DEVELOPMENT AND TESTING OF A FUNCTIONAL MODEL SCALE UNDERWATER GLIDER FOR RESEARCH IN DYNAMICS AND CONTROL

PENG WEN

DEVELOPMENT AND TESTING OF A FUNCTIONAL MODEL SCALE UNDERWATER GLIDER FOR RESEARCH IN DYNAMICS AND CONTROL

by

©Peng Wen

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

October, 2012

St. John's Newfoundland

Abstract

Due to their long endurance and low operational cost, underwater gliders are a proven technology for collecting data from the oceans. The AUV groups at NRC-OCRE (National Research Council Canada, Ocean, Coastal and River Engineering) and MUN-AOSL (Memorial University of Newfoundland - Autonomous Ocean Systems Lab) are designing, building and testing the 'IOT Glider'. It is a test-platform prototype for the development of low-energy-budget gliders for long-duration missions, and has been designed for performing sawtooth-like trajectories in a tow tank which is 7 m deep and 200 m long. The IOT Glider has an overall length of 1.56 m, a hull diameter of 115 mm and a mass of 9.6 kg. This glider is therefore a multi-function vehicle for concept design, assessment of hydrodynamic performance and control systems research. It also provides comprehensive, multi-feature capabilities which are inexpensive to implement, test and verify.

The current IOT Glider has a complete buoyancy engine, an active pitch and roll control mechanism, and, attitude and pressure sensors. The buoyancy engine uses a linear actuator and a D-type diaphragm to change the glider's buoyancy. The pitch and roll mechanism provides independent control of the longitudinal and rotational positions of the 470 $gram$ battery pack, and provides the desired pitch angle and turning rate. This glider has a pair of swept wings which can be adjusted fore-and-aft, a hollow nose section and a hollow tail section made from ABS plastic via a rapid

prototyping machine; these sections can be used for additional sensors, and for ballasting and adjusting trim conditions.

For prediction purposes, a MATLAB-based motion simulator has been developed for mission planning. The six-degree-of-freedom motion equations include the effects of added mass, hydrodynamic forces and moments, and hydrodynamic damping. Several strategies are used to improve the quality of these predictions, including CFD (Computational Fluid Dynamics) analysis, analytical methods and empirical techniques.

Acknowledgements

I would like to thank my supervisors, Dr. Ralf Bachmayer and Dr. Christopher Williams, Dr. Moqin He and Dr. Dong-Cheol Seo of NRC St. John's and all the colleagues working surrounding me, Brian Claus, Mingxi Zhou, Zhi Li, and Haibing Wang. I would also like to thank the technicians in the Technical Services, Machine Shop and OERC (Ocean Engineering Research Centre) at MUN for their instrumentation and experimental support. This dissertation would not have been possible without the guidance, support and help from them.

Table of Contents

List of Tables

List of Figures

Nomenclature

L_{nose}	The length of the nose section of the IOT Glider
L_{tail}	The length of the tail section of the IOT Glider
η	The position and orientation vector of the IOT Glider in earth-fixed coordinate system
υ	The translational and rotational velocity vector of the glider with respect to the body-fixed coordinate system
τ	The total external forces and moments acting on the glider in the body-fixed frame
$\mathbf{I}$	The moment of inertia matrix
B	Buoyancy force of the IOT Glider
xG	The X component of the glider C. G. in the body-fixed frame
yG	The Y component of the glider C. G. in the body-fixed frame
zG	The Z component of the glider C. G. in the body-fixed frame
xB	The X component of the glider C. B. in the body-fixed frame
yB	The Y component of the glider C. B. in the body-fixed frame
zB	The Z component of the glider C. B. in the body-fixed frame
B_0	The buoyancy force on the IOT Glider when the piston is in its neutral position
m	The mass of the IOT Glider (with or without ballast mass)

x_piston	The X-axis position of the piston relative to the neutral position
A_piston	The frontal area of the piston
ρ	The density of fresh water
xB_0	The original X-axis position of the buoyancy force with regard to the body-fixed frame
xG_0	The original X-axis position of the gravity force with regard to the body-fixed frame
m_piston	The mass of the piston
x_batt	The X-axis position of the battery pack relative to the neutral position
m_batt	The mass of the battery pack
$theta\_batt$	The angle the battery pack rotates with regard to its zero position
rG_batt	The distance from the battery pack C. G. to the glider central axis
I'_{yy}	Original I_{yy} value
X_piston	The X-axis present position of the piston in body-fixed frame
X_piston'	The X-axis previous position of the piston in body-fixed frame
X_batt	The X-axis present position of the battery pack in body-fixed frame
X_batt'	The X previous position of the battery pack in body-fixed frame
$\mathbf{B}$	The buoyancy force vector in the body-fixed frame
$\mathbf{G}$	The gravity force vector in the body-fixed frame
$\mathbf{r_G}$	The glider C. G. position vector in body-fixed frame
$\mathbf{r_B}$	The glider C. B. position vector in body-fixed frame
l_{wing}	The wing tip-to-root length
α_w	The wing local angle of attack
Λw	The swept angle of the wings
θ_0	The initial rolling angle offset of the IOT Glider

rG	The distance from the C. G. to the central axis

Chapter 1

Introduction

1.1 Underwater Gliders

The underwater glider is a relatively new type of oceanographic Autonomous Underwater Vehicle (AUV) used for oceanographic measurements or experiments. Underwater gliders rely on buoyancy change to operate and they always have buoyancy engines, fixed wings and internal positioning mass.

Nowadays, there are several commercial gliders available. Different existing underwater glider designs can be found in the following references[17], [5], [12]. At Memorial University of Newfoundland (MUN) and the National Research Council in St. John's (NRC in St. John's) glider related research is still going on. New concept designs and improvements are developed and realized. Figure 1.1 is a photo of the SLOCUM glider in the Autonomous Ocean Systems Lab (AOSL).

1.2 Motivation and Scope of Work

It is always within our interest to better understand the performance of underwater gliders. To develop a platform for in-door research on dynamics and control of this

Figure 1.1: A SLOCUM Glider in AOSL

type of underwater vehicles, we decided to build a laboratory-scale glider which can be used within the lab environment, the IOT Glider. The several tank facilities at MUN and NRC will be capable of conducting experiments and tests on the IOT Glider. The tested and proven concepts and ideas will then be applied to a real underwater glider. The IOT Glider imitates the design idea of the SLOCUM glider. Previous studies and designs include buoyancy engine mechanical and electronic design and the concept design of the battery pack frame mechanism which is used for roll and pitch motion control. Janes [10] started from the buoyancy engine concept design and studied its vertical motion pattern without wings. Skillings [13] designed and constructed the buoyancy engine for the IOT Glider. Warren [15] designed the electrical system for the buoyancy engine and tested the buoyancy engine in water. Williams [19] trimmed and ballasted the buoyancy engine and studied the battery system in the buoyancy engine. Hewitt [8] initially designed the pitch and roll control mechanism for the IOT Glider. The tasks of this thesis project is to complete the IOT Glider design and test it. Next section gives a summary of the work done in this thesis.

1.3 Contributions of Thesis

The following work is considered and conducted in this thesis work.

- Recheck the pitch and roll mechanism design.
- Develop the electronic part of the roll and pitch mechanism and integrate it into the overall glider electric system.
- Integrate the pitch and roll mechanism.
- Test the pitch and roll mechanism.
- Design and assemble back and forth movable wings.
- Design and assemble appropriate fins.
- Design and assemble appropriate glider nose and tail sections.
- Develop software for controlling the whole system.
- Develop numerical motion simulation model for the IOT Glider.
- Measure and calculate the basic parameters of the glider. (Centre of Gravity, Moments of inertia)
- Test the IOT Glider in the trim tank and deep tank at MUN.

The structure of this thesis will be described in the next section.

1.4 Outline of Thesis

This thesis addresses the design and tests of the lab-scale IOT Glider. Chapter one is the introduction to the thesis project. Chapter two summarizes previous designs

and describes new designs for the IOT Glider. Chapter three describes the numerical simulation model development for the IOT Glider. Chapter four addresses the CFD analysis on IOT Glider wings and the bare hull. Chapter five describes the testing procedure of the IOT Glider in open air and the measurements of its mass properties. Chapter six describes a set of tests of the IOT Glider in water. Chapter seven provides a summary of the whole work.

Chapter 2

Completing the IOT Underwater Glider Design

2.1 Introduction

The IOT Glider is designed for research on underwater gliders in a lab environment. It has an overall length of 1.56 m and an overall mass of approximately 10 kg. The main diameter of its torpedo-shaped bare hull is approximately 115 *mm*. The IOT Glider design contains a complete buoyancy engine which is located at the front and an active roll and pitch mechanism inside the glider hull. It also features back-and-forth movable wings which have a 0.7 m wing span. The glider also has four identical fins near the tail section. The glider is controlled by a Rabbit Smartcat^{TM} single board computer. The on-board sensors include two pressure sensors, a two-channel inclinometer, two motor encoders and a linear potentiometer which are all located in the electronic section of the glider. Figure 2.1 shows a perspective view of the IOT Glider in SolidWorks^{TM}, with the main parts labelled. Table 2.1 gives a summary of the main dimensions of the IOT Glider profile.

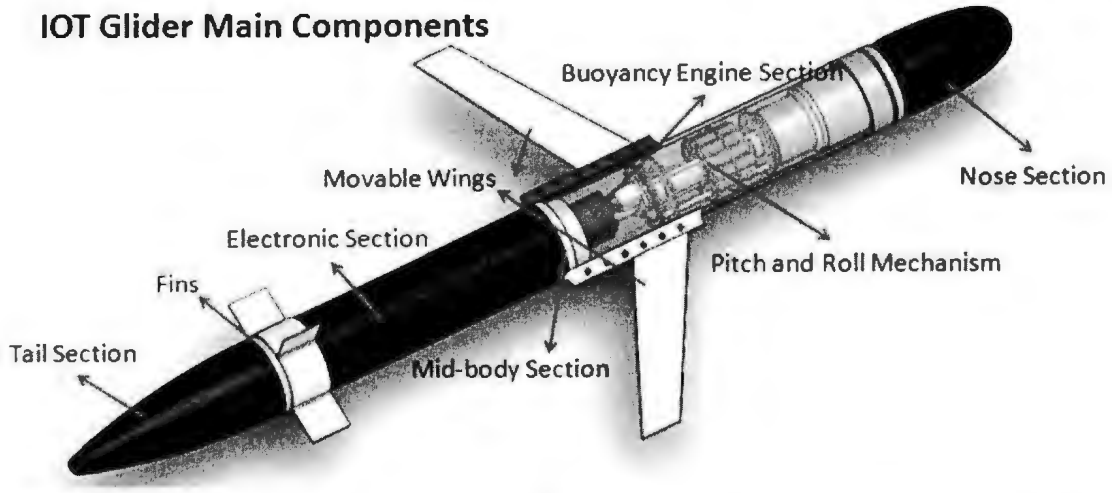


Figure 2.1: SolidWorks Drawing of the IOT Glider with Main Components Labelled

This chapter gives a detailed description of the mechanical and electronic system

Table 2.1: Main Dimensions of IOT Glider

Description	Value	Unit
Overall Length	1.56	m
Net Weight in Air	9.63	kg
Main Diameter	0.115	m
Wing Span	0.64	m

design of the IOT Glider.

2.2 Mechanical Design

The IOT Glider is mainly composed of the nose section, the cylinder hull section, the tail section, the wings, the wing mounts, the fins, the buoyancy engine and the pitch and roll mechanism.

2.2.1 Vehicle Main Profile

The bare hull shape of the IOT Glider is based on the body shape of the Phoenix AUV [18]. The bare hull is composed of three sections: the nose section, the cylinder hull section and the tail section. The cylinder hull section is a circular cylinder shape with a diameter of 0.115 m and a length of 1.01 m. The cylinder's hull section is also composed of three parts: the forward buoyancy engine section, the mid-body section and the aftward electronic section. See Figure 2.1 for the arrangement of the IOT Glider.

Equation 2.1 gives the shape of the nose-section. Nose shape is given by radius r.

$$\frac{r}{D} = 0.8685(\frac{X}{D})^{1/2} - 0.3978(\frac{X}{D}) + 0.006511(\frac{X}{D})^{2} + 0.005086(\frac{X}{D})^{3} \tag{2.1}$$

where X is the distance aft of the Forward Perpendicular, D is the bare hull diameter. The length of the nose section is determined by $L_{nose} = 1.75D$.

Equation 2.2 gives the shape of the tail-section. Tail shape is given by radius r as well.

$$\frac{r}{D} = \frac{1}{2} - \frac{1}{18}(3 - \frac{X'}{D})^{2} \tag{2.2}$$

where X' is the distance ahead of the Aft Perpendicular, D is the bare hull diameter. The length of the tail section is determined by $L_{tail} = 3.00D$.

Figure 2.2 gives a sketch of the outline profile for the vehicle's bare hull. The cylinder hull section is sealed and the nose section and the tail section are attached to it by machine screws. The nose and tail sections are designed as hollow bodies. The benefit is that additional buoyancy or mass could be provided later by putting foam or ballast mass into the hollow sections. Drain holes on their surface were designed for the water to get in and out.

The parts were manufactured using the FDM (Fused Deposition Modelling) machine,

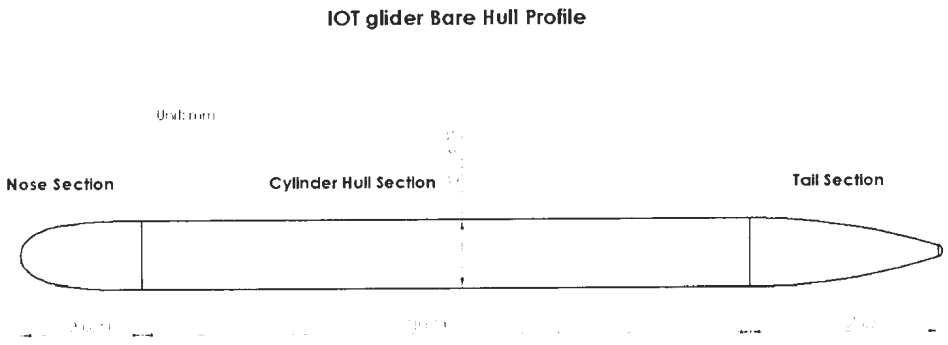


Figure 2.2: SolidWorks Drawing of the IOT Glider

also called the rapid prototyping machine. The material used was ABS plastic. The parts made on this machine are formed from melted plastic material layer by layer according to the specified profile. During the process, some pores may be left between the layers. This results in less density of the final parts than the raw ABS material. The labelled specific gravity of the raw material is 1.04, but the final specific gravity of the parts is around 0.5. This is beneficial because these parts will provide additional buoyancy for the glider.

2.2.2 Glider Fins

The IOT Glider is equipped with four identical fins which are mounted in a cruciform pattern near the forward end of the tail section. Each fin is made of a square flat plate of 6 cm side length. The main purpose of these fixed fins is not for control surfaces.

They are designed for stabilizing the movement of the vehicle. Plus, further studies on its function and performance can be conducted by moving the fin part along the hull or turning it around.

Figure 2.3 shows a perspective view of the fin part 3-D model from SolidWorks. The

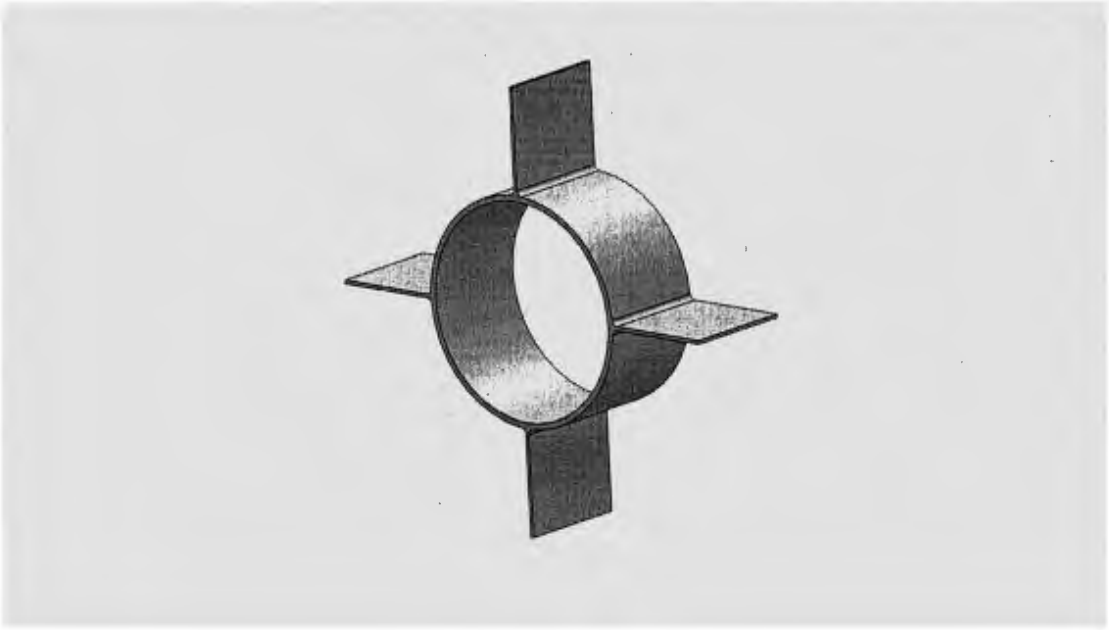


Figure 2.3: IOT Glider Fins

fin part is also made by the rapid prototype machine.

2.2.3 Wings and Wing Mounts

For the purpose of studying the effects of the wing position in the future, the IOT Glider wing design allows the wings to be moved back and forth in a certain range. The wings can be fixed at any position within a range of approximately 90 *mm*. This function is achieved by a special design of wing mounts and wings.

The wings are in the form of swept flat plates. The thickness of the wings is 3 *mm*. Other dimensions of the wings can be found in Figure 2.4. Figure 2.5 gives standard

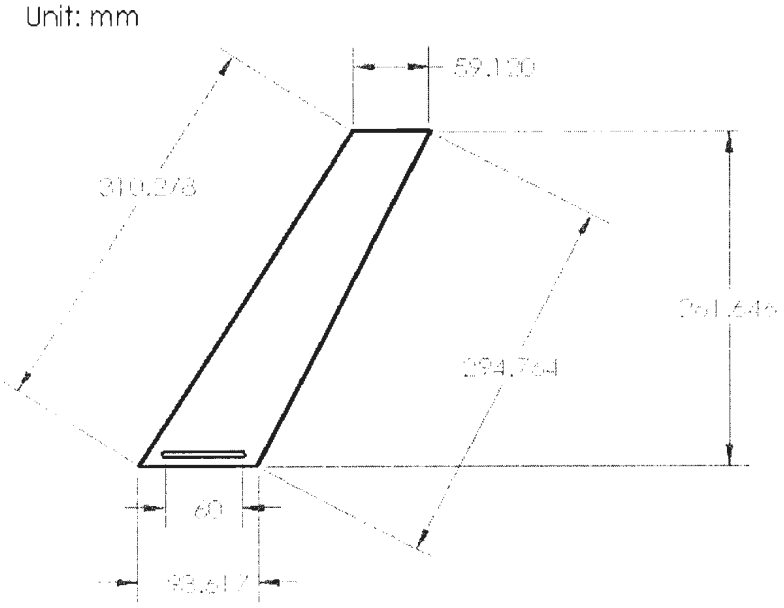


Figure 2.4: Dimensions of the Wings

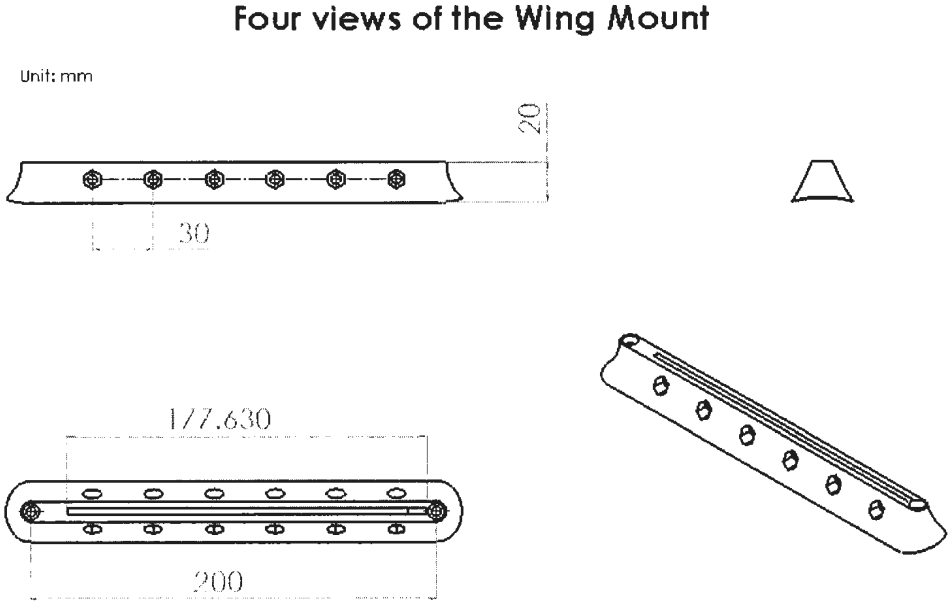


Figure 2.5: Four Views and Dimensions of the Wing Mount

three views and a perspective view of the wing mount design. The groove on the wing mount is designed longer than the wing root so that the wing can be moved along it. The slot near the wing root edge makes it easy to put the wing at any possible position on the wing mount. Machine screws are used to fix the wings.

The forward end of the wing mount is located at 531 *mm* aft of the glider nose tip. As a result, the wings' root leading edge position is capable of moving from 571 *mm* to 661 *mm* aft of the glider nose tip. Two set screws were inserted outside the glider hull pipe to make an attachment for the wings. The outside of the aluminium mid section is drilled and threaded to make an attachment for the wings. Figure 2.6 shows the attachment of the wings on the glider hull. The wings and wing mounts are also made on the rapid prototyping machine. No FEA (Finite Element Analysis) was performed

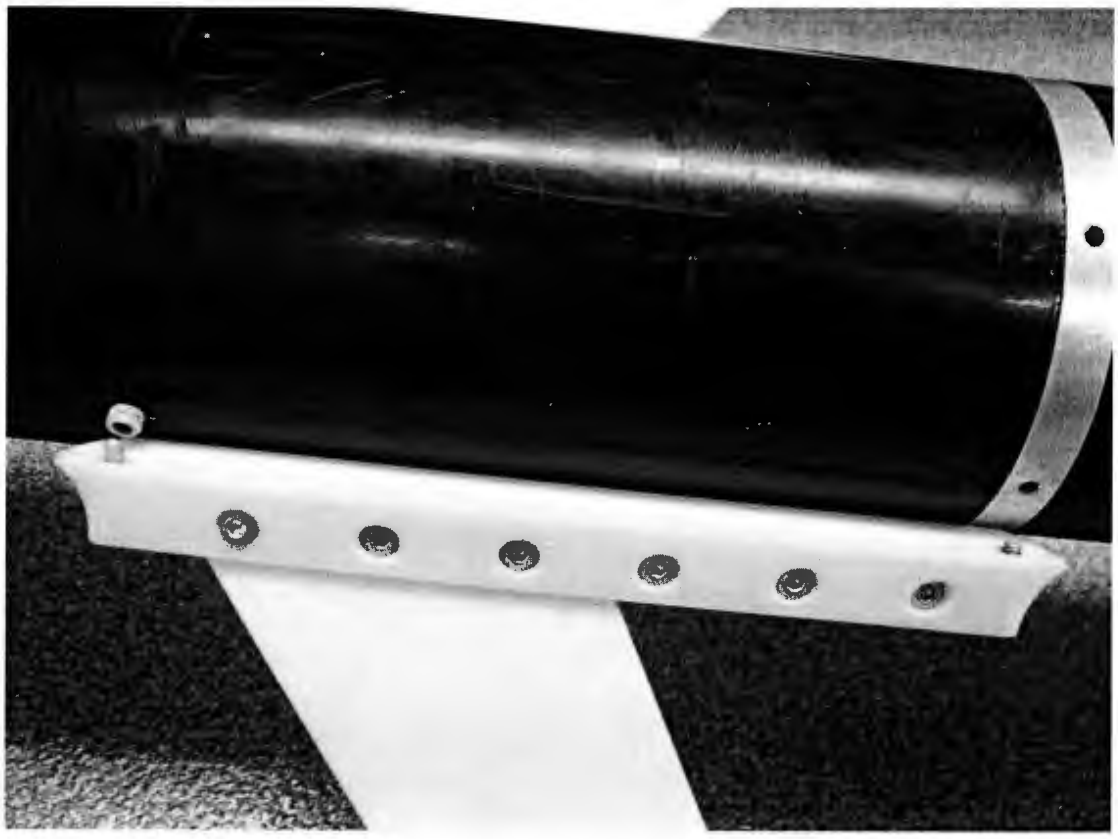


Figure 2.6: Attachment on Glider Hull for the Wings

on any of the structural components of the IOT Glider. The wings seem to be weak, however, from the test observations, the wings did not deflect much. It is reasonable due to the low speed of the vehicle motion.

2.2.4 Buoyancy Engine

Underwater gliders normally rely on changes in buoyancy-gravity-difference to operate. The buoyancy engine of the IOT Glider involves the use of a linear actuator and a rolling diaphragm. Figure 2.7 gives the conceptual drawing of the IOT Glider buoyancy engine [10].

When the linear actuator extends or retracts, the piston which is near the nose part

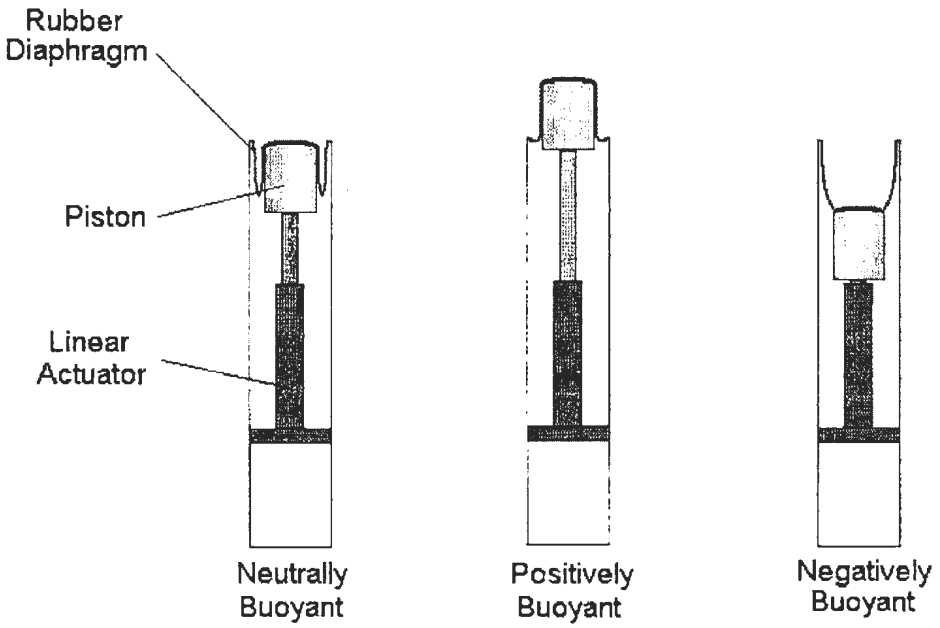


Figure 2.7: Conceptual Drawing of the Buoyancy Engine on IOT Glider

of the glider will make the rolling diaphragm deform. At the same time, a certain amount of water will be expelled or injected through the drain holes on the surface of

the nose part. This action will change the buoyancy force on the vehicle. Usually at the start of a mission, a glider will be ballasted to be neutrally buoyant. Later, when there is a difference created between the buoyancy and gravity forces on the glider, the glider will not be neutrally buoyant and will start to rise or dive.

The diaphragm is a custom made DiaCom^{TM} Type D-300-300 Rolling Diaphragm. The Type D diaphragm is molded with a bead on the flange for easy sealing. A key feature of a rolling diaphragm is that it maintains a constant area as long as it is positioned within its recommended half-stroke. This allows for precise control of volume when position feedback is used.

The linear actuator uses a HT23 stepper motor which allows the piston movement in a 14 cm range. The piston is a circular cylinder with a frontal area of 0.0036 m^2. Setting the initial actuator piston position at the middle of the moving range, the zero position, the buoyancy engine has a capability of expelling or injecting 256.47 cm^3 volume of water. According to the Archimedes' principle, in water with 1000 kg/m^3 density, the buoyancy engine is capable of providing at most about 2.5 N buoyancy-gravity-difference. When the buoyancy engine is in its zero position, the centre of the piston is located at 246 mm aft of the glider nose tip.

2.2.5 Pitch and Roll Mechanism

The active pitch and roll motion control of the IOT Glider is achieved by moving the battery pack. It is moved along the glider's longitudinal axle and rotated around it. The motion of the battery pack will result in the change of the centre of gravity in three directions. The battery pack is mounted around a sleeve bearing which rides on the buoyancy engine actuator tube. An actuator is used for driving the translating motion of the battery pack and a stepper motor with a set of gears rotates it. This allows the batteries to translate for pitch control, and to rotate around the shaft for

roll control. The pitch actuator provides a range of ± 20 *mm* battery pack linear movement; the roll motor provides a range of ± 90 *degree* battery pack rotation. Figure 2.8 gives a description of the pitch and roll mechanism in detail. Hewitt [8]

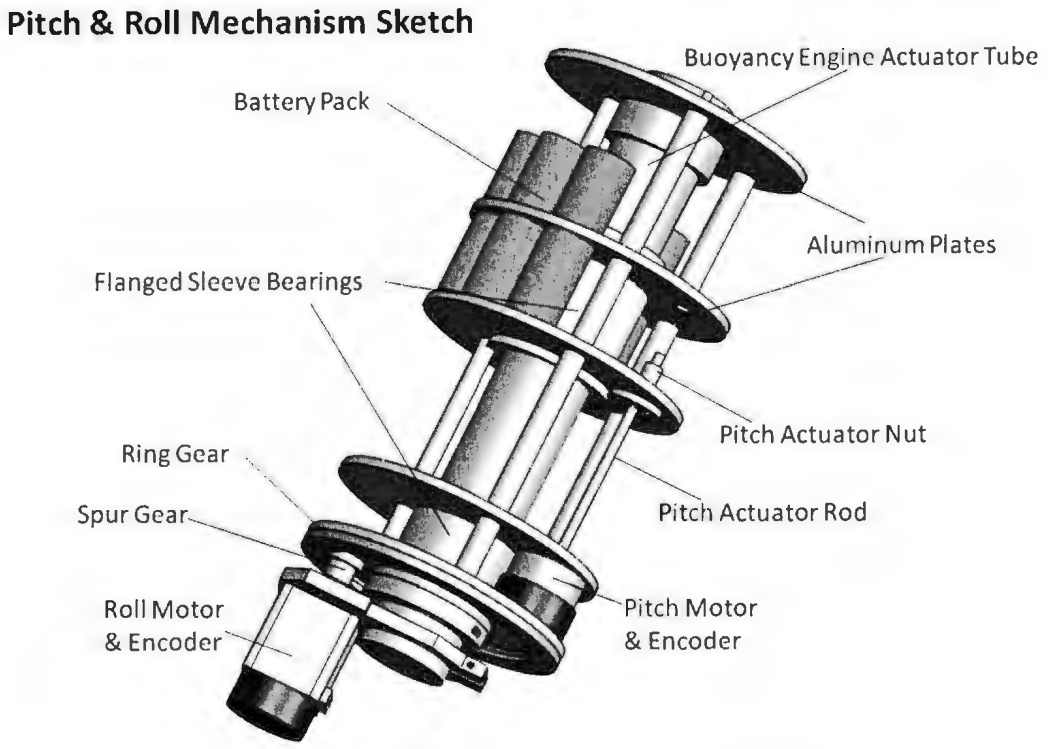


Figure 2.8: Pitch and Roll Mechanism Sketch

originally came up with the concept design of the pitch and roll mechanism. His work was reviewed and revised in some detail. The plates were processed by the Technical Services at MUN. The motor for rolling control was changed to a HT11 step motor; the spur gear was also changed to fit on the new motor.

