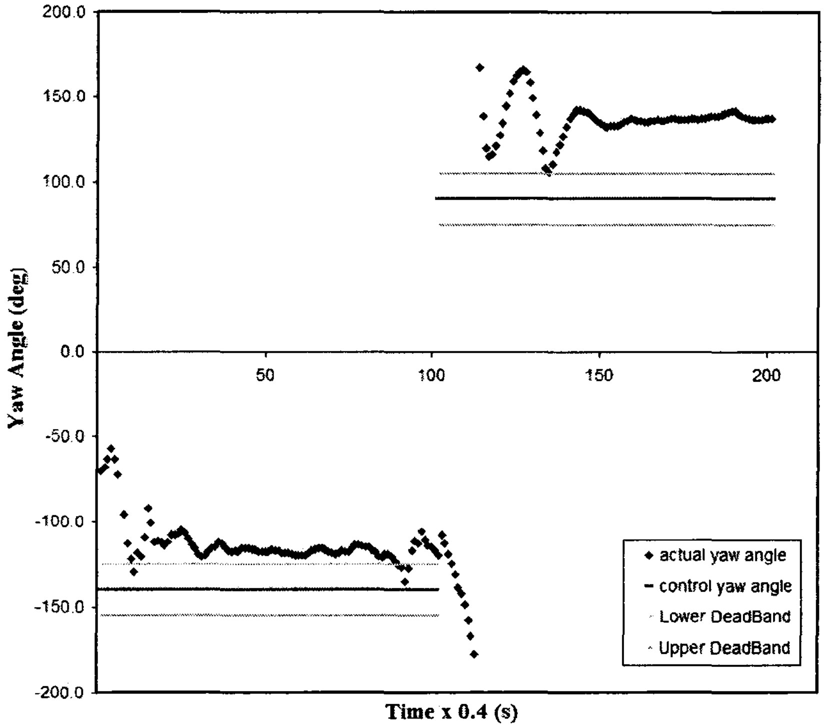


Figure 6.3 Result for Direction Control Experiment

As shown in this figure, the blue dots represent the direction the capsule was facing. As you can see, for the first 40 seconds, the capsule was trying to face itself to the direction which was close to the lower deadband of the control yaw angle. Then it tried to turn itself in counter-clockwise direction, passing minus180.0 degree into the positive angle region, and was trying to keep itself facing at one direction for another 40 seconds.

The direction the capsule facing was quite off from the control yaw angle this time. The reason for this was that there was metal surrounding the Deep Water Tank. It interfered the magnetic field sensed by the digital compass. Figure 6.3 confirms that the model capsule is capable of keeping itself pointing in a specified direction. The computer code for this test is given in Appendix E.

6.5 Complete Mission Test

Finally, the C codes for hovering and direction control were combined together. This code controlled the operation of the model capsule as it was trying to descend down to 1.2 meters below the water surface and get away from the structure. Since this test required space for the capsule to travel along its command trajectory path, the Deep Water Tank was too small for this test. Thus, the test took place in the Offshore Engineering Basin (OEB) at the Institute of Ocean Technologies (IOT), Newfoundland and Labrador.

To begin, all components were installed back to the model capsule. The following steps were necessary to ensure the mission could be completed. First, the proper code was downloaded to the controller. Then the digital compass (Davantech Model CMDS03) was installed in the proper position. Next, batteries for the electronic board and for the motors were fully charged and put back in place. The control chamber was then closed and sealed in the usual way. Two aluminum hooks were tightened to lock the covers in place. This was done to guarantee a water tight seal. In the next step, the control chamber was attached back to the main structure. Tire wraps were used for this purpose. All wires from battery and motors were then connected appropriately. This made the model capsule ready for the test.

An air test was performed to make sure that all systems were working perfectly. The overhead crane was used to lift the model off the floor. The model capsule was turned manually by hand to check if the digital compass (Davantech Model CMDS03) was working effectively. Air was blown through a small plastic tube to compress the air around the pressure transducer (Omega Model P40). The back motor speed and direction of the propellers were changed as the air was blown. This showed that the pressure sensor was working.

Then the model capsule was put into the OEB and was released from the overhead crane at the water surface. The test began as soon as the power switch was plugged in. After a 40 second delay, the model capsule was commanded to descend to 1.2 meters below the water surface. The back motor turned on and off as it tried to maintain itself at the desired depth. At the same time, yaw moment was generated by side motors and the model capsule turned toward the desired direction. As soon as the capsule faced to the desired direction, both motors were turned on. The capsule then traveled forward along this direction. In the second half of the testing period, the capsule was programmed to come back to its starting point. Thus, after approximately 60 seconds since the operation began, the capsule tried to turn itself toward the opposite direction and travel back to the beginning point. Finally, the capsule was programmed to ascend back to the water surface and float as its last stage. The overall mission lasted about 160 seconds.

No attempt was made to get data from the pressure transducer and the compass in OEB. This was because the model would have traveled long distances. In order to get these data, long wires would be needed for data transfer and there was a possibility that these wires would become entangled with the propellers. Instead, only photos and videos were taken of the OEB tests. Some of these pictures are shown in figures below. Figure 6.4 shows a picture of the Offshore Engineering Basin in which the test was taken place. Figure 6.5 shows the model capsule hanging from an overhead crane before the test began. Air was blown through the plastic tube to test the pressure sensor and the capsule was swung around to test the digital compass. Figure 6.6 and Figure 6.7 show the capsule when it was deployed into the OEB. The overhead crane hook was release and the model capsule was ready to be tested. Figure 6.8 and Figure 6.9 show the model capsule when it was started. As shown in the pictures, the capsule was trying to descend into water. Figure 6.10 shows the model capsule when it was hovering below water surface. Lastly, Figure 6.11 shows the model capsule when it turned around and tried to travel back to where it started. Overall the tests show the model can move along a preset trajectory.

6.6 Summary

The model capsule was tested to confirm the results from the 3D motion simulation. The test was divided into 4 steps: (1) Neutral buoyancy and water tight check, (2) Depth Control Test, (3) Direction Control Test, and (4) Complete Mission Test. The first three steps were tested in the Deep Water Tank at Memorial University of Newfoundland. The last step was tested in the Offshore Engineering Basin (OEB) at the Institute for Ocean Technologies (IOT), Newfoundland and Labrador since it required more space for the model capsule to travel. The results from these tests confirm that the model capsule is able to adequately follow any preset command trajectory.

Figure 6.4 The Offshore Engineering Basin (OEB)

Figure 6.5 The model capsule is hung by an overhead crane and ready for being test in the air

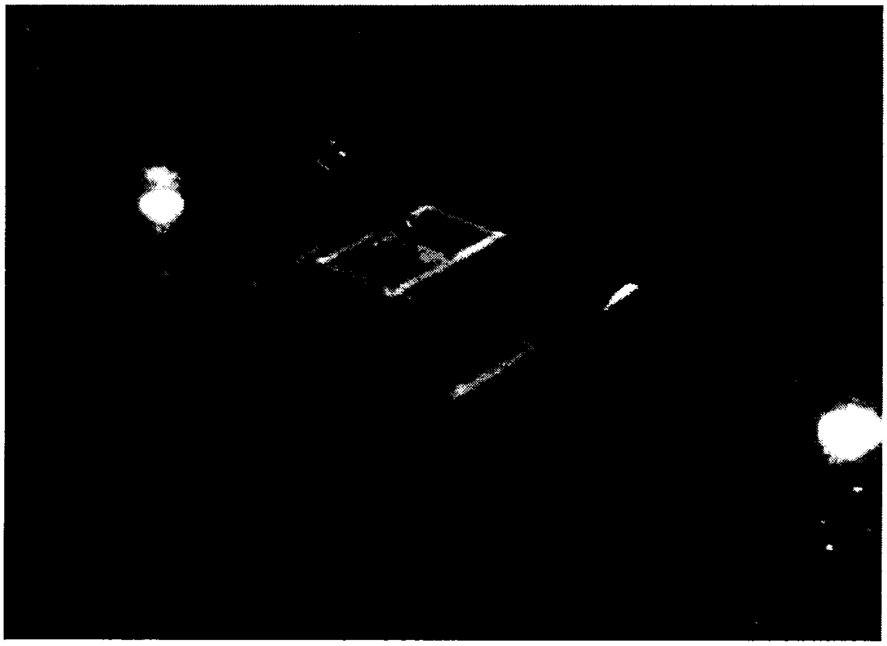


Figure 6.6 The capsule is deployed into the OEB using the overhead crane

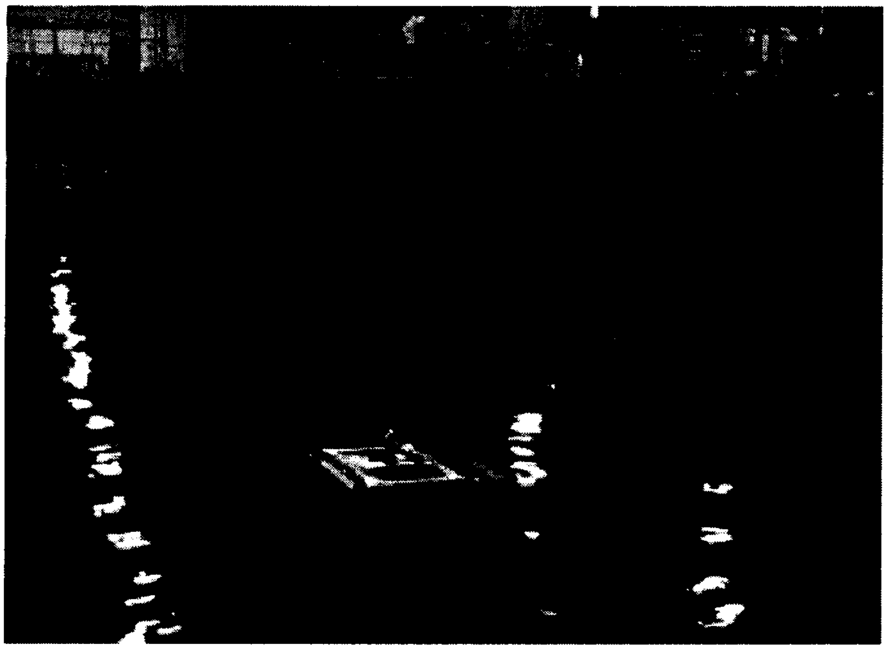


Figure 6.7 The model capsule is floating with neutral buoyancy and ready to start its operation

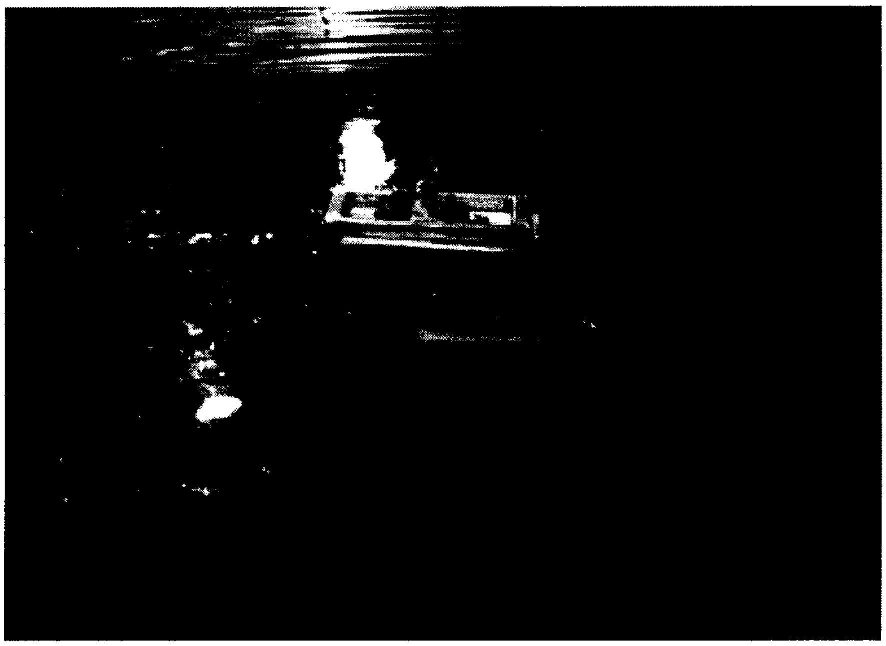


Figure 6.8 The model capsule has been started

Figure 6.9 The model capsule is ascending into water

Figure 6.10 The model capsule is trying to hover itself at 1.2munder water surface

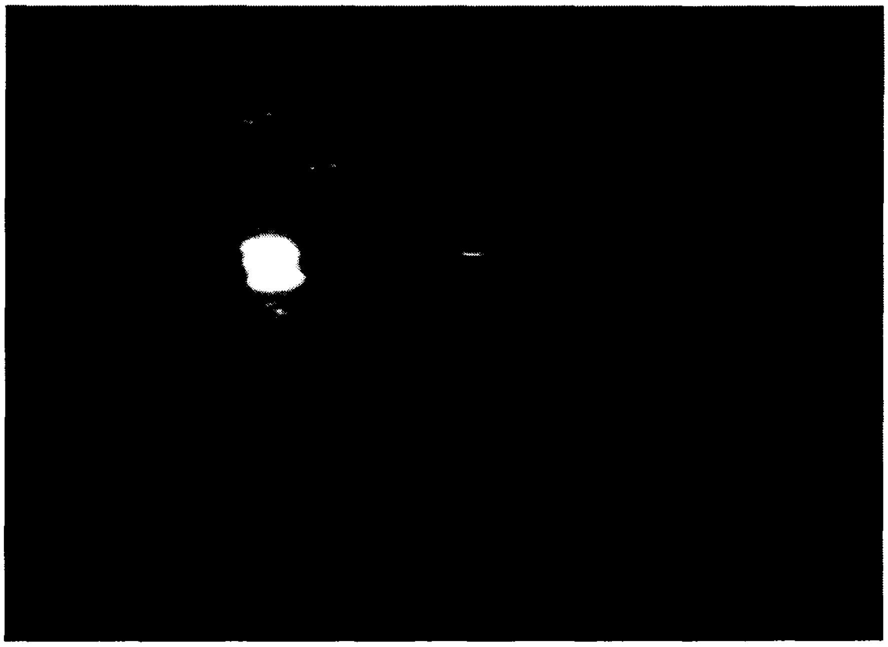


Figure 6.11 The model was turned. It is trying to travel back to where it started

Chapter 7

Conclusions and Future Work

As a conclusion, a new type of evacuation system called the Submersible Escape Capsule was designed in this research. The motivation to this research was the loss of 84 lives at the most tragedy accident on the coast of Newfoundland, the capsizing of the Ocean Ranger in 1982. Whereas, all other evacuation systems developed are for a complete crew, the Submersible Escape Capsule is designed to remove a single person from a structure during emergencies.

As described in early chapters, the model of the Submersible Escape Capsule was decided according to hydromechanic principles. MATLA/Simulink was then used to develop the 3D-Motion Simulation for the model capsule. The equations of motion of the model were used to generate the 3D trajectory path of the capsule when it is underwater. This simulation was designed to have very little coupling between its degrees of freedom. Thus, a combination of semi-empirical analysis and simple tests could be used to find most of the parameters using in the simulation. As shown in Chapter 4, the results from the simulation show that the model capsule was capable of moving along the desired path. An attempt to increase mass of the capsule was done in the simulation. This is to simulate the behavior of the capsule when its envelope is enclosed. The results from these trials show that the increase of mass does not have much effect on the trajectory path but it slows the system down. As a result, it takes more time for the capsule to travel along the desired path when its mass is increase. The results from the 3D-Motion Simulation show that this concept has merit and can be practical. The model capsule was then constructed. The experiment was then set up for testing the capsule in the Deep Water Tank at MUN and the OEB at the Institute for Ocean Technologies (IOT), Newfoundland and Labrador. The experiment was divided into 4 steps: (1) Neutral Buoyancy and Water Tight Check, (2) Depth Control Test, (3) Direction Control Test, and (4) Complete Mission Test. The first three steps were taken in the Deep Water Tank at MUN. The last step was taken in the OEB at IOT since the model capsule needed more space to travel and the Deep Water Tank was too small. The results for these tests show that the model capsule was both statically and dynamically stable. During the tests, the model capsule moved smoothly at high speed through the water. This implies that the thrusters and associated electronics were properly matched. Moreover, the module capsule was able to float at water surface as well as to hover itself at a certain level underwater when all thrusters were turned off. This implies that the model capsule was well designed to be neutrally buoyant.

Good agreement was found between the results from the 3D-Motion Simulation and the experiments. Hence, this research has proved that the idea of the Submersible Escape Capsule has potential for further development.

For further research issue, a more realistic model capsule needs to be constructed. The new model should have its mass and its enclosed envelope closer to the prototype. The digital compass used in the current model is easily distracted by the magnetic field caused by the surrounding metals and the magnetic field generated by its motors. This causes the capsule to go out of track sometimes during its mission while testing in the

Deep Water Tank and the OEB. Thus it should be replaced by the better one. Higher capacity batteries should also be utilized for extension of battery life. A buoyancy engine could be used to make sure the capsule is always neutrally buoyant.

After the new model has been tested, a prototype should be constructed and tested at sea. For the prototype, all of the factors mentioned earlier such as the in air trajectory would have to considered; an emergency-ascent system would be needed in order to bring the capsule back to the water surface in case of a computer or power failure. A good example of this system involves the release of some large item of weight such as the battery pack or drop weight. Potential sources of funds for prototype construction include: Search & Rescue New Initiatives Fund (SAR/NIF), Atlantic Innovation Fund (AIF), and Petroleum Research Atlantic Canada (PRAC).

Finally, the model developed in this thesis is basically an electromechanical system. Thus, it can be adapted for educational purposes. Laboratories could be set up around this model for mechatronics and controls courses for engineering students at Memorial University of Newfoundland. In addition, the idea of the Submersible Escape Capsule can also be found in [20] and [21]. Soft copy of the control and simulation codes, system test video, circuit diagram and mechanical drawings are available by email upon request.

References

[1] Det Norske Veritas. *Evacuation and Rescue Means – Strengths, Weaknesses and Operational Constraints*, Technical Report: NPD and Norwegian Oil Industry Association, Final Report, Report No. 98-5601, Revision No. 03.

[2] Dan O'Brien, Michael Hinchey, and Derek Muggeridge (1993). *Recent Developments in Offshore Rig/Platform Evacuation*, Ocean Engineering, Vol. 20, No. 6, pp. 555-567.

[3] Michael Hinchey, Dan King, and Derek Muggeridge (1991). *A Submersible Escape Capsule for Offshore Oil Rigs*, Ocean Engineering Research Centre, Memorial University of Newfoundland: St. John's, Newfoundland.

[4] António Simões Ré, Brian Veitch, and Scott Newbury (1999). Offshore Evacuation Systems – Summary Report. Prepared for Transportation Development Centre Safety and Security Transport Canada.

[5] Scott Newbury Marine Consulting. (1999). Offshore Evacuation Systems – Survey of Approved Systems for the Canadian Offshore. Prepared for The Institute for Marine Dynamics, National Research Council Canada.

[6] Frank G. Bercha, Milan Cerovšek. Paul Gibbs, and Ernest. Radloff. (2001). *Arctic Offshore EER Systems*, Proceedings of the 16th International Conference on Port and Ocean Engineering under Arctic Conditions, Ottawa, Ontario, Canada, pp. 495-504.

[7] Michael Hinchey, and Raju Goteti (1989). *Submergence Depth Control of a Free-Fall Submersible Escape Capsule for an Offshore Oil Rig*, Proceeding of Ocean '98 An International Conference Addressing Methods For Understanding The Global Ocean, Seattle, Washington, pp. 734-740.

[8] Michael Hinchey (2005). Private Communication.

[9] Allmendinger (1990). Submersible Vehicle Systems Design, the Society of Naval Architects and Marine Engineers: Jersey City, New Jersey.

[10] John Paul Comstock (1967). Principles of Naval Architecture, the Society of Naval Architects and Marine Engineers: New York

[11] Karen Muggeridge, and Michael Hinchey (1991). *Underwater Robot Control*, Newfoundland Electrical and Computer Engineering Conference (NECEC'91): Marine Institute, St.John's, Newfoundland.

[12] Michael Hinchey (1996). *Supervisory Control Strategies for Autonomous Subsea Robots*, Memorial University of Newfoundland: St. John's, Newfoundland.

[13] Reeni Woolgar (1999). Development of the Subsea Robot ANNE (Autonomous Pneumatic Nautical Explorer), Masters Thesis, Faculty of Engineering and Applied Science, Memorial University of Newfoundland: St. John's, Newfoundland.

[14] Michael Hinchey (2003). *Automatic Control Engineering*, Memorial University of Newfoundland: St. John's, Newfoundland.

[15] Bruce Colbourne (2004). *Calculating Wave Force on Offshore Structures*, Memorial University of Newfoundland: St. John's, Newfoundland.

[16] Michael Hinchey (2003). *Steady Foil Theory*, Memorial University of Newfoundland: St. John's, Newfoundland.

[17] John Newman (1980). Marine Hydrodynamics, The MIT Press Cambridge, Massachusetts, and London, England.

[18] Robert W. Fox, Alan T. McDonald, Philip J. Pritchard (2004). Introduction to Fluid Mechanics (6th ed.), John Wiley & Son, Inc.: Hoboken, NewJersey.

[19] Katsuhiko Ogata (1997). Modern Control Engineering (3rd ed.), Prentice-Hall, Inc.: Upper Saddle River, New Jersey.

[20] Michael Hinchey, Rob Norris, Brian Crane and Supangpen Ruangwech (2004). *Development of a Series of Small Versatile Autonomous Underwater Vehicles*, Newfoundland Electrical and Computer Engineering Conference (NECEC'04): St.John's, Newfoundland.

[21] Supangpen Ruangwech and Michael Hinchey (2005). *Development of a Submersible Escape Capsule for Offshore Structures*, Newfoundland Electrical and Computer Engineering Conference (NECEC'05): St.John's, Newfoundland.

Appendix A

Weight and Buoyancy Calculations

As shown in Table A-1, the calculation result is negative buoyancy. This means that the buoyancy of the model capsule is insufficient. A certain volume of Styrofoam had to be added to the model to increase buoyancy. The shape of this Styrofoam was determined by where it had to be positioned on the capsule.

The decision was made and the Styrofoam would be added to the top of the model capsule. The reason for this was to ensure that the center of buoyancy (C_B) of the capsule would be maintained above the center of gravity (C_G) of the capsule. As a result, the model capsule would maintain itself in an upright position.

According to the calculation in Table A-1, the difference between air weight of the capsule and its buoyancy was equal to 19.6020 kilograms. This was then converted to a volumetric displacement of 0.019602 cubic meters or 1199.1874 cubic inches. Thus, this amount of Styrofoam had to be added to the model capsule to make it neutrally buoyant. Several trials were required to obtain satisfactory shape for Styrofoam to fit with the main frame of the model capsule. Figure A-1 shows its final shape and dimension. Moment calculations were then used to position the batteries and motors to get zero trim. To obtain zero trim, the center of gravity (C_G) must be beneath the center of buoyancy (C_B) on the same vertical line normal to submersible's xy-plane. The center of gravity (C_G) can be determined from

$$C_G \sum W_n = W_1 d_1 + W_2 d_2 + W_3 d_3 + \ldots \qquad \textbf{(a.1)}$$

where $W_n =$ weight of each components

$d_n =$ distance from each component to the reference line and the center of buoyancy can be determined from

$$C_B \sum \rho V_n = \rho V_1 d_1 + \rho V_2 d_2 + \rho V_3 d_3 + \ldots \qquad \textbf{(a.2)}$$

where ρ = density fluid. In the case of fresh water, it is equal to 1000kg/m^3

$V_n =$ volumetric displacement of each components

$d_n =$ distance from each component to the reference line

In order to locate C$_G$ on the same vertical line as C$_B$,

$$\frac{W_1 d_1 + W_2 d_2 + W_3 d_3 + \ldots}{\sum W_n} = \frac{V_1 d_1 + V_2 d_2 + V_3 d_3 + \ldots}{\sum V_n} \qquad \textbf{(a.3)}$$

The location of control chamber had been fixed. Iteration method had been used to get the satisfactory location of battery and motors. Table A-2 shows all components location on an XY-plane. The schematic details of these components location is shown in Figure A-2.

Calculations for center of buoyancy (C_B) and center of gravity (C_G) are shown below.

Center of buoyancy (C_B) calculation

From equation (a.2) to get the location of the center of buoyancy (C_B) calculation, this equation has to be rearranged and the result gives

$$\therefore \quad C_B = \frac{\rho V_1 d_1 + \rho V_2 d_2 + \rho V_3 d_3 + \ldots}{\sum \rho V_n}$$

Consider center of buoyancy (C_B) on x-axis

$$C_{BX} = \frac{\rho V_1 d_{X1} + \rho V_2 d_{X2} + \rho V_3 d_{X3} + \dots}{\sum \rho V_n}$$

$$C_{BX} = \frac{1000[(180.4102)(10.8750) + (175.2855)(7.8735) + (1005.4508)(10.6545) + (1175.9869)(5.0297)]}{1000(180.4102 + 175.2855 + 1005.4508 + 1175.9869)}$$

$$C_{BX} = \frac{1000[1961.9609 + 1380.1104 + 10712.5756 + 5914.8613]}{1000(2537.1334)}$$

$$C_{BX} = \frac{1000(19969.5082)}{1000(2537.1334)}$$

$$\therefore \quad C_{BX} = 7.8709$$

Consider center of buoyancy (C_B) on y-axis

$$C_{BY} = \frac{\rho V_1 d_{Y1} + \rho V_2 d_{Y2} + \rho V_3 d_{Y3} + \dots}{\sum \rho V_n}$$

$$C_{BY} = \frac{1000[(180.4102)(-21.2168) + (175.2855)(11.4580) + (1005.4508)(-5.6250) + (1175.9869)(18.7732)]}{1000(180.4102 + 175.2855 + 1005.4508 + 1175.9869)}$$

$$C_{BY} = \frac{1000[(-3827.7271) + (2008.4213) + (-5655.6608) + (22077.0373)]}{1000(2537.1334)}$$

$$C_{BY} = \frac{1000(14602.0707)}{1000(2537.1334)}$$

$$\therefore C_{BY} = 5.7553$$

Center of gravity (C_G) calculation

From equation (a.1) to get the location of the center of gravity (C_G) calculation, this equation has to be rearranged and the result gives

$$\therefore \quad C_G = \frac{W_1 d_1 + W_2 d_2 + W_3 d_3 + \dots}{\sum W_n}$$

Consider center of gravity (C_G) on x-axis

$$C_{GX} = \frac{W_1 d_{X1} + W_2 d_{X2} + W_3 d_{X3} + \ldots}{\sum W_n}$$

$$C_{GX} = \frac{[(8.7217)(1.6375) + (5.8967)(10.8750) + (13.6077)(7.8735) + (13.1542)(10.6545)]}{1000(8.7217 + 5.8967 + 13.6077 + 13.1542)}$$

$$C_{GX} = \frac{[14.2818 + 64.1266 + 107.1402 + 140.1514]}{1000(41.3803)}$$

$$C_{GX} = \frac{(325.7000)}{(41.3803)}$$

$$\therefore \quad C_{GX} = 7.8709$$

Consider center of gravity (C_G) on y-axis

$$C_{GY} = \frac{W_1 d_{Y1} + W_2 d_{Y2} + W_3 d_{Y3} + \ldots}{\sum W_n}$$

$$C_{GY} = \frac{[(8.7217)(-3.1714) + (5.8967)(-21.2168) + (13.6077)(11.4580) + (13.1542)(-5.6250)]}{1000(8.7217 + 5.8967 + 13.6077 + 13.1542)}$$

$$C_{GY} = \frac{[(-27.6600) + (-125.1091) + 155.9170 + (-73.9924)]}{1000(41.3803)}$$

$$C_{GY} = \frac{(-70.7845)}{(41.3803)}$$

$$\therefore \quad C_{GY} = -1.7106$$

Table A-1 Weight and Displacement Calculation

Components	Weight/unit (kg)	Displacement/unit (cubic inches)	unit	Total weight (kg)	Total displacement (cubic inches)	Displacement (cubic meters)
1. Structural and Frame						
1.1 Main Body	8.7217	negligible	1	8.7217	negligible	
1.2 Thrust Producing Units' Holder						
2.Propulsion and Maneuvering System						
2.1 Energy Generating System	5.8967	180.4102	1	5.8967	180.4102	0.0029
2.2 Thrust Producing Units	4.5359	58.4285	3	13.6077	175.2855	0.0028
2.3 Subconn Connector and wires	negligible					
3. Operation Control System						
3.1 Control Chamber	13.1542	1005.4508	1	13.1542	1005.4508	0.0161
Total Air Weight of the model capsule (kg)	41.3803					
Total Displacement of the model capsule (cubic meters)	0.0218					
Total Buoyancy of the model capsule (kg)	21.7783					
Difference between air weight and buoyancy (kg)	-19.6020					

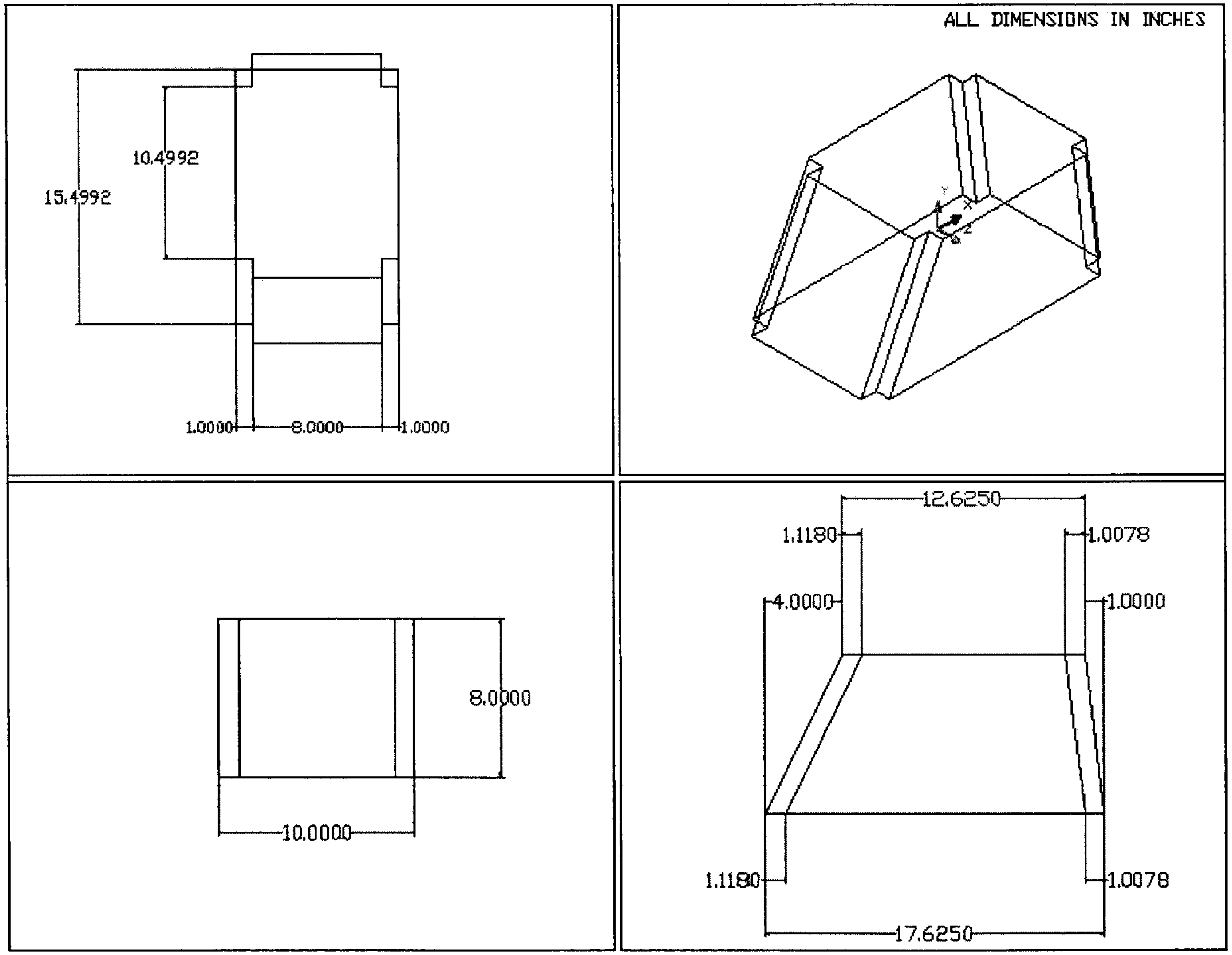


Figure A-1 Styrofoam Dimensions

Table A-2Capsule Components Location, Center of Buoyancy and Center of Gravity of the capsule