The battery pack contains seven lithium-ion batteries. Each of these batteries has a mass of 47 g, a diameter of 18 *mm* and length of 65 *mm*. When the battery pack is in its zero position (central position), the position of the battery pack centre is longitudinally located at 466 *mm* aft of the glider nose tip. To allow for more

effective vehicle roll motion, it is important to set the centre of gravity of the battery pack apart from the glider central axis. The arrangement of the batteries (end view) is shown in Figure 2.9. This battery arrangement alone gives the centre of gravity of

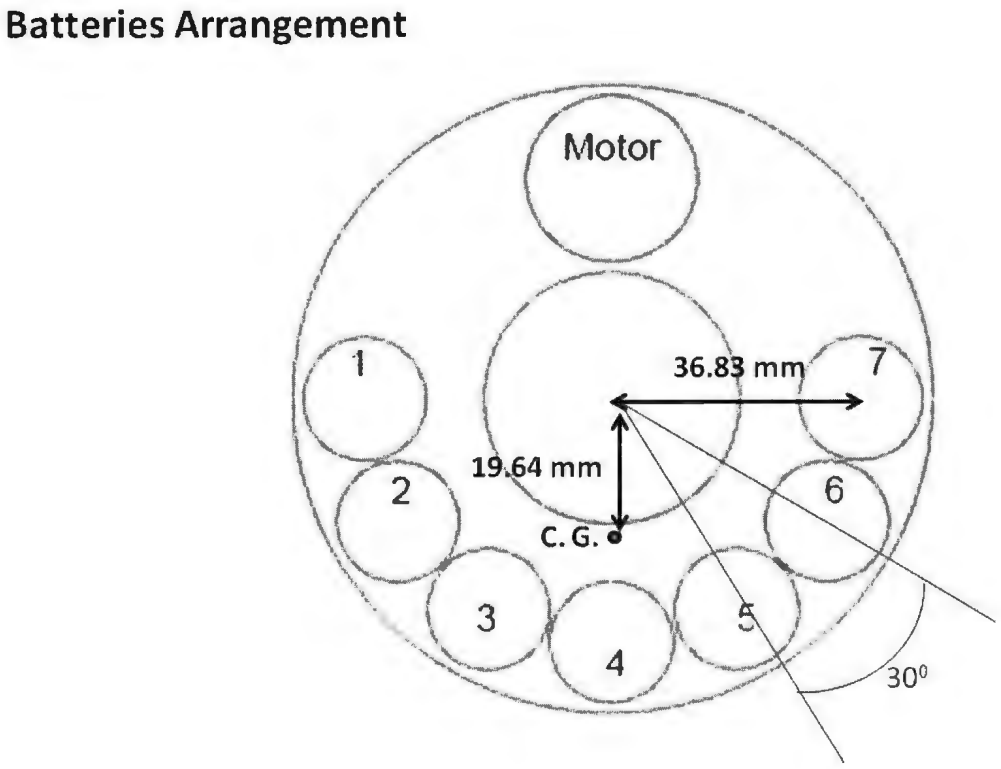


Figure 2.9: The Arrangement of the Batteries

the battery pack 19.64 *mm* distance from the central axis.

2.3 Electronic Components on the IOT Glider

2.3.1 Electronic Tray

The electronic components on the IOT Glider are mainly set on two sides of an aluminium plate which is attached in the electronic section of the vehicle. Figure 2.10 shows the two sides of the electronic tray on the IOT Glider, with the main electronic components labelled. The aluminum tray is not coated with anything but rubber pads are put between the electronics and the tray to prevent shorts. Plastic cable connectors are used for easy connection and insulation. Plastic coats are used for coating wires when necessary. To mount the electronic tray, two rails glued inside the hull wall provide attachment for the plate. Figure 2.11 shows the aluminium tray and the two attachment rails. The electronic system sketch on the IOT Glider is shown in Figure 2.12.

2.3.2 Rabbit Single Board Computer

The IOT Glider uses a high-performance C-programmable single board computer (SBC) BL2100 for controlling the entire glider system. The BL2100 offers several built-in digital and analogue I/O ports with serial communications. A Rabbit 2000 microprocessor operating at $22.1MHz$ provides fast data processing. The BL2100 is also provided with its own file system which can be used for run-time data storage in stand-alone applications, supported by its $256K$ flash memory and the $128K$ static RAM. The usage of the I/O ports is listed in Table 2.2.

The BL2100 is programmed using Rabbit's Dynamic C. Dynamic C is an integrated development system for writing embedded software and is designed for use with SBCs and other devices based on the Rabbit microprocessor.

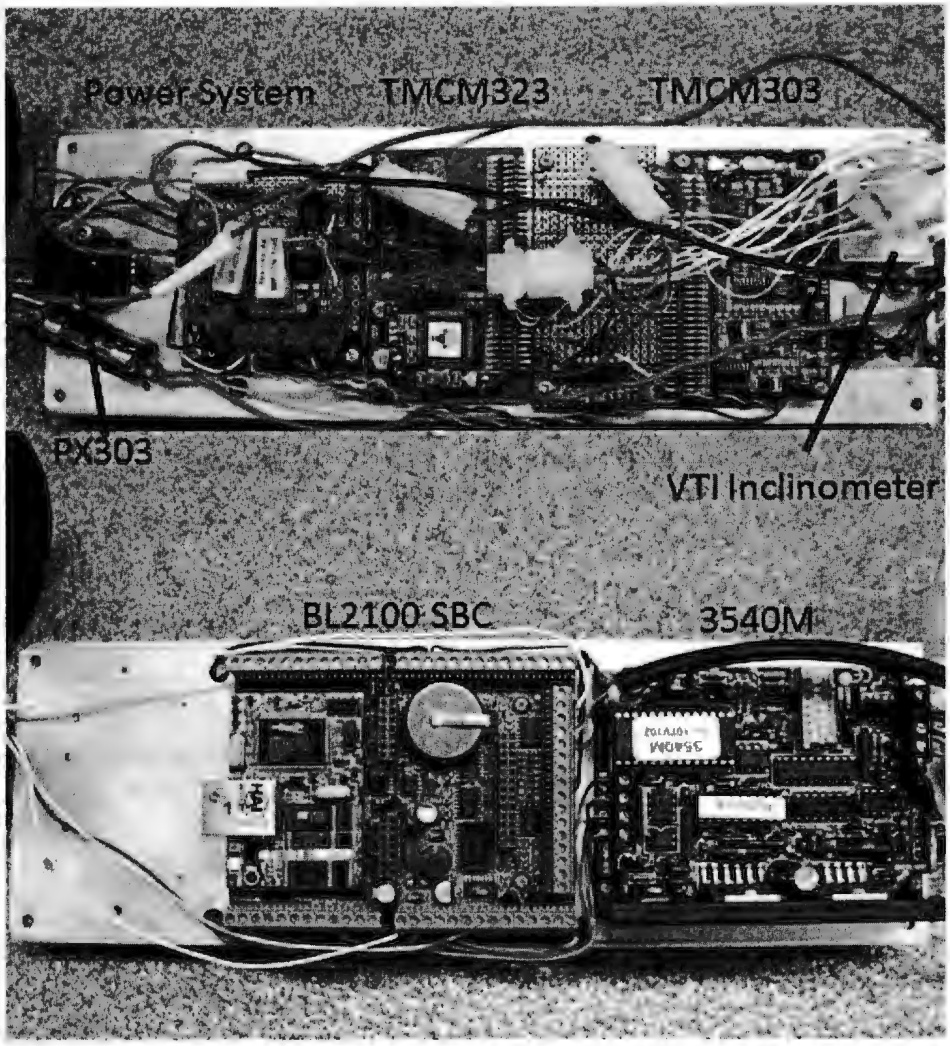


Figure 2.10: The Electronic Tray (Up: the upper-side of the tray; Down: the under-side of the tray)

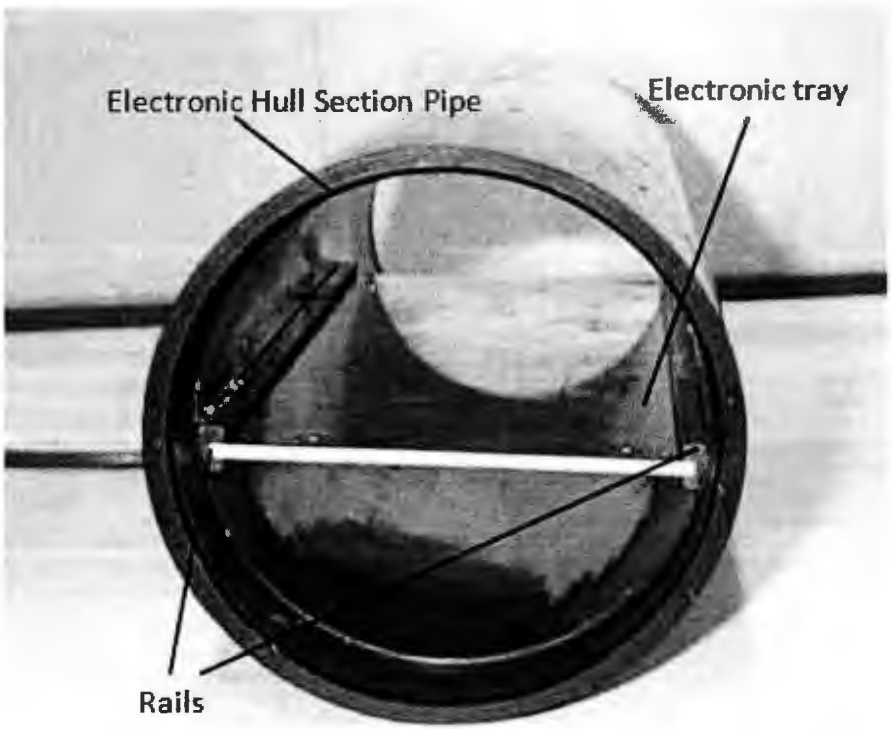


Figure 2.11: Electronic Tray on IOT Glider

Table 2.2: I/O Port Usage on the BL2100

I/O Ports	Type	Function
ADC00	Analogue Input	Buoyancy Engine Potentiometer Input Channel
ADC01	Analogue Input	PX303 Pressure Sensor Signal Input Channel
ADC02	Analogue Input	PX138 Pressure Sensor Signal Input Channel
ADC03	Analogue Input	Inclinometer X-axis Signal Input Channel
ADC03	Analogue Input	Inclinometer Y-axis Signal Input Channel
OUT00	Digital Output	Enable Channel on 3540M Motor Driver
OUT01	Digital Output	Direction Channel on 3540M Motor Driver
OUT02	Digital Output	Step Channel on 3540M Motor Driver

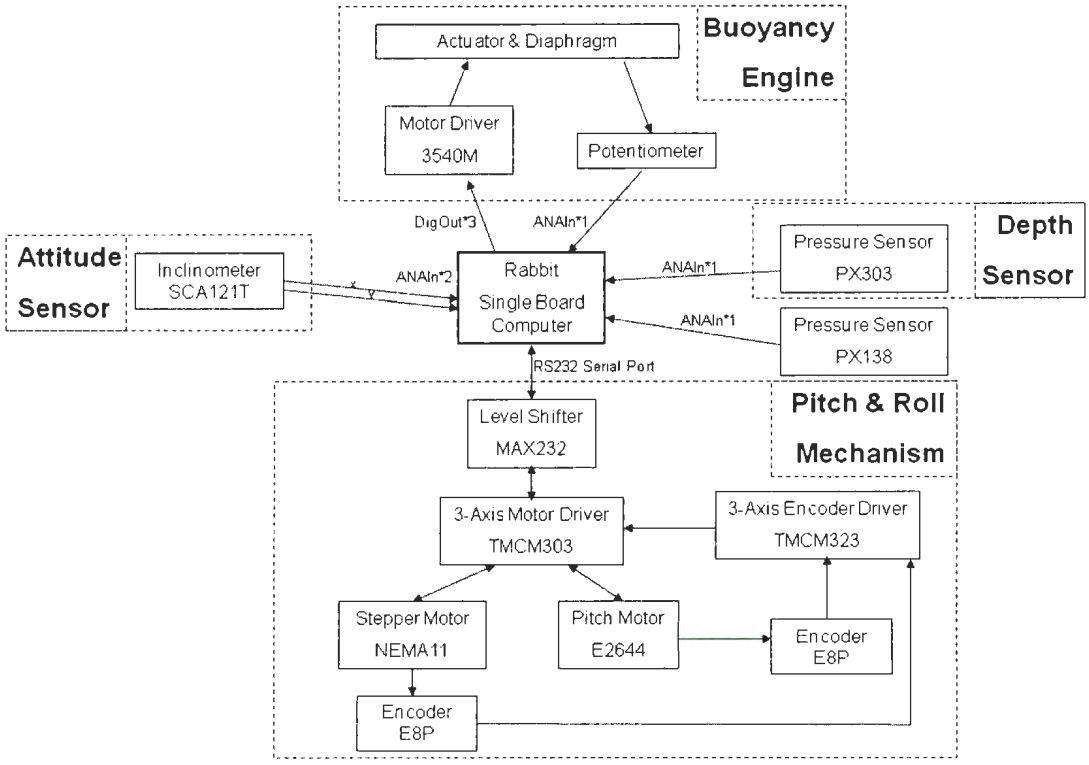


Figure 2.12: Electronic System of IOT Glider

2.3.3 Buoyancy Engine

As stated before, the buoyancy engine design consists of a rolling diaphragm driven by a linear actuator. To move the diaphragm, an UltraMotion^{TM} Digit 8 inch stroke Linear Actuator with NEMA HT23-401 size stepper motor is used. To obtain position feedback the linear actuator is fitted with a Precision Linear Potentiometer. The analogue output from the potentiometer is attached to the BL2100 SBC. With this set-up, precise control of the volume of ballast water is possible. For motor control the system uses an Applied Motion^{TM} 3540M Stepper Motor Driver. Inputs to the driver are optically isolated signals for step, direction, and enabling the motor. To control the buoyancy engine, the signals are sent out from three digital output channels on the BL2100 SBC. The construction and testing of the buoyancy engine have been done by Skilling [13] and Warren [15]. Refer to their reports for more detail on this.

2.3.4 Pitch and Roll Motors and Encoders

The battery pack rotating motion is actuated by a NEMA HT11 bipolar stepper motor, while the translating motion is actuated by a Haydon Kerk^{TM} E2644 Can-Stack stepper motor linear actuator which has a threaded shaft and a nut. Both the stepper motors can be controlled by a TRINAMIC TMCM303 3-axis step motor driver at the same time. The two E8P optical encoders attached to the motors are connected to a TRINAMIC TMCM323 3-axis encoder interface module. The TMCM303 can talk to the TMCM 323 using SPI communication. These result in accurate control and feedback of the position and orientation of the battery pack. Figure 2.13 describes the electronic system of the pitch and roll mechanism.

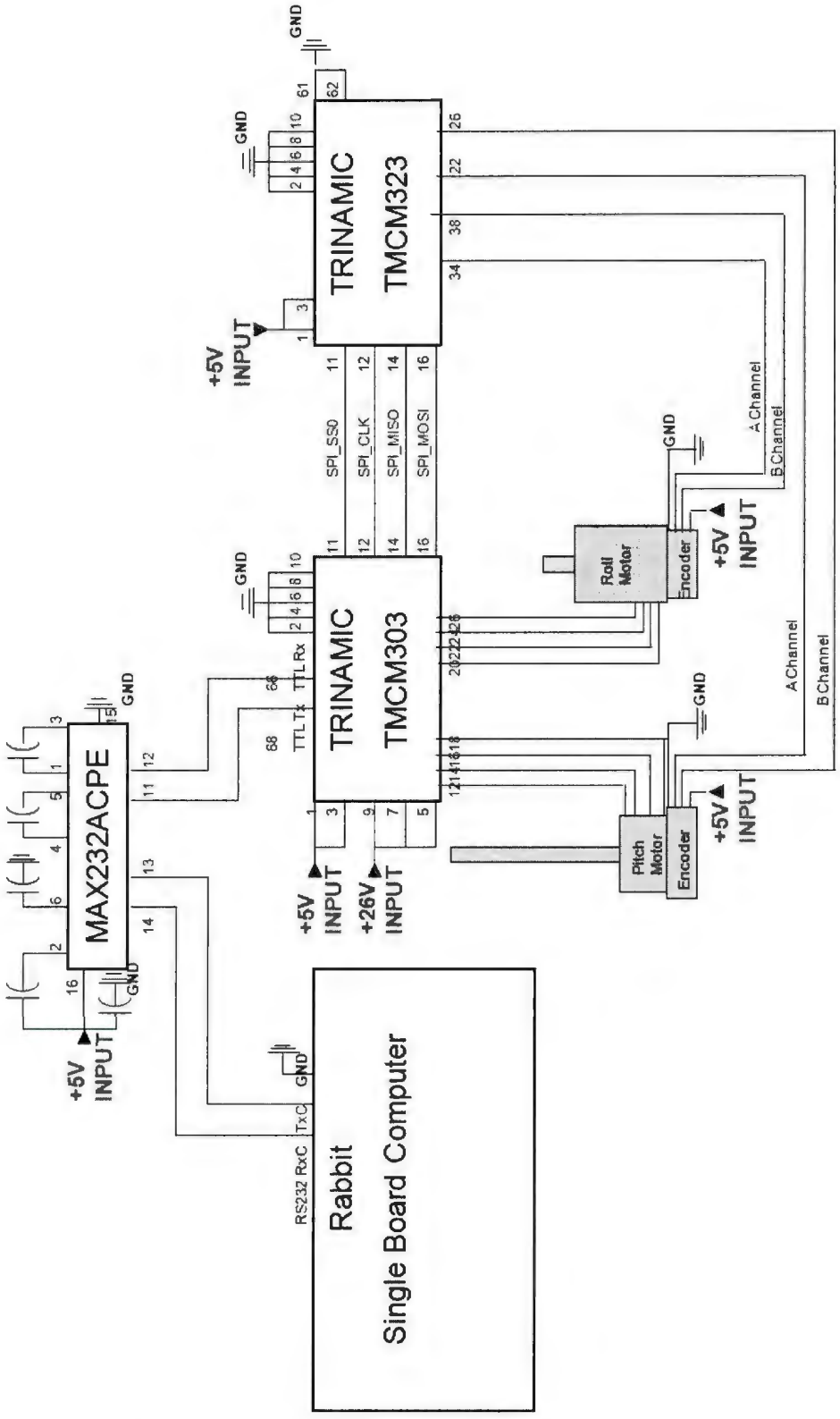


Figure 2.13: Sketch of the Pitch and Roll Mechanism Electrical System

The NEMA HT11 stepper motor is a standard 2 phase 1.8 degree hybrid step motor which can offer high holding torque (minimum 7 $oz \cdot in$, thus 0.0494 $N \cdot m$) which is sufficient[8].

The Haydon Kerk E26449 Can-Stack stepper motor is a 7.5 degree captive bipolar external linear actuator. The linear travel per step is 0.00025 $inch$. As a result, the linear travel per revolution is:

$$0.00025\ inch \times \frac{360}{7.5} = 0.012\ inch$$

The E8P encoders have a 500 CPR (cycles per revolution). As a result, we expect the linear travel of the pitch motor in $mm/resolution$ to be:

$$MMPerEncoderRes = \frac{0.012 \times 25.4}{2000} = 0.000154\ mm/resolution \tag{2.3}$$

Additionally, the rotating angle of the roll motor in $degree/resolution$ is:

$$DegreePerEncoderRes = \frac{360}{2000} = 0.18\ degree/resolution \tag{2.4}$$

Considering the gear ratio of the gear system, the relationship between battery pack rotation and encoder reading is:

$$BPDegree2EncoderRes = \frac{0.18\ degree}{6} = 0.03\ degree/resolution \tag{2.5}$$

The above two parameters will be used as constants in the future software development.

As an inverting level shifter is required by use of the TMCM303 RS232 communication, a MAX232ACPE is used for level shifting.

2.3.5 Inclinometer

The VTI SCA121T dual axis inclinometer is used for measuring the pitching angle and rolling angle of the IOT Glider. It has a measuring range of $\pm 90°$ and a high shock durability of $20000g$. The two analogue outputs from the inclinometer are connected to two channels on the BL2100 SBC. See Table 2.2 for details. Figure 2.14 shows a photo of the sensor.

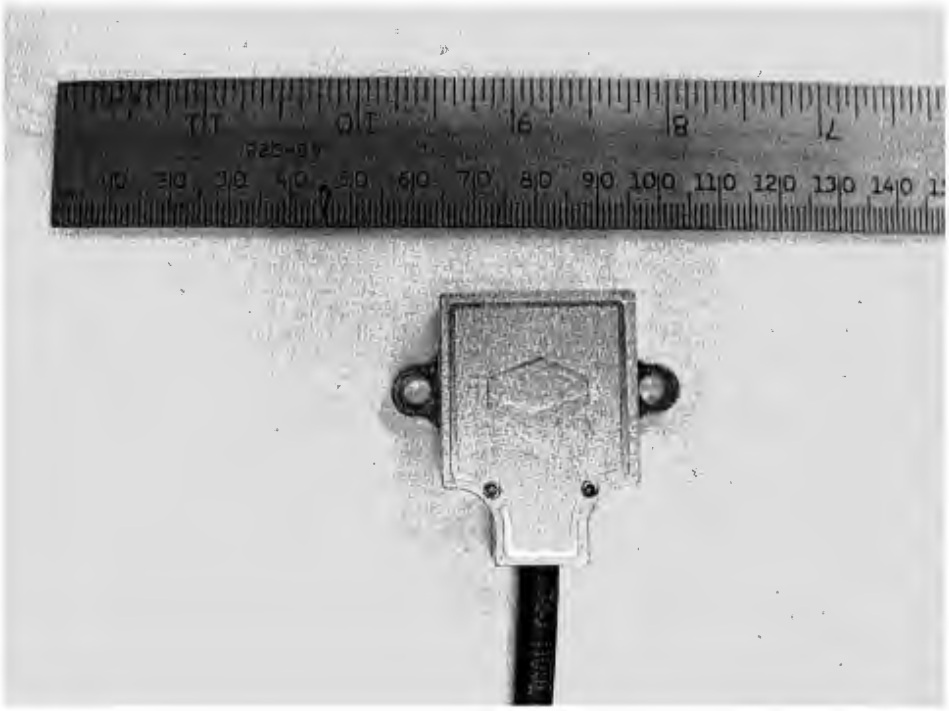


Figure 2.14: Inclinometer VTI SCA121T

2.3.6 Pressure Sensors

The two pressure sensors used on the IOT Glider both come from OMEGA Engineering Inc. The PX138 absolute pressure sensor is used for monitoring the vacuum inside the glider; the PX303 absolute pressure sensor measures the pressure external to the vehicle. Using the pressure reading, the depth of the vehicle can be determined using

Equation 2.6.

$$h(m) = \frac{PressureReading(Pa) - P_0}{\rho g} \tag{2.6}$$

where h is the depth value in water, ρ is the density of the water, P_0 is the pressure in the open air on the ground and g is the gravity acceleration.

The output pins of the PX138 and PX303 are connected to two analogue input channels on the BL2100 SBC.

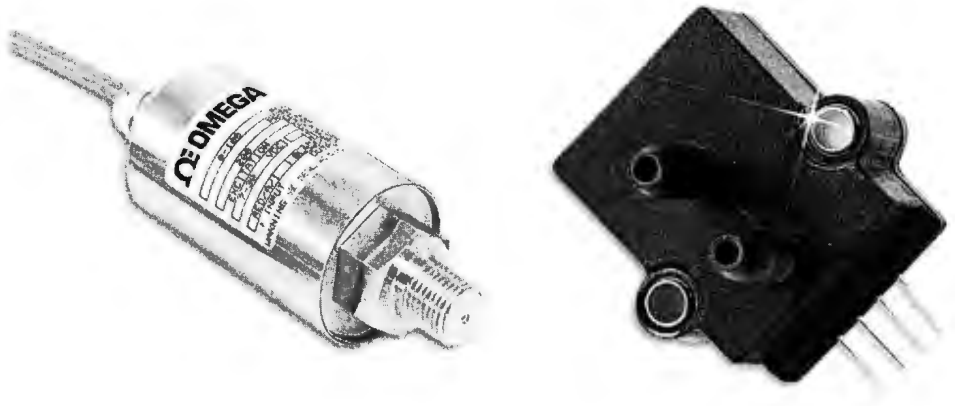


Figure 2.15: Pressure Sensors (Left PX303, Right PX138)

2.4 Software Development

The IOT Glider is controlled by the BL2100 single board computer which can be programmed by the Dynamic C on a PC. Several custom library files for better use of the equipment on the glider are developed for future applications.

2.4.1 Functions in 'INCLINOMETER.LIB'

This library helps with communication between the BL2100 and the inclinometer. The following is the function list and the associated descriptions.

float readPitch(void) This function reads the X-axis voltage output of the inclinometer. It then calculates the pitch angle of the glider and returns the data in degrees. The following relationship is used from the data sheet.

$$PitchAngle(degree) = arcsin((ADCRead(V) - 2.5V) * INC\_GAIN) \quad (2.7)$$

where INC_GAIN is 0.5 which can be obtained from the data sheet. 2.5 V is given as the offset on the data sheet.

float readRoll(void) This function reads the Y-axis voltage output of the inclinometer. It then calculates the roll angle of the glider and returns the data in degrees using the following relationship from the data sheet.

$$RollAngle(degree) = arcsin((ADCRead(V) - 2.5V) * INC\_GAIN) \quad (2.8)$$

where INC_GAIN is also 0.5 which can be obtained from the data sheet. $2.5V$ is given as the offset on the data sheet.

2.4.2 Functions in 'TMCMcommunication.LIB'

This library is written for communication between the BL2100 SBC and the TMCM driver modules and for control of the two stepper motors. The functions include sending commands to TMCM 303 via RS232 port C on BL2100 and reading replies from the TMCM303 via RS232.

void setupSerialPortC(void) This function sets up serial port C which is connected with the TMCM303 module. This function also opens serial port C at 9600 baud rate with no flow control. It allows communication with the TCMC303.

void msDelay(unsigned int delay) This function allows for delay. The program will be paused for a period of time in milliseconds.

char* serCgets(char* buffer) This function reads in a string of characters from serial port C into a buffer. It's used for receiving reply information from the TMCM303 module.

char ROR(char Motor, int Speed) This function sends a command to the TMCM303 via serial port C to make one of the motors rotate clockwise at a specified speed.

char ROL(char Motor, int Speed) This function sends a command to the TMCM303 via serial port C to make one of the motors rotate counter-clockwise at a specified speed.

char MST(char Motor) This function sends a command to the TMCM303 via serial port C to make one of the motors stop.

char MVP(char Type, char Motor, int Value) This function sends a command to the TMCM303 via serial port C to move motors. There are two types of movement: one is moving to an absolute position and the other is moving to a relative position.

char SAP(char Paramter, char Motor, int Value) This function sends a command to the TMCM303 via serial port C to adjust axis parameters of the TMCM303 module. The six parameters which would be used during any future application are listed below as Table 2.3. Adjusting these parameters for both of the motors can better control their movement.

int GAP(char Paramter, char Motor) This function sends a command to the TMCM303 via serial port C to get an axis parameter of the TMCM303 module.

Table 2.3: Axis Parameters for TMCMs

Defined Name	Description	Axis parameter number in TMCM 303 module
ActualPosition	The present position of the motor	1
MaxSpeed	Maximum positioning speed	4
MaxAcce	Maximum acceleration (deceleration)	5
MaxCurrent	Absolute maximum current value	6
StandCurrent	The current limit two seconds after the motor has stopped	7
Microstep	Microstep resolution	140

The six parameters which will be used during a application are the same as the SAP function.

float readRMAngle(void) This function reads the roll motor position from the attached encoder and determines the battery pack present rotation angle in degrees.

float readPMPos(void) This function reads the pitch motor position from the attached encoder and determines the battery pack present linear position in millimetres.

char MTP(char Motor, float Pos) This function moves or rotates one of the motors and moves the battery pack to the specified position, judging by feedback from encoder signals.

2.4.3 Other Custom Libraries

Another four custom libraries were also developed. The 'BE_Pressure.LIB' provides functions used for reading the two pressure sensors. The 'BE.LIB' contains functions for controlling the buoyancy engine [15]. The 'PCcommunication.LIB' provides functions used for talking between the BL2100 SBC and a PC via serial communication. The 'ADC.LIB' helps with converting an analogue signal to digital volts data.

2.5 Summary

In this chapter, the design details of the IOT Glider mechanical systems, electrical systems and the software development are described. Now the IOT Glider is operational in the saw-tooth mode and is also capable of active pitch and roll motion control using the internal pitch and roll mechanism.

To better study the movement of gliders, a six-degree of freedom motion simulation model is developed.

Chapter 3

A Six-Degree of Freedom Simulation Model for the IOT Underwater Glider

3.1 Introduction

To better understand the motion patterns of underwater gliders, a numerical simulation model is necessary. As the motion equations already exist [6], for simulation of underwater vehicles, the main work is the evaluation of the external forces on them while they move. Previous underwater vehicle simulation work [11] evaluated the external forces and moments on the vehicle based on the assumption that they come from hydrostatics, hydrodynamic damping and added mass effects. The hydrodynamic damping should be from both translational motion and rotational motion. In [11], the discussed AUV doesn't possess wings and the author treated the hydrodynamic damping force as separated axial drag , rolling drag and cross-flow drag. In Stante's simulation work [14] on the SLOCUM glider, one of the weak points is that

the author didn't consider the hydrodynamic damping from the rotational motion. In this work, damping effects from the translational motion will be treated as drag and lift forces on the main hull, fins and wings using CFD analysis. The evaluation of damping effects from rotational motions will be using the strip theory and empirical formulas as in Prestero's work. The added mass and added moments of inertia will be evaluated by an existing program, 'ESAM'.

The model which will be discussed in this chapter differs from a propeller driven AUV. The differences include:

- As stated before, there are two swept wings on the IOT Glider.
- The IOT Glider doesn't have propellers.
- Operation of the buoyancy engine and the pitch and roll mechanism on the IOT Glider will change its centre of gravity, moment of inertia tensor, and buoyancy.
- There is no rudder for turning the IOT Glider.

For accurate simulation of the glider, all of these factors should be considered properly.

3.2 Governing Equations

3.2.1 Vehicle Body-Fixed Coordinate System

The origin of the body fixed coordinate for the IOT Glider is fixed with regard to the glider hull. The origin point is 0.72 m aft of the nose tip, and it is at the center point of the cross section. Figure 3.1 shows the vehicle body-fixed coordinate system in the simulation model.