Components	Center of Buoyancy and/or Center of Gravity		
	x	y	Z
1. Structural and Frame			
1.1 Main Body	1.6375	-3.1714	0.0000
1.2 Thrust Producing Units' Holder			
2.Propulsion and Maneuvering System			
2.1 Energy Generating System	10.8750	-21.2168	0.0000
2.2 Thrust Producing Units	7.8735	11.4580	0.0000
2.3 Subconn Connector and wires	Negligible		
3. Operation Control System			
3.1 Control Chamber	10.6545	-5.6250	0.0000
4. Buoyancy adjusted			
4.1 Added Styrofoam	5.0297	18.7732	0.0000
4.2 Lead weight	Negligible		
Center of Buoyancy of the capsule	7.8709	5.7553	0.0000
Center of Gravity of the capsule	7.8709	-1.7120	0.0000

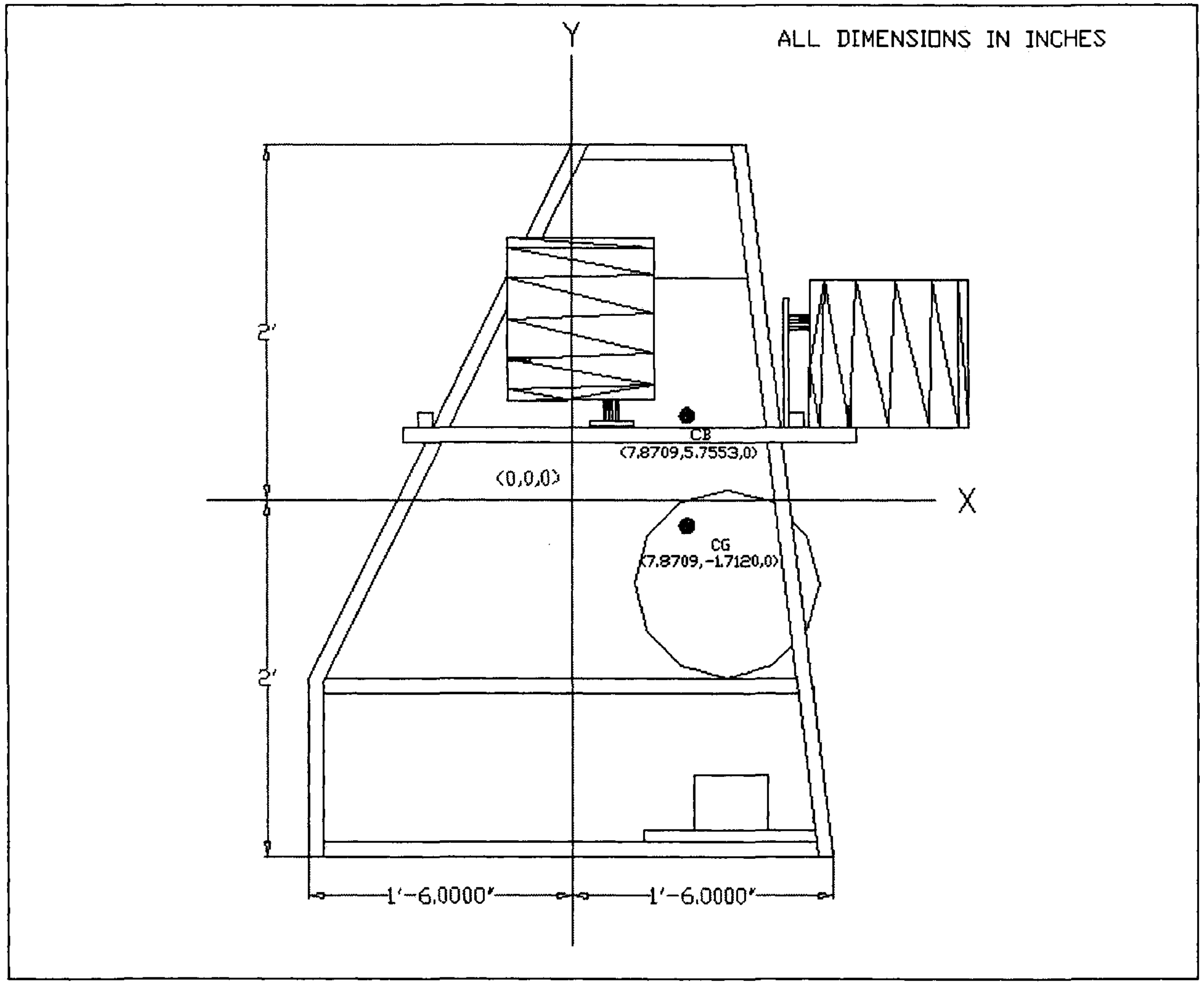


Figure A-2 Location of the center of gravity and center of buoyancy on an XY-plane

Appendix B

Estimation of Parameters used in 3D-Simulation

Determination of all parameters using in 3D-simulation

All parameters used in 3D simulation of the Submersible Escape Capsule were determined by estimations or experiments.

Masses (M) and Mass Moments of Inertia (I)

The first two parameters to be determined in this appendix are mass and mass moment of inertia. Mass of the model capsule is used in the governing equations for heave and surge motions. Mass moment of inertia of the model capsule is used in the governing equations for pitch and yaw motions. These two parameters are basically equal to summations of the submersible weight and its loads, the water entrained in the submersible's envelope and its appendages, and the amount of water accelerated by the submersible's motion. This implied that mass and mass moment of inertia of the model capsule varies with the capsule envelope type and its motion through water. There was no facility available to determine the actual value of the water accelerated by the submersible's motion. According to former experiments done by many engineers, the amount of water accelerated by its motion can be estimated to be 50 percents of the submersible actual weight. Thus mass and mass moment of inertia of the model capsule can be estimated and the calculations are as shown below.

Mass of the model capsule (M)

$$\textit{The total air weight of the capsule} \quad = \quad 41.3803 \quad kg$$

$$\therefore \quad \textit{Mass of the capsule when accounted with the added mass}$$

$$= \quad (41.3803\,kg) + (41.3803 \times 0.50\,kg)$$

$$= \quad 41.3803 + 20.6902\,kg$$

$$= \quad 62.0705\,kg$$

Mass moments of inertia of the capsule (I)

Mass moments of inertia for the model capsule can be calculated by applying equations in Chapter 4. The detail calculations for mass moment of inertia using in the governing equations for pitch and yaw are as shown below.

Mass moment of inertia for pitch motion (rotation about y-axis)

$$I = \sum mR^2$$

$$I = [(13.6877)(0.1265)^2 + (13.1542)(0.2976)^2 + (5.8967)(0.6893)^2 + (8.7217)(0.2811)^2]$$

$$I = [(0.2190) + (1.1650) + (2.8017) + (0.6892)]$$

$$I = 4.8748 \quad (kg)(m)^2$$

$$\therefore \quad \textit{Mass moment of inetia when accounted with the added mass}$$

$$= \quad (4.8748\ kgm^2) + (4.8748 \times 0.50\ kgm^2)$$

$$= \quad 4.8748 + 2.4374\ kgm^2$$

$$= \quad 7.3122\ kgm^2$$

Mass moment of inertia for yaw motion (rotation about z-axis)

$$I = \sum mR^2$$

$$I = [(13.6877)(0.0042)^2 + (13.1542)(0.0707)^2 + (5.8967)(0.0763)^2 + (8.7217)(0.1600)^2]$$

$$I = [(2.4145 \times 10^{-4}) + (0.0658) + (0.0343) + (0.2233)]$$

$I = 0.3236 \quad (kg)(m)^2$

$\therefore \quad Mass\, moment\, of\, inetia\, when\, accounted\, with\, the\, added\, mass$

$= \quad (0.3236 \; kgm^2) + (0.3236 \times 0.50 \; kgm^2)$

$= \quad 0.3236 + 0.1618 \; kgm^2$

$= \quad 0.4854 \; kgm^2$

Drag Factor (K)

Drag factors to be determined in this appendix are ones use to calculate form drag. Form drag is created by the fluid that flowing over the body. The separation of the body then creates turbulence and results in pocket of low and high pressure that leaves a wake behind the body. Form drag can be calculated by applying Morrison's Equation

$$F_{Drag} = \frac{1}{2} C_D \rho A U \,|\, U \,| \qquad \textbf{(b.1)}$$

Thus, it can be written as

$$F_{Drag} = KU \,|\, U \,| \qquad \textbf{(b.2)}$$

where F_{Drag} = form drag

C_D = drag coefficient of the body

ρ = fluid density

A = projected area of the underwater portion of the body

U = relative velocity of the body

K = drag factor = $\frac{1}{2} C_D \rho A$

Experiments were set up to determine the values of K use in the governing equations for heave surge and yaw. The value of K use in the governing equations for pitch was estimated based on projected area of the underwater portion of the body.

Drag factor for heave

To get drag factor for heave, an experiment was set up in the Deep Water Tank at MUN. First, the model capsule was lifted by an overhead crane and deployed into the tank. It was floated by neutral buoyancy. In the next step, masses of 150g, 200g and 250g were added on top of the capsule, one at a time, to increase its weight. As the mass was added, weight of the capsule became larger than its buoyancy; this made the capsule sink right to the bottom of the tank. An overall depth of the tank was measured. This was subtracted by the height of the capsule to give a distance (s) required for the capsule to travel. The time taken for the capsule since it started to sink until it reached the bottom of the tank (t) was noted. Four trials were done for each value of added mass. The data taken from this experiment is shown in Table (B-1). The average time taken for each value of added mass to make the capsule sink to the bottom was then calculated. Next, these values were used to calculate velocity of the capsule as it was traveling to the bottom of the tank (v) using the following equation.

$$U = v = \frac{s}{t} \qquad \textbf{(b.3)}$$

Next, the velocities calculated from equation (b.3) were substituted in equation (b.2). In this case, drag force for each value of velocity is equal to the added mass times gravitational acceleration $g = 9.81 m/s^2$. The results for these calculations are as shown in Table (B-2) as well as the average value for drag factor (K). After finished the experiment, the model capsule was retrieved from the tank and rested until it dried.

Table B-1 Time taken for the model capsule to travel to the bottom of the Deep Water Tank

Added mass (g)	Time (s)				
	Trial 1	Trial 2	Trial 3	Trial 4	Average
150	42	46	41	45	43.5
200	36	38	37	36	36.75
250	32	33	33	33	32.75

Table B-2 Drag factor calculation for heave

Added mass (g)	Drag force (N)	Velocity (m/s^2)	(Velocity)^2 (m/s^2)^2	Drag factor K
150	1.4715	0.0574	0.0033	447.2987
200	1.9620	0.0679	0.0046	425.6696
250	2.4525	0.0762	0.0058	422.5621
			Average drag factor	431.8435

Drag factor for surge

Another experiment was set up in the Deep Water Tank to find drag factor for surge motion. The model capsule was again lifted by the overhead crane and deployed into the tank. The hook was then released and the capsule was floated in the tank by its buoyancy.

In this experiment, a rope was attached to the frontal part of the capsule. This rope was then passed through the pulley and a mass was attached to another end of it. Three different values of mass were used as the experiment required many trials to ensure that the value of drag factor was closed to the actual one. These masses include 500g, 1000g, and 1500g. Each of these weights was then attached to the rope one at a time. As the weight was let go, the rope pulled the capsule and moved it forward along x-direction.

The width of the tank was measured. This was subtracted by the width of the capsule to obtain the distance required for the capsule to travel (s). The time taken for the capsule since it started to move until it reached the opposite edge of the tank (t) was measured. Four trials were done for each value of added mass. The data taken from this experiment is shown in Table (B-3). The average time taken for each value of added mass to make the capsule travel from one end of the tank to another end was then calculated. Next, these values were used to calculate velocity of the capsule as it was traveling to the edge of the tank (v) using equation (b.3). Next, the velocities calculated from equation (b.3) were substituted in equation (b.2). In this case, drag force for each value of velocity is equal to the added mass times gravitational acceleration $g = 9.81 m/s^2$. The results for these calculations as well as the average value for drag factor (K) are as shown in Table (B-4). After finished the experiment, the model capsule was retrieved from the tank and rested until it dried.

Table B-3 Time taken for the model capsule to travel to the edge of the Deep Water Tank

Added mass (g)	Time (s)				
	Trial 1	Trial 2	Trial 3	Trial 4	Average
500	21	22	21	20	21
1000	15	14	15	15	14.75
1500	12	12	12	12	12

Table B-4 Drag factor calculation for surge

Added mass (g)	Drag force (N)	Velocity (m/s^2)	$(Velocity)^2$ $(m/s^2)^2$	Drag factor K
500	4.9050	0.1321	0.0175	280.8999
1000	9.8100	0.1881	0.0354	277.1578
1500	14.7150	0.2313	0.0535	275.1673
			Average drag factor	277.7417

Drag factor for yaw

An experiment was also set up in the Deep Water Tank to obtain drag factor for yaw motion. This time, an aluminum bar was attached in the top part of the capsule; its center was closed to where the crane hook holder is. A rope was then attached to each end of the aluminum bar. The model capsule was again lifted by the overhead crane and deployed into the tank. The hook was then released and the capsule was floated in the tank by its buoyancy. The two ropes were passed through the pulleys and a mass of 150 g was applied at the end of each rope. As masses were released to fall from their starting positions, moments would be generated; the model capsule would be turned counter clockwise by these moments. The time taken for the capsule to turn from 0 degree to 90 degrees (t) was measured. The angular velocity (ω) of the capsule, as it rotated, was then calculated using equation (b.4). Next, the velocity was substituted in equation (b.2). In this case, drag force is equal to the added mass times gravitational acceleration $g = 9.81 m/s^2$. This results in the drag factor (K) of 21.4695. This calculation is as shown below.

$$\omega = \frac{\Omega}{t} \qquad \textbf{(b.4)}$$

where ω = angular velocity (rad/s)

Ω = angular displacement (rad)

t = time (s)

In this case $\Omega = \frac{\pi}{2}\, rad$ and $t = 6\, s$

$$\omega = \left(\frac{\pi}{2} \times \frac{1}{6}\right) \frac{rad}{s}$$

$$\therefore \quad \omega = 0.2618 \frac{rad}{s}$$

To calculate drag factor (K) for yaw motion

$$K = \frac{F_{drag}}{U\,|\,U\,|}$$

$$K = \frac{150 \times 0.100 \times 9.81}{0.2618 \times |\, 0.2618 \,|}$$

$$\therefore \quad K = 21.4695$$

After finished the experiment, the model capsule was retrieved from the tank and rested until it dried.

Drag factor for pitch

Since there was no facility available to set up the experiment to determine drag factor for pitch motion, a draft estimate of drag factor for yaw motion was done according to equations (b.5). The values of drag factor K was then equal to

$$K = \frac{1}{2} C_D \rho A \qquad \textbf{(b.5)}$$

where $C_D =$ drag coefficient of the body

ρ = fluid density

A = projected area of the underwater portion of the body

The value of C_D is normally obtained by an experiment. In this case, Reynolds number was used to determine drag coefficient (C_D) for pitch motion. Reynolds number for pitch motion can be calculated from

$$\mathrm{Re} = \frac{\rho v D}{\mu} \tag{b.6}$$

where ρ = fluid density

v = velocity of the capsule

D = diameter of the body

μ = viscosity of the fluid

Next, the parameter values were then substituted in equation (b.6):

$\rho = 1000 kg/m^3$, $v = 0.2 m/s$, $D = 0.254 m$, and $\mu = 0.001569 kg/(m \cdot s)$.

Thus,

$$\mathrm{Re} = \frac{(1000 kg/m^3)(0.2 m/s)(0.254 m)}{0.001569 kg/(m \cdot s)}$$

$$\therefore \quad \mathrm{Re} = 3.2377 \times 10^4$$

As shown in Figure B-1, drag coefficient (C_D) for pitch motion is equal to 1. Substitute this value in equation (b.5) to get drag factor K.

$$K = \frac{1}{2}(1)(1000 kg/m^3)(0.0516 m^2)$$

$$\therefore \quad K = 25.8$$

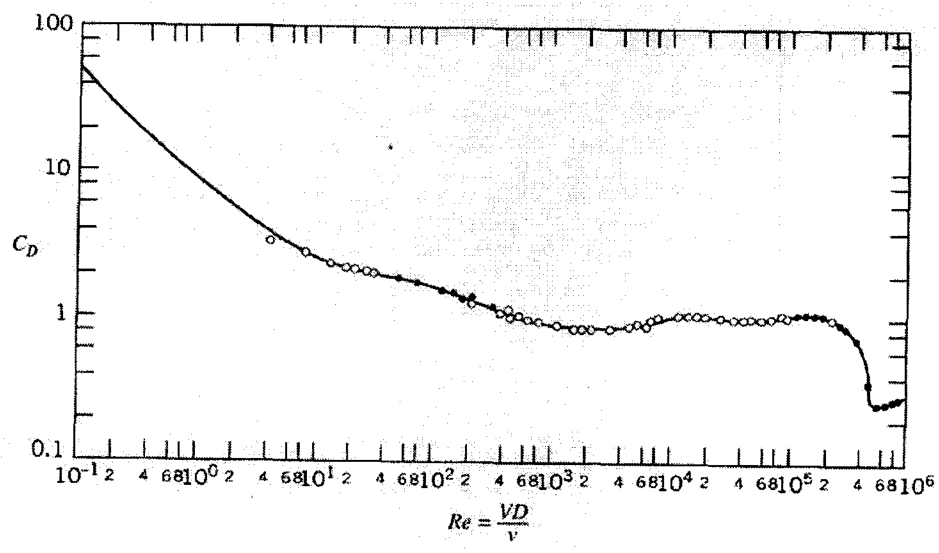


Figure B-1 Drag coefficient as a function of Reynolds Numbers [18]

H-Bridge Gain

The H-Bridge gain uses in the 3D-motion simulation can be calculated from the output and input voltage ratio of the H-Bridge module (Davantech Model MD03), equation (b.7).

$$H\,Bridge\;Gain = \frac{V_{output}}{V_{input}} \qquad \textbf{(b.7)}$$

where $H\,Bridge\;Gain$ = H-Bridge gain uses in the 3D-motion simulation

V_{output} = output voltage from H-Bridge module

V_{input} = input voltage to H-Bridge module

According to the data sheet for H-Bridge module (Davantach Model MD03) shown in Appendix F, the required input voltage for this H-Bridge module is equal to 5VDC. The maximum output from the H-Bridge module is equal to 12VDC since the thrusters use in the model capsule required the maximum voltage of 12VDC in order to be operated at its full speed. These values are then substituted in equation (b.7) and the result is as shown below.

$$H\,Bridge\;Gain = \frac{12\,VDC}{5\,VDC}$$

$$\therefore\quad H\,Bridge\;Gain = 2.4$$

Thus, the H-Bridge gain uses is the 3D-motion simulation is equal to 2.4.

Ziegler-Nichols Tuning Rule (Second Method) for PID Controller

Ziegler-Nichols Tuning Rule (Second Method) for PID Controller is based on Critical Gain (K_{CR}) and Critical Period (P_{CR}) of the system [19]. This method is normally used when the system opened loop cannot be easily defined; or there is no time delay in the system reaction. To proceed through this method, we start by setting the gain for the integral controller (K_I) to be 0 and the gain for the derivative controller (K_D) to be infinity (∞). Next, gradually increase the proportional gain (K_P) from 0 until oscillations start. Slowly adjust this gain until the output show signs of constant amplitude oscillation. The proportional gain at this output value is called the critical gain (K_{CR}); and the corresponding period is called the critical period (P_{CR}) [19]. According to Katsuhiko Ogata [19], the values of the parameters K_P, T_i ,and T_D can then be set according to Table (B-5).

Table B-5 Ziegler-Nichols Tuning (Second Method) [19]

Type of Controller	K_P	T_i	T_D
P	$0.5K_{CR}$	∞	0
PI	$0.45K_{CR}$	$\frac{1}{1.2}P_{CR}$	0
PID	$0.6K_{CR}$	$0.5P_{CR}$	$0.125P_{CR}$

Thus, the integral gain (K_I) and the proportional gain (K_D) can be determined from the following formula.

$$K_I = K_p\left(\frac{1}{Ti}\right) \quad \text{(b.7)}$$

$$K_D = K_P T_D \quad \text{(b.8)}$$

This method only gives the first approximation for these gains. The values of these parameters can be adjusted later for better result [19].