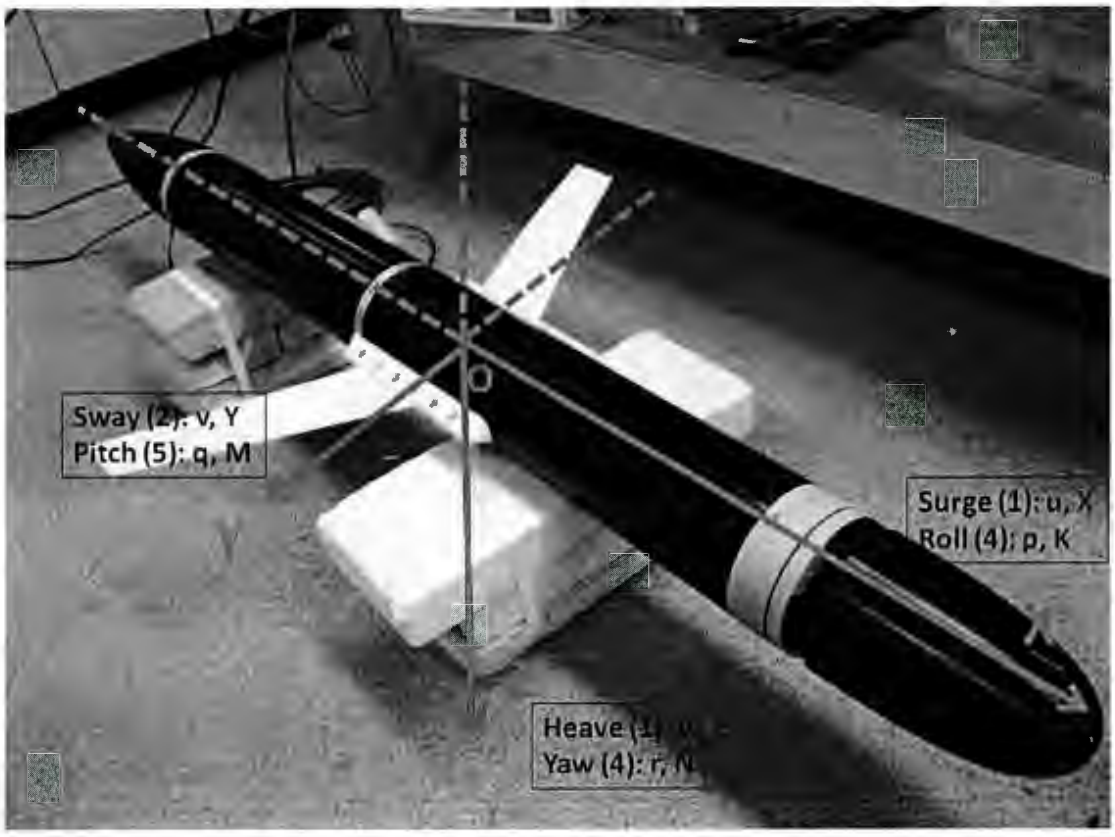


Figure 3.1: The Vehicle Body-Fixed Coordinate System

3.2.2 Vehicle Kinematics

The general motion of a rigid body in six degrees of freedom includes translational motion and rotational motion. The following vectors describe the general motion of the IOT Glider:

$$\eta_1 = [x\ y\ z]^T; \qquad \eta_2 = [\phi\ \theta\ \psi]^T$$
$$\upsilon_1 = [u\ v\ w]^T; \qquad \upsilon_2 = [p\ q\ r]^T$$
$$\tau_1 = [X\ Y\ Z]^T; \quad \tau_2 = [K\ M\ N]^T$$

where η describes the position and orientation of the glider with respect to the inertial coordinate system, υ the translational and rotational velocities of the glider with respect to the body-fixed coordinate system, and τ the total external forces and moments acting on the glider with respect to the body-fixed coordinate system. The following coordinate transform relates translational velocities between body-fixed and inertial coordinate systems:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = J_1(\eta_2) \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{3.1}$$

where $J_1(\eta_2) =$

$$\begin{bmatrix} \cos\psi\cos\theta & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi & \sin\phi\sin\psi + \cos\phi\sin\theta\cos\psi \\ \sin\psi\cos\theta & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\cos\phi\sin\psi + \sin\phi\sin\theta\cos\psi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \tag{3.2}$$

Note that $J_1(\eta_2)$ is orthogonal:

$$(J_1(\eta_2))^{-1} = (J_1(\eta_2))^T \tag{3.3}$$

In addition, the following coordinate transform relates rotational velocities between body-fixed and inertial coordinate systems:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = J_2(\eta_2) \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{3.4}$$

where

$$J_2(\eta_2) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi/\cos\theta & \cos\phi/\cos\theta \end{bmatrix} \tag{3.5}$$

3.2.3 Vehicle Rigid Body Dynamics

As stated in the previous section, the origin of the body-fixed coordinate system is located at a fixed point in the glider. The equations of motion for the IOT Glider in

six degrees of freedom can be written as:

$$
\begin{aligned}
&m[\dot{u} - vr + wq - xG(q^2 + r^2) + yG(pq - \dot{r}) + zG(pr + \dot{q})] = \sum X_{ext}, \\
&m[\dot{v} - wp + ur - yG(r^2 + p^2) + zG(qr - \dot{p}) + yG(qp + \dot{r})] = \sum Y_{ext}, \\
&m[\dot{w} - uq + vp - zG(p^2 + q^2) + xG(rp - \dot{q}) + yG(rq + \dot{p})] = \sum Z_{ext}, \\
&I_{xx}\dot{p} + (I_{zz} - I_{yy})qr - (\dot{r} + pq)I_{xz} + (r^2 - q^2)I_{yz} + (pr - \dot{q})I_{xy} \\
&+m[yG(\dot{w} - uq + vp) - zG(\dot{v} - wp + ur)] = \sum K_{ext}, \\
&I_{yy}\dot{q} + (I_{xx} - I_{zz})rp - (\dot{p} + qr)I_{xy} + (p^2 - r^2)I_{xz} + (qp - \dot{r})I_{yz} \\
&+m[zG(\dot{u} - vr + wq) - xG(\dot{w} - uq + vp)] = \sum M_{ext}, \\
&I_{zz}\dot{r} + (I_{yy} - I_{xx})pq - (\dot{q} + rp)I_{yz} + (q^2 - p^2)I_{xy} + (rq - \dot{p})I_{xz} \\
&+m[xG(\dot{v} - wp + ur) - yG(\dot{u} - vr + wq)] = \sum N_{ext}.
\end{aligned}
\tag{3.6}
$$

where the location of the glider centre of gravity is expressed as:

$$
\mathbf{r_G} = \begin{bmatrix} xG \\ yG \\ zG \end{bmatrix}
$$

The moment of inertia tensor with respect to the body fixed frame is:

$$
\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}
$$

Note that for the IOT Glider, expelling or injecting water does not change the vehicle's mass. As a result, the mass m is the vehicle dry mass and is considered as

constant.

The external forces and moments are functions of the velocities and other state variables. The external forces and moments will be derived and evaluated in the following sections.

3.3 Change of Mass Properties During Operations

As mentioned in previous sections, the IOT Gliders relies on moving its piston and battery pack to dive or rise in the water. The movement of these devices will change the buoyancy force and mass properties of the glider. It is necessary to re-evaluate these properties at each time step. The following variables in Table 3.1 are considered in the simulation model.

Table 3.1: List of Changing Mass Properties

Syntax	Description	Cause of changes
B	Buoyancy force	Piston movement in buoyancy engine
xB	Longitudinal position of buoyancy force	Piston movement in buoyancy engine
xG	Longitudinal position of gravity force	Piston movement in buoyancy engine, translational movement of battery pack
yG	Position of gravity force in Y direction	Rotation of battery pack
zG	Position of gravity force in Z direction	Rotation of battery pack
I	Inertia tensor	Piston movement in buoyancy engine, translational movement of battery pack, rotation of battery pack

3.3.1 B and xB Evaluation

Moving of the buoyancy engine will cause the overall buoyancy force B and center of buoyancy xB to change. The change is from extending or retracting the piston. As a result, the buoyancy force and centre of buoyancy can be determined by the following equations.

$$B = B_0 + x\_piston \times A\_piston \times \rho \times g \tag{3.7}$$

where B_0 is the buoyancy force when the piston is in its neutral position (corresponding to the neutrally buoyant state in Figure 2.7 and might differ from its zero position), and $B_0 = W$, where W is the weight of the glider when submerged; x_piston is the X-axis position of the piston movement relative to the neutral position; A_piston is the frontal area of the piston.

$$xB = \frac{B_0 \times xB\_0 + x\_piston \times A\_piston \times \rho \times g \times x'}{B} \tag{3.8}$$

where xB_0 is the original X-axis position of the buoyancy force with regard to the body-fixed frame; x' is the X-axis position of the geometry center of the expelled or injected ballast water with regard to the body-fixed frame.

3.3.2 xG, yG and zG Evaluation

The moving of the buoyancy engine and battery pack both have effects on the change of the center of gravity position. Taking these factors into account, the center of gravity position in three directions can be evaluated by the following equations.

$$xG = \frac{m \times xG\_0 + x\_piston \times m\_piston + x\_batt \times m\_batt}{m} \tag{3.9}$$

where xG_0 is the X-axis position of the glider center of gravity when both the buoyancy engine piston and the battery pack are in their neutral positions (which might differ from its zero position).

$$yG = \frac{m \times yG\_0 + rG\_batt \times \sin(theta\_batt) \times m\_batt}{m} \tag{3.10}$$

where yG_0 is the Y-axis position of the glider's center of gravity when both the buoyancy engine piston and the battery pack are in their zero positions; rG_batt is the distance from the overall battery pack center of gravity to the glider's central axis; $theta\_batt$ is the angle the battery pack rotates with regard to their zero positions.

$$zG = \frac{m \times zG\_0 - rG\_batt \times \cos(theta\_batt) \times m\_batt}{m} \tag{3.11}$$

where zG_0 is the Z position of the glider center of gravity when both the buoyancy engine piston and the battery pack are in their zero positions. Figure 2.9 shows that rG_batt is 19.64 mm.

3.3.3 Moment of Inertia Evaluation

Due to the moving pattern of the buoyancy engine piston and battery pack, it is obvious that Ixx never changes. Considering the profile of the IOT glider and the symmetry of the glider arrangement, Iyy and Izz of the IOT glider are always close. It is assumed that $Iyy = Izz$. They can be evaluated by the following equation.

$$Iyy = Iyy' + m\_piston \times (X\_piston^2 - X\_piston'^2) + m\_batt \times (X\_batt^2 - X\_batt'^2) \tag{3.12}$$

where Iyy' is the original Iyy value; X_piston is the X position of the piston in the body-fixed frame; X_piston' is the previous X position of the piston in the body-fixed

frame; X_{batt} is the X position of the battery pack in the body-fixed frame; $X_{batt'}$ is the previous X position of the battery pack in the body-fixed frame.

3.4 Hydrostatics

The hydrostatic forces come from the effects of the glider's weight and buoyancy. The hydrostatic forces can be evaluated using the equations in the previous section. Then the hydrostatic forces and moments on the glider can be expressed as:

$$\begin{aligned} \mathbf{F_{HS}} &= \mathbf{G} - \mathbf{B} \\ \mathbf{M_{HS}} &= \mathbf{r_G} \times \mathbf{G} - \mathbf{r_B} \times \mathbf{B} \end{aligned} \tag{3.13}$$

Note that all of the above vector terms should be expressed in the body fixed frame.

3.5 Hydrodynamic Drag and Lift Forces

The evaluation of the hydrodynamic effect on the IOT Glider uses a CFD analysis technique. In the analysis, we treat the glider hull, wings and fins separately, using a component-buildup strategy.

3.5.1 Hull/Main Body

The evaluation of the hydrodynamic forces and moments on the IOT Glider's hull is based on the experimental results from the Phoenix AUV model tests [18]. In the Phoenix tests, the overall length to diameter ratio is between 8.5 and 12.5, while the ratio for the IOT Glider is 13.55. However, as the hydrodynamic performance turned out to be almost the same at different length to diameter ratios, we assume that when the ratio reaches 13.55, the results are still applicable.

Beside the experimental data, CFD analysis on the hull is conducted as well. Comparison between the data from the two strategies shows close results. The results from experimental data are used in the simulation model. Details on the CFD analysis and a comparison can be found in the next chapter.

3.5.2 Wings

As there are no available hydrodynamic drag and lift data found for a swept flat plate, the evaluation of the hydrodynamic forces and moments on the glider wings is based on the CFD method. This part of the work is detailed in the next chapter. The hydrodynamic effects on the wings will be expressed as drag forces, lift forces and pitch moments. The relationship between them and the local angle of attack was found.

3.6 Hydrodynamic Damping

The hydrodynamic rotational damping coefficient evaluation is based on summing up the drag of two-dimensional cylindrical cross-sections along the glider's body. This method is analogous to strip theory. This results in the following expression for the damping forces and moments. An example is shown below. Note that wings and fins are considered separately.

$$Y = \int_{tail}^{nose} -\frac{1}{2}\rho c_{dc} v|v| d(x) dx - 2 \times (\frac{1}{2}\rho v_{fin}|v_{fin}| S_{fin} c_{df}) \tag{3.14}$$

where c_{dc} is the drag coefficient for the two-dimensional circular cylinder cross-section, Hoerner [9] estimates it to be 1.1; x is the location of the cross section in the body-fixed frame; v is the local body velocity relative to the surrounding water; $d(x)$ is the

diameter of the local cross-section; S_{fin} is the area of a single fin; and c_{df} is the drag coefficient of a thin rectangular plate perpendicular to the flow. It is estimated to be 1.05 [3].

The damping forces and moments on the vehicle can be also expressed using damping coefficients. For example the lateral force Y can be expressed as:

$$Y = Y_{r|r|} r|r| \tag{3.15}$$

where $Y_{r|r|}$, the damping coefficient, can be determined by the following relationship.

$$Y_{r|r|} = -\frac{1}{2}\rho c_{dc} \int_{tail}^{nose} x|x|d(x)dx - 2x_{fin}|x_{fin}| \left(\frac{1}{2}\rho S_{fin} c_{df}\right) \tag{3.16}$$

Using the same strategy, we can get other damping coefficients $Z_{q|q|}$, $N_{r|r|}$, $M_{q|q|}$ and $K_{p|p|}$ as follows.

$$\begin{aligned} Z_{q|q|} =& \frac{1}{2}\rho c_{dc} \int_{tail}^{nose} x|x|d(x)dx + 2x_{fin}|x_{fin}| \left(\frac{1}{2}\rho S_{fin} c_{df}\right) \\ &+ 2x_{WingCenter}|x_{WingCenter}| \left(\frac{1}{2}\rho S_{Wing} c_{dw}\right) \end{aligned} \tag{3.17}$$

The wings are assumed to be rectangular plates here, c_{dw} is the drag coefficient of a thin rectangular plate perpendicular to flow, and is estimated to be 1.12 [3]. $x_{WingCenter}$ is the X position of the geometry centre of the wings in the glider body-fixed frame.

$$N_{r|r|} = -\frac{1}{2}\rho c_{dc} \int_{tail}^{nose} x^3 d(x)dx - 2x_{fin}^3 \left(\frac{1}{2}\rho S_{fin} c_{df}\right) \tag{3.18}$$

$$\begin{aligned} M_{q|q|} =& -\frac{1}{2}\rho c_{dc} \int_{tail}^{nose} x^3 d(x)dx - 2x_{fin}^3 \left(\frac{1}{2}\rho S_{fin} c_{df}\right) \\ &- 2x_{WingCenter}^3 \left(\frac{1}{2}\rho S_{Wing} c_{dw}\right) \end{aligned} \tag{3.19}$$

$$
\begin{aligned}
K_{p|p|} = &-\frac{1}{2}\rho c_{dl}\int_{tip}^{root} 2x^3 l(x)dx \\
&- 4h_{fin}^3\left(\frac{1}{2}c_{df}\rho S_{fin}\right) \\
&- 2\left(\frac{D}{2}+\frac{h_{WingMount}}{2}\right)^3\left(\frac{1}{2}c_{dl}\rho S_{WingMount}\right)
\end{aligned}
\tag{3.20}
$$

Here c_{dl} is the drag coefficient of a thin flat plate two-dimensional bluff section. It is estimated to be 1.98 [3]; h_{fin} is the radial distance of the fin center from the longitudinal axis of the IOT Glider; $h_{WingMount}$ is the hight of the wing mounts. The glider's constant parameters for these equations evaluating damping coefficients are summarized in Table 3.2. Using MATLAB codes shown in the Appendix E, the

Table 3.2: Glider Constant Parameters for Hydrodynamic Damping Calculation

Parameters	Value	Unit
x_{fin}	-0.3	m
S_{fin}	0.0036	m^2
$x_{WingCenter}$	0.1	m
S_{Wing}	0.02	m^2
h_{fin}	0.09	m
$S_{WingMount}$	0.004	m^2
$h_{WingMount}$	0.2	m
l_{wing}	0.3	m

hydrodynamic damping coefficients are summarized in Table 3.3.

Table 3.3: List of Damping Coefficients

Parameter	Value	Units
Yrr	1.20	$kg \cdot m/rad^2$
Kpp	-0.355	$kg \cdot m^2/rad^2$
Mqq	-8.38	$kg \cdot m^2/rad^2$
Nrr	-8.35	$kg \cdot m^2/rad^2$
Zqq	-0.922	$kg \cdot m/rad^2$

3.7 Added Mass Evaluation

As the added mass effect is essential for the accurate prediction of underwater vehicle performance, it is also evaluated for the IOT Glider.

This part of the work was accomplished numerically by means of ESAM (Estimation of Submarine Added Mass) program which was developed by the Defence Research Establishment Atlantic (DREA) in Dartmouth, Nova Scotia, Canada[16]. The ESAM method considers a submarine as a multi-component rigid body and takes the interaction between the hull and appendages into account. The components of the analysed submarine are treated as best fit ellipsoids. Then the added mass is evaluated based on the replaced ellipsoids which build up the overall added mass of the whole vehicle. From the output data from the ESAM, a 3-D SolidWorks drawing in Figure 3.2 is made to compare the real glider shape and the actual analysed shape in the ESAM. The estimated added mass matrix is :

Figure 3.2: IOT Glider in ESAM analysis

$$M\_AM = \begin{bmatrix} -0.1708 & 0 & 0 & 0 & 0 & 0 \\ 0 & -14.20 & 0 & 0 & 0 & 0.201 \\ 0 & 0 & -16.66 & 0 & -0.2612 & 0 \\ 0 & 0 & 0 & -0.0706 & 0 & 0 \\ 0 & 0 & -0.2612 & 0 & -2.161 & 0 \\ 0 & 0.201 & 0 & 0 & 0 & -2.152 \end{bmatrix} \quad (3.21)$$

Note that the result is based on the most aft wing position. Change of wing position results in a slight difference in the values of the m_{35}, and m_{53} elements. By moving the wings from its most aft position to its most forward position, m_{53} changes from $-0.2612\ kg \cdot m/rad$ to $-0.0795\ kg \cdot m/rad$.

3.8 Summary

This chapter describes the development of a numerical simulation model for the IOT Glider. It also presents the evaluation of mass properties, hydrostatic forces, hydrodynamic drag and lift forces, hydrodynamic damping forces, and added mass. Different strategies were used in these evaluations and comparisons are made. The MATLAB codes developed are shown in Appendix F. The details of the CFD analysis will be provided in the next chapter.

Chapter 4

CFD Analysis on the IOT Glider Wings and the Bare Hull

4.1 Introduction

The hydrodynamic forces and moments on the IOT Glider include forces and moments on the glider wings and its bare hull. The drag force, lift force and pitch moment are evaluated by means of a set of numerical simulations utilizing CFD methodology. The relationship between the non-dimensional coefficients and angle of attack are determined using curve fitting. The CFD commercial code COMSOL is used in this analysis. This chapter presents the details of the CFD analysis. The results for the bare hull are compared with experimental results from the Phoenix tests.

4.2 Numerical Models

4.2.1 CFD Code, COMSOL

COMSOL Multiphysics is an integrated environment for solving multi-physics time dependent or stationary systems in one, two, and three dimensions. It provides sophisticated and convenient tools for choosing the physics, the study mode and for defining the turbulence model. The Autonomous Oceans Systems Laboratory has a licence for the COMSOL package which was taken advantage of in this component of the work.

4.2.2 Turbulence Models

For analysing the hydrodynamic coefficients of the wings and the bare hull, the 3D stationary studies were chosen. The analysed objects are fixed in the fluid domain space, not attached to anything. The uniform flow is passing through the objects by a specified speed and angle of attack. The objects were arranged in a duct, in which a uniform flow passes through. The duct has enough size to allow the flow to fully develop in it. The flow velocity was defined in the inlet of the duct in this model, as the incoming flow. The outlet was defined such that pressure there was zero and there was no viscous stress. To change the angle of attack, the duct geometry was rotated by a certain angle. The object was modelled by using the wall functions in COMSOL to generate boundary layers. The surrounding four walls of the duct were set as slip walls.

Within the whole flow domain, the flow was assumed as incompressible. The standard $k-\varepsilon$ turbulence model was chosen as it is used in the prediction of most turbulence flows because of its robustness, economy and reasonable accuracy for a wide range of flows.

4.3 CFD Analysis of the Wings

4.3.1 Design of Numerical Experiments

The hydrodynamic forces and moments acting on the glider wings when the glider is moving in the water are equivalently divided into drag force, lift force and pitch moment. The drag force is the component defined as in the direction of the incoming flow, while lift force is perpendicular to the incoming flow. In most cases, the yaw moment and roll moment will cancel out due to the symmetry of wing location.

The hydrodynamic drag and lift forces and pitch moment acting on each wing can be calculated as in Equation 4.1 - 4.3.

$$L = \frac{1}{2}\rho V^2 S_{Wing} C_L(\alpha_w, V) \tag{4.1}$$

$$D = \frac{1}{2}\rho V^2 S_{Wing} C_D(\alpha_w, V) \tag{4.2}$$

$$M = \frac{1}{2}\rho V^2 S_{Wing} l_{wing} C_M(\alpha_w, V) \tag{4.3}$$

where ρ is the density of the water, V is the incoming flow velocity, S_{Wing} is the area of the wing, l_{wing} is the wing tip-to-root length along the mid-chord line which can be found in Table 3.2 and $C_L(\alpha_w, V)$, $C_D(\alpha_w, V)$, $C_M(\alpha_w, V)$ are lift, drag and pitch moment coefficients of the wing at the local angle of attack (AOA) α_w respectively. The work completed in this chapter is finding an accurate relationship between these coefficients and the angle of attack, by means of CFD simulations.

To achieve this goal, a set of simulations was designed. Two independent variables were studied. One is the incoming flow velocity and the other is the angle of attack of the flow relative to the wing. During normal operations, the gliders will have a velocity of no more than $1m/s$ and an angle of attack of less than 10 *degree* [4] [7].

The experiment design is shown as below. Taking the l_{Wing} as the characteristic length, the Reynolds Number can be evaluated using Equation 4.4.

$$Re_{wing} = \frac{V l_{wing}}{\nu} \tag{4.4}$$

where the kinematic viscosity ν is taken at $20°C$ to be 1.308×10^{-6} for fresh water; value of l_{wing} can be found in Table 3.2.

Table 4.1: Numerical Experiment Design for the Wings

Run Number	Inclined Angle (deg)	Flow Velocity (m/s)	Re Number
1	2	0.5	1.147E5
2	2	1.0	2.294E5
3	2	1.5	3.440E5
4	5	0.5	1.147E5
5	5	1.0	2.294E5
6	5	1.5	3.440E5
7	10	0.5	1.147E5
8	10	1.0	2.294E5
9	10	1.5	3.440E5
10	15	0.5	1.147E5
11	15	1.0	2.294E5
12	15	1.5	3.440E5
13	20	0.5	1.147E5
14	20	1.0	2.294E5
15	20	1.5	3.440E5

4.3.2 Simulation Results and Data Processing

Below is an example case study result. The inclined angle of the flow is 20° and the velocity is $1m/s$. Figure 4.1 shows the velocity distribution around the wing profile.

In the meshes of the wing CFD model, the maximum element size is 0.06 m; the minimum element size is 0.019 m. The results obtained from COMSOL stationary studies include three velocity components, in x y and z directions and pressure values

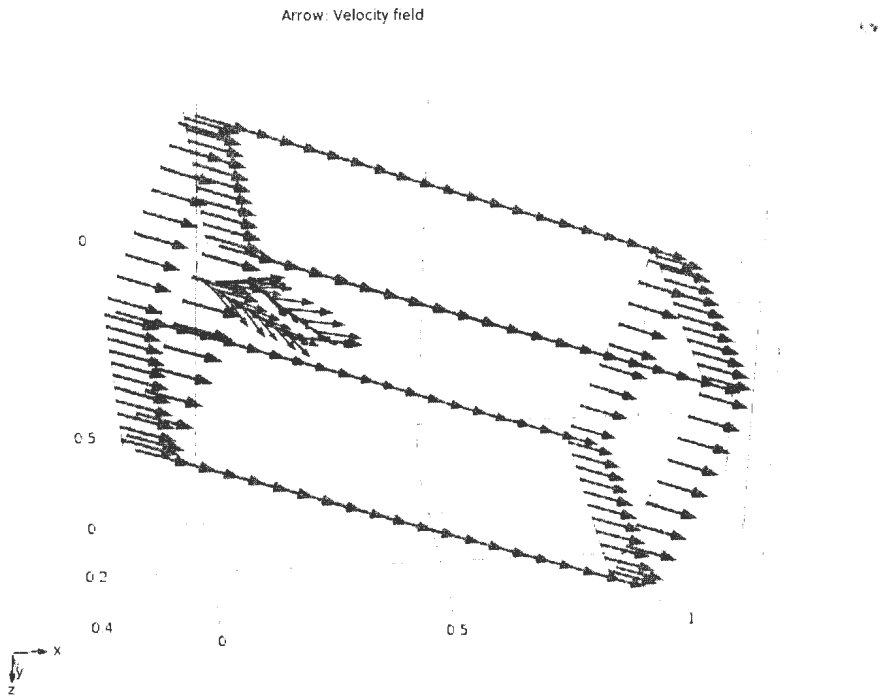


Figure 4.1: Example CFD case result (V=1m/s, $\alpha = 20°$)

for each element within the whole fluid domain. It was found no significant difference on the results using a 'fine' or 'normal' mesh in COMSOL. A finer mesh will result in more accurate prediction but will involve a much longer consumption of time and more memory usage.

As it is unable to get forces or moment value acting on the wing directly from the CFD code because the CFD code targets the fluid domain, thus we need to calculate them by using the pressure distribution surrounding the wing. Note that the origin point of the coordinate system in the model is at the leading edge at the root.

First we derive the force acting on the wing. In theory, it is the surface integral of the pressure. Note that a pressure value is a scalar and a force value is a vector.

$$\vec{\mathbf{F}} = \iint_{S_0} p_i \cdot \vec{\mathbf{n}}_i dS \tag{4.5}$$

where S_0 is the whole wing surface, p_i is the pressure distribution, and $\vec{\mathbf{n}_i}$ is the unit vector normal to the wing surface. If we present $\vec{\mathbf{n}_i}$ as $\vec{\mathbf{n}_i} = (n_{ix}, n_{iy}, n_{iz})$, the force acting on the wing can be expressed as:

$$F_x = \iint_{S_0} p_i \cdot n_{ix} dS \tag{4.6}$$

$$F_y = \iint_{S_0} p_i \cdot n_{iy} dS \tag{4.7}$$

$$F_z = \iint_{S_0} p_i \cdot n_{iz} dS \tag{4.8}$$

where $\vec{\mathbf{F}} = (F_x, F_y, F_z)$.

Now we translate the force vector into drag and lift forces.

$$D = F_x \cos(\alpha) + F_z \sin(\alpha) \tag{4.9}$$

$$L = F_x \sin(\alpha) + F_z \cos(\alpha) \tag{4.10}$$

where α is the inclined angle of the incoming flow in the test design. The moments can be derived using the following equation.

$$\vec{\mathbf{M}} = \iint_{S_0} \vec{\mathbf{r}_i} \times (\vec{\mathbf{n}_i} \cdot p_i) dS \tag{4.11}$$

where $\vec{\mathbf{r}_i}$ is the location of the elements on the wing surface. Note that the calculated moment is about the origin point in the COMSOL geometry model, which is located at the leading edge of the wing root. Later transfer of these moments to the glider body-fixed frame is necessary.

If we write $\vec{\mathbf{r}_i}$ as (r_{ix}, r_{iy}, r_{iz}) and again write $\vec{\mathbf{n}_i}$ as (n_{ix}, n_{iy}, n_{iz}), then we get:

$$M_x = \iint_{S_0} p_i \cdot (r_{iy} \cdot n_{iz} - r_{iz} \cdot n_{iy}) dS \tag{4.12}$$

$$M_y = \iint_{S_0} p_i \cdot (r_{iz} \cdot n_{ix} - r_{ix} \cdot n_{iz}) dS \tag{4.13}$$

$$M_z = \iint_{S_0} p_i \cdot (r_{ix} \cdot n_{iy} - r_{iy} \cdot n_{ix}) dS \tag{4.14}$$

After evaluating the drag, lift and pitch moment, the hydrodynamic non-dimensional coefficients are determined from equations 4.1, 4.2 and 4.3.

$$C_D(\alpha_w, V) = \frac{D(\alpha_w)}{\frac{1}{2}\rho V^2 S_{wing}} \tag{4.15}$$

$$C_L(\alpha_w, V) = \frac{L(\alpha_w)}{\frac{1}{2}\rho V^2 S_{wing}} \tag{4.16}$$

$$C_M(\alpha_w, V) = \frac{M(\alpha_w)}{\frac{1}{2}\rho V^2 S_{wing} l_{wing}} \tag{4.17}$$

Before curve fitting, we first define the local angle of attack on the wings.

As the wing is swept, tapered and in the form of a flat plate 3 mm thick, the wing local coordinate system is used for calculating the AOA on the wings. In order to more accurately capture its behaviour, we define a local wing body-fixed frame. The origin of this local frame is coincident with the origin of the body-fixed frame. The wing local frame is oriented along the wing's sweep line. That is to say, the body-fixed frame is rotated 33° about its Z-axis. Figure 4.2 shows the wing local coordinate system. The local angle of attack for the wing is determined from the following equation using the vehicle velocity components in the body-fixed frame.

$$\alpha_w = tan^{-1}(\frac{sin(tan^{-1}\frac{w}{u})}{cos(tan^{-1}\frac{w}{u})} sin(\Lambda_w)) \tag{4.18}$$

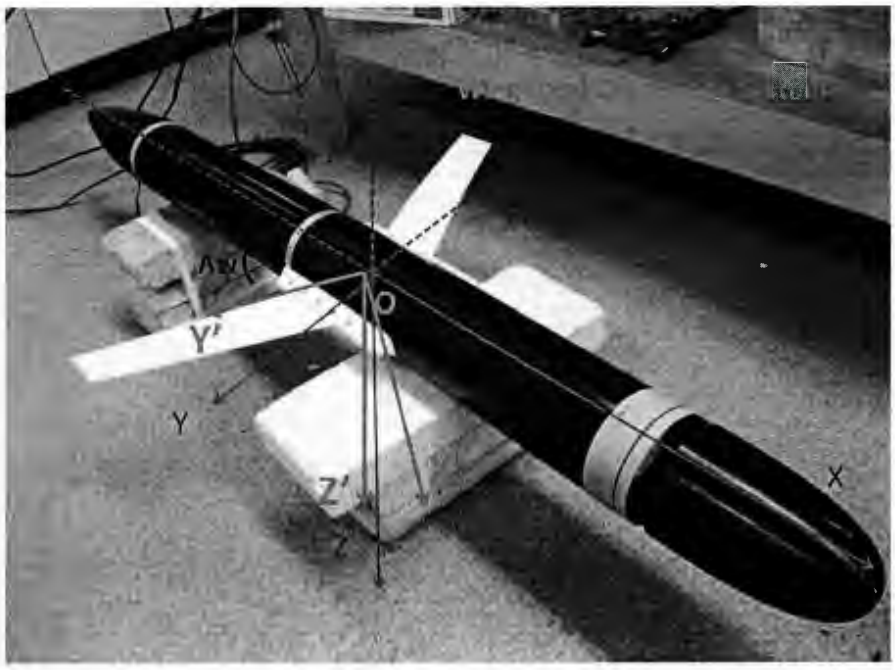


Figure 4.2: The Wing Local Coordinate System

where α_w is the local wing AOA, and Λ_w is the wing swept angle.

After running all the CFD cases, we can put all the results together as Table 4.2.

From the results we can see that the flow velocity has an insignificant effect on the hydrodynamic coefficients. As a result, we define them as functions of local wing AOA: $C_D, C_L, C_M = f(\alpha_w)$. Here, we use polynomials to present the relationships. As the movement velocity of underwater gliders is near 1 m/s, the final results from the 1 m/s tests are used for the simulation model. Due to the symmetry pattern of the drag, lift and pitch moment, we can easily determine the coefficients for a negative angle of attack. After extending the whole data, polynomials are used for the curve fitting, and the results are shown in Figures 4.3, 4.4 and 4.5. The polynomial expressions for the hydrodynamic coefficients are expressed as the following equations.

$$C_D = 0.00018551\alpha_w^2 + 0.0123 \tag{4.19}$$

Table 4.2: Wing CFD Analysis Results

Flow Inclined Angle(deg)	Wing Local AOA(deg)	V(m/s)	CD	CL	CM
2	2.38	0.5	0.0130	0.0386	-0.0146
2	2.38	1.0	0.0127	0.0317	-0.0120
2	2.38	1.5	0.0125	0.0282	-0.0107
5	5.96	0.5	0.0186	0.0880	-0.0303
5	5.96	1.0	0.0181	0.0830	-0.0283
5	5.96	1.5	0.0178	0.0805	-0.0273
10	11.87	0.5	0.0407	0.1748	-0.0583
10	11.87	1.0	0.0398	0.1702	-0.0564
10	11.87	1.5	0.0393	0.1680	-0.0555
15	17.72	0.5	0.0737	0.2324	-0.0759
15	17.72	1.0	0.0723	0.2277	-0.0738
15	17.72	1.5	0.0716	0.2255	-0.0729
20	23.46	0.5	0.1150	0.2842	-0.0951
20	23.46	1.0	0.1131	0.2793	-0.0930
20	23.46	1.5	0.1122	0.2769	-0.0920

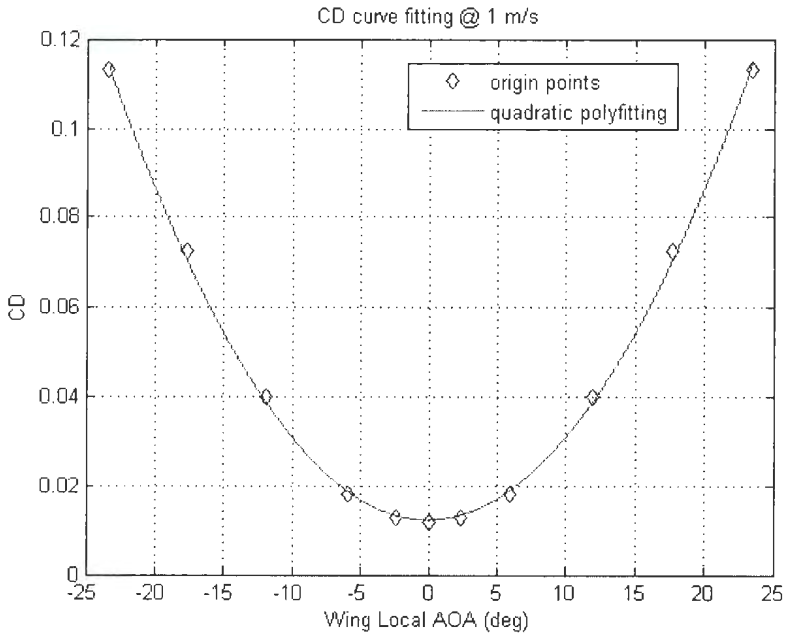


Figure 4.3: Wing Drag Coefficient Curve Fitting

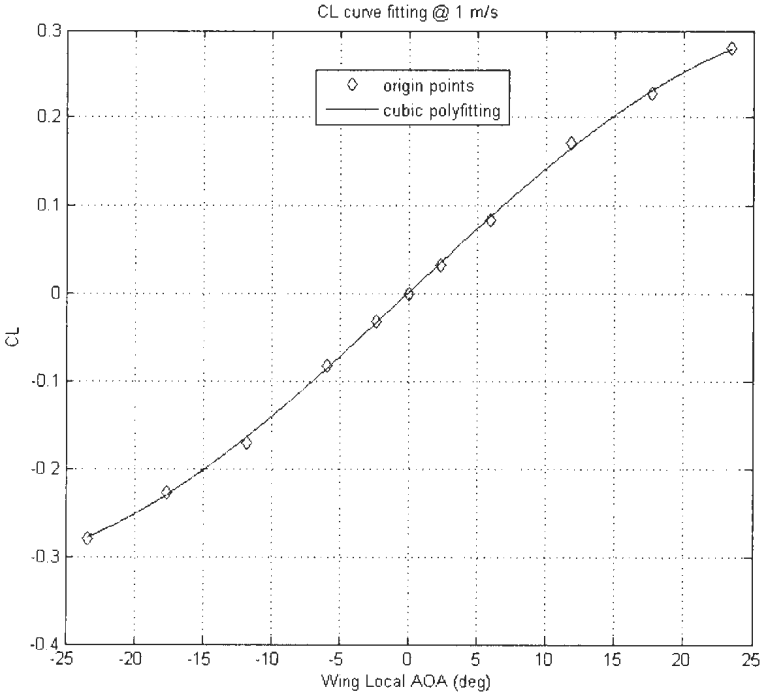


Figure 4.4: Wing Lift Coefficient Curve Fitting

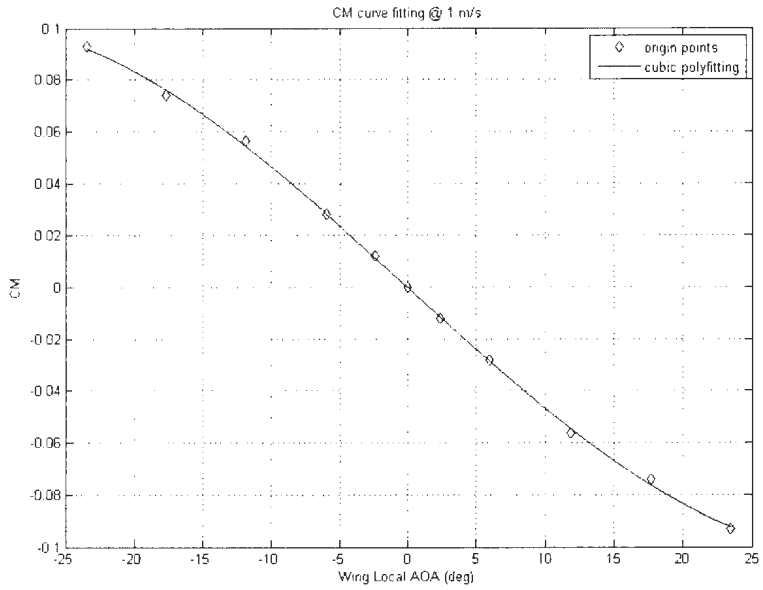


Figure 4.5: Wing Pitch Moment Curve Fitting (About the leading point of the wing root)

$$C_L = -0.000004922\alpha_w^3 + 0.0146\alpha_w \tag{4.20}$$

$$C_M = 0.0000016\alpha_w^3 - 0.0048\alpha_w \tag{4.21}$$

where α_w is measured in *degrees*.

4.4 CFD analysis of the Bare Hull

4.4.1 Design of Numerical Experiments

The hydrodynamic forces and moments acting on the glider bare hull when the glider is moving in the water are also equivalently divided into drag force, lift force and pitch moment. The drag force is the component defined as being in the direction of the incoming flow, while the lift force is perpendicular to the incoming flow.

The hydrodynamic drag and lift forces and pitch moment acting on the glider bare

hull can also be expressed as Equations 4.1 - 4.3. The work completed in this section is finding the accurate relationship between these coefficients and the angle of attack, by means of CFD simulations.

To achieve this goal, a set of simulations was designed. As can be seen in the previous section and in the results from the Phoenix tests[18], the incoming flow velocity variable has no effect on the dimensionless coefficients. Only one independent variable was studied for the bare hull, the angle of attack. We chose the AOA range of $\pm 20°$. Assuming that the performance of these coefficients is symmetrical, five cases are studied, 0°, 5°, 10°, 15°, 20°. All cases were studied using the incoming flow at $1m/s$. Taking the overall length as the characteristic length, the Reynolds Number is:

$$Re_{hull} = \frac{VL_{oa}}{\nu} = \frac{1 \times 1.56}{1.308 \times 10^{-6}} = 1.19 \times 10^{6} \tag{4.22}$$

where the kinematic viscosity ν is taken at a temperature of $20°C$ for fresh water.

4.4.2 Simulation Results and Data Processing

Below is an example case study result. The inclined angle of the flow is 20°. Figure 4.6 shows the velocity distribution around the bare hull profile. In the meshes of the bare hull CFD model, the maximum element size is 0.09 m; the minimum element size is 0.017 m.

Again, use Equations 4.5, 4.6, 4.11 and 4.12 to calculate forces and moments acting on the hull from the pressure distribution. Transfer the force into the body-fixed frame using Equations 4.9 and 4.10. Then use Equation 4.23, 4.24, and 4.25 to find the non-dimensional coefficients. The results can be found in Table 4.3.

$$C_D(\alpha) = \frac{D}{\frac{1}{2}\rho V^2 Volum^{2/3}} \tag{4.23}$$

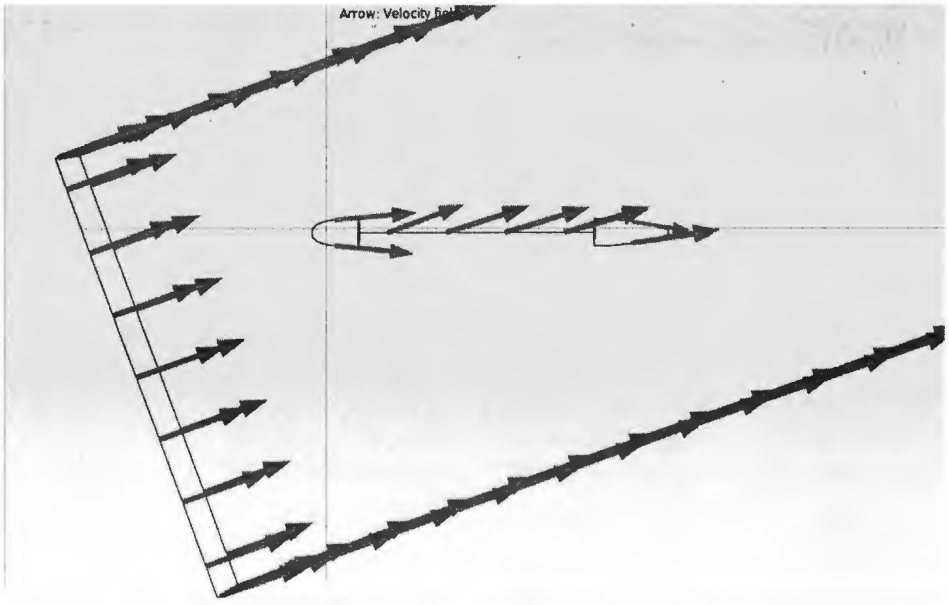


Figure 4.6: Example CFD case result (V=1m/s,$\alpha = 20°$)

$$C_L(\alpha) = \frac{L}{\frac{1}{2}\rho V^2 Volum^{2/3}} \tag{4.24}$$

$$C_M(\alpha) = \frac{M}{\frac{1}{2}\rho V^2 Volum} \tag{4.25}$$

Note that the moment calculated here is at the glider bare hull geometry center,

Table 4.3: Bare Hull CFD Analysis Results

Angle of Attack (deg)	CD	CL	CM
0	0.00976	-0.00052	0.00016
5	0.0129	0.0439	0.1249
10	0.0297	0.1176	0.2296
15	0.0536	0.1906	0.3271
20	0.1063	0.2965	0.4233

which is also the origin point of the body-fixed frame, located at $0.72m$ aft of the glider nose tip. This allows us to compare the result from the Phoenix experiments. Again, using polynomial curve fitting, we have Figures 4.7, 4.8 and 4.9 showing the

hydrodynamic performance of the glider bare hull. The results from Phoenix experiments are also plotted to make a comparison.

The polynomial coefficients are summarized in Table 4.4.

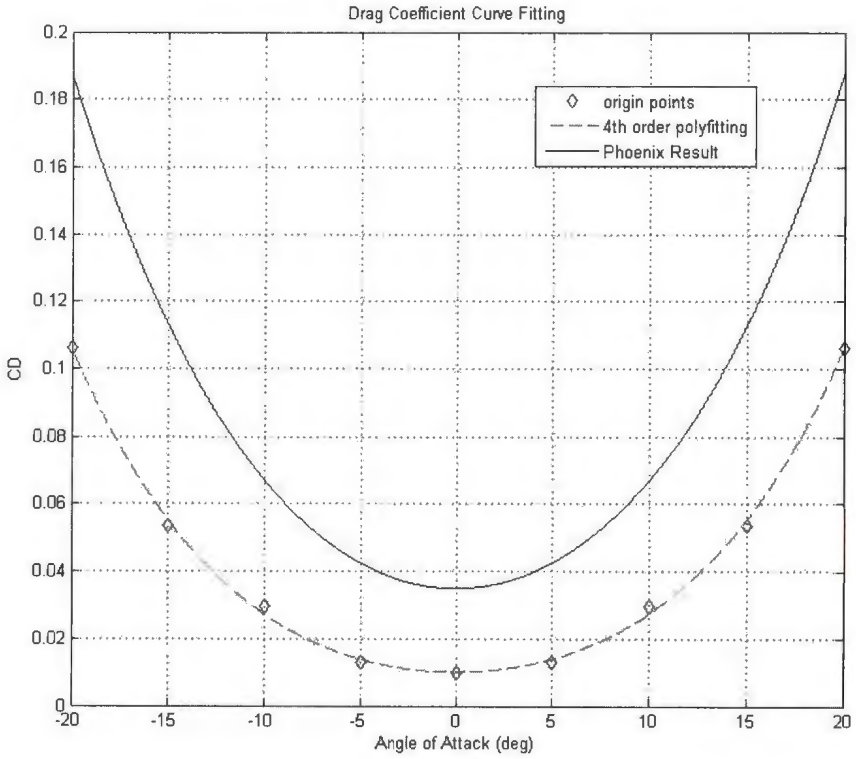


Figure 4.7: Glider Bare Hull Drag Coefficient Curve Fitting

We can see that lift and pitching moment results match well and that the difference happens in the drag result. The explanation is that in CFD modelling, the drag force is more sensitive to the mesh size and turbulence model [2]. To better predict the hydrodynamic performance, a finer mesh is necessary.

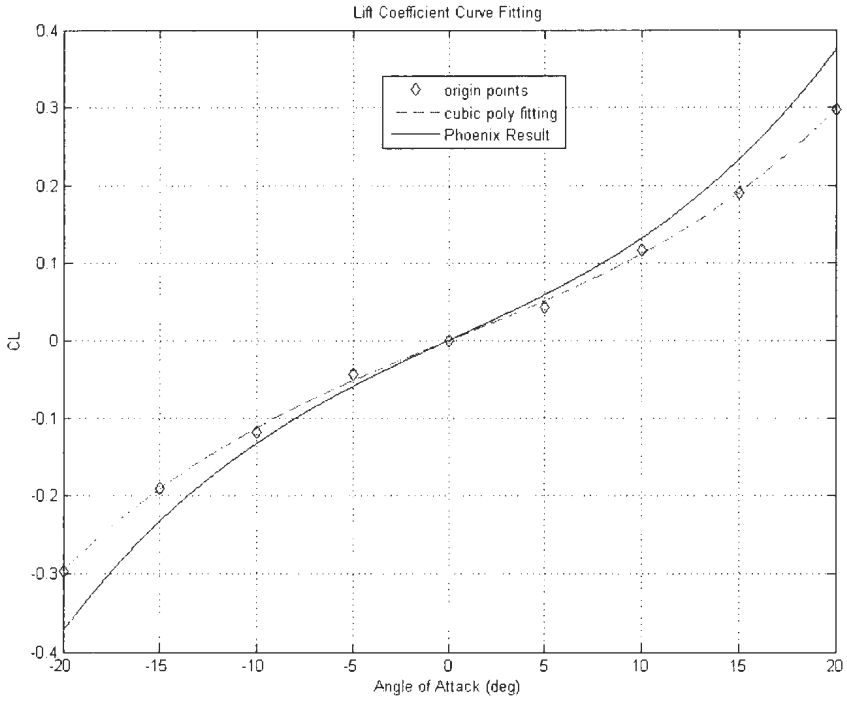


Figure 4.8: Glider Bare Hull Lift Coefficient Curve Fitting

Table 4.4: Curve Fitting Results for Bare Hull CFD Analysis

	Drag Coefficient		Lift Coefficient		Pitching Moment Coefficient	
	CFD Analysis	Phoenix Result	CFD Analysis	Phoenix Result	CFD Analysis	Phoenix Result
a_4	2.24E-7	2.02E-7	0	0	0	0
a_3	0	0	1.202E-5	1.822E-5	-6.35E-6	-9.99E-6
a_2	1.05E-6	3.01E-4	0	0	0	0
a_1	0	0	0.010	0.011	0.0236	0.0223
a_0	0.010	0.035	0	0	0	0

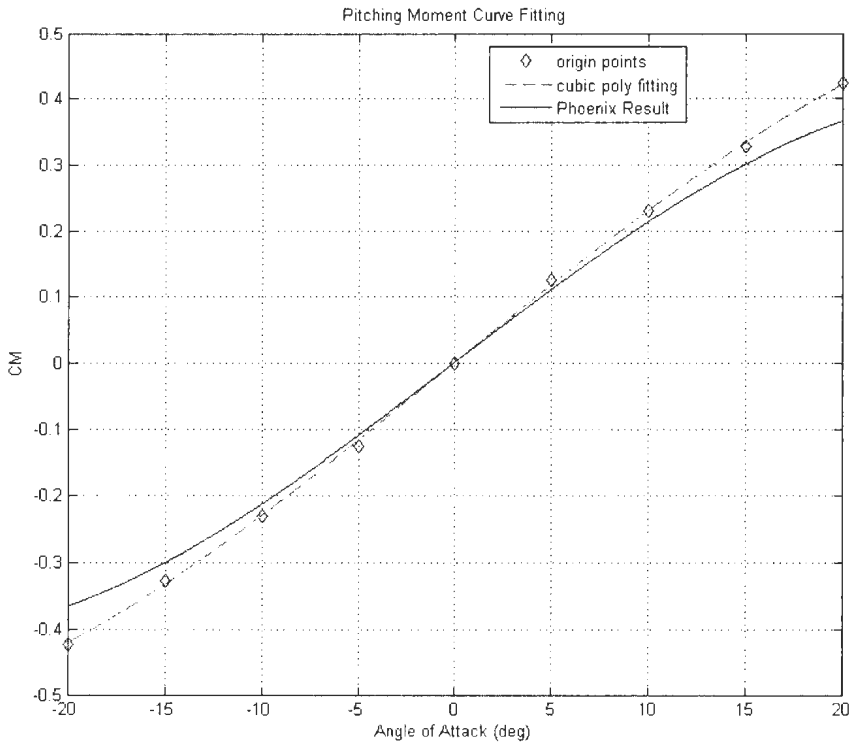


Figure 4.9: Glider Bare Hull Pitching Moment Coefficient Curve Fitting

4.5 Summary

This chapter presents a detailed study of the hydrodynamic forces on the glider bare hull and the wings. CFD methodology is used. The relationship is found between the angle of attack and the hydrodynamic coefficients using polynomial curve fitting. Part of the prediction is confirmed by comparison with experimental data. The results from the CFD analysis will be integrated into the glider simulation model to predict the forces and moments acting on the vehicle.

Chapter 5

Testing of the Pitch and Roll Mechanism

5.1 Introduction

After the pitch and roll mechanism was integrated into the IOT Glider, a set of tests was conducted in the open air. The tests verified the proper working of the whole system. At the same time, some of the mass properties were measured. This chapter presents the preparation work and tests before putting the IOT Glider into the water.

5.2 Assembly of the Glider

Warren[15] detailed the procedure of the buoyancy engine assembly for the IOT Glider. As the current IOT Glider possesses the pitch and roll mechanism, the installation should be done after the buoyancy engine actuator is mounted to the aluminium mid-section.

1. First attach the rolling motor on the roll motor mount plate with four screws,

then put the spur gear on the motor shaft.

2. Put the roll motor mount plate on the bottom of the buoyancy engine actuator, and tighten it up using a screw.

3. Put a stop clamp above the rolling motor mount plate to hold the whole carriage.

4. Put one of the end plates on a flanged sleeve bearing; tighten up using set screws. Attach the ring gear on the bottom of the end plate using four screws. Screw on the three pillars. Refer to Figure 2.8 for the components on the pitch and roll mechanism.

5. Attach the pitching motor to the pitch motor mount plate with two screws. Put the plate on the sleeve bearing mentioned in the previous step and tighten it using set screws.

6. Now the frame for the two motors is ready. Put it on the buoyancy engine actuator. Match the gears.

7. Attach the pitching motor nut to the battery carriage aft plate using three screws. Put the plate on the second sleeve bearing and tighten it using set screws.

8. Put the battery carriage mid plate on the sleeve bearing mentioned in the previous step; tighten it using set screws.

9. Put the seven batteries into the battery carriage.

10. Put the other end plate onto the third sleeve bearing; tighten it using set screws.

11. Put the third sleeve bearing onto the buoyancy engine, to the ends of the three pillars. Tighten up the pillars using screw nuts.

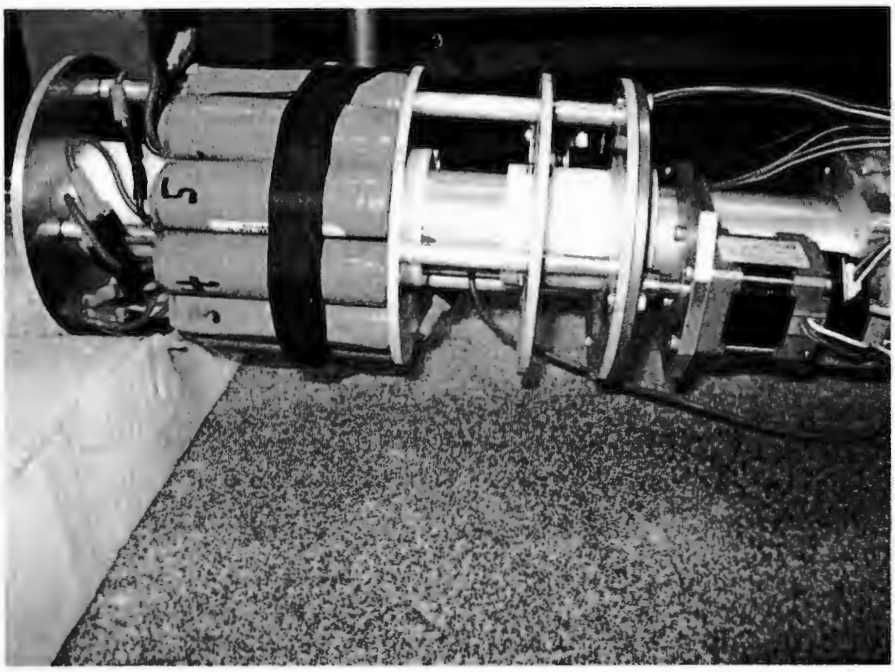


Figure 5.1: Pitch and Roll Mechanism Installed on the Buoyancy Engine Actuator

After assembly of all the glider elements, the air inside the glider hull is pumped out until the internal pressure reaches about $30kPa$ vacuum. The air vacuum will keep the glider from bending. The nose section and the tail section can be attached to the glider's main body using screws. The wing mounts can be attached to the set screws and the threaded hole on the mid section of the glider.

5.3 Calculation and Testing of Mass Properties

Before testing the glider in water, it is important to know where the overall centre of gravity is located and what are some other mass properties such as moments of inertia. For this purpose, a set of experiments were conducted to determine these parameters. Besides the experiments, calculation results from component details are provided to make a comparison.

5.3.1 Longitudinal Centre of Gravity

The mass and the longitudinal center of gravity position of the IOT Glider without fins, wings or any ballast will be measured and calculated fundamental information for future use.

5.3.1.1 Tests Using Weighing Platforms

The IOT Glider was supported by two steel angles which are put on top of two weighing platforms. Before putting the glider on top, an effort was made to make

Figure 5.2: Longitudinal CG Measure Using Two Weighing Platforms

the angles level. A bubble level was put on top to make sure the glider would be totally horizontal. After levelling, the glider was put on top. The position of the front support is measured from the nose tip of the glider as d_1; the position of the second support is measured from the nose tip of the glider as d_2; measurements from

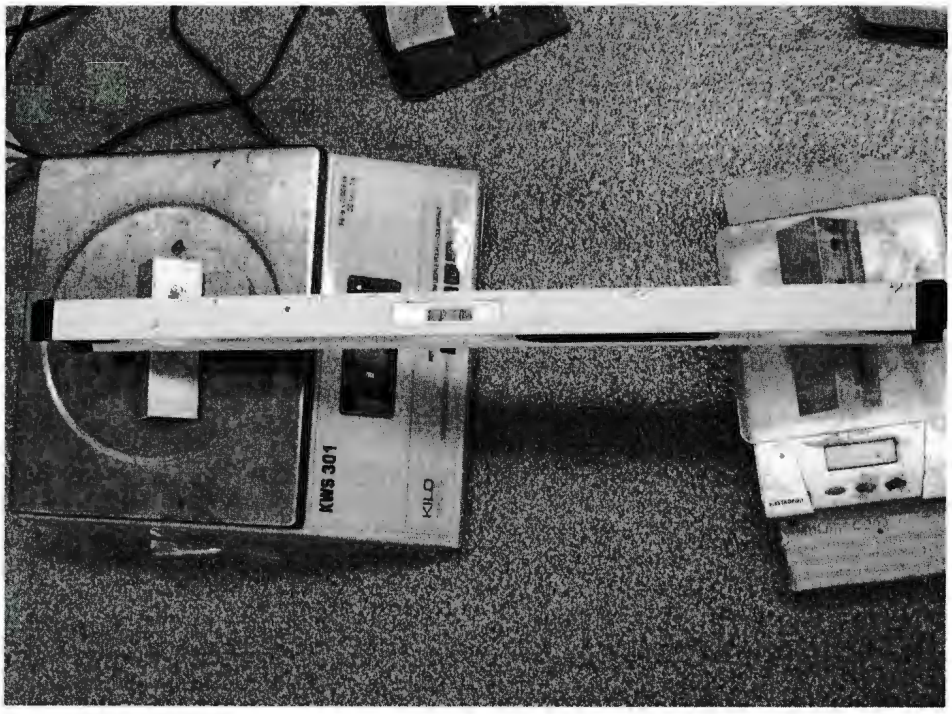


Figure 5.3: Levelling the Two Weighing Platforms

the weighing platforms were recorded as m_1 and m_2. The total mass of the glider can be determined by Equation 5.1. The position of the center of gravity from the glider nose tip can be determined by Equation 5.2.

$$m = m_1 + m_2 \tag{5.1}$$

$$xG = \frac{d_1 \times m_1 + d_2 \times m_2}{m} \tag{5.2}$$

Two sets of tests were conducted. The results are provided in Table 5.3.

5.3.1.2 Tests Using Load Cells

To make comparisons, several tests were conducted measuring xG using two load cells with the glider hanging from them. Again, before actually measuring, an attempt was

Figure 5.4: Testing Longitudinal CG Using Two Weighing Platforms

made to make sure that the hanging string was perpendicular. A mass was suspended next to the string to easily check the perpendicularity of the string. After checking the perpendicularity of both string, the position of the front hanging point is measured from the nose tip of the glider as d_1; the position of the second hanging point is measured from the nose tip of the glider as d_2; measurements from the load cells were recorded as m_1 and m_2.

The total mass of the glider can be determined by Equation 5.1. The position of the center of gravity from the glider nose tip can be determined by Equation 5.2. Three sets of tests were conducted.

5.3.1.3 Test Results

The measurements from tests by the weighing platforms and load cells are summarized in Table 5.1 with the calculation from components mass and position details using

Figure 5.5: Checking the Perpendicularity

Figure 5.6: Testing Longitudinal CG Using Load Cells

Equation 5.3. The masses and the measured or assumed centre of gravity positions of each component on the IOT Glider are provided in Appendix G.

$$xG = \frac{\sum m_i \times x_i}{\sum m_i} \tag{5.3}$$

Note that all results are for the glider without wings or tail fins. The reason is that

Table 5.1: Longitudinal CG Measurements

	Total mass m (kg)	Longitudinal CG position xG (mm)
Components build up	9.50	651
Weighing platform (1)	9.61	649
Weighing platform (2)	9.58	646
Load cell results (1)	9.61	649
Load cell results (2)	9.63	651
Load cell results (3)	9.64	650

the wings are able to be fixed in different positions. As long as we have the mass and exact position of the wings, we can calculate the overall CG. The measurements do not include any ballast mass either.

5.3.2 Center of Gravity in Y-Z Plane

To evaluate the CG position in the Y-Z plane, it is expressed by θ_0 and r_G in an angular coordinate system fixed on the center of the glider's bare hull cross section. For measuring the θ_0 and r_G parameters, a mass is suspended from the wing mount on one side. See Figure 5.7. The rolling angle the glider rotates θ_m with regard to the zero position and the weight of the mass m' are recorded, the rolling angle of the glider without a mass is also recorded, which is just θ_0. The angle information is given

by the on board inclinometer. Then r_G can be determined by Equation 5.4.

$$r_G = \frac{m' \times d \times \cos\theta_m}{m \times \sin(\theta_m - \theta_0)} \tag{5.4}$$

where d is the distance from the mass hanging point to the center of the angular coordinate system.

Figure 5.7 shows one of the r_G tests. The glider is suspended on two pivots so that it can rotate freely about its central axis. Four different mass blocks were used for these tests. The results are shown in Table 5.2.

Table 5.2: Measurements for CG in Y-Z Plane Position

	Testing Mass (kg)	θ_0 (deg)	θ_m (deg)	r_G (mm)
Test (1)	1.035	12	80	1.6
Test (2)	0.247	11	54	1.7
Test (3)	0.323	10	59	1.8
Test (4)	0.463	10	67	1.7

5.4 Testing on Pitch and Roll Mechanism

After assembling the pitch and roll mechanism into the glider, several tests were performed to check the subsystem.

5.4.1 Test Set-Up

To facilitate the tests of the glider in the open air, a pair of multi-functional supports are designed and made for holding the glider hull to let it roll or pitch freely. In Figure 5.8 is a SolidWorks drawing of one piece. The support is composed of three parts: a platform, a stand-up pipe and a rod. The rod is welded to the middle of the stand-up

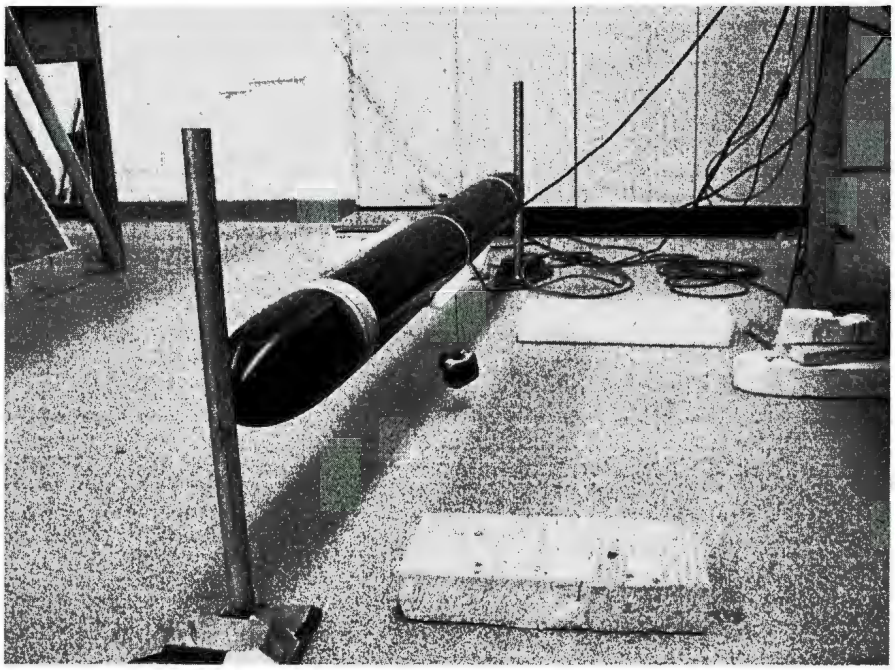


Figure 5.7: Measuring CG in the Y-Z plane

Figure 5.8: SolidWorks Drawing for One of the Supports

pipe. To make the glider hull pitch only, the rods are inserted into the holes on the aluminium mid-section; to make the glider hull roll only, the rods are inserted into the holes in the nose tip and tail tip.

Due to the friction generated by the rod, ball bearings are introduced to reduce the friction. Figure 5.9 shows the improved glider supports. As the holes on the

Figure 5.9: Improved Glider Supports with Ball Bearings

aluminium are located 734.34 mm aft of the nose tip, an appropriate ballast mass is necessary to be inserted in the tail section to make the overall glider centre of gravity be located exactly at that position. As the distance from the ballast mass will be 1320.2 mm aft of the nose tip, the mass of the ballast is 1.37 kg. Rechecking:

$$\frac{9.63kg \times 651mm + 1.37kg \times 1320.2mm}{9.63kg + 1.37kg} = 734.34mm$$

5.4.2 Steady-State Tests on Pitch and Roll

The first step tests after the pitch and roll mechanism has been integrated and calibrated are steady-state tests. The pitch actuator (roll actuator) for the battery pack is moved to various positions and the pitch (roll) attitude is measured by the inclinometer and recorded by the SBC and transferred to a PC.

The test results are summarized in Tables 5.3 and 5.4 below. In the pitching tests, the sign-convention system is used different from the normally used because of the signal given by the inclinometer.

Figure 5.10 and Figure 5.11 show plots about the steady-state pitching angle and

Table 5.3: Pitching Steady-State Test Results

Test Number	Battery pack position (mm)	Pitching angle (deg)
1	-18.3	-5.2
2	-15.3	-3.5
3	-10.3	-1.2
4	-5.3	0.8
5	-0.2	2.4
6	5.3	3.5
7	10.3	5.0
8	15.2	6.6
9	19.7	9.1

Table 5.4: Rolling Steady-State Test Results

Test Number	Battery pack rotation (deg)	Rolling angle (deg)
1	-93.2	0.7
2	-62.3	-1.2
3	-31.7	-3.5
4	-0	-7.2
5	33.1	-11.4
6	62.2	-14.9
7	91.7	-17.8

steady-state rolling angle versus the battery pack position, respectively. Straight lines were expected for small pitching and rolling angles.