Appendix C
Detailed Fabrication Drawings

Submersible Escape Capsule

Fabrication Drawings

Structure and Frame

Drawing A Structure and Frame

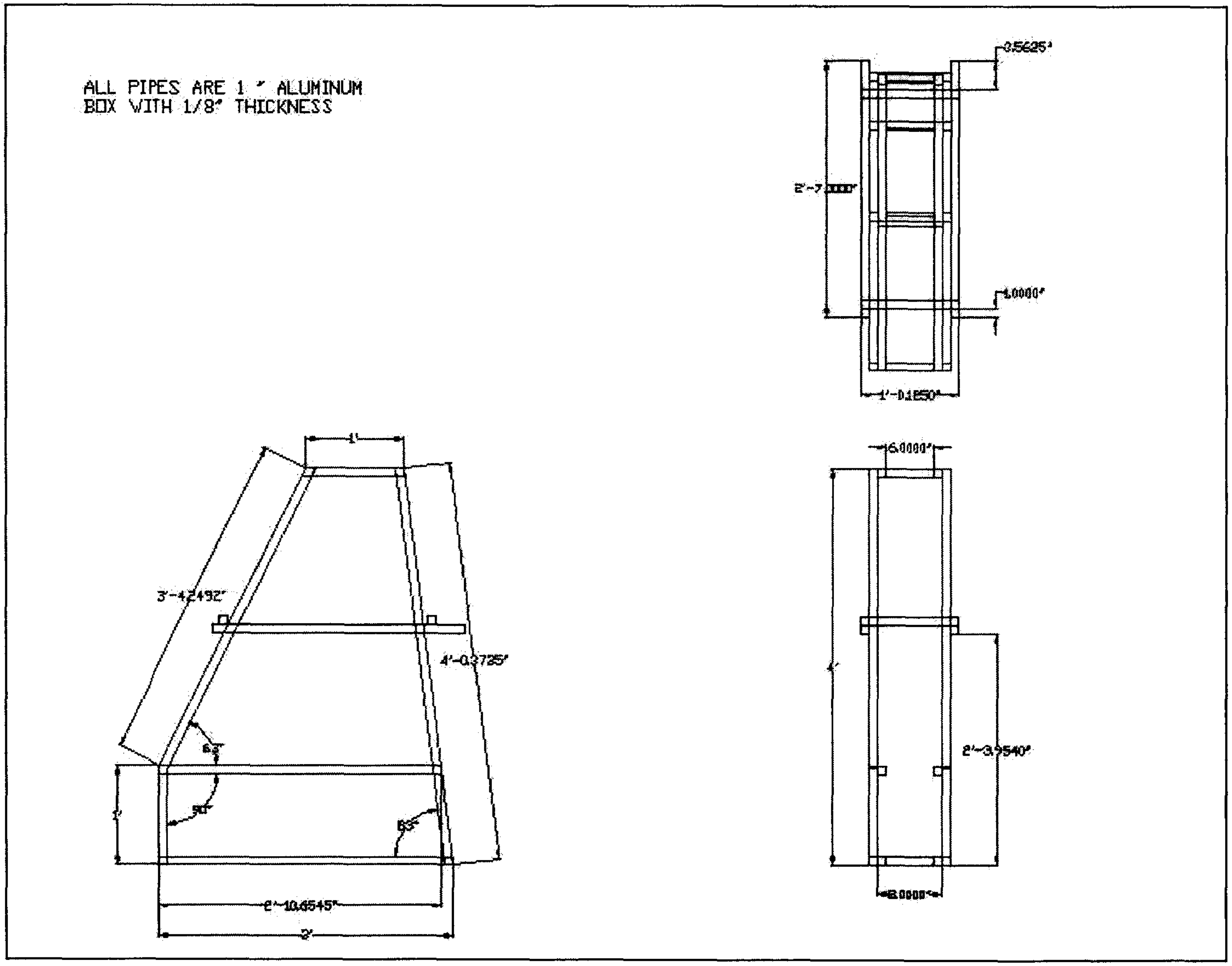
Propulsion and Maneuvering System

Drawing B Thruster Shroud

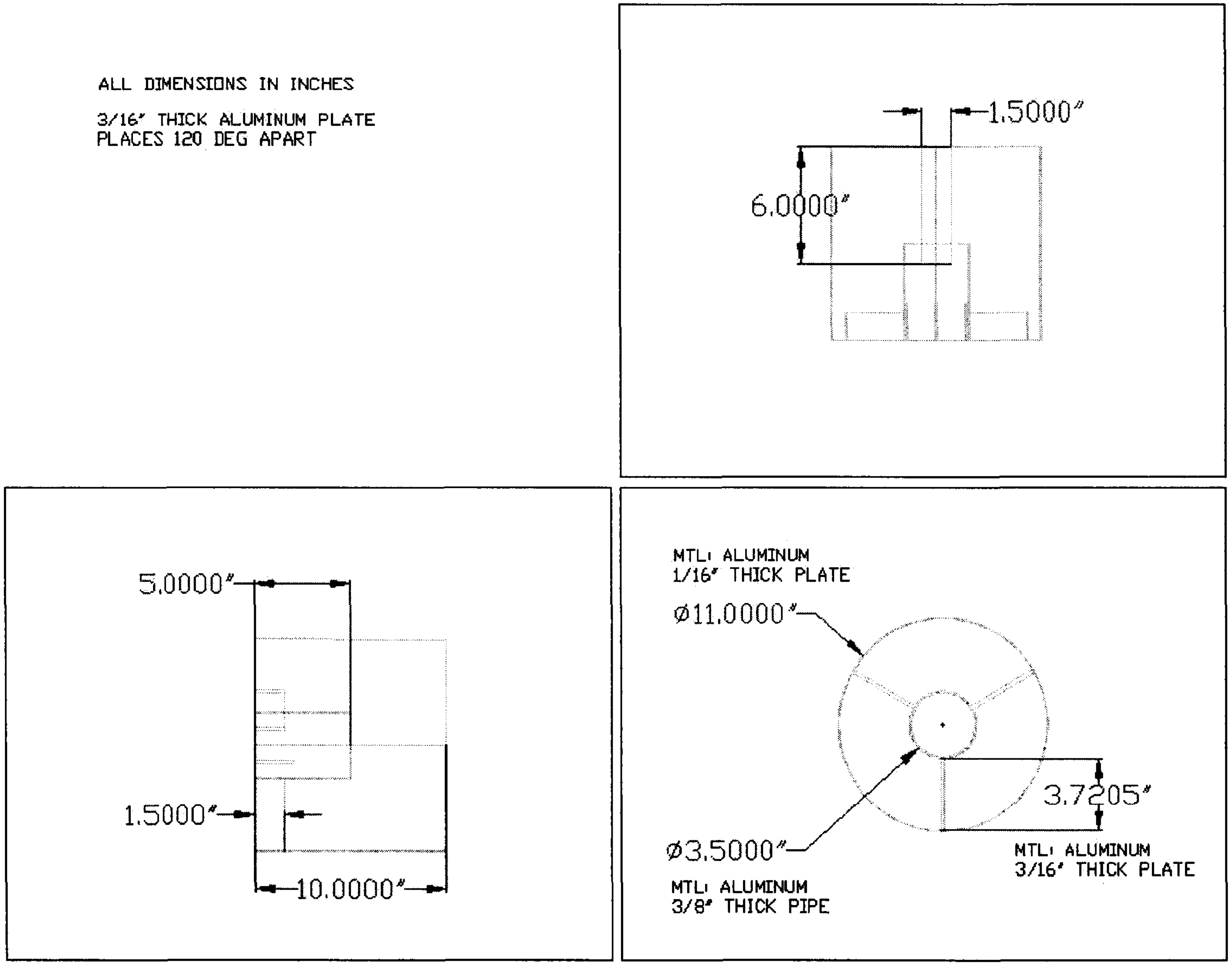
Controller Box

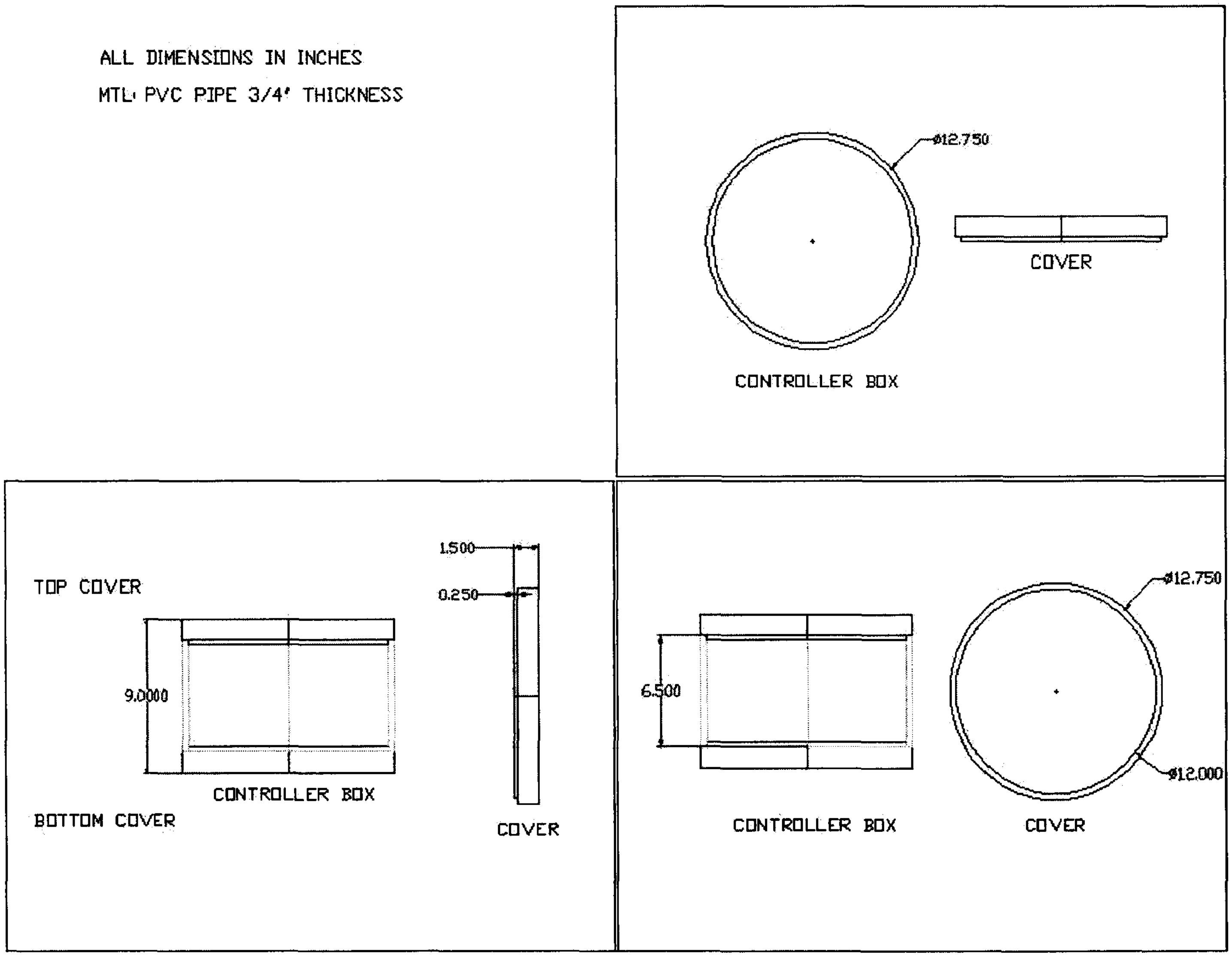
Drawing C Controller Box

Drawing A: Structure and Frame



Drawing B: Thruster Shroud

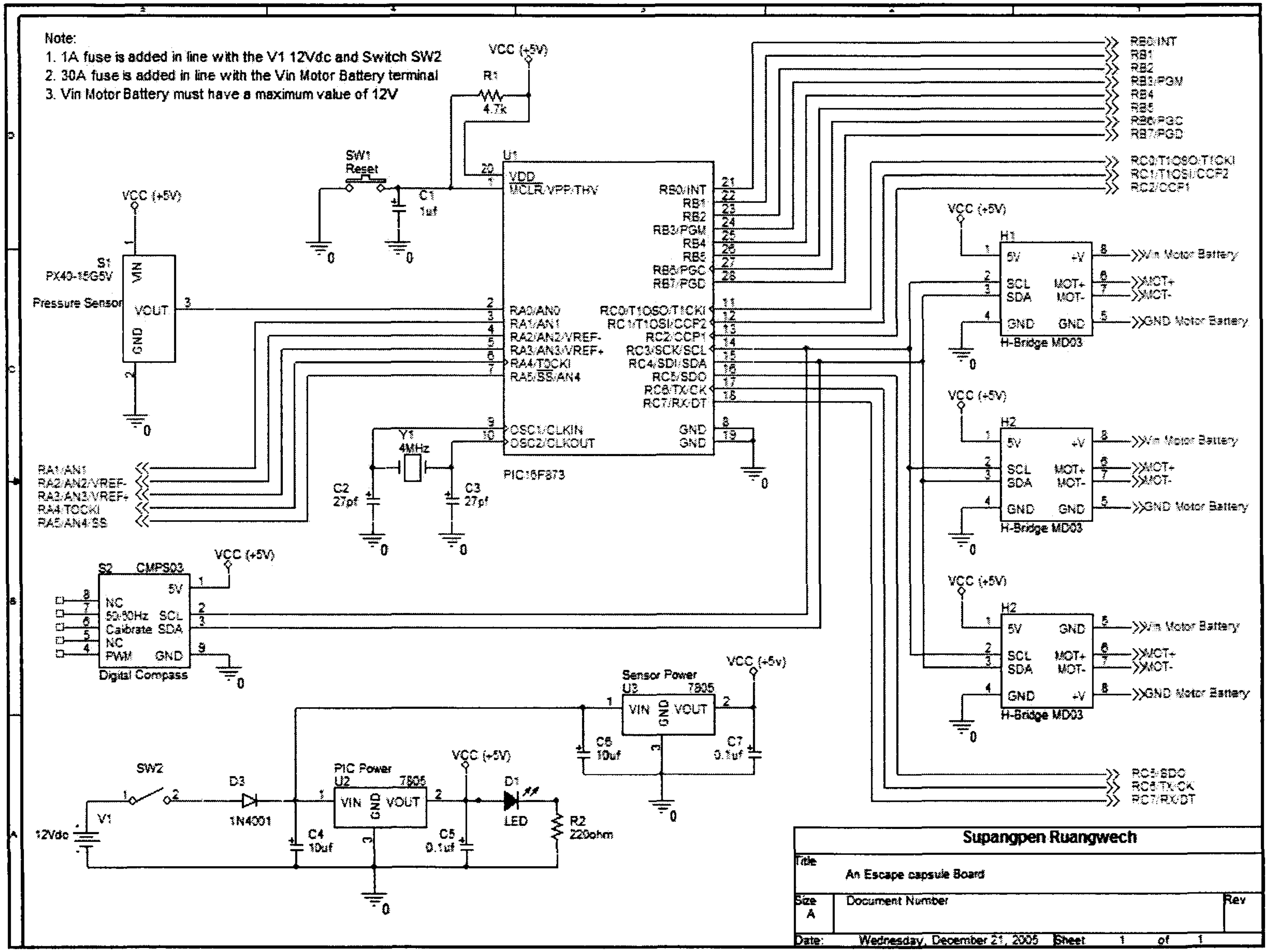




Drawing C: Controller Box

Appendix D

Circuit Diagram for Controller Box



Note:
1. 1A fuse is added in line with the V1 12Vdc and Switch SW2
2. 30A fuse is added in line with the Vin Motor Battery terminal
3. Vin Motor Battery must have a maximum value of 12V

Supangpen Ruangwech

Title
An Escape capsule Board

Size	Document Number	Rev
A		

Date: Wednesday, December 21, 2005 Sheet 1 of 1

Appendix E

PIC Code for Experiments

Peripheral Interface Control (PIC) C-programming code

for Depth Control Experiment

```
/*****************************************************************************************
*****************************************************************************************
                         AN ESCAPE CAPSULE FOR OFFSHORE STRUCTURE
                                  Using PID Control Strategy

                                   Author:  S.Ruangwech
                                            M.Hinchey
                                   Date: June 29th, 2005
*****************************************************************************************
*****************************************************************************************/
/*
This program was written to control the Submersible Escape Capsule in order that it will descend toward
the control depth and ascend back to water surface safely after a certain period of time. The pressure
transducer (OMEGA model: PX40-15G5V) is used to read the distance that the capsule's traveling to. This
pressure transducer has a calibration factor of 0.4 volt/meter. It can read a pressure from 0-15 psi.

During an operation, the Submersible Escape Capsule is programmed to descend toward the control depth
of 1.2 meters (or 100 in 10 bits) then ascend back to the water surface. An overall operation will last about
40 seconds.
*/

#include <16F873.h>
#fuses HS,NOWDT,NOPROTECT
#device ADC=10                                                        //Set up ADC = 10 bit
#use delay(clock=4000000)                                             //Set up clock = 4MHz
#use i2c(master, sda=PIN_C4, scl=PIN_C3, FAST, FORCE_HW)
#use rs232(BAUD=9600,XMIT=PIN_C6,RCV=PIN_C7,PARITY=N,BITS=8)  //Setup Serial Connection
                                                                      //via RS232
#org 0x0F00, 0x0FFF{}                                                 //Reserve Memory for
                                                                      //Bootloader

#opt 9
#include <stdlib.h>

/**********************************************/
/************** Variables Initialize **************/
/**********************************************/

/** Declaring variables for ThrusterCmd Function **/

enum ThrustUnit {Back, Left, Right} ThrustPos;          //Defining all thruster units
enum Dir {Up, Down, Forward, Reverse, Stop} TurnDir;    //Defining all possible capsule's movements

int ThrusterSpeed = 0;                                  //Representing thrusters' speed

int ThrusterAccel = 50;                                 //Representing thrusters' acceleration rate

/** Declaring variables for Heave motion control **/

unsigned long    ZeroDepth = 0,
                 CurrentD = 0;
```

```
signed long     ControlD = 100,     //Setup the control depth, 100 in 10 bit digital number = 1.2
                UpperBand = 40,     //Setup the upper band for the control depth is 0.5 m above the ControlDepth
                LowerBand = -40,    //Setup the upper band for the control depth is 0.5 m below the ControlDepth
                HDeadBand = 16,     //Setup the upper deadband at 20 cm above the ControlDepth
                LDeadBand = -16,    //Setup the upper deadband at 20 cm below the ControlDepth
                Error,
                P_Error;

float           Kp = 1.0,
                P_Controller,
                PWM_Signal= 0.0,
                AbsPWM = 0.0;

unsigned int8   int8_PWM;

/** Declaring variables for Counter **/

int8            Loop_Counter = 0,
                        Time_Counter = 0,
                        Heave_Counter = 0,
                        Up_Counter = 0;


/**********************************************/
/******** ThrusterCommand Function **************/
/**********************************************/

//Sending signal to H-bridge drivers via I2C bus

void ThrusterCmd(ThrustUnit ThrustPos, Dir TurnDir, int ThrusterSpeed, int ThrusterAccel)

{
        switch(ThrustPos)
        {
        case Back:    /**** Sending address for the Back Thruster (0xB4) ***/

                      /************** Setup Thruster's Speed ***************/
                                i2c_start();
                                i2c_write(0xB4);
                                delay_ms(20);
                                i2c_write(2);                   //Register Name   : Speed
                                                                //Register Address: 2
                                delay_ms(20);
                                i2c_write(ThrusterSpeed);
                                i2c_stop();

                                delay_ms(30);

                      /********* Setup Thruster's Accelerated Rate *********/
                                i2c_start();
                                i2c_write(0xB4);
                                delay_ms(20);
                                i2c_write(3);                   //Register Name   : Acceleration
```

```
                                                //Register Address: 3
                delay_ms(20);
                i2c_write(ThrusterAccel);
                i2c_stop();

                delay_ms(30);


        /*************** Issuing Turn Direction **************/
                i2c_start();
                i2c_write(0xB4);
                delay_ms(20);
                i2c_write(0);                   //Register Name   : Command
                                                //Register Address: 0
                delay_ms(20);
switch(TurnDir)
{
        case Up:        i2c_write(01);break;    //Ascend
        case Down:      i2c_write(02);break;    //Descend
        case Stop:      i2c_write(00);break;    //Stop
        default:        i2c_write(00);break;    //If wrong command were given, did nothing
}
i2c_stop();

break;

case Left:      /**** Sending address for the Left Thruster (0xB0) ***/

        /************** Setup Thruster's Speed ***************/
                i2c_start();
                i2c_write(0xB0);
                delay_ms(20);
                i2c_write(2);                   //Register Name   : Speed
                                                //Register Address: 2
                delay_ms(20);
                i2c_write(ThrusterSpeed);
                i2c_stop();

                delay_ms(30);

        /********* Setup Thruster's Accelerated Rate *********/
                i2c_start();
                i2c_write(0xB0);
                delay_ms(20);
                i2c_write(3);                   //Register Name   : Acceleration
                                                //Register Address: 3
                delay_ms(20);
                i2c_write(ThrusterAccel);
                i2c_stop();

                delay_ms(30);

        /*************** Issuing Turn Direction **************/
```

```
                    i2c_start();
                    i2c_write(0xB0);
                    delay_ms(20);
                    i2c_write(0);           //Register Name   : Command
                                            //Register Address: 0
                    delay_ms(20);
switch(TurnDir)
{
        case Forward:   i2c_write(01);break;    //Moving Forward
        case Reverse:   i2c_write(02);break;    //Moving Backward
        case Stop:      i2c_write(00);break;    //Stop
        default:        i2c_write(00);break;    //If wrong command were given, did nothing
}
i2c_stop();

break;

case Right:     /*** Sending address for the Right Thruster (0xB2) ***/

                /************** Setup Thruster's Speed ***************/
                    i2c_start();
                    i2c_write(0xB2);
                    delay_ms(20);
                    i2c_write(2);           //Register Name   : Speed
                                            //Register Address: 2
                    delay_ms(20);
                    i2c_write(ThrusterSpeed);
                    i2c_stop();

                    delay_ms(30);

                /********* Setup Thruster's Accelerated Rate *********/
                    i2c_start();
                    i2c_write(0xB2);
                    delay_ms(20);
                    i2c_write(3);           //Register Name   : Acceleration
                                            //Register Address: 3
                    delay_ms(20);
                    i2c_write(ThrusterAccel);
                    i2c_stop();

                    delay_ms(30);

                /*************** Issuing Turn Direction **************/
                    i2c_start();
                    i2c_write(0xB2);
                    delay_ms(20);
                    i2c_write(0);           //Register Name   : Command
                                            //Register Address: 0
                    delay_ms(20);
switch(TurnDir)
{
        case Forward:   i2c_write(01);break;    //Moving Forward
```

```
                case Reverse:     i2c_write(02);break;       //Moving Backward
                case Stop:        i2c_write(00);break;       //Stop
                default:          i2c_write(00);break;       //If wrong command were given, did nothing
        }
        i2c_stop();

        break;

        }       //Ending switch statements

}

/***********************************************/
/**************** Start Main Program *************/
/***********************************************/

void main()

{       /*** STARTING MAIN LOOP ***/

setup_adc_ports(ALL_ANALOG);                            //Initialize A/D ports
setup_adc(ADC_CLOCK_DIV_8);                             //Setup clock for A/D conversion

delay_ms(5000);

/************* Starting Operation *************/

        ThrusterCmd(Back, Stop, 0, 50);
        ThrusterCmd(Left, Stop, 0, 50);
        ThrusterCmd(Right, Stop, 0, 50);
        delay_ms(20);

set_adc_channel(0);                     //Set ADC channel RA0 for pressure sensor
delay_us(21);
ZeroDepth = read_adc();                 //Read ZeroDepth

delay_ms(15000);
delay_ms(20000);

/***** Descending toward The Control Depth ****/

while(Heave_Counter<=200)               //Total of 40s for Heave Motion before ascending to the water surface
{       //Starting while loop

        /**********************/
        /**** Heave Motion ****/
        /**********************/

        set_adc_channel(0);             //Setup RA0 to keep reading CurrentDepth using pressure sensor
        delay_us(21);
        CurrentD = read_adc();          //Read CurrentDepth

        CurrentD=CurrentD-ZeroDepth;
```

```
if(CurrentD<0)
{
CurrentD=0;
}

Error=ControlD-CurrentD;

/****** Activating PID Controller when the capsule is within the control depth band ******/

if((Error<=UpperBand)&&(Error>=LowerBand))
//The capsule is within the control depth band

{

        /*********** Calculating PWM Signal **********/

        P_Error = Error;
        P_Controller = Kp*abs(P_Error);
        PWM_Signal = P_Controller;
        AbsPWM = abs(PWM_Signal);

        //Set the upper and the lower limit of the H-bridge input signal
        if(AbsPWM<75.0)
        {
                AbsPWM = 75.0;
        }
        else
        {
                if(AbsPWM>200.0)
                {
                        AbsPWM = 200.0;
                }
        }
        int8_PWM = (int8)AbsPWM;

        if((Error<=HDeadBand)&&(Error>=LDeadBand) )
        //The capsule is in the deadband for the control depth
        {
                ThrusterCmd(Back, Stop, 0, 50);
                delay_ms(20);
        }

        else
        {
                if(Error>HDeadBand)
                {
                        ThrusterCmd(Back, Down, int8_PWM, 50);
                        delay_ms(20);
                }

                else
                {
                if(Error<LDeadBand)
```

```
                        {
                                ThrusterCmd(Back, Up, int8_PWM, 50);
                                delay_ms(20);
                        }
                    }
                }

            }

            /***** Deactivating PID Controller when the capsule is outside the control depth band ****/

            else

                /***** Setup constant rate for PWM_Signal ****/
            {

                if(Error>UpperBand)
                //The capsule is above the upper band
                //Descend toward the control depth band at constant speed
                {
                    ThrusterCmd(Back, Down, 200, 50);
                    delay_ms(20);
                }

                else /*** Error<LowerBand ***/
                //The capsule is under the lower band
                //Ascend toward the control depth band at constant speed
                {
                if(Error<LowerBand)
                    {
                        ThrusterCmd(Back, Up, 200, 50);
                        delay_ms(20);
                    }
                }
            }

                Heave_Counter = Heave_Counter + 1;

            Loop_Counter = Loop_Counter + 1;
            if(Loop_Counter=1)
            {
                Time_Counter=Time_Counter+1;
                delay_ms(20);
                printf("\r\n %U", Time_Counter);
                printf("\t %lu", CurrentD,"\n\r");
                Loop_Counter = 0;
            }

        }       //Ending while loop

                ThrusterCmd(Right, Stop, 0, 50);
                ThrusterCmd(Left, Stop, 0, 50);
                ThrusterCmd(Back, Stop, 0, 50);
                delay_ms(20);
/*** Ascending toward water surface ***/

while(Up_Counter<25)
{

                ThrusterCmd(Right, Stop, 0, 50);
                ThrusterCmd(Left, Stop, 0, 50);
                ThrusterCmd(Back, Up, 200, 50);
                delay_ms(20);

                Up_Counter = Up_Counter + 1;

}

                ThrusterCmd(Back, Stop, 0, 50);
                delay_ms(20);

        break;

}       /*** ENDING MAIN LOOP ***/

/**************************************************/
/*************** END OF PROGRAM ****************/
/**************************************************/
```

Peripheral Interface Control (PIC) C-programming code

for Direction Control Experiment

```
/*************************************************************************************
**************************************************************************************
                    AN ESCAPE CAPSULE FOR OFFSHORE STRUCTURE
                           Using PID Control Strategy

                              Author: S.Ruangwech
                                      M.Hinchey
                              Date: June 29th, 2005
**************************************************************************************
*************************************************************************************/
/*
This program was written to control the Submersible Escape Capsule in order that it will keep itself under it
control path. The digital compass (OMEGA model: CMDS03) is used for determining the direction that the
capsule is facing to. The output from the digital compass will be any number range from 0-3599 (16-bits
data). This number will be directly sent to the PIC; and will be used to calculate the command response in
which will be used for corrective action.

During an operation, the Submersible Escape Capsule is programmed to travel from the edge of the Deep
Water Tank. Then turn itself 180 degree and travel back to where it has been released. An overall operation
will last about 80 seconds.
*/

#include <16F873.h>
#fuses HS,NOWDT,NOPROTECT
#device ADC=10                                                         //Set up ADC = 10 bit
#use delay(clock=4000000)                                              //Set up clock = 4MHz
#use i2c(master, sda=PIN_C4, scl=PIN_C3, FAST, FORCE_HW)
#use rs232(BAUD=9600,XMIT=PIN_C6,RCV=PIN_C7,PARITY=N,BITS=8)  //Setup Serial Connection
                                                                       //via RS232
#org 0x0F00, 0x0FFF{}                                                  //Reserve Memory for
                                                                       //Bootloader
#opt 9
#include <stdlib.h>

/*********************************************/
/************* Variables Initialize **************/
/*********************************************/

/** Declaring variables for DCompass Function **/

unsigned int8       HighByte,
                    LowByte;

unsigned long       CompassBearing;

float               YawAngle;

/** Declaring variables for ThrusterCmd Function **/

enum ThrustUnit {Back, Left, Right} ThrustPos;         //Defining all thruster units
enum Dir {Up, Down, Forward, Reverse, Stop} TurnDir;   //Defining all possible capsule's movements
int ThrusterSpeed = 0;                                 //Representing thrusters' speed
int ThrusterAccel = 50;                                //Representing thrusters' acceleration rate
```

```
/** Declaring variables for Heave motion control **/

unsigned long      ZeroDepth = 0;

/** Declaring variables for Surge motion control **/

float              ControlYaw = -140.0,     //Setup a set point for Yaw Angle
                   RightDBand = 15.0,       //Setup the right deadband at 5 degree to the right of the setpoint
                   LeftDBand = -15.0,       //Setup the left deadband at 5 degree to the left of the setpoint
                   Yaw_Kp = 0.5,
                   CurrentYaw,
                   Yaw_Error,
                   Yaw_Controller,
                   Yaw_PWM_Signal,
                   Yaw_AbsPWM;

unsigned int8      int8_YawPWM,
                   PWM1,
                   PWM2;

/** Declaring variables for Counter **/

int8               Loop_Counter = 0,
                   Time_Counter = 0,
                   Heave_Counter = 0,
                   Yaw_Counter = 0,
                   Up_Counter = 0;

/**********************************************/
/*************** DCompass Function **************/
/**********************************************/

//Recieving Signal from a Digital Compass via I2C bus

float DCompass()
{

i2c_start();
i2c_write(0xC0);
delay_us(50);
i2c_write(0x02);                  //Register Address  :2
                                  //Register Function :Compass Bearing as a word(first byte)

delay_us(50);
i2c_write(0x03);                  //Register Address  :3
                                  //Register Function :Compass Bearing as a word(second byte)

delay_us(50);

i2c_start();                      //Starting to read signal for the digital compass
i2c_write(0xC1);
HighByte=i2c_read(1);             //Reading HighByte for compass bearing
LowByte=i2c_read(0);              //Reading LowByte for compass bearing
i2c_stop();
```

```
/** Combining two 8-bit data to get 16-bit result for Compass Bearing **/

CompassBearing = make16(HighByte, LowByte);

YawAngle = CompassBearing*0.1;

if((YawAngle>=0.0)&&(YawAngle<=180.0))

        {
        //NOTE:   if YawAngle < 0, the capsule is on the left of 0 degree
        //        if YawAngle > 0, the capsule is on the left of 0 degree

        YawAngle = YawAngle;

        }

else
        {
        if((YawAngle>180.0)&&(YawAngle<=360.0))
                {
                        YawAngle = YawAngle-360.0;
                }
        }

return YawAngle;

}

/*********************************************/
/******** ThrusterCommand Function **************/
/*********************************************/

//Sending signal to H-bridge drivers via I2C bus

void ThrusterCmd(ThrustUnit ThrustPos, Dir TurnDir, int ThrusterSpeed, int ThrusterAccel)

{
        switch(ThrustPos)
        {
        case Back:   /**** Sending address for the Back Thruster (0xB4) ***/

                        /************** Setup Thruster's Speed ***************/
                                i2c_start();
                                i2c_write(0xB4);
                                delay_ms(20);
                                i2c_write(2);            //Register Name   : Speed
                                                         //Register Address: 2
                                delay_ms(20);
                                i2c_write(ThrusterSpeed);
                                i2c_stop();

                                delay_ms(30);
                /********* Setup Thruster's Accelerated Rate *********/
                        i2c_start();
                        i2c_write(0xB4);
                        delay_ms(20);
                        i2c_write(3);                   //Register Name   : Acceleration
                                                        //Register Address: 3
                        delay_ms(20);
                        i2c_write(ThrusterAccel);
                        i2c_stop();

                        delay_ms(30);


                  /*************** Issuing Turn Direction **************/
                        i2c_start();
                        i2c_write(0xB4);
                        delay_ms(20);
                        i2c_write(0);                   //Register Name   : Command
                                                        //Register Address: 0
                        delay_ms(20);
switch(TurnDir)
{
        case Up:        i2c_write(01);break;            //Ascend
        case Down:      i2c_write(02);break;            //Descend
        case Stop:      i2c_write(00);break;            //Stop
        default:        i2c_write(00);break;            //If wrong command were given, did nothing
}
i2c_stop();

break;

case Left:        /**** Sending address for the Left Thruster (0xB0) ***/

                  /************** Setup Thruster's Speed ***************/
                        i2c_start();
                        i2c_write(0xB0);
                        delay_ms(20);
                        i2c_write(2);                   //Register Name   : Speed
                                                        //Register Address: 2
                        delay_ms(20);
                        i2c_write(ThrusterSpeed);
                        i2c_stop();

                        delay_ms(30);

                  /********* Setup Thruster's Accelerated Rate *********/
                        i2c_start();
                        i2c_write(0xB0);
                        delay_ms(20);
                        i2c_write(3);                   //Register Name   : Acceleration
                                                        //Register Address: 3
                        delay_ms(20);
                        i2c_write(ThrusterAccel);
```

```
                    i2c_stop();

                    delay_ms(30);

          /*************** Issuing Turn Direction **************/
                    i2c_start();
                    i2c_write(0xB0);
                    delay_ms(20);
                    i2c_write(0);           //Register Name   : Command
                                            //Register Address: 0
                    delay_ms(20);
switch(TurnDir)
{
     case Forward:  i2c_write(01);break;    //Moving Forward
     case Reverse:  i2c_write(02);break;    //Moving Backward
     case Stop:     i2c_write(00);break;    //Stop
     default:       i2c_write(00);break;    //If wrong command were given, did nothing
}
i2c_stop();

break;

case Right:    /*** Sending address for the Right Thruster (0xB2) ***/

          /************** Setup Thruster's Speed ***************/
                    i2c_start();
                    i2c_write(0xB2);
                    delay_ms(20);
                    i2c_write(2);           //Register Name   : Speed
                                            //Register Address: 2
                    delay_ms(20);
                    i2c_write(ThrusterSpeed);
                    i2c_stop();

                    delay_ms(30);

          /********* Setup Thruster's Accelerated Rate *********/
                    i2c_start();
                    i2c_write(0xB2);
                    delay_ms(20);
                    i2c_write(3);           //Register Name   : Acceleration
                                            //Register Address: 3
                    delay_ms(20);
                    i2c_write(ThrusterAccel);
                    i2c_stop();

                    delay_ms(30);

          /*************** Issuing Turn Direction **************/
                    i2c_start();
                    i2c_write(0xB2);
                    delay_ms(20);
                    i2c_write(0);           //Register Name   : Command
```

```
                                                                    //Register Address: 0
                                     delay_ms(20);
        switch(TurnDir)
        {
                case Forward:        i2c_write(01);break;           //Moving Forward
                case Reverse:        i2c_write(02);break;           //Moving Backward
                case Stop:           i2c_write(00);break;           //Stop
                default:             i2c_write(00);break;           //If wrong command were given, did nothing
        }
        i2c_stop();

        break;

        }       //Ending switch statements

}

/*********************************************/
/*************** Start Main Program *************/
/*********************************************/

void main()

{       /*** STARTING MAIN LOOP ***/

setup_adc_ports(ALL_ANALOG);                                        //Initialize A/D ports
setup_adc(ADC_CLOCK_DIV_8);                                         //Setup clock for A/D conversion

delay_ms(5000);

/************* Starting Operation *************/

        ThrusterCmd(Back, Stop, 0, 50);
        ThrusterCmd(Left, Stop, 0, 50);
        ThrusterCmd(Right, Stop, 0, 50);
        delay_ms(20);

set_adc_channel(0);                         //Set ADC channel RA0 for pressure sensor
delay_us(21);
ZeroDepth = read_adc();                     //Read ZeroDepth

delay_ms(15000);
delay_ms(20000);

/***** Moving Toward The Control Direction ****/

while(Heave_Counter<=200)                   //An overall operation takes about 80 seconds to complete
{       //Starting while loop

        if(Yaw_Counter>100)
        {
        ControlYaw = 90.0;
        }
```

```
/**********************/
/**** Yaw and Surge  ****/
/**********************/

CurrentYaw = DCompass();         //Measuring Current YawAngle of the capsule

Yaw_Error= ControlYaw-CurrentYaw;

        /*********** Calculating PWM Signal **********/

        Yaw_Controller = Yaw_Kp*abs(Yaw_Error);
        Yaw_PWM_Signal = Yaw_Controller;
        Yaw_AbsPWM = abs(Yaw_PWM_Signal);
        int8_YawPWM = (int8)Yaw_AbsPWM;

        PWM1=200+int8_YawPWM;
        if(PWM1>210)
        {
                PWM1=210;
        }

        PWM2=200-int8_YawPWM;
        if(PWM2<190)
        {
                PWM2=190;
        }

        if((Yaw_Error<=RightDBand)&&(Yaw_Error>=LeftDBand) )
        //The capsule is in the controller deadband
        {

                ThrusterCmd(Left, Stop,0, 50);
                ThrusterCmd(Right, Stop, 0, 50);
                delay_ms(20);
        }

        else
        {
                        if(Yaw_Error<0.0)
                        {
                                ThrusterCmd(Right, Forward, PWM1, 50);
                                ThrusterCmd(Left, Forward, PWM2, 50);
                                delay_ms(20);
                        }
                        else
                        {
                                if(Yaw_Error>0.0)
                                {
                                        ThrusterCmd(Left, Forward, PWM1, 50);
                                        ThrusterCmd(Right, Forward, PWM2, 50);
                                        delay_ms(20);
                                }
                        }
                }

                Yaw_Counter = Yaw_Counter + 1;
                Heave_Counter = Heave_Counter + 1;

        Loop_Counter = Loop_Counter + 1;
        if(Loop_Counter=1)
        {
                Time_Counter=Time_Counter+1;
                delay_ms(20);
                printf("\r\n %U", Time_Counter);
                printf("\t %f", CurrentYaw,"\n\r");
                Loop_Counter = 0;

        }

}       //Ending while loop

                ThrusterCmd(Right, Stop, 0, 50);
                ThrusterCmd(Left, Stop, 0, 50);
                ThrusterCmd(Back, Stop, 0, 50);
                delay_ms(20);

/*** Ascending toward water surface ***/

while(Up_Counter<25)

{
                ThrusterCmd(Right, Stop, 0, 50);
                ThrusterCmd(Left, Stop, 0, 50);
                ThrusterCmd(Back, Stop, 0, 50);
                delay_ms(20);

                Up_Counter = Up_Counter + 1;

}

                ThrusterCmd(Back, Stop, 0, 50);
                delay_ms(20);

        break;

}       /*** ENDING MAIN LOOP ***/

/**************************************************/
/*************** END OF PROGRAM ****************/
/**************************************************/
```