The plots indicate that the pitch and roll mechanism has a capacity of changing

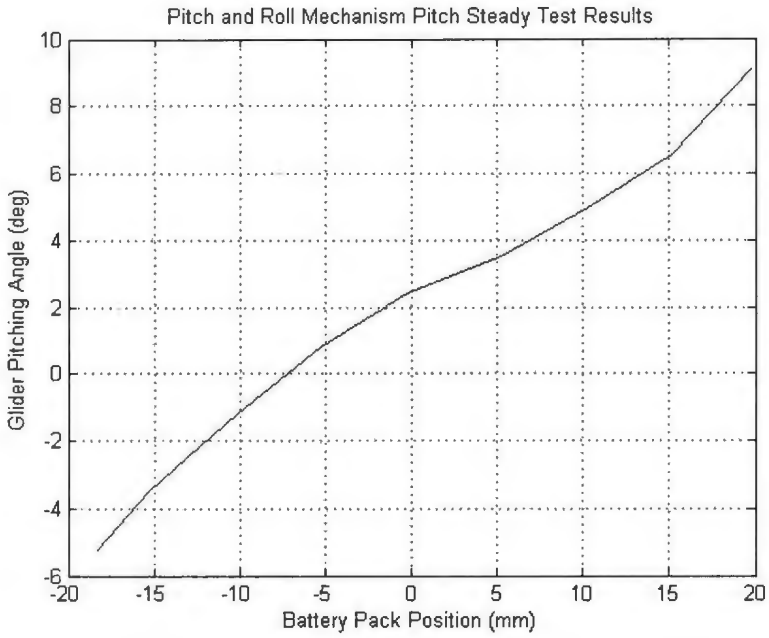


Figure 5.10: Steady Pitch Test Results

pitch angle within 14.3 *deg* and roll angle 18.5 *deg* in air, with a glider overall mass of 11 *kg*.

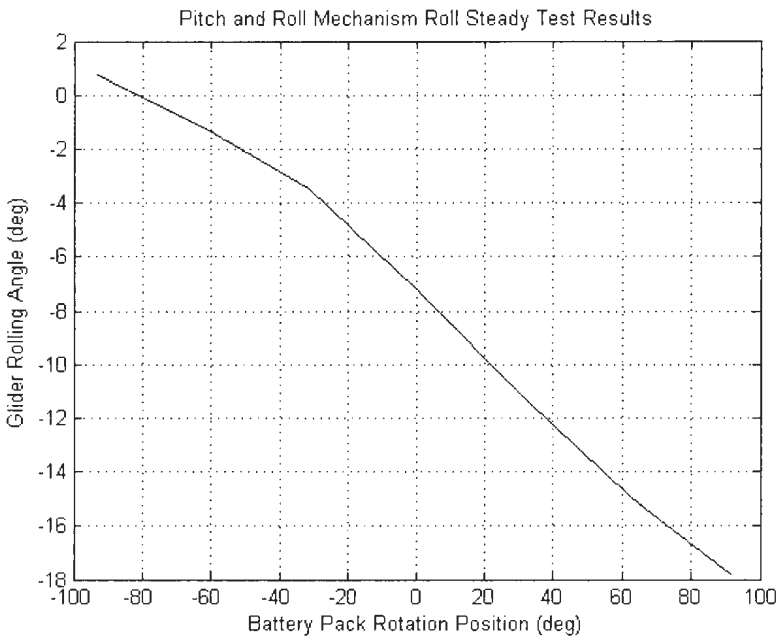


Figure 5.11: Steady Roll Test Results

5.4.3 Pitching Angle PID Control Tests

PID control loops for controlling the pitching angle and rolling angle of the IOT Glider when suspended in air were developed. The pitch motor and roll motor speed signal are determined using a PID algorithm from the error between the set point and current pitch angle and roll angle feedback, respectively.

The set point of the pitch angle control is 3 *deg*. Small pitching angle set point was used for preventing the glider from hitting the ground in the tests. Firstly, P control is applied, with I gain and D gain both set as 0. Various P gain values and saturation values of motor speed are tried. PID control is then applied with suitable gain values. Two P control cases are shown as Figure 5.12 and 5.13. Both cases use ± 600 saturation on the motor speed signal. The result in Figure 5.12 uses a P gain of 150; the pitching angle reaches the set point gradually in about 60 *sec*. Figure 5.13 uses a P gain of 4500; as it is too large, the pitching angle turns out to be vibrating around the set point.

After trials on the P control, a group of PID control tests were conducted. Figure 5.14 gives an example of a successful PID control case with a settling time of about 20 *sec*. It is obvious that the glider pitching angle reaches the set point much faster than the P control in Figure 5.12.

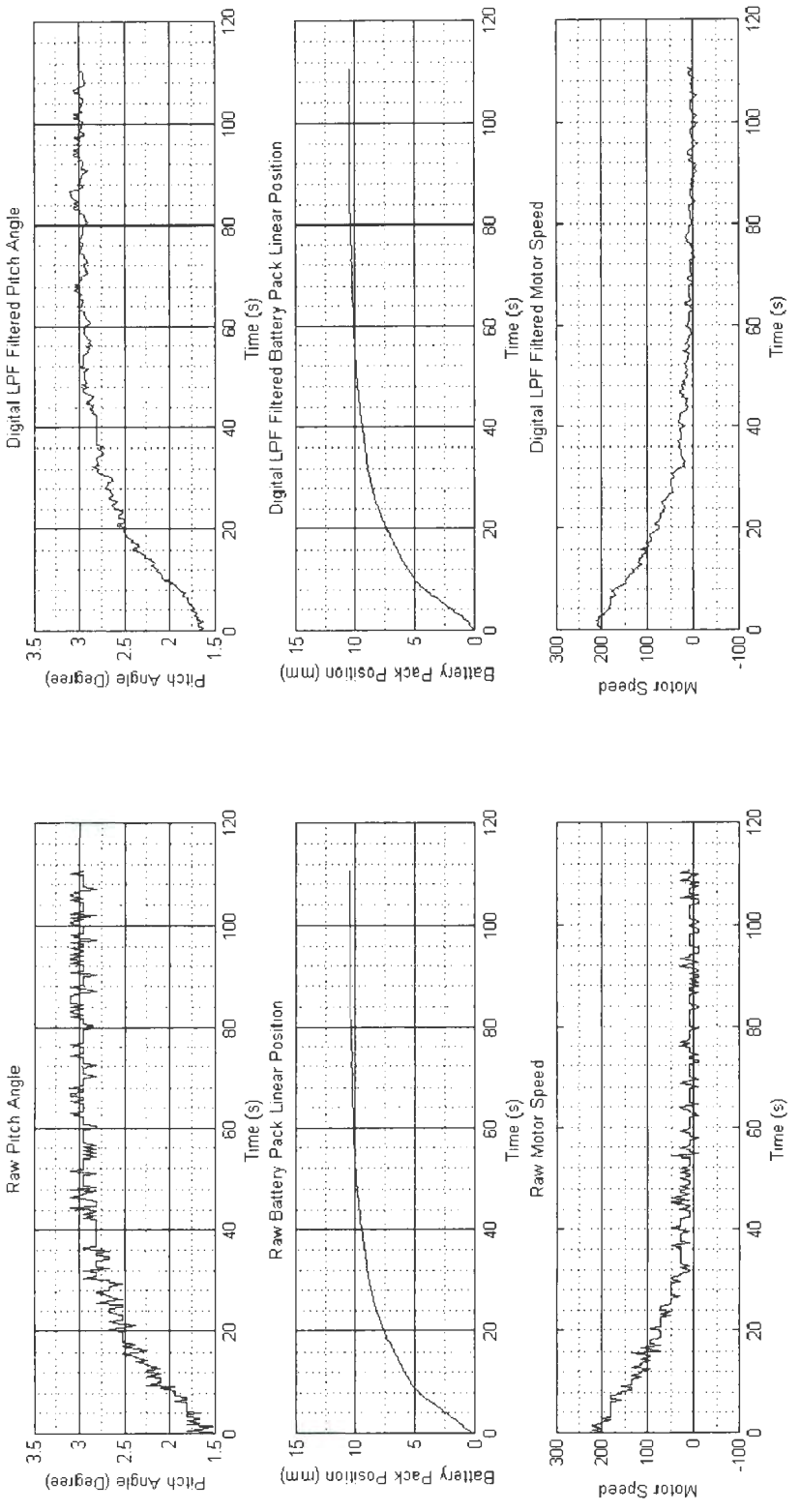


Figure 5.12: P Gain 150, I Gain 0, D Gain 0, Saturation ±600

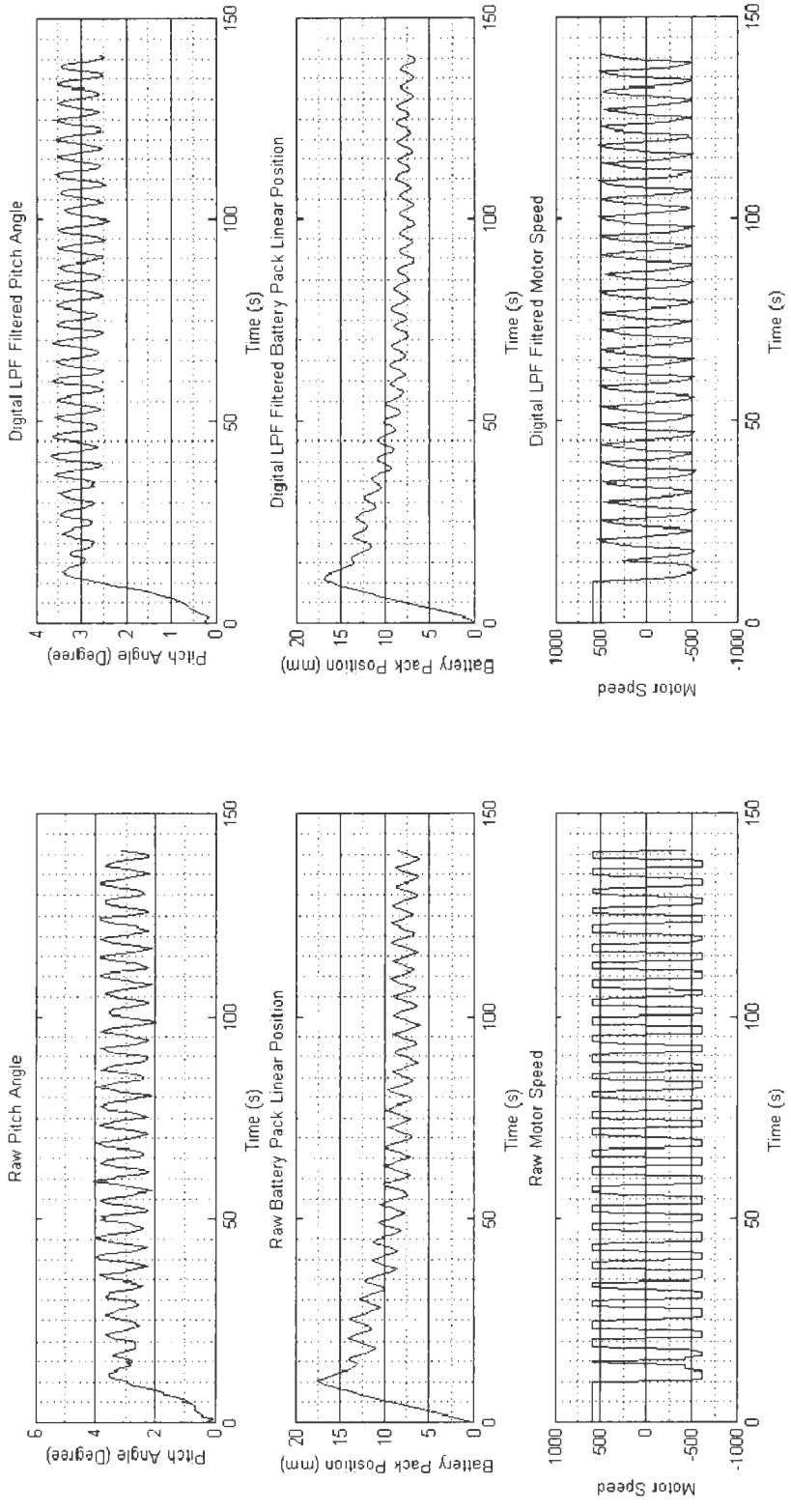


Figure 5.13: P Gain 4500, I Gain 0, D Gain 0, Saturation ±600

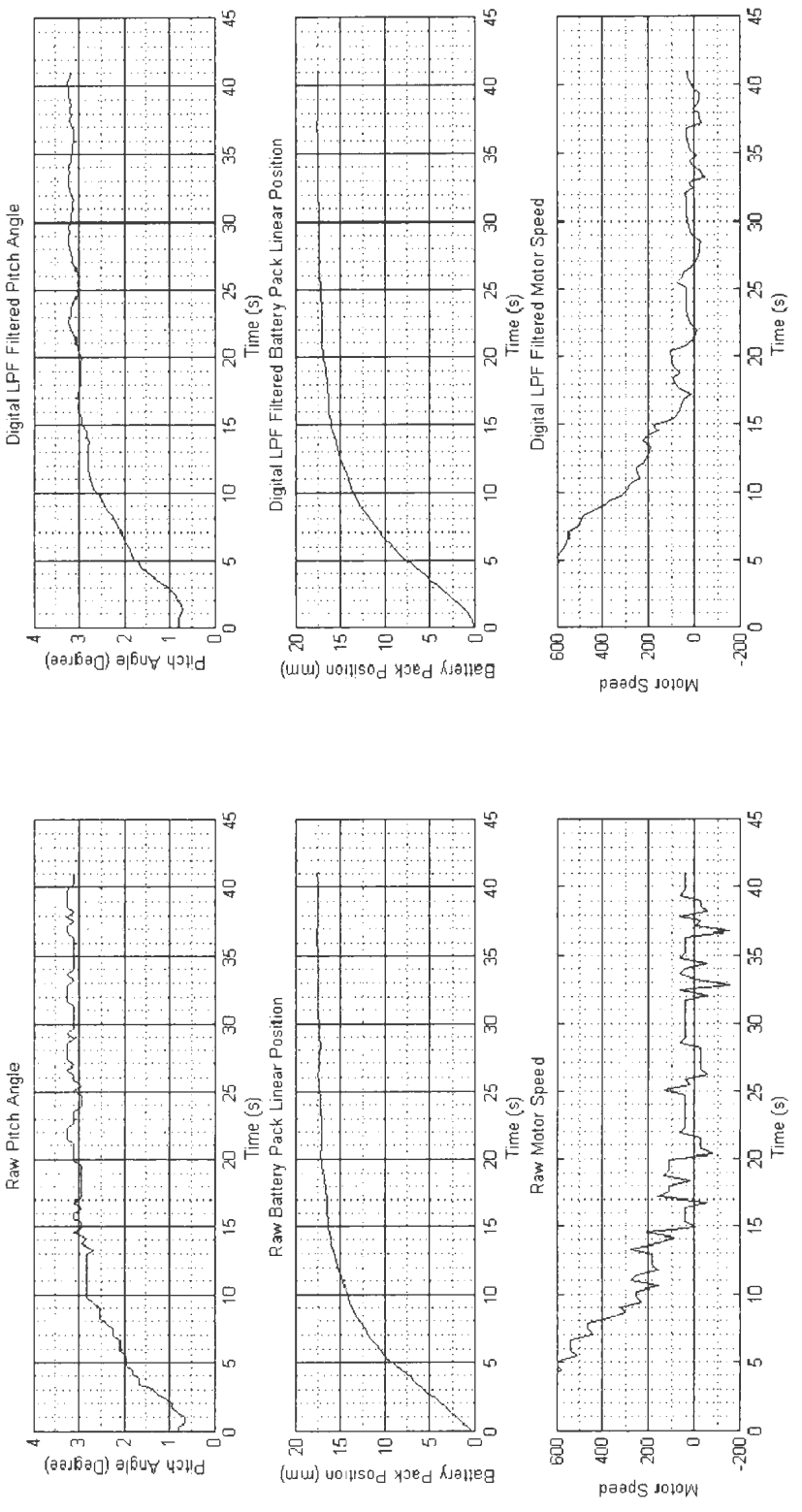


Figure 5.14: P Gain 500, I Gain 10, D Gain 50, Saturation ±600

The effects and observations from the tests are selected and summarized below.

Table 5.5: PID Pitch Control Results

Saturation of Motor Speed	P Gain	I Gain	D Gain	Time Used	Observation
200	50	0	0	150	No overshoot, no vibration.
200	450	0	0	40	Overshoot, no vibration.
400	100	0	0	130	No overshoot, no vibration.
400	900	0	0	50	Overshoot, no vibration.
400	3000	0	0	inf	Overshoot, vibration.
600	150	0	0	60	No overshoot, no vibration.
600	4500	0	0	inf	Overshoot, vibration.
600	810	100	200	inf	Unstable.
600	500	10	50	22	No overshoot, no vibration.

The time history of the pitch angle, battery pack longitudinal position and the motor speed signal are plotted for all the tests. The raw data is also filtered using a simple digital low pass filter. The plots can be found in Appendix A.

5.5 Calculation and Test of Mass Moments of Inertia

The IOT Glider was supported by two frames with ball bearings to determine the moments of inertia about the X-axis and Y-axis.

The glider was set to a small initial pitching angle (rolling angle) and released to perform free oscillations. The pitching angle (rolling angle) time history was recorded by measurements with the inclinometer. Figure 5.15 shows the rolling angle time history in the Ixx test; Figure 5.16 shows the pitching angle time history in the Iyy test.

The rotational motion period T was captured from the plots and the moment of

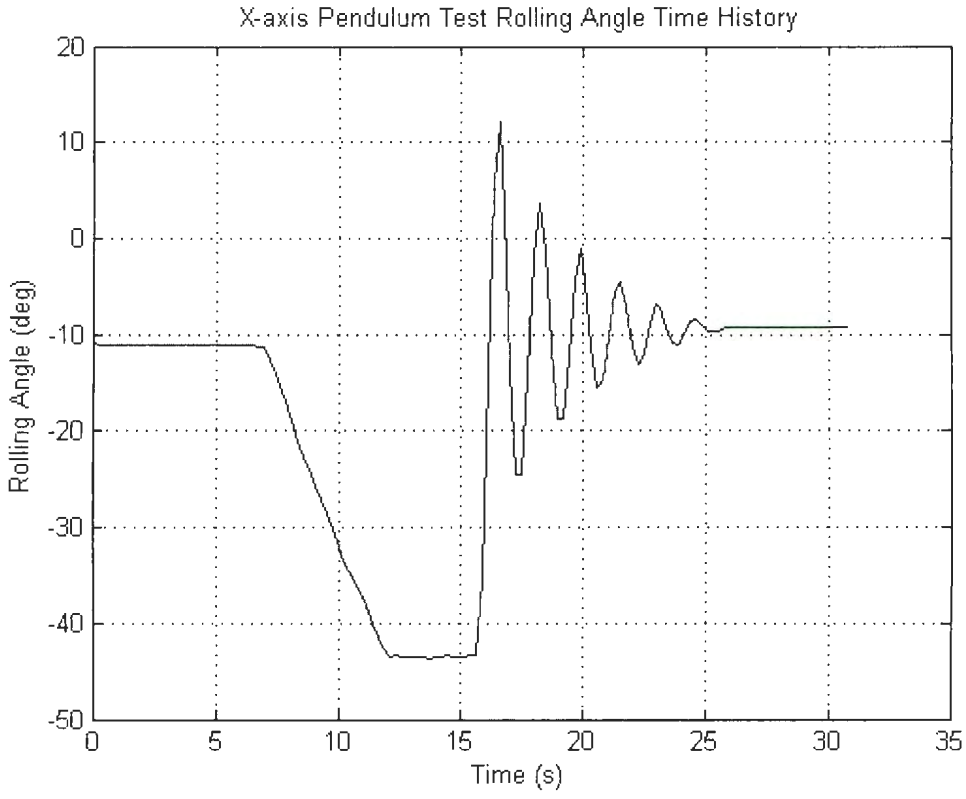


Figure 5.15: Rolling Angle Time History in the Ixx Test

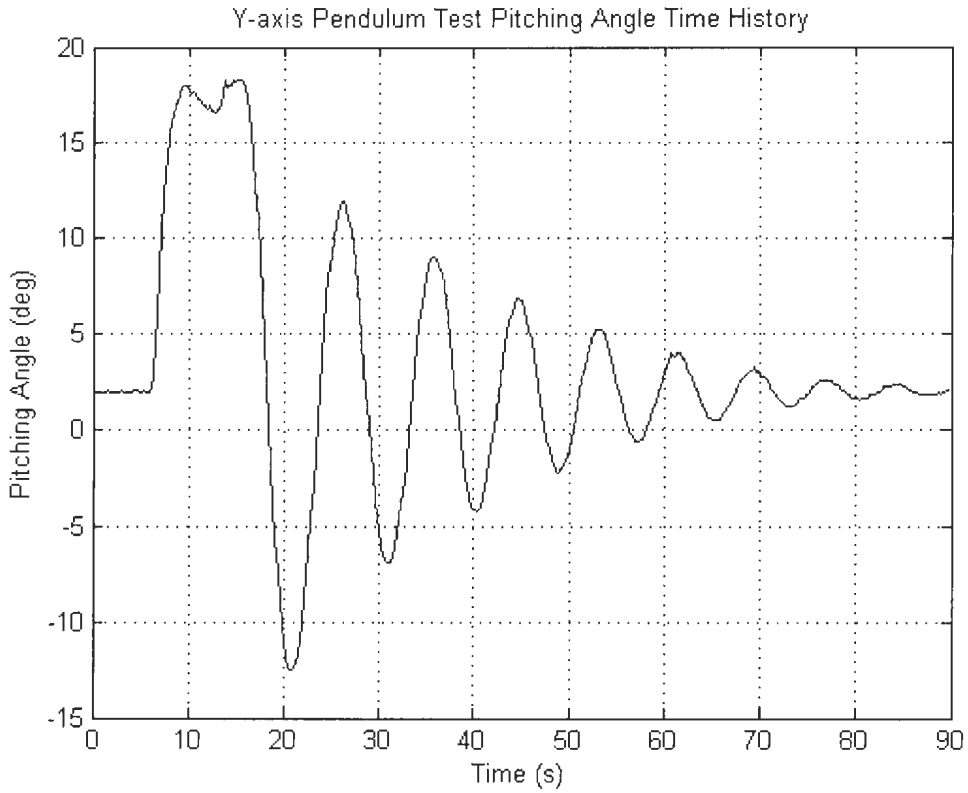


Figure 5.16: Pitching Angle Time History in the Iyy Test

inertia can be evaluated from the small angle pendulum theory. The derived equation for calculating moments of inertia is Equation 5.5

$$I = (\frac{T}{2\pi})^2 \times M \times zG \tag{5.5}$$

where M is the total mass of the glider, as a $1.37kg$ ballast mass was attached in the tail section, $M = 11kg$; zG is the CG position from the centre of rotation axle, which is taken as rG.

The estimated rotation period and the evaluated moments of inertia are shown in Table 5.6.

Table 5.6: Moment of Inertia Measurement Results

	Period (s)	Moment of Inertia ($kg \cdot m^2$)
About X-axis	1.7	0.016
About Y-axis	9.8	0.525

5.6 Summary

This chapter demonstrates the proper working of the pitch and roll mechanism on the IOT Glider. Several tests were conducted, including steady-state pitch and roll tests and pitching angle PID control tests. Measurements were also applied to determine several mass properties of the glider. Rolling angle PID control tests were not conducted because they are using similar control algorithm and experimental set-up. By now, the IOT Glider is equipped with its buoyancy engine, pitch and roll mechanism and several on-board sensors. The vehicle is working properly and has been tested. It is now ready to be put underwater.

Chapter 6

Deep Tank Testing of the IOT Glider

6.1 Introduction

The glider components have all been assembled together; the pitch and roll mechanism is now working properly. A group of tests in water was performed next. The IOT Glider was made to fly with various combinations of buoyancy engine actuator position and positions of the pitch actuator for the battery pack. Then a look-up table was made to study the effects of the two varying factors.

This chapter presents preparation work before the tests, the procedure of the tests and results from the tests.

6.2 Test Set-up and Water Tanks at MUN

6.2.1 Water Tanks at MUN

The OERC (Ocean Engineering Research Centre) at MUN is equipped with several water tanks, including a 52 m long wave tank, a $12' \times 12' \times 13'$ deep water tank, a $30' \times 17'' \times 22''$ open water flume, a $12' \times 3' \times 1'8''$ trim tank and a riverbed flume. These facilities help with the tests of the IOT Glider in water. The tests presented in this chapter were conducted in the tanks at MUN.

6.2.2 Leak Check

The IOT Glider is assembled from a number of sections joined together using O-ring seals. Moreover, the end cap also contains three ports for sensors, wire, and the air valve [15]. As a result, before putting the IOT Glider underwater, a leak test is necessary for verifying these seals and for ensuring the vehicle is watertight.
As it has been proved that the main air leakage appears through the cable [15], a section of cable with one of its ends sealed and blocked was used instead. Figure 6.1 shows the end cap and the sealed cable section.

All electronic components were moved out of the glider. After assembly of the glider's bare hull, a vacuum pump was used to establish a $24''$ Hg vacuum inside. It was then left in the bottom of a 1 m deep tank. A total of $27''$ Hg pressure difference was generated between inside and outside of the glider hull. The inside pressure was tested after several days. The results are shown in Table 6.1.

As the IOT Glider is designed for lab use only, a normal test will take place within a few minutes; this is long enough time in the leak test to verify its sealing. As shown, there was just $3\,''Hg$ drop of the inside vacuum in such a long duration, so we conclude that the sealing on the IOT Glider is ready for the future underwater tests.

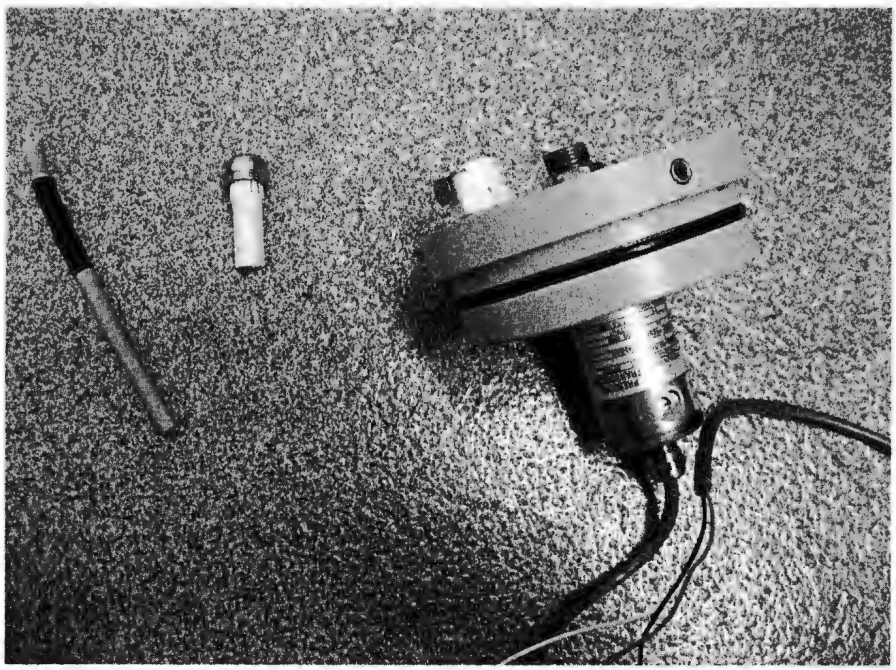


Figure 6.1: The End Cap and the Dummy Cable

Table 6.1: Leak Test Records

Test Date and Time	Time after Placement in Water (hour)	Vacuum Inside Hull
Jun. 28, 2012 13:30	0	24" Hg
Jun. 29, 2012 10:37	21	22" Hg
Jul. 2, 2012 9:05	92.5	21" Hg

Figure 6.2: Leak Tests

6.2.3 Communication and Power Cable

Although the IOT Glider has its own internal battery pack and the BL2100 SBC is capable of recording data on its own file system, we decided to use an external communication and power cable instead. Using the external cable, the IOT Glider is powered by a power supply; the real time data is transmitted to a desktop PC during applications. The benefits are that we can send commands from the desktop PC during applications. We can also monitor the status of the glider by reading data received from the glider. The cable is going through the tail section wall and into the glider through the end cap. A cord grip is used to seal the cable.

The disadvantage of using an external cable communication and powering is that a long cable is needed. In our case, a 9 m long cable is used. The cable has a higher density than water. As a result, it will affect the movement of the glider when the glider is underwater with the cable.

Figure 6.3: Powering and Monitoring the IOT Glider

To solve this problem, small foam blocks are attached on the cable to provide additional buoyancy. The foam blocks are $2cm \times 2cm \times 2cm$ cubic blocks with a 0.51 cm diameter hole through them. The density of the foam used is 42.1 kg/m^3. The cable has a diameter of 0.51 cm and a length of 9 m. It was derived that a 0.32 m interval between them will make the cable neutrally buoyant in fresh water.

30 foam blocks were made in the Machine Shop at MUN using a band saw and a milling machine. Figure 6.4 and 6.5 demonstrate the procedures for making the foam block.

Figure 6.4: Making Foam Blocks on a Milling Machine

Figure 6.5: Making Foam Blocks Using a Band Saw

6.3 Procedures and Results

6.3.1 Ballasting

After assembly (remove the ballast mass during the tests in air), the IOT Glider floats when put in water. Suitable ballast mass needs to be mounted on the glider. At the same time, to approximately level the glider, the appropriate amount of mass put in both nose section and tail section is necessary. After several trials in the Trim Tank at MUN, a 0.727 kg mass was fixed in the nose section, located at 101.6 mm aft of the glider nose tip; a 1.510 kg mass was fixed in the tail section, located at 1117.6 mm aft of the glider nose tip.

The rest ballasting was accomplished by using the buoyancy engine and the pitch and roll mechanism following the steps shown below.

1. Use the buoyancy engine actuator to make the glider neutrally buoyant when fully submerged in the Deep Tank.

2. Move the pitch actuator for the battery pack to achieve a level-pitch attitude of the glider while it is fully submerged.

3. Move the roll actuator for the battery pack to achieve a level-roll attitude of the glider while it is fully submerged.

Figure 6.6 shows a photo for ballasting and trimming in the trim tank at MUN. It was found that the buoyancy engine in the 2.1 cm position makes the glider neutrally buoyant. The battery pack in the -8 mm linear position makes the glider have a level pitching attitude. The battery pack in the -35 $degree$ rolling position makes the glider level in its rolling attitude. Based on this configuration, using Equation 3.9 we can obtain the longitudinal position of the glider's centre of gravity. It is 677 mm aft of the glider's nose tip. It is also concluded that the centre of buoyancy is located

at 677 mm aft of the glider nose tip and the buoyancy force is 116 N.

At this configuration, the glider is neutrally buoyant, in level pitching attitude and

Figure 6.6: Ballasting and Trimming in the Trim Tank at MUN

in level rolling attitude.

6.3.2 Experiment Design and Results

Both rising tests and diving tests were conducted with different buoyancy engine position and battery pack linear position combinations. The wings are settled in their most forward positions in order to generate more hydrodynamic pitching moment on the vehicle from the wings.

At the start of each diving run, the desired buoyancy engine and battery pack positions are set. Then the glider is held at the water surface and fully submerged. When the water calms down, we set the glider free and let it fly. During the flights, the glider real-time pitch attitude in *deg* and the depth information (pressure outside the glider) among other data are measured and recorded by the electronic equipment and are transmitted and shown on a desktop PC. Figure 6.7 shows a photo of one of the diving tests.

At the start of each rising run, the desired battery pack position is first set. Then the

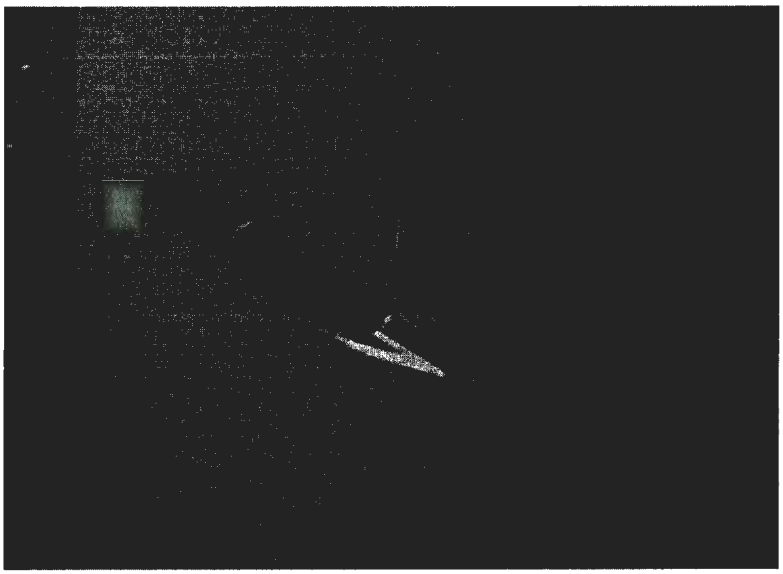


Figure 6.7: A Diving Run

glider is adjusted to dive mode by retracting the buoyancy engine. Put the glider into

water and settle it down at the bottom of the tank using the cable. After the glider is set at the bottom of the tank, set the buoyancy engine to the desired position. Then the vehicle begins to rise. Figure 6.8 shows a photo of one of the rising tests.

The buoyancy engine position is varied by -1 cm, -0.5 cm, 0.5 cm, and 1 cm

Figure 6.8: A Rising Run

with regard to its neutral position 2.1 cm; the battery pack position is varied by -8 mm, 0 mm, 8 mm, and 16 mm with regard to its neutral position -8 mm. Note that positive numbers mean forward movements of both the buoyancy engine and the battery pack. Using Equations 3.7, 3.8, and 3.9, the buoyancy-gravity difference, longitudinal centre of gravity, and longitudinal centre of buoyancy can be evaluated for each combination. These parameters for all the tests are summarized in Table 6.2, 6.3, and 6.4.

In each diving (rising) run, the glider pitching angle keeps increasing (decreasing) at the start; then it turns stable, which demonstrates the reaching of the steady state in the glider's typical motion[1].

The barometric pressure time history can be turned into depth information using

Table 6.2: Buoyancy-Gravity Difference (W-B) Status for the Tests (unit: N)

		Buoyancy Engine Position (cm)			
		Rising		Diving	
		1	0.5	-0.5	-1
Battery Pack Position (mm)	16	-0.3532	-0.1766	0.1766	0.3532
	8	-0.3532	-0.1766	0.1766	0.3532
	0	-0.3532	-0.1766	0.1766	0.3532
	-8	-0.3532	-0.1766	0.1766	0.3532

Table 6.3: xG Status for the Tests (aft of the glider's nose tip; unit: m)

		Buoyancy Engine Position (cm)			
		Rising		Diving	
		1	0.5	-0.5	-1
Battery Pack Position (mm)	16	0.6757	0.6760	0.6766	0.6769
	8	0.6761	0.6764	0.6770	0.6773
	0	0.6764	0.6767	0.6773	0.6776
	-8	0.6767	0.6770	0.6776	0.6779

Table 6.4: xB Status for the Tests (aft of the glider's nose tip; unit: m)

		Buoyancy Engine Position (cm)			
		Rising		Diving	
		1	0.5	-0.5	-1
Battery Pack Position (mm)	16	0.6756	0.6763	0.6777	0.6784
	8	0.6756	0.6763	0.6777	0.6784
	0	0.6756	0.6763	0.6777	0.6784
	-8	0.6756	0.6763	0.6777	0.6784

Equation 2.6. The pressure in open air is also measured using the same sensor to eliminate error; it is typically $P_0 = 101.6k\ Pa$. Then the time histories of the pitching angle and depth are plotted. Two example cases are shown in Figures 6.9 and 6.10. Figure 6.9 shows a rising case and Figure 6.10 shows a diving case.

The experiment design and results are summarized in Tables 6.5 and 6.6.

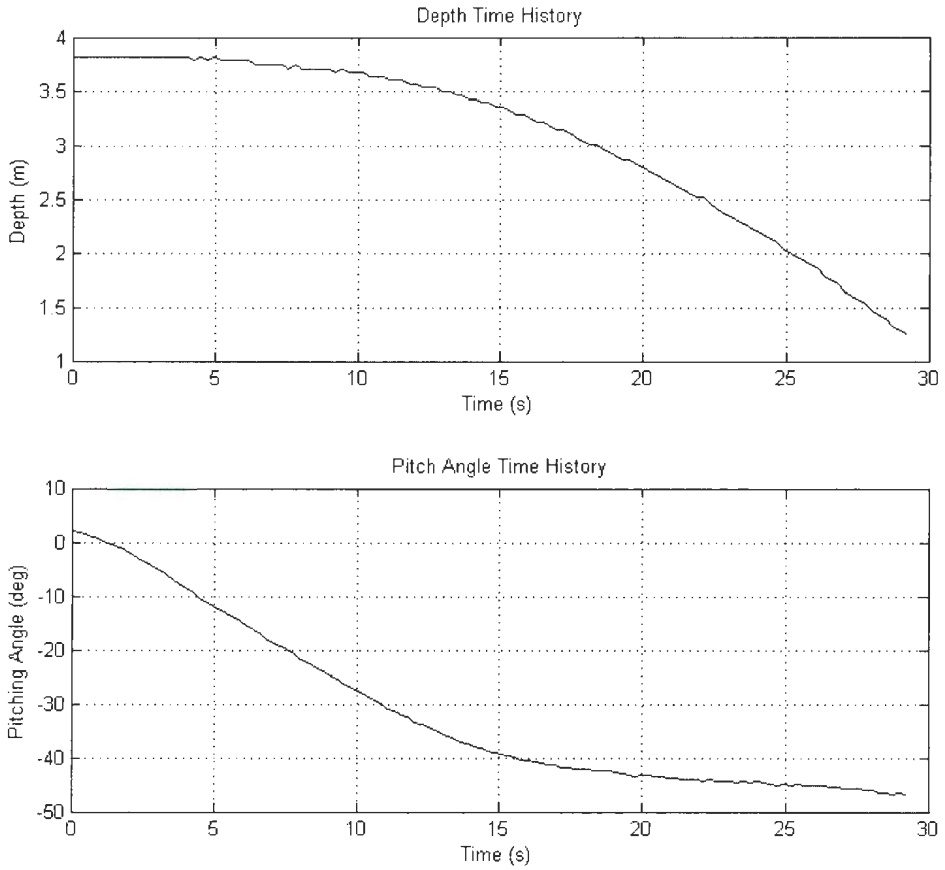


Figure 6.9: Buoyancy Engine 0.5 cm, Battery Pack -8 mm, Rising

From the test results we can see that the IOT Glider has a motion pattern similar to the SLOCUM glider. It will pitch into a steady state in about 10 *seconds*. Two of the main features of its steady state, depth rate and pitching angle, are dependent on

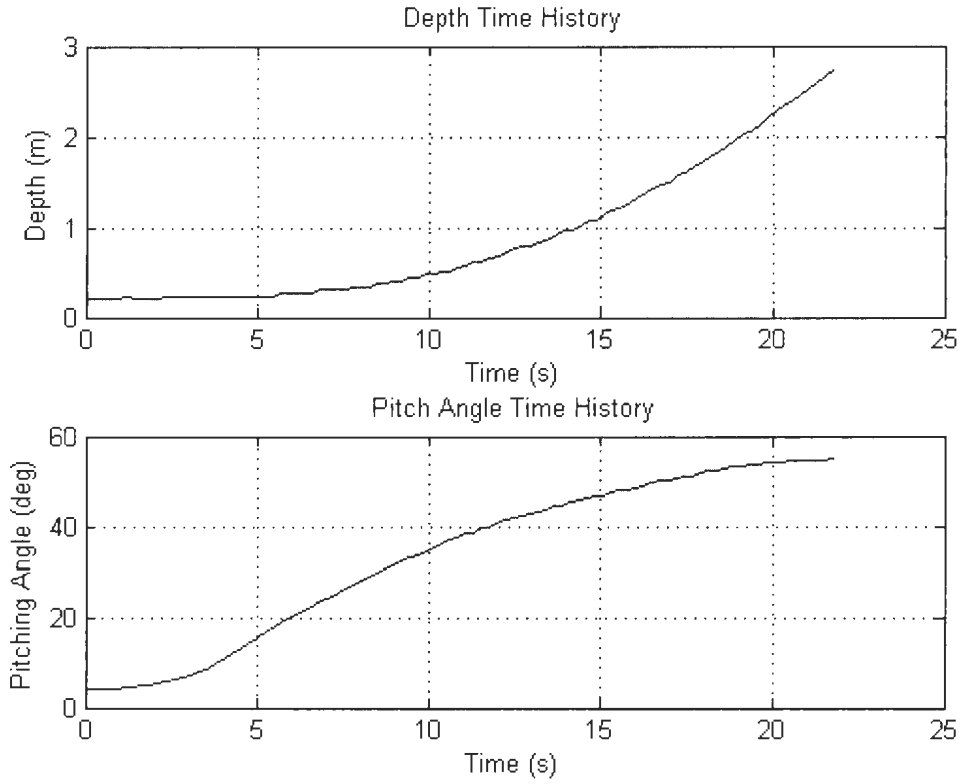


Figure 6.10: Buoyancy Engine -1 cm, Battery Pack 0 mm, Diving

Table 6.5: Rate of Ascent or Descent Results from the Tests (unit: m/s)

		Buoyancy Engine Position (cm)			
		Rising		Diving	
		1	0.5	-0.5	-1
Battery Pack Position (mm)	16	-0.055	-0.05	0.167	0.341
	8	-0.073	-0.057	0.25	0.326
	0	-0.115	-0.055	0.16	0.27
	-8	-0.225	-0.13	0.072	0.15

Table 6.6: Pitching Angle Results from the Tests (unit: *deg*)

		Buoyancy Engine Position (cm)			
		Rising		Diving	
		1	0.5	−0.5	−1
Battery Pack Position (mm)	16	-10	-6	55	63.25
	8	-17.5	-13	55.37	59
	0	-31	-20	56	54
	-8	-38	-16.7	32	53

both the buoyancy engine position and the battery pack position.

6.3.3 Steady State Results From Numerical Model

To make a comparison with the steady state experimental data from the IOT Glider water tank tests, numerical results are obtained from a simplified version of the simulation model described in Chapter Three and Four.

As seen from test results, in the steady states of the IOT Glider, the turning rates of the vehicle in all three directions are zero and the velocity of the vehicle is constant. As a result, in the steady state the added mass terms are making no contributions to its motion any more. In addition, as no rolling motion is involved in the tests, the motion is reduced to only three degrees of freedom, surge, heave and pitch. The motion equations are simplified to be:

$$\begin{aligned} X_{HS}(x\_piston, x\_batt, \theta) + X_{Wing}(V_A, \alpha) + X_{Hull}(V_A, \alpha) = 0 \\ Z_{HS}(x\_piston, x\_batt, \theta) + Z_{Wing}(V_A, \alpha) + Z_{Hull}(V_A, \alpha) = 0 \\ M_{HS}(x\_piston, x\_batt, \theta) + M_{Wing}(V_A, \alpha) + M_{Hull}(V_A, \alpha) = 0 \end{aligned} \tag{6.1}$$

where HS means hydrostatic forces which can be evaluated from 3.13, $Wing$ means hydrodynamic drag and lift forces on the wings and $Hull$ means hydrodynamic drag

and lift forces on the bare hull. V_A is the total velocity of the IOT Glider, α is the angle of attack of the IOT Glider, θ is the pitching angle of the IOT Glider. Only the X and Z directions forces and the pitching moment in the body-fixed frame are involved.

The equation system has three unknown variables, V_A, α, θ, which can fully define the moving status of the vehicle in its steady state. x_piston and x_batt are the initial input variables. Using 'fsolve' function in MATLAB, the equation system can be solved. The evaluation of the hydrostatic and hydrodynamic forces and moments can be referred to Chapter 3 and Chapter 4 for details. The MATLAB scripts for solving the steady state variables can be found in Appendix F. The two input variables are varied as same as in the water tank test design. The results are summarized in the following tables.

Table 6.7: Rate of Ascent or Descent Results from Numerical Model (unit: m/s)

		Buoyancy Engine Position (cm)			
		Rising		Diving	
		1	0.5	-0.5	-1
Battery Pack Position (mm)	16	-0.1449	-0.0483	0.1743	0.3447
	8	-0.2093	-0.0622	0.1331	0.3068
	0	-0.2646	-0.0867	0.0855	0.2600
	-8	-0.3106	-0.1341	0.0619	0.2038

Table 6.8: Pitching Angle Results from Numerical Model (unit: deg)

		Buoyancy Engine Position (cm)			
		Rising		Diving	
		1	0.5	-0.5	-1
Battery Pack Position (mm)	16	-24.1	-8.6	36.4	48.4
	8	-31.9	-14.7	29.4	43.6
	0	-38.5	-21.4	21.2	38
	-8	-44.1	-29.6	14.5	31.3

From the prediction results and the experimental results, a comparison is available. Figures 6.11, 6.12, 6.13 and 6.14 show the comparison between prediction from the model and the result from the water tank tests. Clearly the water tank test results confirm the trend in the predictions. Although most results are well predicted from numerical model, the agreement between the figures is not perfect. It might be explained by the effects from the existence of the communication cable during the water tests. The communication cable is not included in the numerical simulation model. However, the cable still has inertia and will involve added mass and hydrodynamic drag forces during the motion of the IOT Glider, even if foam blocks have been attached to make it neutrally buoyant. The comparison gives confidence on both the numerical model and the water tests. It also stimulates future improvement of the water tank tests. Future experiments should eliminate the communication cable by allowing the IOT Glider be pre-programmed to dive and rise autonomously.

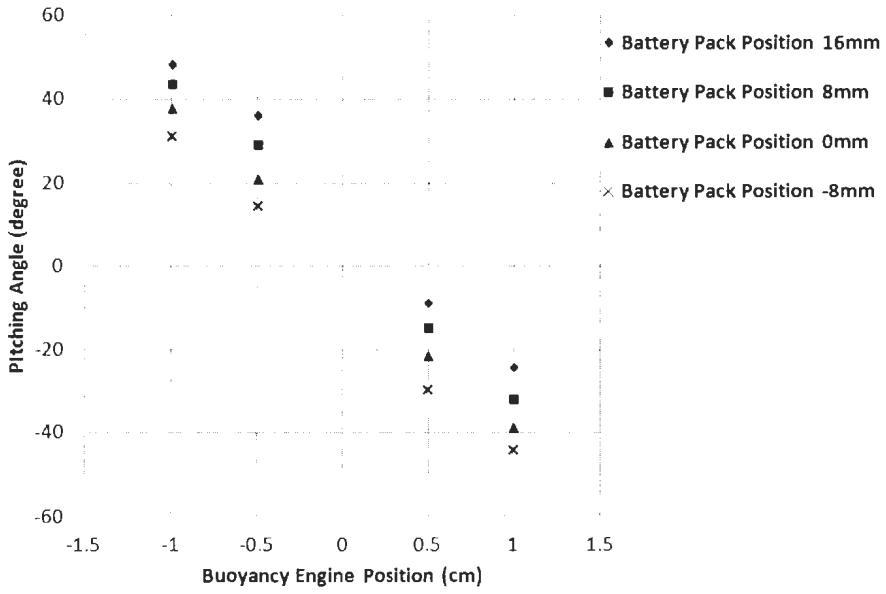


Figure 6.11: The Pitching Angle Results from Simulation

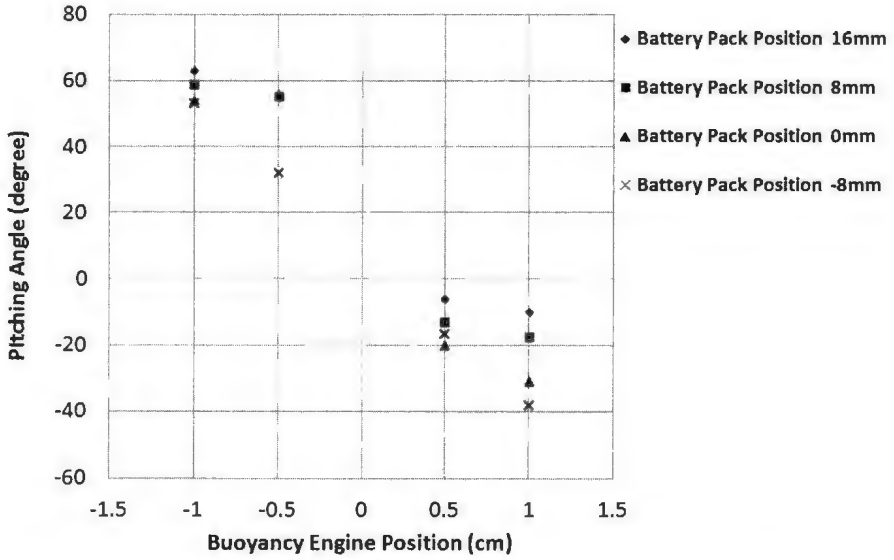


Figure 6.12: The Pitching Angle Results (Left: from Simulation; Right: from Experiments)

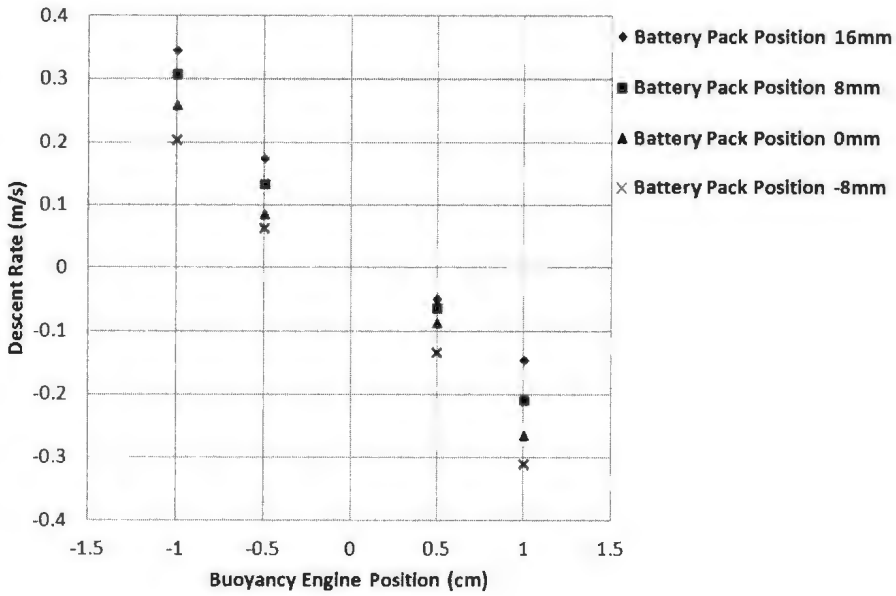


Figure 6.13: The Descent Rate Results from Simulation

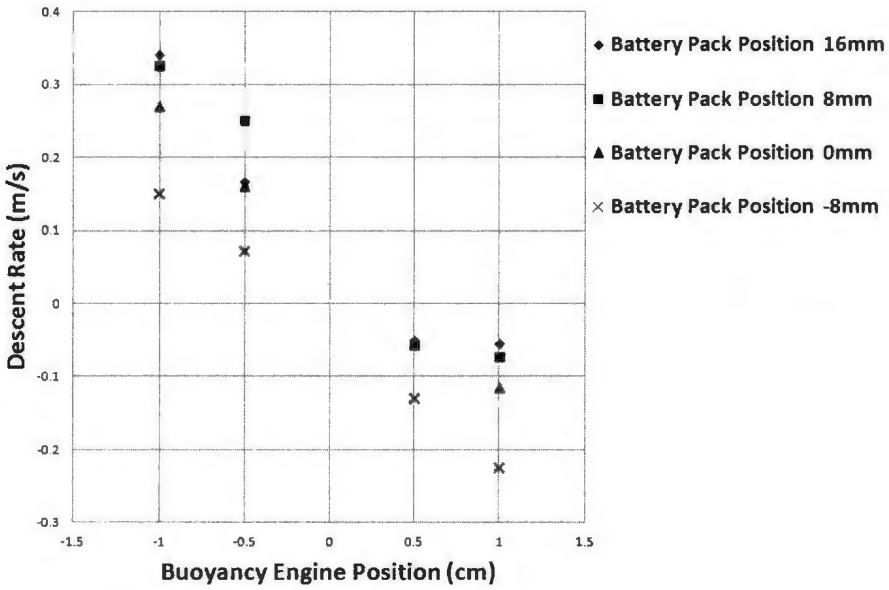


Figure 6.14: The Descent Rate Results from Experiments)

6.4 Summary

A set of diving and rising experiments on the IOT Glider was conducted varying its buoyancy engine position and battery pack position. Before the tests, a leak check was performed after sealing the glider. Foam blocks were attached to the communication and power cable to eliminate buoyancy effects due to the cable. Test data were compared from numerical model predictions. The results showed same trend and similar steady state variables. The tests proved that the IOT Glider is a useful tool for hydrodynamic study of this type of vehicle.

Chapter 7

Conclusions

The design and assembly of the lab-scale underwater glider, the IOT Glider has been completed. The pitch and roll mechanism design has been rechecked and realized. The wing and wing mount design allows the wing position to be adjustable. The nose section and tail section shapes make the IOT Glider torpedo-shaped. Software development for the glider has been accomplished. Equipped with various sensors on board, the IOT Glider is able to dive and rise underwater and is capable of active pitch and roll control. Tests of the glider pitch and roll mechanism has been conducted. The mechanism is working properly. Some mass properties of the glider have been measured by testing the glider in air. PID control has been conducted for pitching angle control in air. Testing in water has been conducted varying the buoyancy engine position and battery pack longitudinal position.

From the tests in air and the tests underwater, it was found that the IOT Glider is sufficient for a research platform. From the water tests we found that during the motions of this glider, its rate of ascent and descent and pitching angle are dependent on both its buoyancy engine position and battery pack position.

To study the hydrodynamic performance of the glider, CFD analysis has been con-

ducted on the glider bare hull and the wings. The results were also compared to available experiment data. Hydrodynamic damping coefficients have been evaluated. The added mass matrix of the IOT Glider has been evaluated by ESAM. Calculation of the mass properties when the buoyancy engine and battery pack positions change has been completed.

Future study might be on the items below:

- Study the wing position effects.
- Complete the development of a PID controller for the roll mechanism.
- Install the MicroStrain inertial measurement unit (IMU) and record the pitch rate, yaw rate and roll rate during turns.
- Install an underwater connector on the outer surface of the hull so that the data cable can be attached and removed quickly for transferring data to and from the glider.
- Add a 314 MHz RF modem and RF whip antenna to the tail section so that the glider can be controlled by an RF transmitter while it is submerged.
- Install a simple grid on the far wall of the Flume Tank at the Marine Institute. Perform dives without turning. Use multiple video cameras to record the glider's trajectories. Analyse the video recordings to measure (i) the glide speed, (ii) the glide angle, (iii) the angle of attack for straight-ahead dives and climbs. From these measurements deduce suitable values for the drag and lift forces which act on the glider.
- Design, fabricate and test a nose section which is equipped with multiple pressure sensors so that the forward end of the glider can be used as a real-time

5-hole or 7-hole probe for measuring the instantaneous angle of attack (AOA). Calibrate this AOA sensor. Perform dives and turns and measure the instantaneous AOA.

- Upgrade the 6-DOF simulation code to include the results from the tests. Design some specific manoeuvres for the glider which will enhance the presence of certain hydrodynamic coefficients. Perform those experiments and compare the values of the experimentally-determined hydrodynamic coefficients with the value predicted by the CFD process, analytical and other methods.

Bibliography

[1] R. Bachmayer, N. E. Leonard, J. Graver, E. Fiorelli, P. Bhatta, and D. Paley. Underwater gliders: Recent developments and future applications. *Underwater Technology*, 20-23 April:195–200, April 2004.

[2] Christopher Baker. Estimating drag forces on submarine hulls. Technical Report CR 2004-125, Defence R&D Canada - Atlantic, September 2004.

[3] Robert D. Blevins. *Applied Fluid Dynamics Handbook*. Van Nostrand Reinhold Company Inc., 1984.

[4] Brian Claus. Development of an underwater glider equipped with an auxiliary propulsion module. Master's thesis, Memorial University of Newfoundland, 2010.

[5] Charles C. Eriksen, T. James Osse, Russell D. Light, Timothy Wen, Thomas W. Lehman, Peter L. Sabin, John W. Ballard, and Andrew M. Chiodi. Seaglider: A long-range autonomous underwater vehicle for oceanographic research. *IEEE JOURNAL OF OCEANIC ENGINEERING*, VOL. 26:424–436, 2001.

[6] Thor I. Fossen. *Guidance and Control of Ocean Vehicles*. John Wiley & Sons, 1994.

[7] Moqin He and Christopher D. Williams. A simple method for hydrodynamic prediction of the manoeuvring performance of a twin-pod vehicle. In *The 29th American Towing Tank Conference*, 2010.

[8] Gerald Hewitt. Design of pitch and roll control for the IOT Glider. Technical Report SR-2005-28, National Research Council Canada, December 2005.

[9] Sighard F. Hoerner. *Fluid Dynamic Drag.* Published by author, 1965.

[10] Nicholas Janes. Design of a buoyancy engine for an underwater glider. Technical Report LM-2004-13, NRC-IOT, April 2004.

[11] Timothy Prestero. Verification of a six-degree of freedom simulation model for the remus autonomous underwater vehicle. Master's thesis, MIT, August 2001.

[12] Jeff Sherman, Russ E. Davis, W. B. Owens, and J. Valdes. The autonomous underwater glider spray. *IEEE JOURNAL OF OCEANIC ENGINEERING*, VOL. 26:437–446, 2001.

[13] Ben Skillings. Buoyancy engine construction and design for an underwater glider. Technical Report LM-2004-21, NRC-IOT, August 2004.

[14] Giorgio F. Stante. Dynamic simulation of the slocum glider. Technical report, McGill University, 2007.

[15] Christopher Warren. Testing and evaluation of a buoyancy engine for an underwater glider. Technical Report SR-2005-03, NRC-IOT, April 2005.

[16] George D. Watt. Estimates for the added mass of a multi-component, deeply submerged vehicle part one: Theory and program description. Technical report, Defence Research Establishment Atlantic (DREA), 1988.

[17] Douglas C. Webb, Paul J. Simonetti, and Clayton P. Jones. Slocum: An underwater glider propelled by environmental energy. volume 26, pages 447–452, 2001.

[18] Christopher D. Williams, Ralf Bachmayer, and Brad deYoung. Progress in predicting the performance of ocean gliders from at-sea measurements. In *Oceans*, 2008.

[19] Steven Williams. Trim, ballast and battery systems in a buoyancy engine for a underwater glider. Technical Report SR-2005-16, NRC-IOT, 2005.