Peripheral Interface Control (PIC) C-programming code

for Complete Mission Experiment

```
/*************************************************************************************
**************************************************************************************
                    AN ESCAPE CAPSULE FOR OFFSHORE STRUCTURE
                            Using PID Control Strategy

                                  Author:  S.Ruangwech
                                           M.Hinchey
                                  Date: June 29th, 2005
**************************************************************************************
*************************************************************************************/


#include <16F873.h>
#fuses HS,NOWDT,NOPROTECT
#device ADC=10                                                        //Set up ADC = 10 bit
#use delay(clock=4000000)                                             //Set up clock = 4MHz
#use i2c(master, sda=PIN_C4, scl=PIN_C3, FAST, FORCE_HW)
#use rs232(BAUD=9600,XMIT=PIN_C6,RCV=PIN_C7,PARITY=N,BITS=8)  //Setup Serial Connection
                                                              //via RS232
#org 0x0F00, 0x0FFF{}                                                 //Reserve Memory for
                                                                      //Bootloader
#opt 9
#include <stdlib.h>

/**********************************************/
/************** Variables Initialize ***************/
/**********************************************/

/** Declaring variables for DCompass Function **/

unsigned int8     HighByte,
                  LowByte;

unsigned long     CompassBearing;

float             YawAngle;

/** Declaring variables for ThrusterCmd Function **/

enum ThrustUnit {Back, Left, Right} ThrustPos;          //Defining all thruster units
enum Dir {Up, Down, Forward, Reverse, Stop} TurnDir;    //Defining all possible capsule's movements
int ThrusterSpeed = 0;                                  //Representing thrusters' speed
int ThrusterAccel = 50;                                 //Representing thrusters' acceleration rate

/** Declaring variables for Heave motion control **/

unsigned long     ZeroDepth = 0,
                  CurrentD = 0;

signed long       ControlD = 100,    //Setup the control depth, 100 in 10 bit digital number = 1.2
                  UpperBand = 40,    //Setup the upper band for the control depth is 0.5 m above the ControlDepth
                  LowerBand = -40,   //Setup the upper band for the control depth is 0.5 m below the ControlDepth
                  HDeadBand = 16,    //Setup the upper deadband at 20 cm above the ControlDepth
                  LDeadBand = -16,   //Setup the upper deadband at 20 cm below the ControlDepth
```

```
                Error,
                P_Error;

float           Kp = 1.0,
                P_Controller,
                PWM_Signal= 0.0,
                AbsPWM = 0.0;

unsigned int8   int8_PWM;

/** Declaring variables for Surge motion control **/

float           ControlYaw = -140.0,    //Setup a set point for Yaw Angle
                RightDBand = 15.0,      //Setup the right deadband at 5 degree to the right of the setpoint
                LeftDBand = -15.0,      //Setup the left deadband at 5 degree to the left of the setpoint
                Yaw_Kp = 0.5,
                CurrentYaw,
                Yaw_Error,
                Yaw_Controller,
                Yaw_PWM_Signal,
                Yaw_AbsPWM;

unsigned int8   int8_YawPWM,
                PWM1,
                PWM2;

/** Declaring variables for Counter **/

int8            Heave_Counter = 0,
                Yaw_Counter = 0,
                Up_Counter = 0;

/***********************************************/
/*************** DCompass Function **************/
/***********************************************/

//Recieving Signal from a Digital Compass via I2C bus

float DCompass()
{

i2c_start();
i2c_write(0xC0);
delay_us(50);
i2c_write(0x02);                //Register Address  :2
                                //Register Function :Compass Bearing as a word(first byte)
delay_us(50);
i2c_write(0x03);                //Register Address  :3
                                //Register Function :Compass Bearing as a word(second byte)
delay_us(50);

i2c_start();                    //Starting to read signal for the digital compass
i2c_write(0xC1);
```

```
HighByte=i2c_read(1);              //Reading HighByte for compass bearing
LowByte=i2c_read(0);               //Reading LowByte for compass bearing
i2c_stop();

/** Combining two 8-bit data to get 16-bit result for Compass Bearing **/

CompassBearing = make16(HighByte, LowByte);

YawAngle = CompassBearing*0.1;

if((YawAngle>=0.0)&&(YawAngle<=180.0))

        {
        //NOTE:   if YawAngle < 0, the capsule is on the left of 0 degree
        //        if YawAngle > 0, the capsule is on the left of 0 degree

        YawAngle = YawAngle;

        }

else
        {
        if((YawAngle>180.0)&&(YawAngle<=360.0))
                {
                        YawAngle = YawAngle-360.0;
                }
        }

return YawAngle;

}

/***********************************************/
/******** ThrusterCommand Function **************/
/***********************************************/

//Sending signal to H-bridge drivers via I2C bus

void ThrusterCmd(ThrustUnit ThrustPos, Dir TurnDir, int ThrusterSpeed, int ThrusterAccel)

{
        switch(ThrustPos)
        {
        case Back:   /**** Sending address for the Back Thruster (0xB4) ***/

                     /************** Setup Thruster's Speed ***************/
                                i2c_start();
                                i2c_write(0xB4);
                                delay_ms(20);
                                i2c_write(2);             //Register Name   : Speed
                                                          //Register Address: 2
                                delay_ms(20);
                                i2c_write(ThrusterSpeed);
```

```
            i2c_stop();

            delay_ms(30);

      /********* Setup Thruster's Accelerated Rate *********/
            i2c_start();
            i2c_write(0xB4);
            delay_ms(20);
            i2c_write(3);                   //Register Name   : Acceleration
                                            //Register Address: 3
            delay_ms(20);
            i2c_write(ThrusterAccel);
            i2c_stop();

            delay_ms(30);


        /*************** Issuing Turn Direction **************/
            i2c_start();
            i2c_write(0xB4);
            delay_ms(20);
            i2c_write(0);                   //Register Name   : Command
                                            //Register Address: 0
            delay_ms(20);
switch(TurnDir)
{
    case Up:        i2c_write(01);break;    //Ascend
    case Down:      i2c_write(02);break;    //Descend
    case Stop:      i2c_write(00);break;    //Stop
    default:        i2c_write(00);break;    //If wrong command were given, did nothing
}
i2c_stop();

break;

case Left:      /**** Sending address for the Left Thruster (0xB0) ***/

        /************** Setup Thruster's Speed ***************/
            i2c_start();
            i2c_write(0xB0);
            delay_ms(20);
            i2c_write(2);                   //Register Name   : Speed
                                            //Register Address: 2
            delay_ms(20);
            i2c_write(ThrusterSpeed);
            i2c_stop();

            delay_ms(30);

        /********* Setup Thruster's Accelerated Rate *********/
            i2c_start();
            i2c_write(0xB0);
            delay_ms(20);
```

```
                        i2c_write(3);           //Register Name  : Acceleration
                                                //Register Address: 3
                        delay_ms(20);
                        i2c_write(ThrusterAccel);
                        i2c_stop();

                        delay_ms(30);

                /*************** Issuing Turn Direction **************/
                        i2c_start();
                        i2c_write(0xB0);
                        delay_ms(20);
                        i2c_write(0);           //Register Name  : Command
                                                //Register Address: 0
                        delay_ms(20);
switch(TurnDir)
{
        case Forward:   i2c_write(01);break;    //Moving Forward
        case Reverse:   i2c_write(02);break;    //Moving Backward
        case Stop:      i2c_write(00);break;    //Stop
        default:        i2c_write(00);break;    //If wrong command were given, did nothing
}
i2c_stop();

break;

case Right:     /*** Sending address for the Right Thruster (0xB2) ***/

                /************** Setup Thruster's Speed ***************/
                        i2c_start();
                        i2c_write(0xB2);
                        delay_ms(20);
                        i2c_write(2);           //Register Name  : Speed
                                                //Register Address: 2
                        delay_ms(20);
                        i2c_write(ThrusterSpeed);
                        i2c_stop();

                        delay_ms(30);

                /********* Setup Thruster's Accelerated Rate *********/
                        i2c_start();
                        i2c_write(0xB2);
                        delay_ms(20);
                        i2c_write(3);           //Register Name  : Acceleration
                                                //Register Address: 3
                        delay_ms(20);
                        i2c_write(ThrusterAccel);
                        i2c_stop();

                        delay_ms(30);

                /*************** Issuing Turn Direction **************/
```

```
                                i2c_start();
                                i2c_write(0xB2);
                                delay_ms(20);
                                i2c_write(0);           //Register Name   : Command
                                                        //Register Address: 0
                                delay_ms(20);
        switch(TurnDir)
        {
                case Forward:   i2c_write(01);break;    //Moving Forward
                case Reverse:   i2c_write(02);break;    //Moving Backward
                case Stop:      i2c_write(00);break;    //Stop
                default:        i2c_write(00);break;    //If wrong command were given, did nothing
        }
        i2c_stop();

        break;

        }       //Ending switch statements

}

/***********************************************/
/**************** Start Main Program **************/
/***********************************************/

void main()

{       /*** STARTING MAIN LOOP ***/

setup_adc_ports(ALL_ANALOG);                            //Initialize A/D ports
setup_adc(ADC_CLOCK_DIV_8);                             //Setup clock for A/D conversion

delay_ms(5000);

/************* Starting Operation *************/

        ThrusterCmd(Back, Stop, 0, 50);
        ThrusterCmd(Left, Stop, 0, 50);
        ThrusterCmd(Right, Stop, 0, 50);
        delay_ms(20);

set_adc_channel(0);                     //Set ADC channel RA0 for pressure sensor
delay_us(21);
ZeroDepth = read_adc();                 //Read ZeroDepth

delay_ms(15000);
delay_ms(20000);

/***** Descending toward The Control Depth ****/

while(Heave_Counter<=200)       //Total of 120s for Heave Motion before ascending to the water surface
{       //Starting while loop
```

```
if(Yaw_Counter>100)
{
ControlYaw = 90.0;
}

/***********************/
/**** Yaw and Surge ***/
/***********************/
CurrentYaw = DCompass();          //Measuring Current YawAngle of the capsule

Yaw_Error= ControlYaw-CurrentYaw;




        /*********** Calculating PWM Signal **********/

        Yaw_Controller = Yaw_Kp*abs(Yaw_Error);

        Yaw_PWM_Signal = Yaw_Controller;

        Yaw_AbsPWM = abs(Yaw_PWM_Signal);

        int8_YawPWM = (int8)Yaw_AbsPWM;

        PWM1=200+int8_YawPWM;
        if(PWM1>210)
        {
                PWM1=210;
        }

        PWM2=200-int8_YawPWM;
        if(PWM2<190)
        {
                PWM2=190;
        }

        if((Yaw_Error<=RightDBand)&&(Yaw_Error>=LeftDBand) )
        //The capsule is in the controller deadband
        {

                ThrusterCmd(Left, Forward,200, 50);
                ThrusterCmd(Right, Forward, 200, 50);
                delay_ms(20);
        }

        else
        {
                        if(Yaw_Error<0.0)
                        {
```

```
                        ThrusterCmd(Right, Forward, PWM1, 50);
                        ThrusterCmd(Left, Forward, PWM2, 50);
                        delay_ms(20);
                    }

                    else
                    {
                        if(Yaw_Error>0.0)
                        {
                            ThrusterCmd(Left, Forward, PWM1, 50);
                            ThrusterCmd(Right, Forward, PWM2, 50);
                            delay_ms(20);
                        }
                    }
    }

    Yaw_Counter = Yaw_Counter + 1;



/***********************/
/**** Heave Motion ****/
/***********************/

set_adc_channel(0);             //Setup RA0 to keep reading CurrentDepth using pressure sensor
delay_us(21);
CurrentD = read_adc();          //Read CurrentDepth

CurrentD=CurrentD-ZeroDepth;
if(CurrentD<0)
{
CurrentD=0;
}

Error=ControlD-CurrentD;

/****** Activating PID Controller when the capsule is within the control depth band ******/

if((Error<=UpperBand)&&(Error>=LowerBand))
//The capsule is within the control depth band

{

    /*********** Calculating PWM Signal **********/

    P_Error = Error;
    P_Controller = Kp*abs(P_Error);
    PWM_Signal = P_Controller;
    AbsPWM = abs(PWM_Signal);

    //Set the upper and the lower limit of the H-bridge input signal
    if(AbsPWM<75.0)
    {
        AbsPWM = 75.0;
    }
    else
    {
        if(AbsPWM>200.0)
        {
            AbsPWM = 200.0;

        }
    }
    int8_PWM = (int8)AbsPWM;

    if((Error<=HDeadBand)&&(Error>=LDeadBand) )
    //The capsule is in the deadband for the control depth
    {
        ThrusterCmd(Back, Stop, 0, 50);
        delay_ms(20);
    }




    else
    {
        if(Error>HDeadBand)
        {
            ThrusterCmd(Back, Down, int8_PWM, 50);
            delay_ms(20);
        }

        else
        {
        if(Error<LDeadBand)
            {
                ThrusterCmd(Back, Up, int8_PWM, 50);
                delay_ms(20);
            }
        }
    }

}

/***** Deactivating PID Controller when the capsule is outside the control depth band ****/

else
    /***** Setup constant rate for PWM_Signal ****/
{

    if(Error>UpperBand)
    //The capsule is above the upper band
    //Descend toward the control depth band at constant speed
```

```
        {
            ThrusterCmd(Back, Down, 200, 50);
            delay_ms(20);
        }

        else /*** Error<LowerBand ***/
        //The capsule is under the lower band
        //Ascend toward the control depth band at constant speed
        {
        if(Error<LowerBand)
            {
                ThrusterCmd(Back, Up, 200, 50);
                delay_ms(20);
            }
        }
    }

        Heave_Counter = Heave_Counter + 1;

}       //Ending while loop

        ThrusterCmd(Right, Stop, 0, 50);
        ThrusterCmd(Left, Stop, 0, 50);
        ThrusterCmd(Back, Stop, 0, 50);
        delay_ms(20);


/*** Ascending toward water surface ***/

while(Up_Counter<25)
{

        ThrusterCmd(Right, Stop, 0, 50);
        ThrusterCmd(Left, Stop, 0, 50);
        ThrusterCmd(Back, Up, 200, 50);
        delay_ms(20);

        Up_Counter = Up_Counter + 1;

}


        ThrusterCmd(Back, Stop, 0, 50);
        delay_ms(20);

    break;

}       /*** ENDING MAIN LOOP ***/

/**************************************************/
/*************** END OF PROGRAM  ****************/
/**************************************************/
```

Appendix F

Specification Sheets for Various Components

Peripheral Interface Control (PIC) Microcontroller – PIC16F873

Specifications Sheets

MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

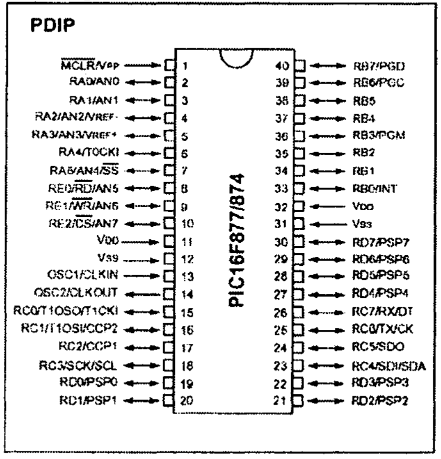
Devices Included in this Data Sheet:

- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 μA typical @ 3V, 32 kHz
  - < 1 μA typical standby current

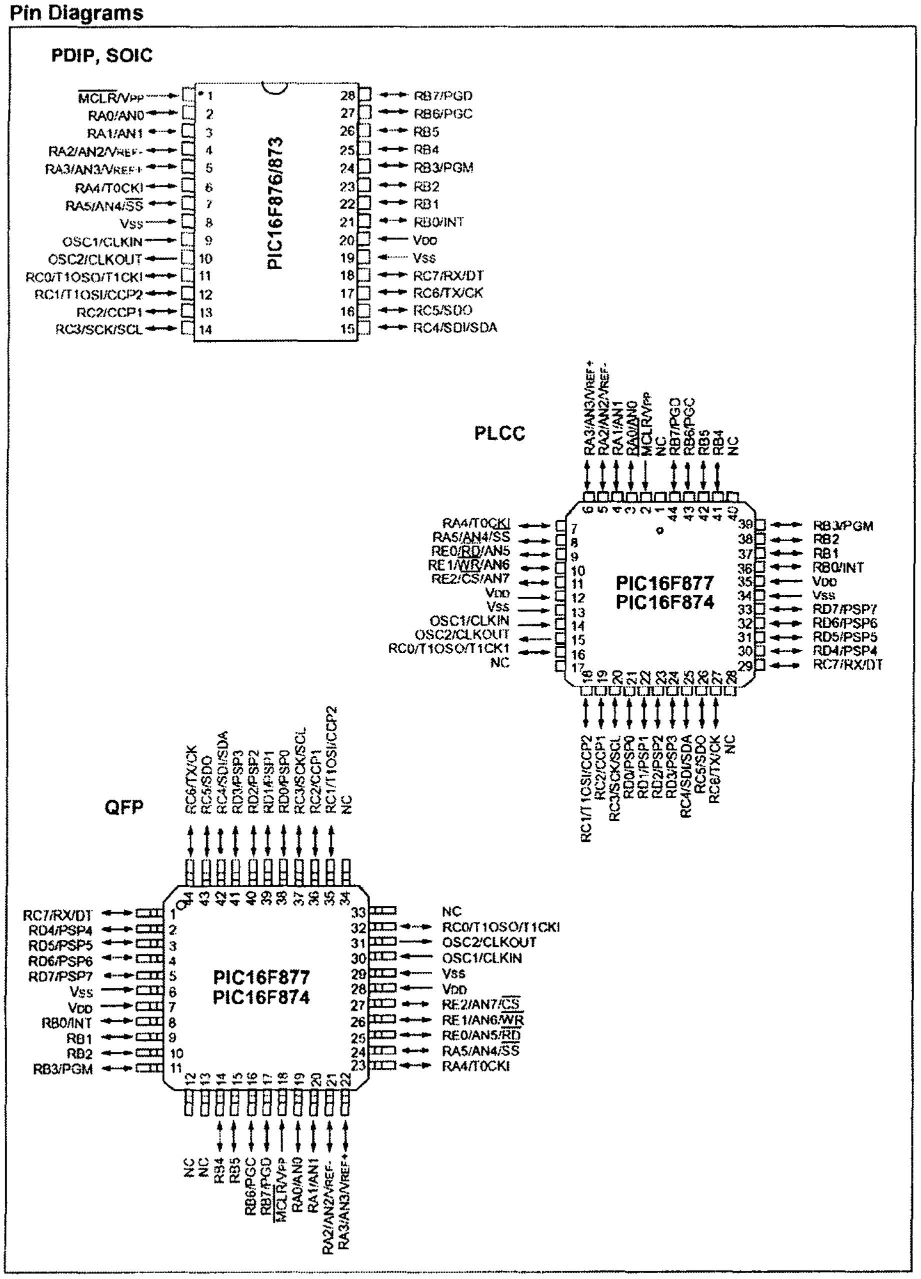
Pin Diagram



Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Pin Diagrams



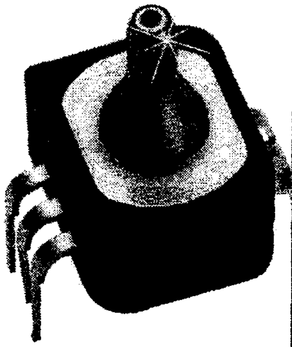
Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

Miniature Voltage Output Pressure Sensor – Model PX40 15G5V

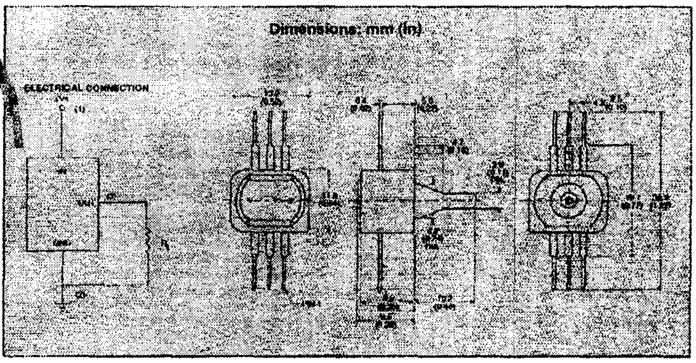
Specifications Sheets

MINIATURE VOLTAGE OUTPUT PRESSURE SENSORS

FULLY TEMPERATURE COMPENSATED



PX40-15G5V, $60, shown much larger than actual size.