Appendix A

Pitch Angle PID Control Plots

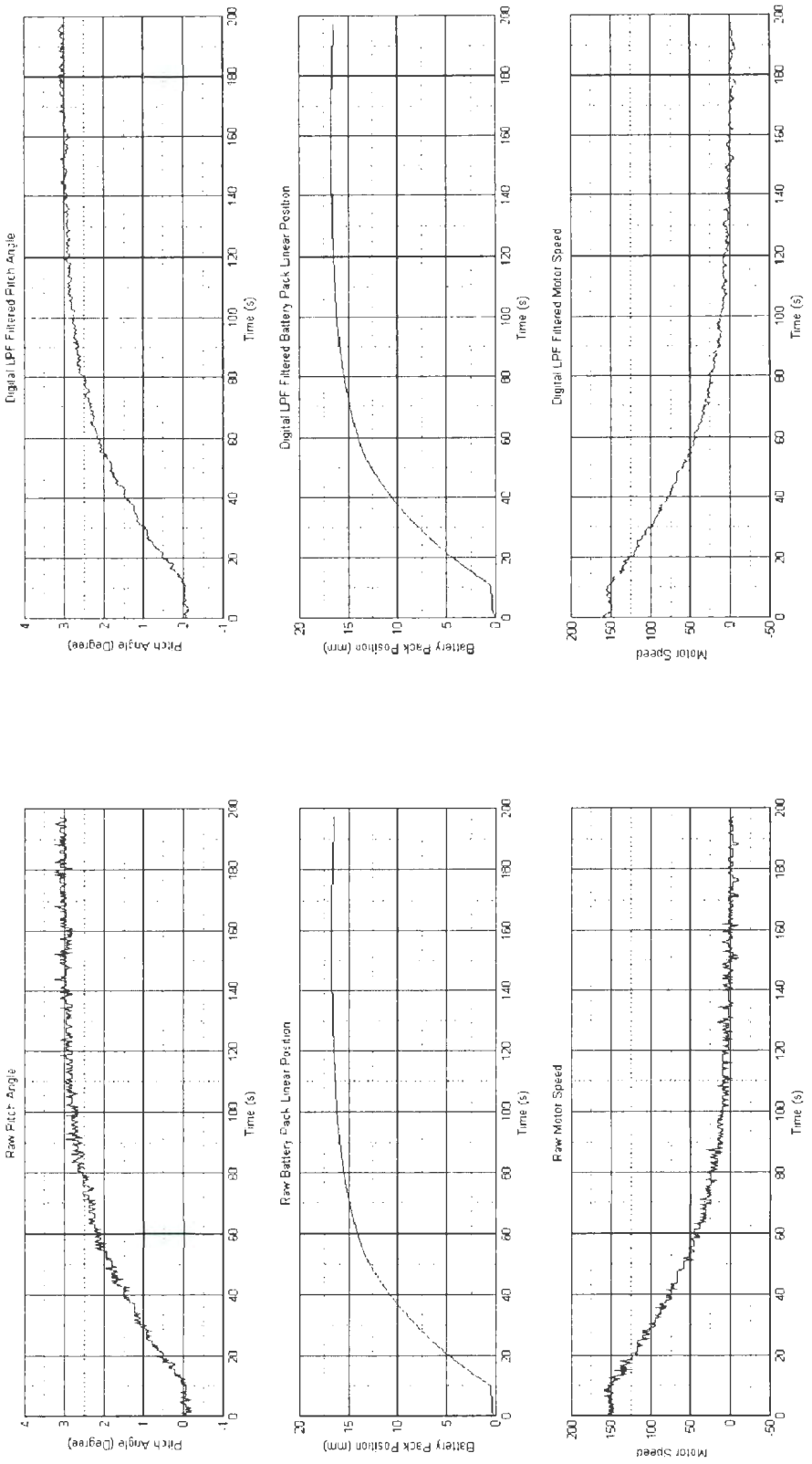


Figure A.1: P Gain 50, I Gain 0, D Gain 0, Saturation ±200

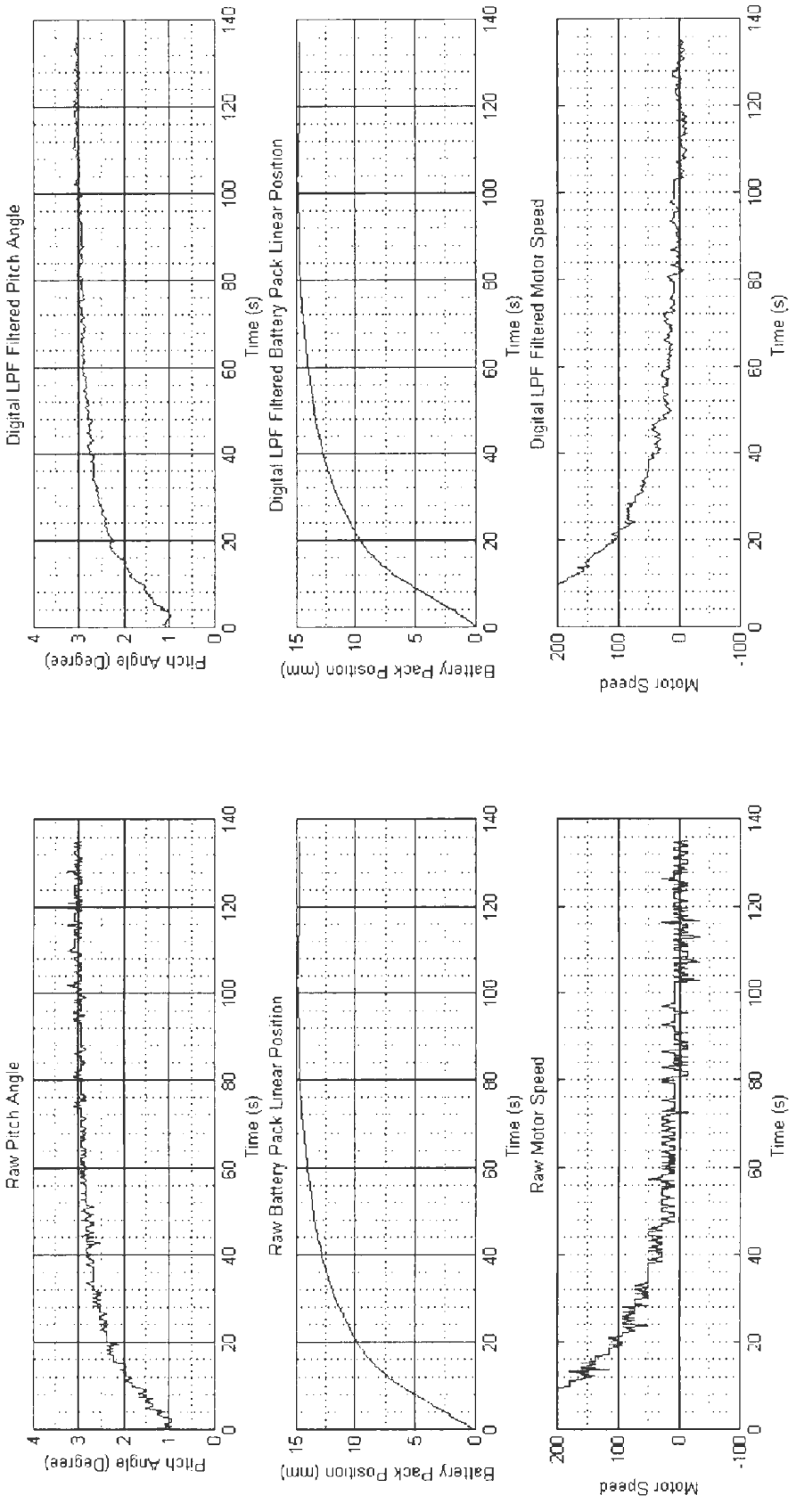


Figure A.2: P Gain 150, I Gain 0, D Gain 0, Saturation ±200

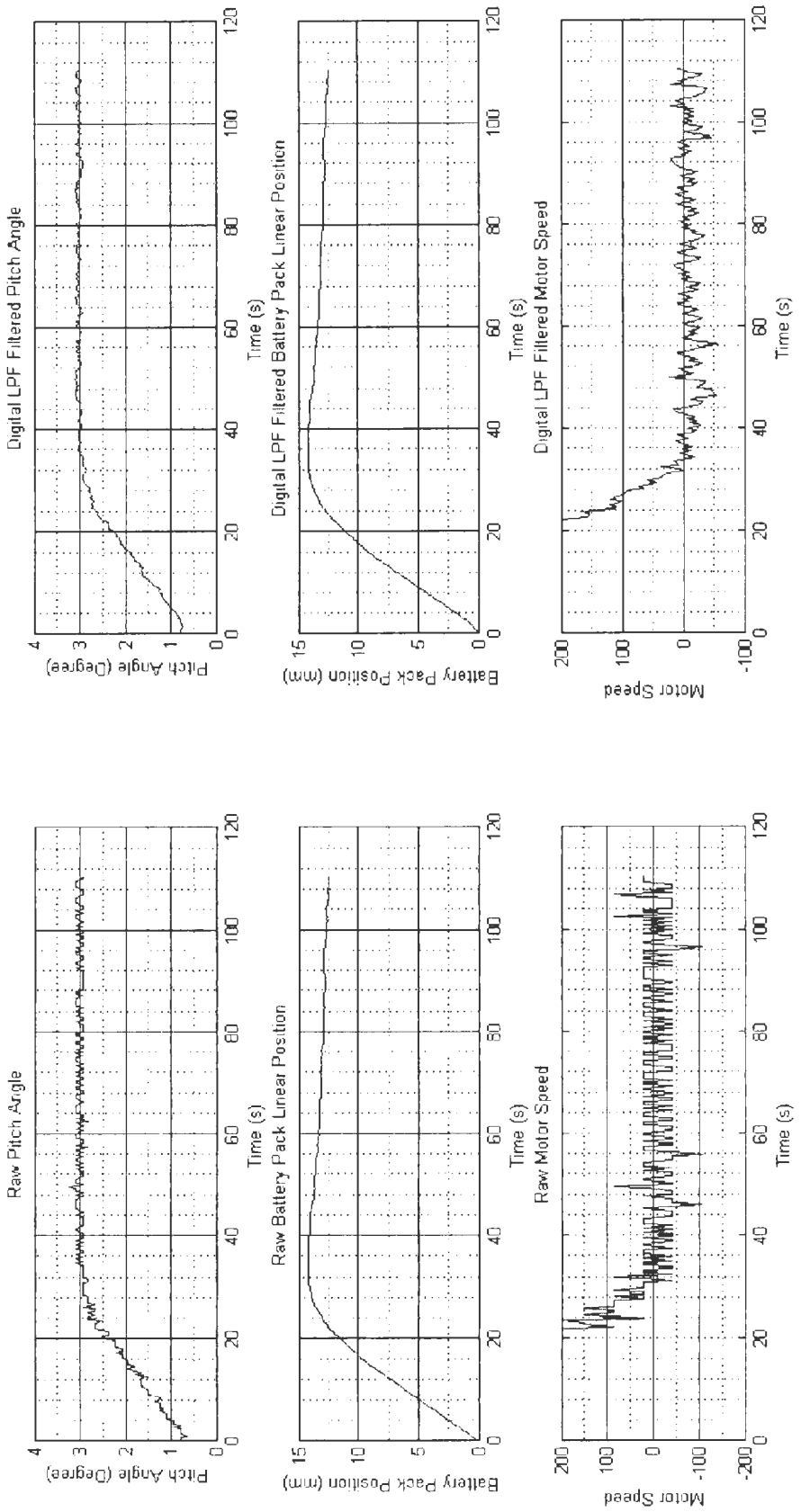


Figure A.3: P Gain 450, I Gain 0, D Gain 0, Saturation ±200

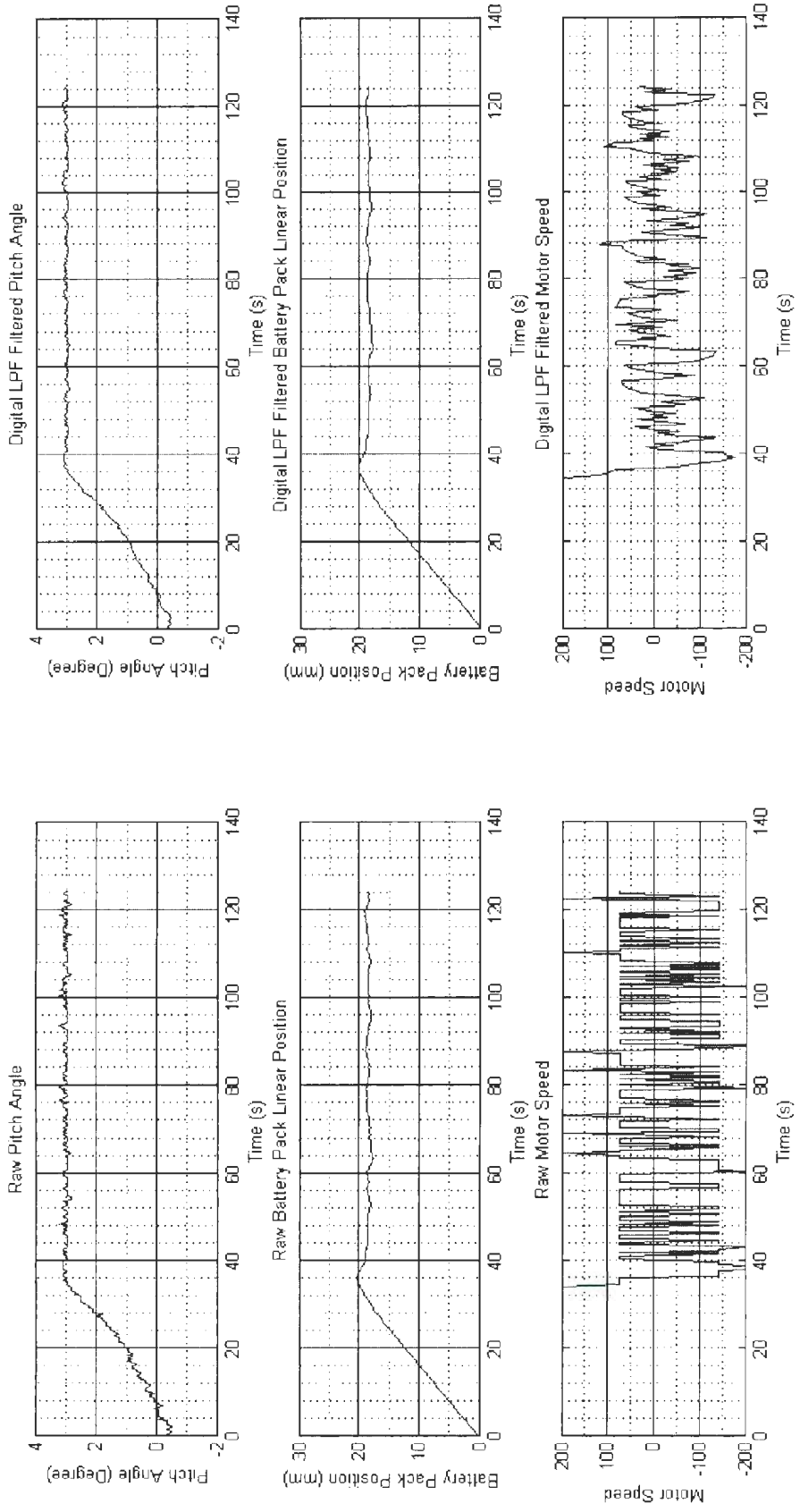


Figure A.4: P Gain 1500, I Gain 0, D Gain 0, Saturation ±200

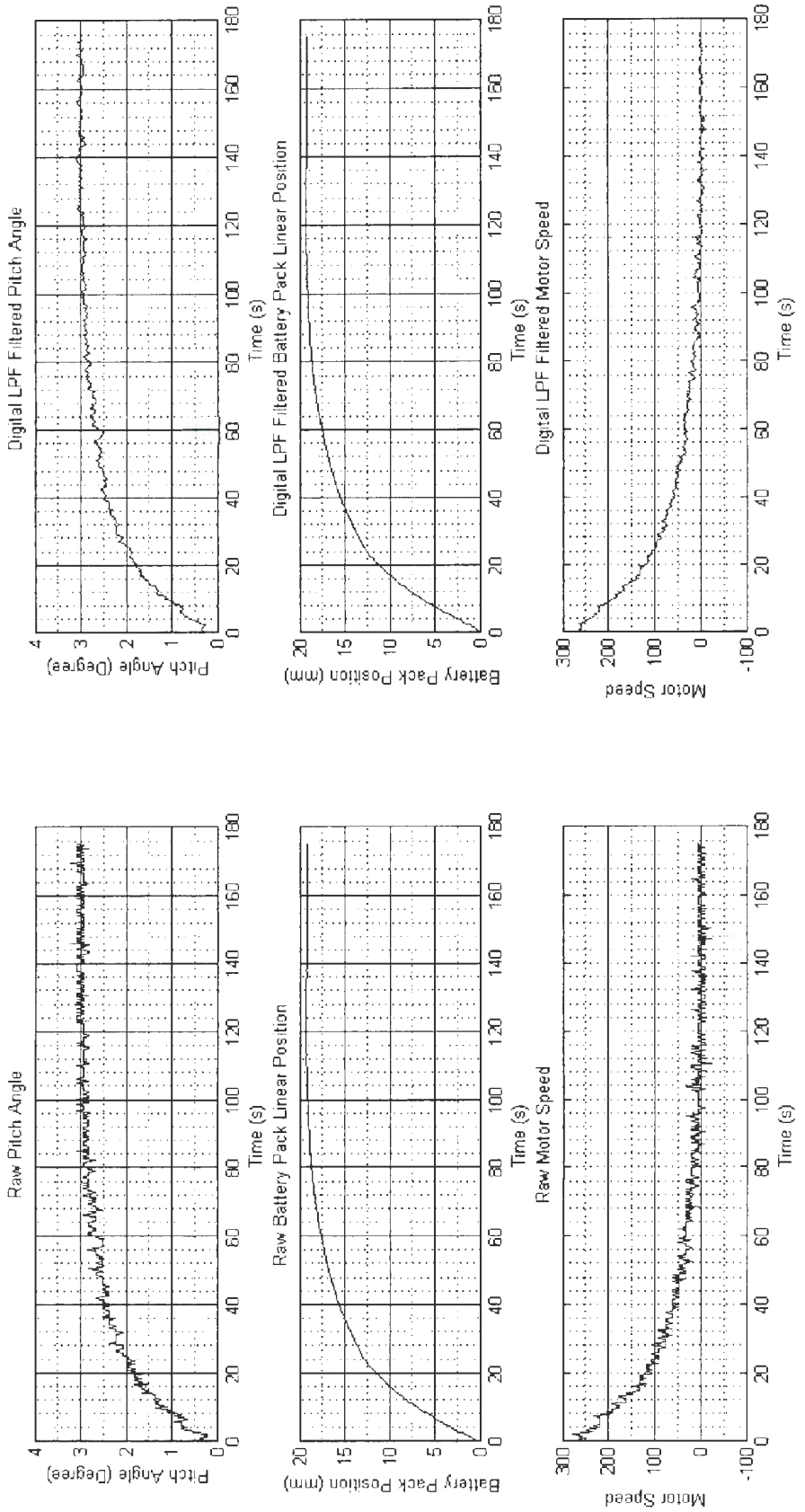


Figure A.5: P Gain 100, I Gain 0, D Gain 0, Saturation ±400

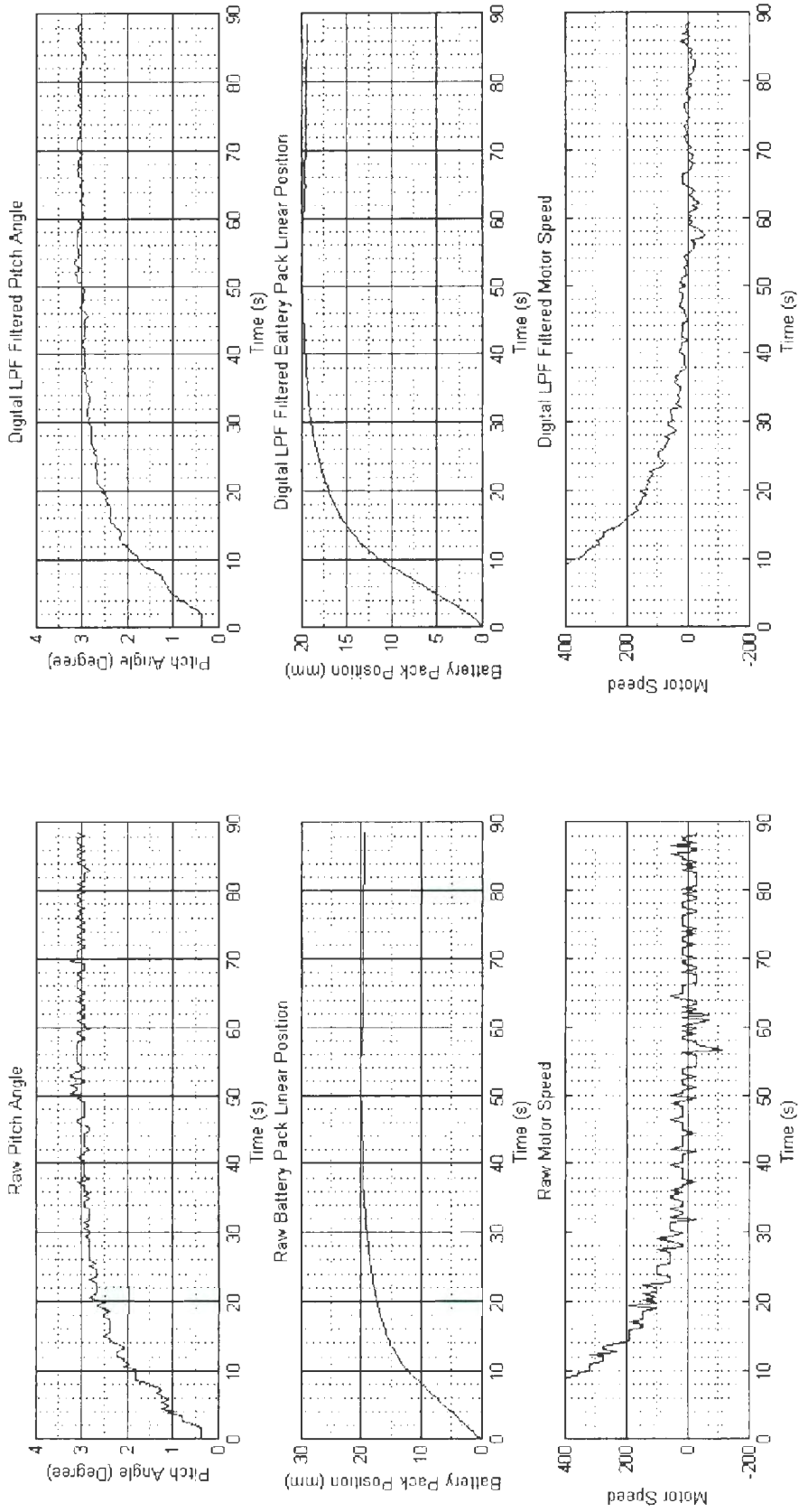


Figure A.6: P Gain 300, I Gain 0, D Gain 0, Saturation ±400

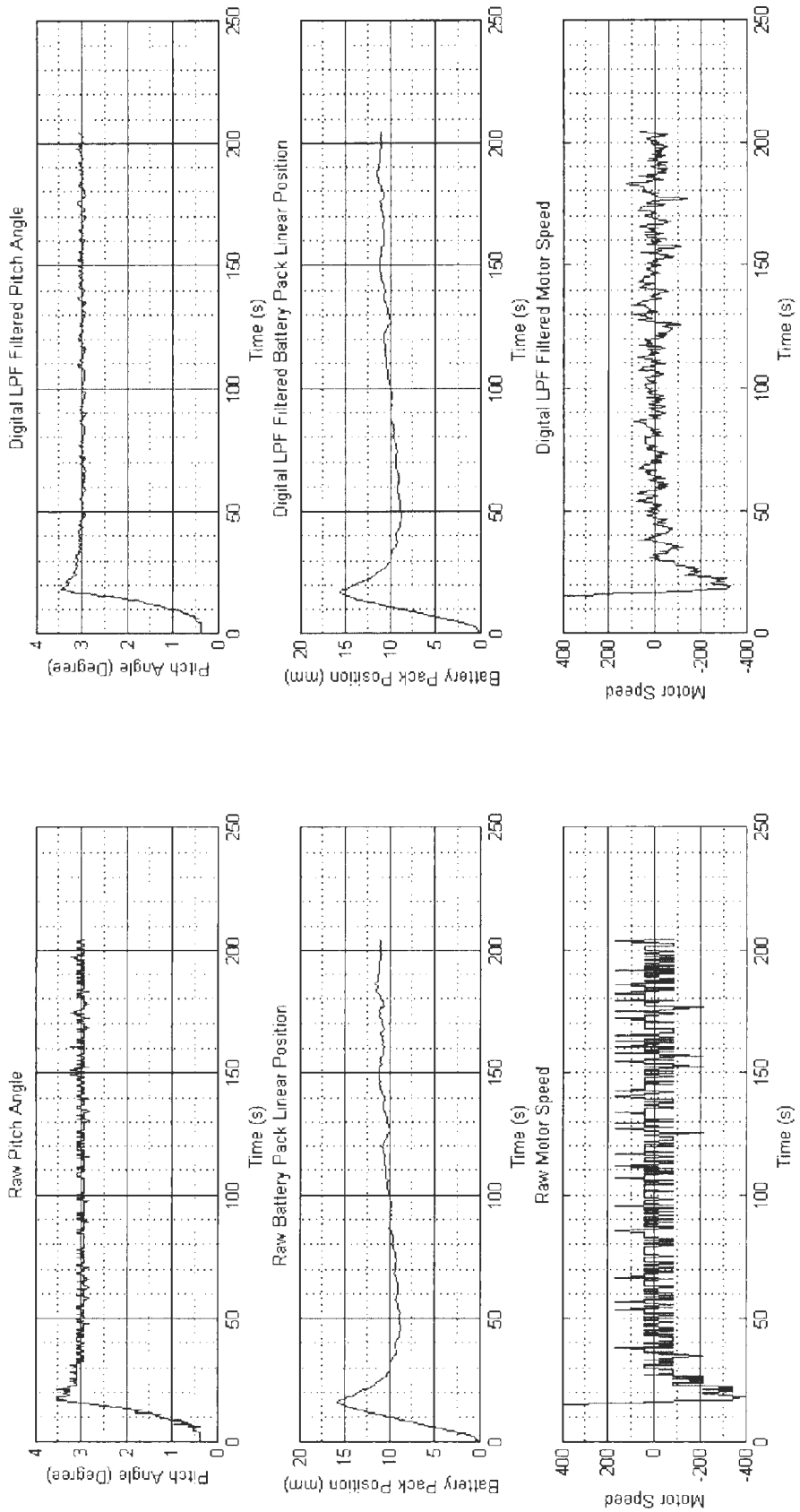


Figure A.7: P Gain 900, I Gain 0, D Gain 0, Saturation ±400

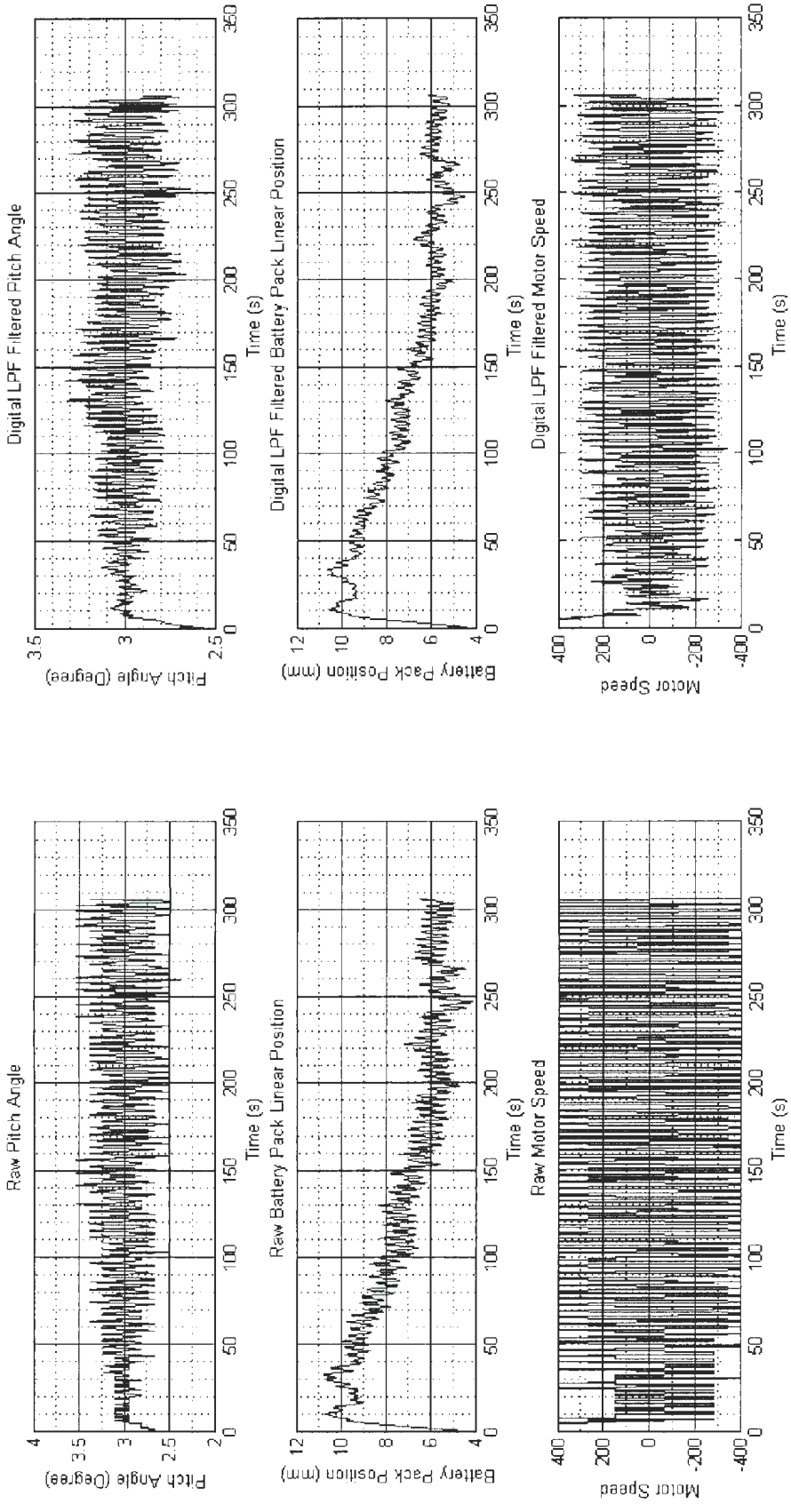


Figure A.8: P Gain 3000, I Gain 0, D Gain 0, Saturation ±400

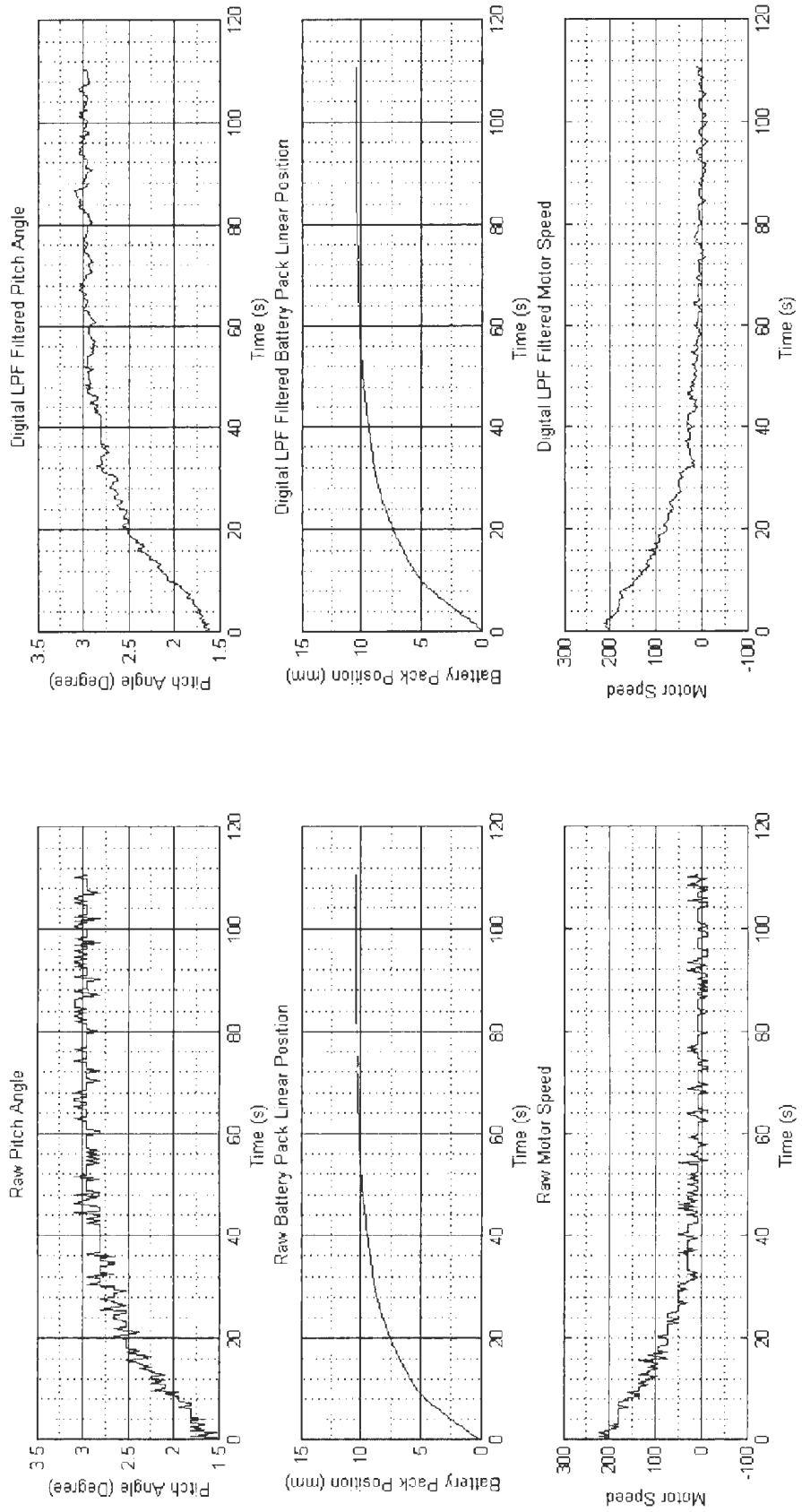


Figure A.9: P Gain 150, I Gain 0, D Gain 0, Saturation ±600

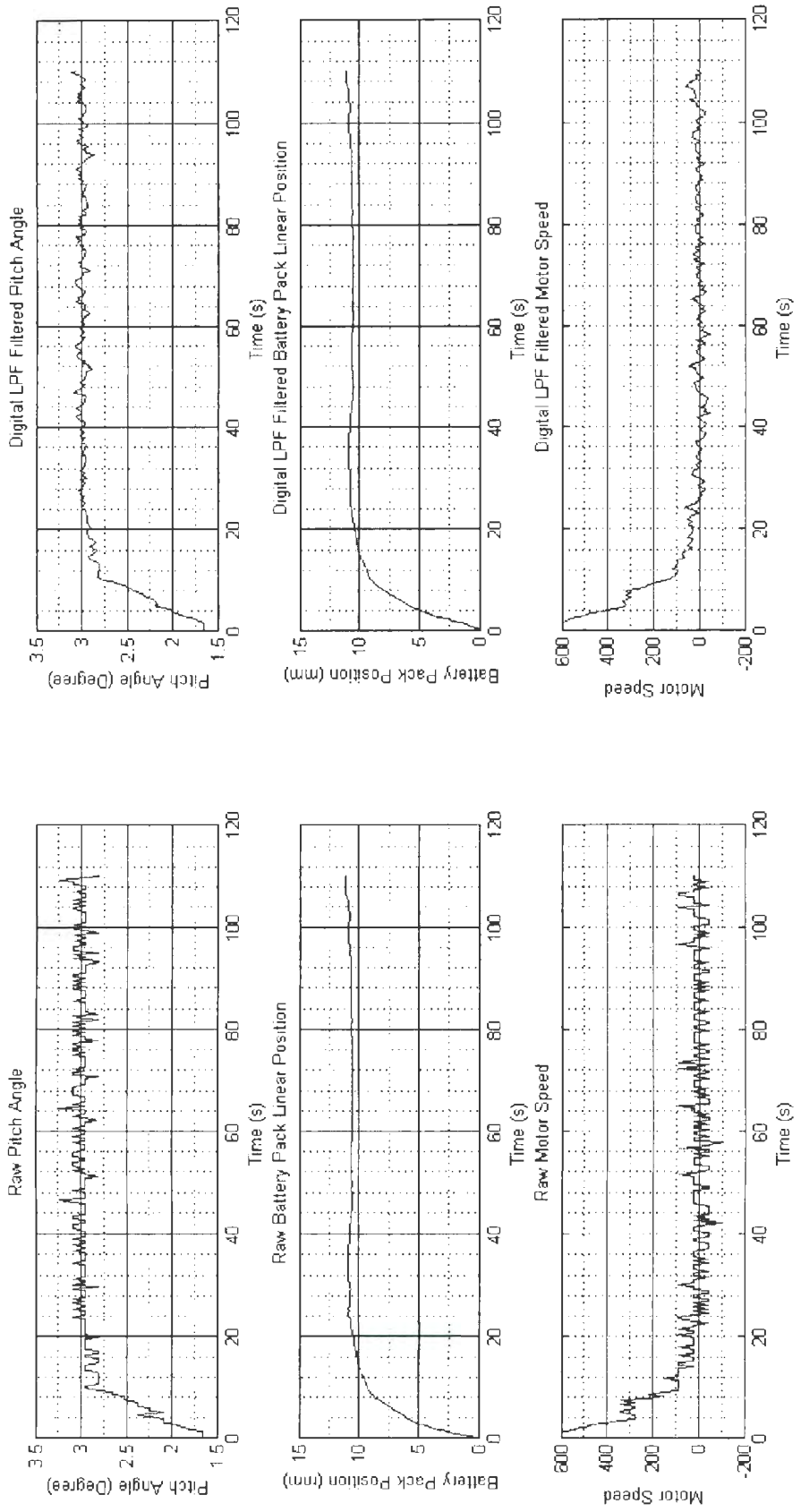


Figure A.10: P Gain 450, I Gain 0, D Gain 0, Saturation ±600