$60 All Models

MADE IN USA

- ✓ Smallest Amplified Package
- ✓ Small Light Weight Package
- ✓ Fully Signal Conditioned
- ✓ Temperature Compensated
- ✓ Port Designed for O-ring Interface
- ✓ Excellent Media Compatibility
- ✓ Wet or Dry Industrial Applications

Typical Applications

- ✓ Laboratory Equipment
- ✓ Electronic Brake Systems
- ✓ Engine Oil Level
- ✓ Transmission Fluid Level
- ✓ Air Conditioning Systems
- ✓ Industrial Fluid Level

SPECIFICATIONS

Excitation: 5 Vdc @10 mA
Output Source Current: 0.5 mA max
Output Sink Current: 1.0 mA max
Hysteresis & Repeatability: 0.15% FS

Span:

±50 mmHg	4.00 Vdc Typ
0-15	4.00 ±0.11 Vdc
0-100	4.00 ±0.09 Vdc
0-150	4.00 ±0.07 Vdc
0-250	4.00 ±0.07 Vdc

Null:

±50 mmHg	2.50 ±0.05 Vdc
0-15	0.50 ±0.11 Vdc
0-100	0.50 ±0.04 Vdc
0-150	0.50 ±0.04 Vdc
0-250	0.50 ±0.04 Vdc

Operating Temp.:
-45 to 125°C (-49 to 257°F)
Compensated Temp.:
0 to 70°C (32 to 158°F);
(0 to 50°C 4 inH_2O)

Overpressure:

±50 mmHg	±170 mmHg
0-15	45 psi
0-100	200 psi
0-150	300 psi
0-250	500 psi

Response Time: 1 ms
Gage Type: Silicon
Media Compatibility: Limited to those media which will not attack invar, copper, silicon, stainless steel, glass and solder: i.e. air, water, refrigerants, engine fuel
Weight: 0.18 oz (5 g)

▭ ***MOST POPULAR MODELS HIGHLIGHTED!***

To Order *(Specify Model Number)*

GAGE MODELS (One Port)

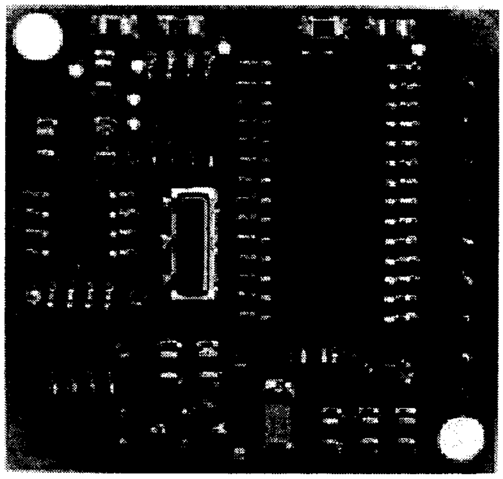
RANGE	MODEL NO.	PRICE	OUTPUT	COMPATIBLE METERS
±50 mmHg	**PX40-50BHG5V**	**$60**	20.8 mV	DP24-E, DP25-E, DP41-E
0-15 psi	**PX40-15G5V**	**60**	16.6	DP24-E, DP25-E, DP41-E
0-30 psi	**PX40-030G5V**	**60**	75	DP24-E, DP25-E, DP41-E
0-100 psi	**PX40-100G5V**	**60**	15	DP24-E, DP25-E, DP41-E
0-150 psi	**PX40-150G5V**	**60**	50	DP24-E, DP25-E, DP41-E
0-250 psi	**PX40-250G5V**	**60**	75	DP24-E, DP25-E, DP41-E
0-500 psi	**PX40-500G5V**	**60**	75	DP24-E, DP25-E, DP41-E

Ordering Example: *PX40-15G5V is a 0-15 psi transducer with .5 to 4.5 Vdc output,* ***$60.***

Devantech Magnetic Compass – Model CMPS03

Specifications Sheets

CMPS03 Magnetic Compass



CMPS03 Magnetic Compass

Description

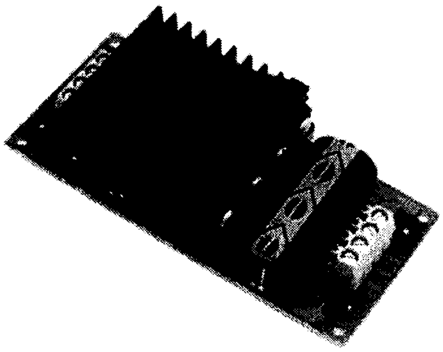
This compass module has been specifically designed for use in robots as an aid to navigation. The aim was to produce a unique number to represent the direction the robot is facing. The compass uses the Philips KMZ51 magnetic field sensor, which is sensitive enough to detect the Earths magnetic field. The output from two of them mounted at right angles to each other is used to compute the direction of the horizontal component of the Earths magnetic field.

Voltage	5v
Current	20mA Typ.
Resolution	0 1 Degree
Accuracy	3-4 Degrees approx (after calibration)
Output 1	Timing Pulse 1mS to 37mS in 0.1mS increments
Output 2	I2C Interface, 0-255 and 0-3599
SCL Speed	up to 1MHz
Weight	0 03 oz.
Size	32mm x 35mm

Devantech HBridge Motor Drive – Model MD03

Specifications Sheets

Devantech HBridge Motor Driver



Description:

The MD03 by Devantech is a robust medium power motor driver designed to supply power and control for one motor. Several control modes. 20A current capacity for 5V to 50V motors.

Details:

The Devantech MD03 is a medium power motor driver, designed to supply power beyond that of any of the low power single chip H-Bridges that exist. Main features are ease of use and flexibility. It also has several different control modes.

Specifications

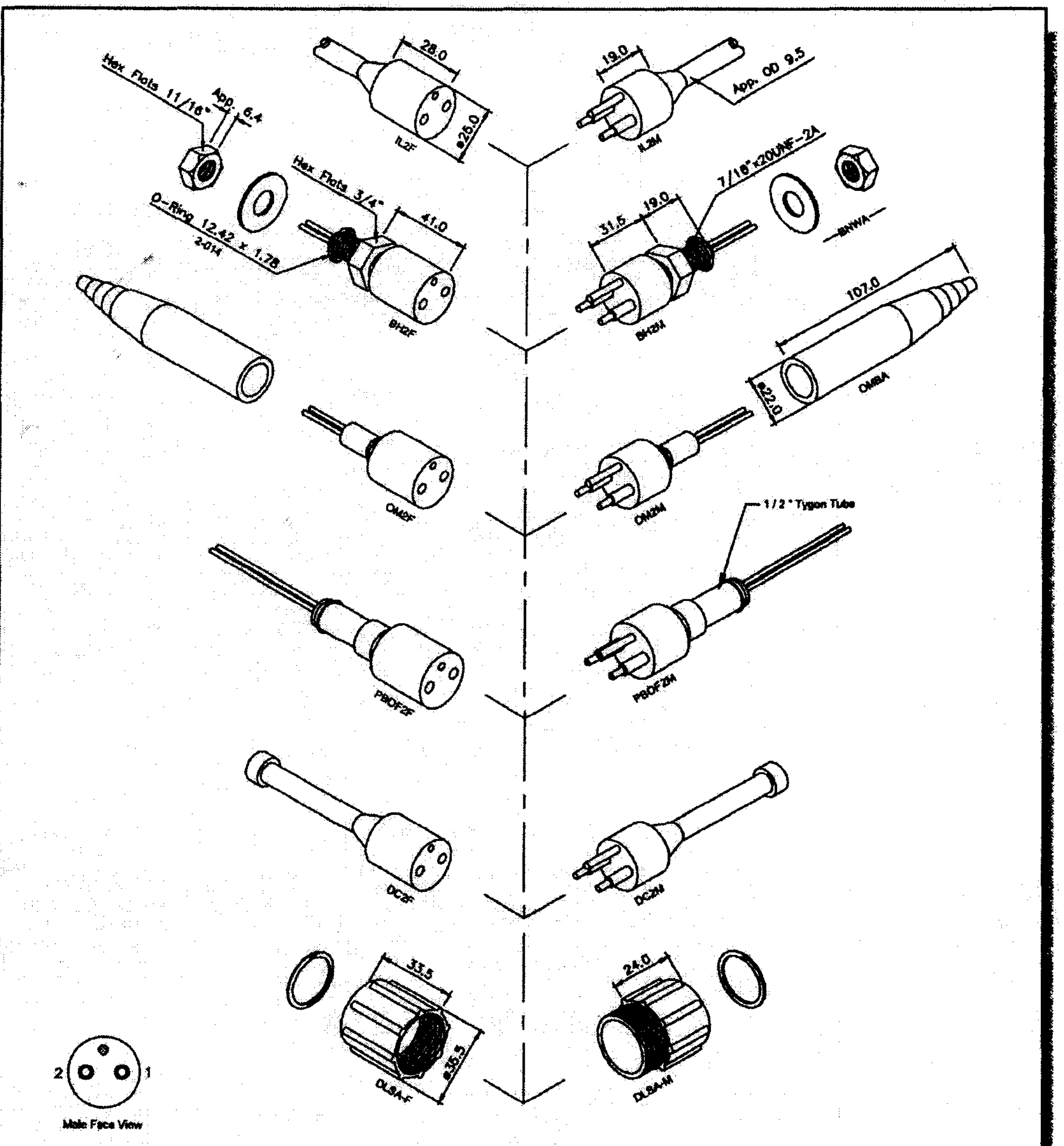
Voltage	5v logic supply and 5v - 50v for the motor
Current	50mA Max for logic and up to 20a for the motor
Mode 1	0v-2.5v-5v Analog, 0v full reverse, 2.5v stop, 5v full forward
Mode 2	0v - 5v (or PWM equiv.) with seperate direction control
Mode 3	RC Mode - Controlled by standard Radio Control system. Direct connection to RC receiver. (1mS - 2mS pulse with 1.5mS center off).
Mode 4	I2C Interface, full control, acceleration and status reporting. Up to 8 modules on bus. SCL speed up to 1MHz
Current limiter	Preset at 20A
Temperature Limiter	Reduces power gracefully if module overheats
Size	113mm x 52mm x 30mm

Additional application notes, schematics, and tech support can be found at Robot Electronics.

Subconn Connectors Schematics and Data

Subconn®

Circular 2 Contacts



Inline Cable Colour Code

1 Black
2 White

(Bulkhead Leads Number Tagged)

Connector Specifications

Single Contact Rating	: 600 V @ 15 Amp
Connector Rating 2 pins	: 600 V @ 10 Amp
Insulation Resistance	: >200 megaohm
Contact Resistance	: <0,01 ohm
Wet Matings	: >500
Temp. Rating	: -4° to +60° C 25° to 140° F
Depth Rating	: 1.400 bar - 20,000 psi

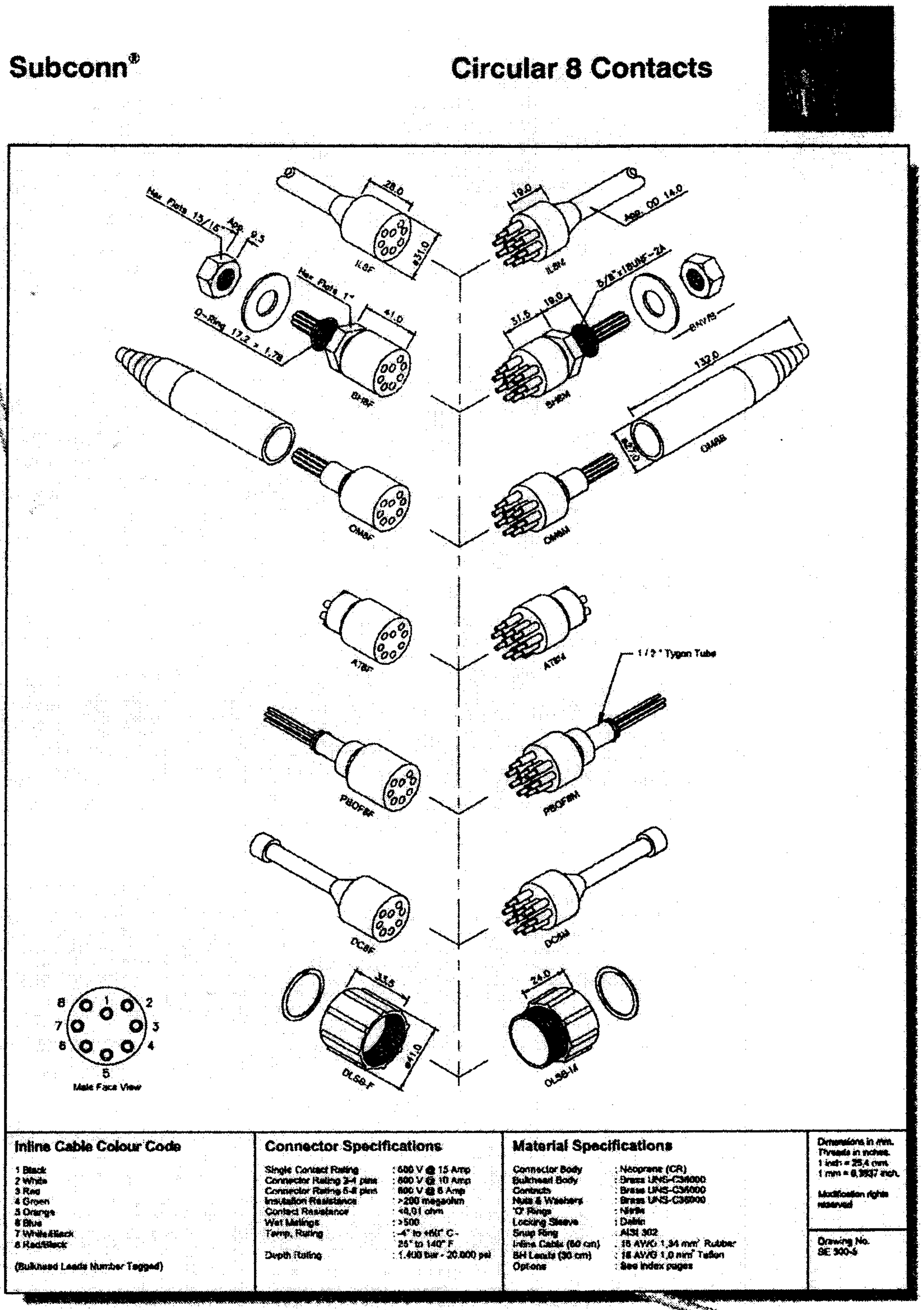
Material Specifications

Connector Body	: Neoprene (CR)
Bulkhead Body	: Brass UNS-C36000
Contacts	: Brass UNS-C36000
Nuts & Washers	: Brass UNS-C36000
Location pin	: AISI 303
'O' Rings	: Nitrile
Locking Sleeve	: Delrin
Snap Ring	: AISI 302
Inline Cable (80 cm)	: 16 AWG 1,34 mm² Rubber
BH Leads (30 cm)	: 18 AWG 1,8 mm² Teflon
Options	: See index pages

Dimensions in mm.
Threads in inches.
1 inch = 25,4 mm.
1 mm = 0,03937 inch.

Modification rights reserved

Drawing No.
SE 300-4



Subconn®

Circular 8 Contacts

Inline Cable Colour Code

1 Black
2 White
3 Red
4 Green
5 Orange
6 Blue
7 White/Black
8 Red/Black

(Bulkhead Leads Number Tagged)

Connector Specifications

Single Contact Rating	: 600 V @ 15 Amp
Connector Rating 2-4 pins	: 600 V @ 10 Amp
Connector Rating 5-8 pins	600 V @ 5 Amp
Insulation Resistance	: >200 megaohm
Contact Resistance	: <0,01 ohm
Wet Matings	: >500
Temp. Rating	: -4° to +60° C - 25° to 140° F
Depth Rating	: 1.400 bar - 20.000 psi

Material Specifications

Connector Body	: Neoprene (CR)
Bulkhead Body	: Brass UNS-C36000
Contacts	: Brass UNS-C36000
Nuts & Washers	: Brass UNS-C36000
'O' Rings	: Nitrile
Locking Sleeve	: Delrin
Snap Ring	: AISI 302
Inline Cable (60 cm)	: 16 AWG 1,34 mm² Rubber
BH Leads (30 cm)	: 18 AWG 1,0 mm² Teflon
Options	: See index pages

Dimensions in mm.
Threads in inches.
1 inch = 25,4 mm.
1 mm = 0,3937 inch.

Modification rights reserved

Drawing No.
SE 300-5