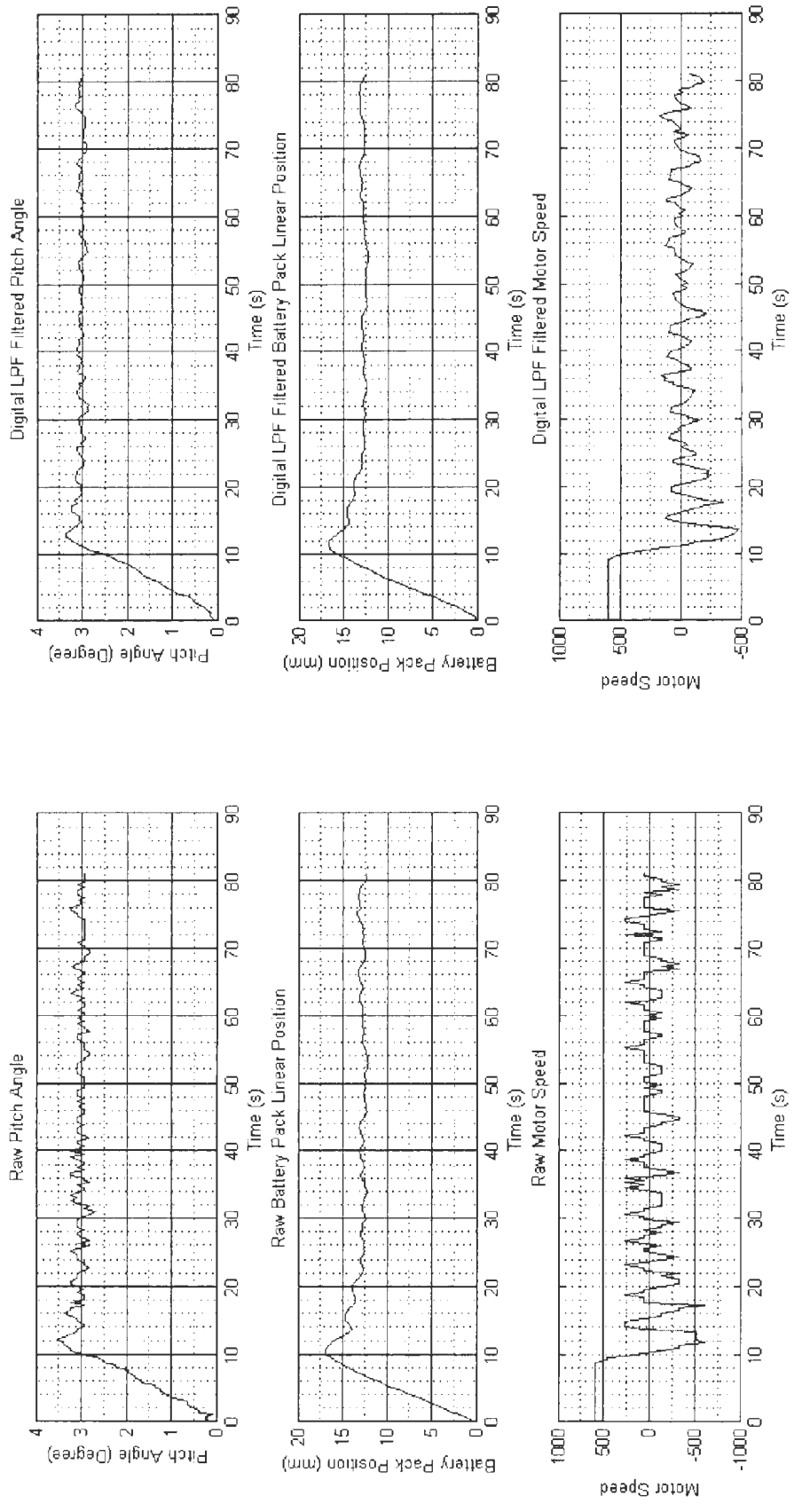


Figure A.11: P Gain 1350, I Gain 0, D Gain 0, Saturation ±600

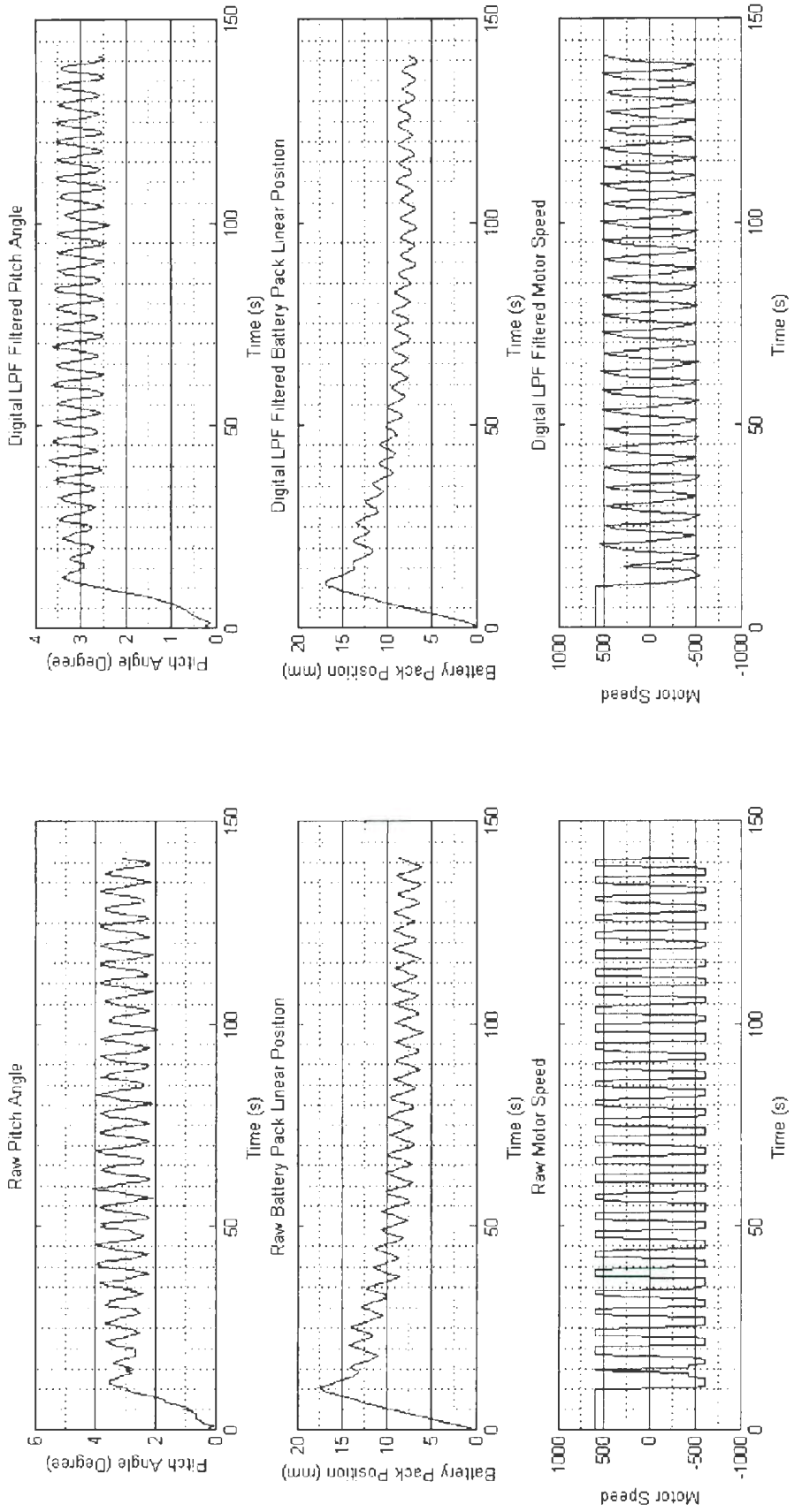


Figure A.12: P Gain 4500, I Gain 0, D Gain 0, Saturation ±600

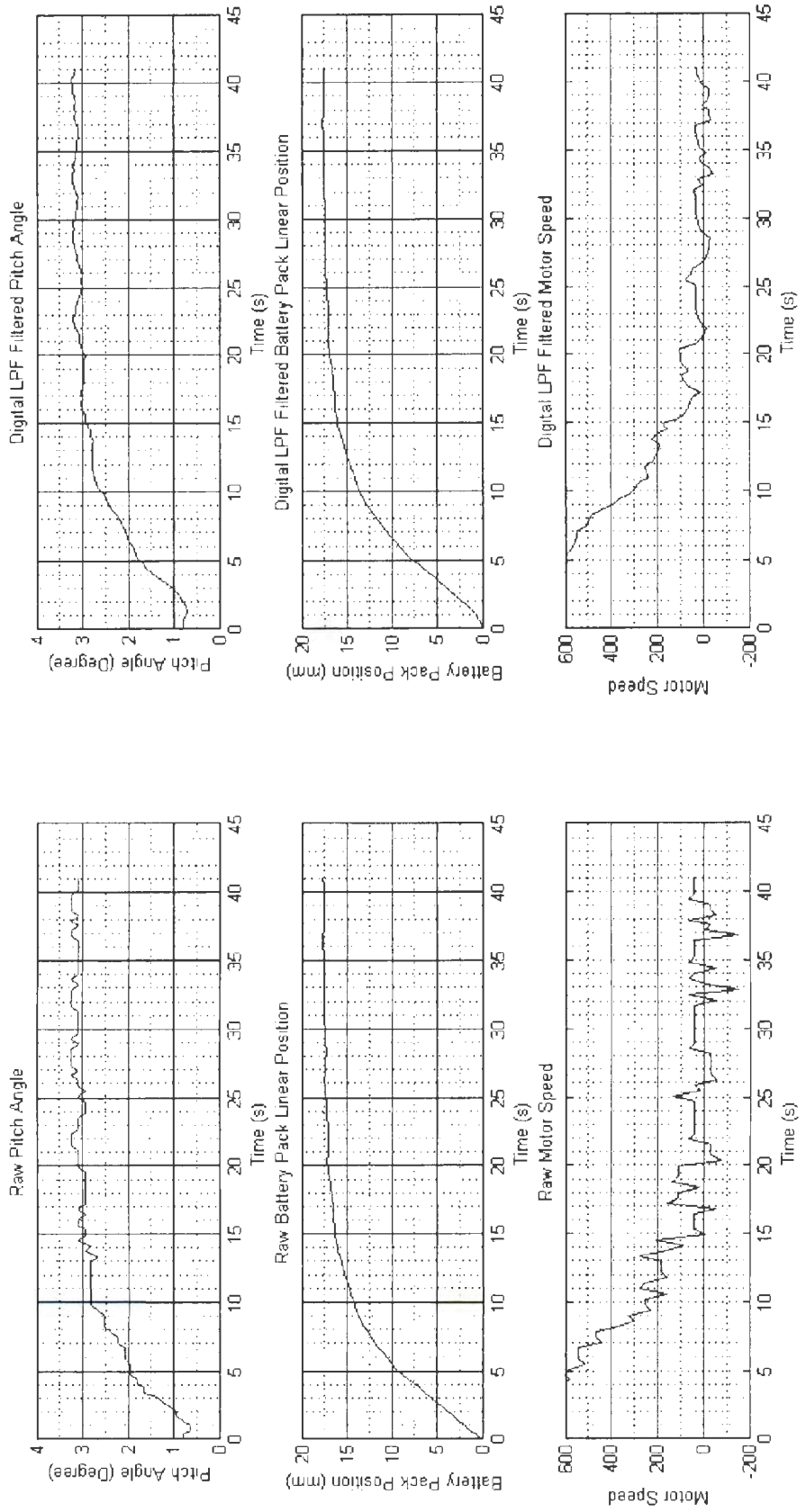


Figure A.13: P Gain 500, I Gain 10, D Gain 50, Saturation ±600

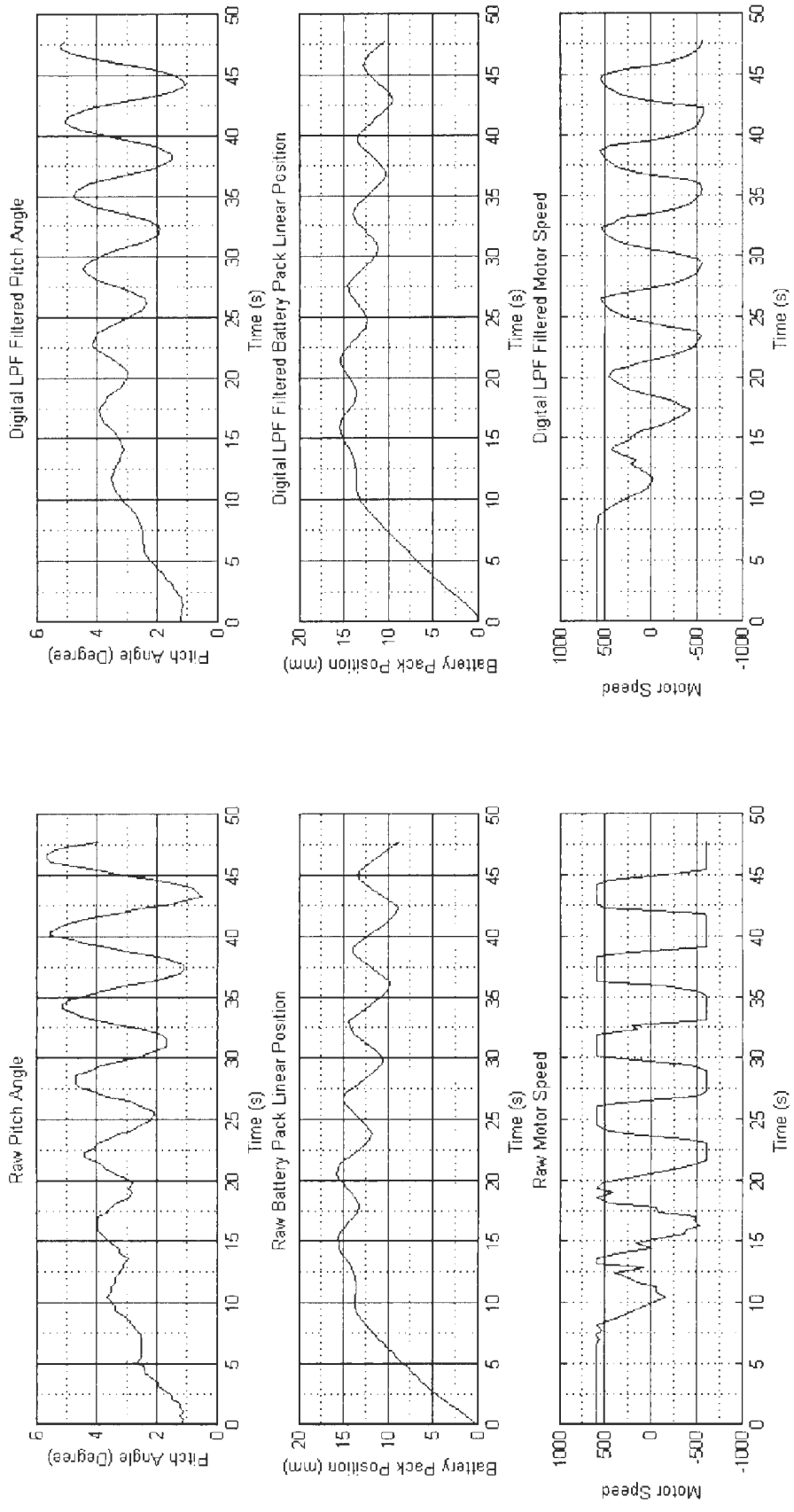


Figure A.14: P Gain 810, I Gain 100, D Gain 200, Saturation ±600

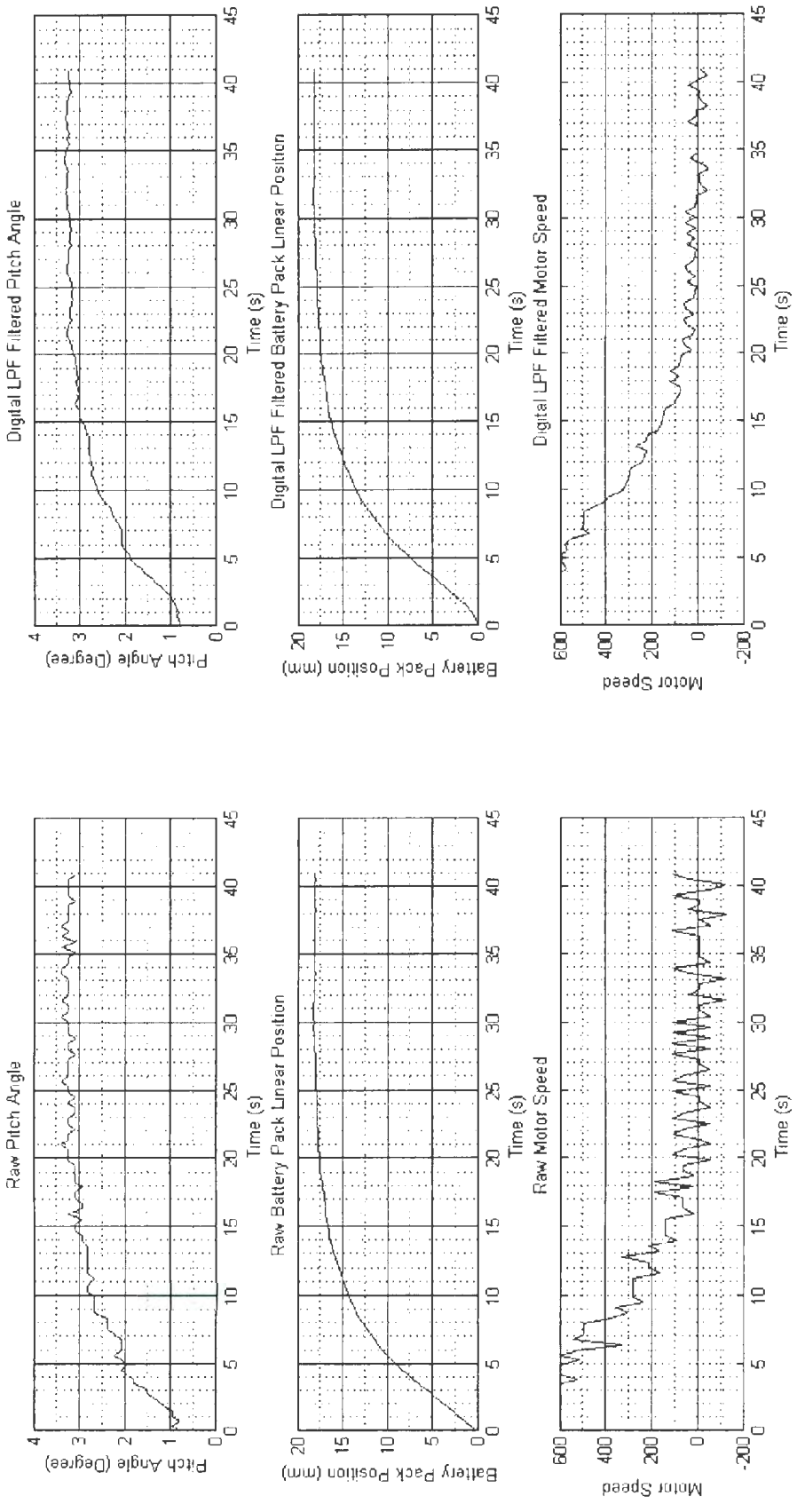


Figure A.15: P Gain 500, I Gain 15, D Gain 100, Saturation ±600

Appendix B

Deep Water Tank Test Plots

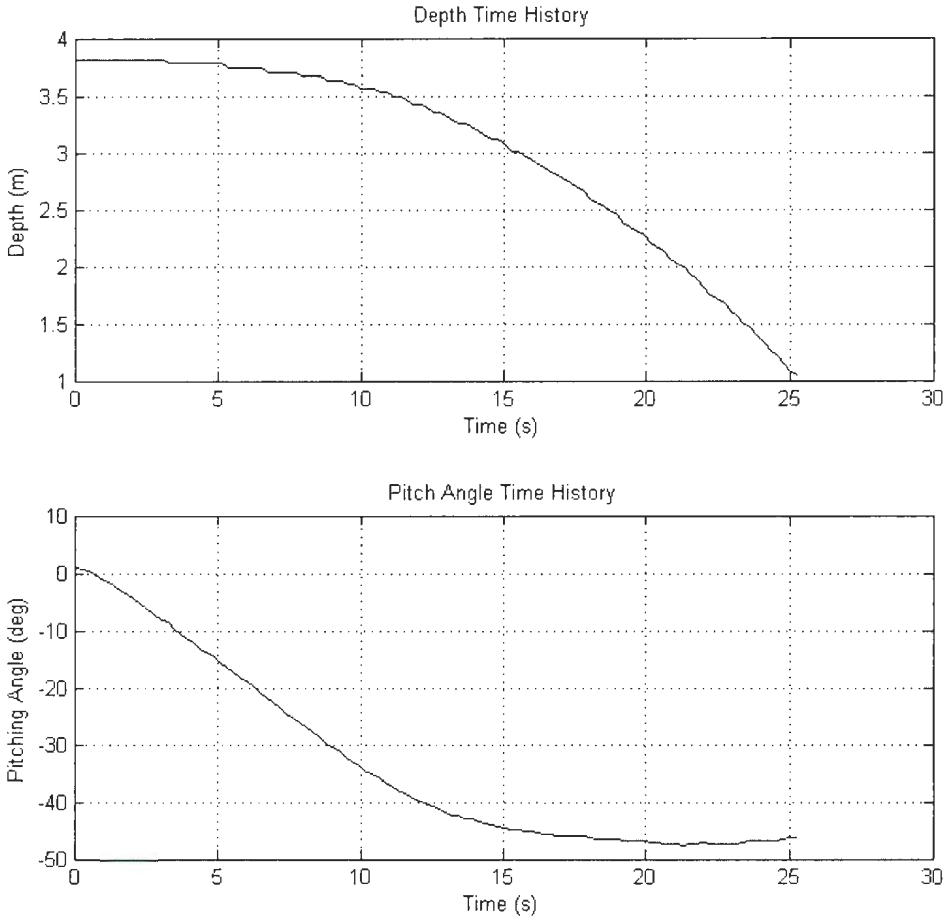


Figure B.1: Buoyancy Engine 1 *cm*, Battery Pack 0 *mm*, Rising

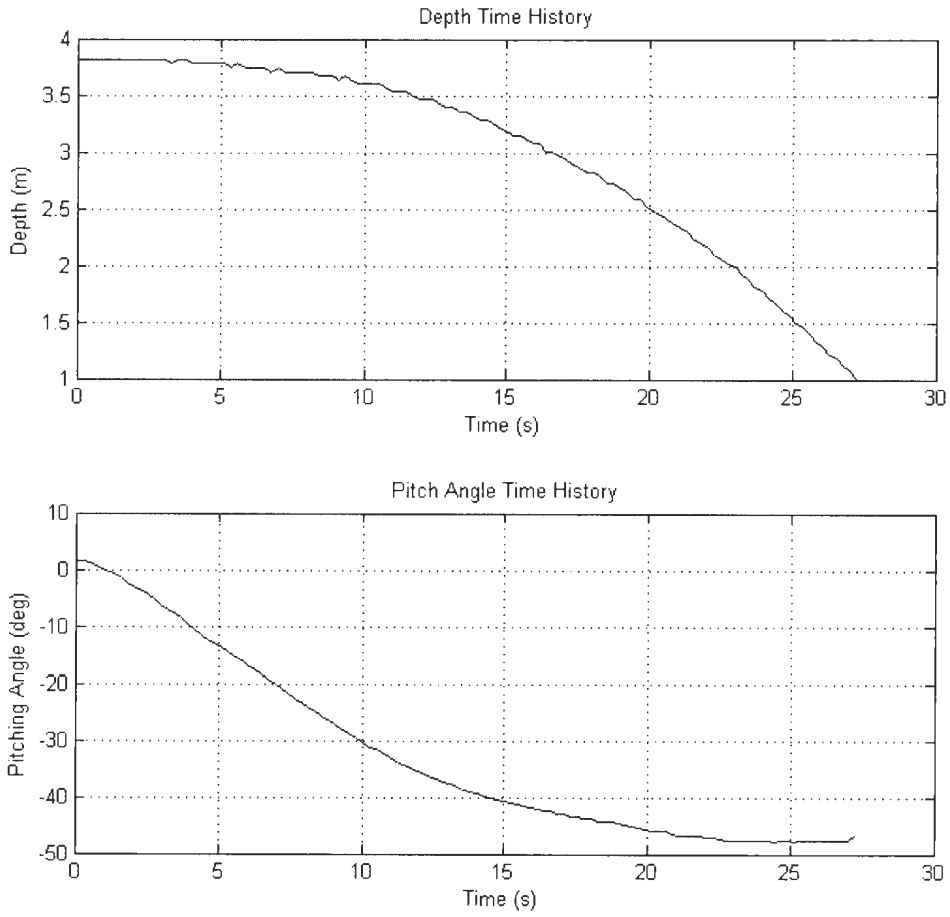


Figure B.2: Buoyancy Engine 1 *cm*, Battery Pack 8 *mm*, Rising

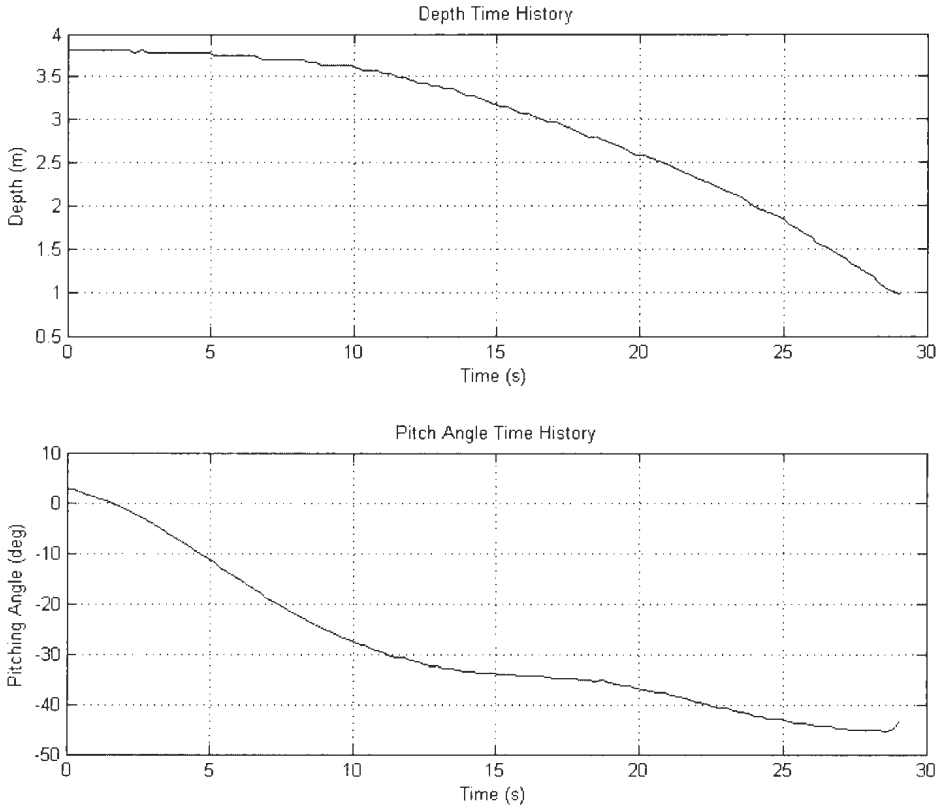


Figure B.3: Buoyancy Engine 1 cm, Battery Pack 16 mm, Rising

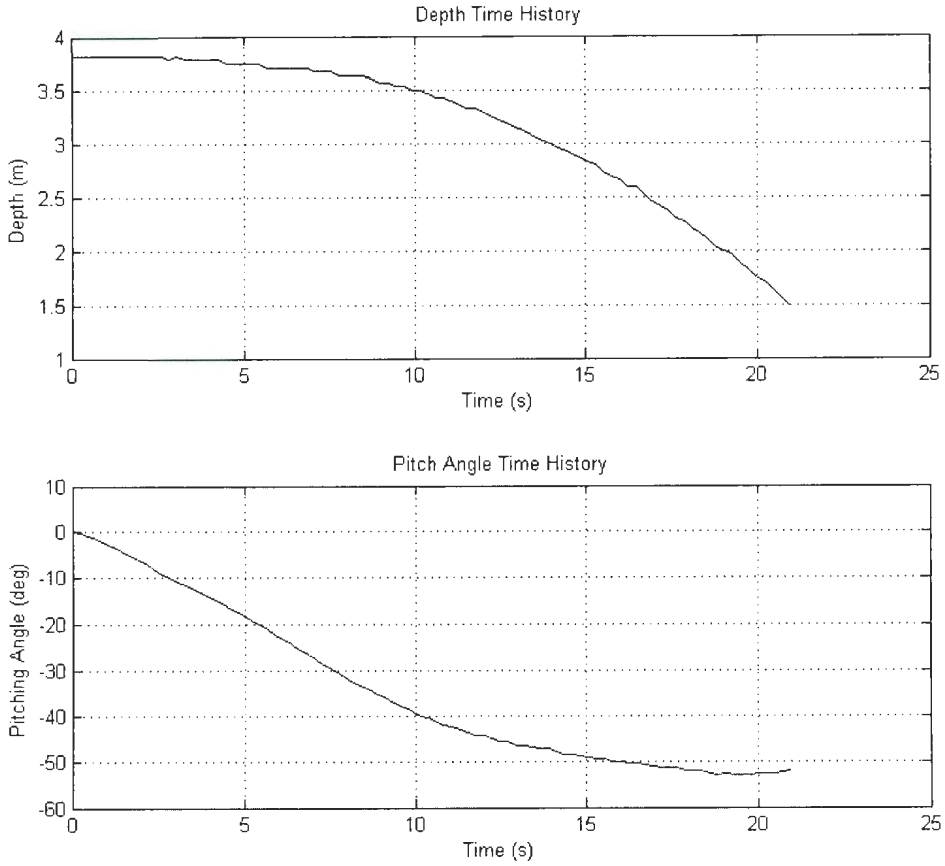


Figure B.4: Buoyancy Engine 1 cm, Battery Pack -8 mm, Rising

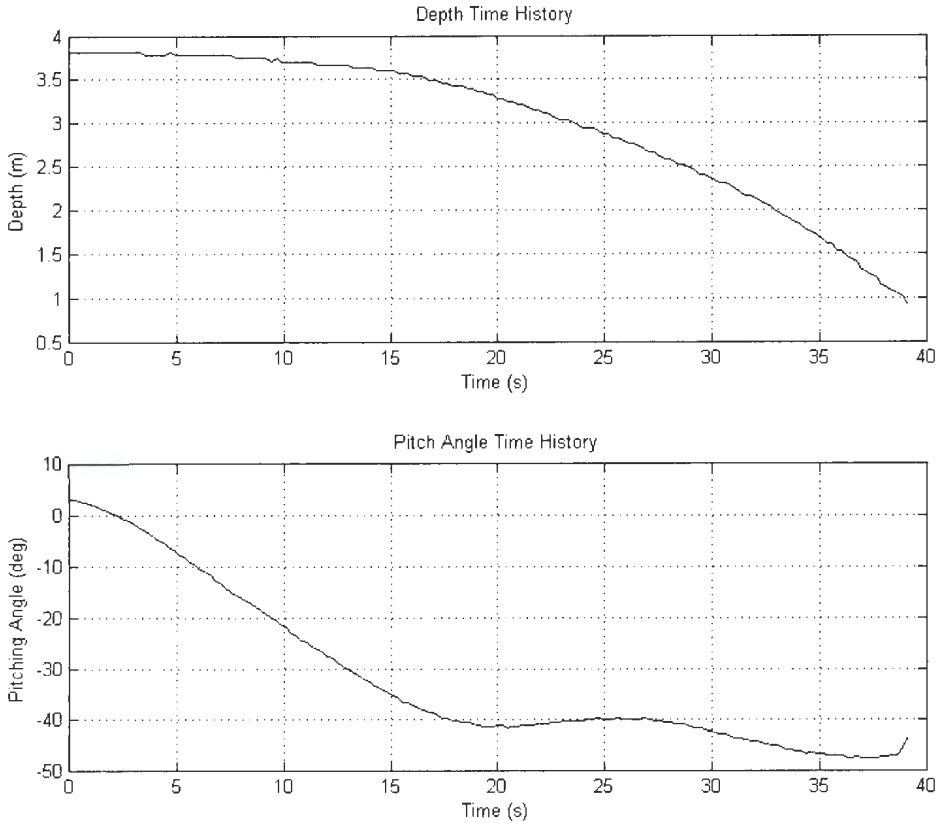


Figure B.5: Buoyancy Engine 0.5 cm, Battery Pack 0 mm, Rising

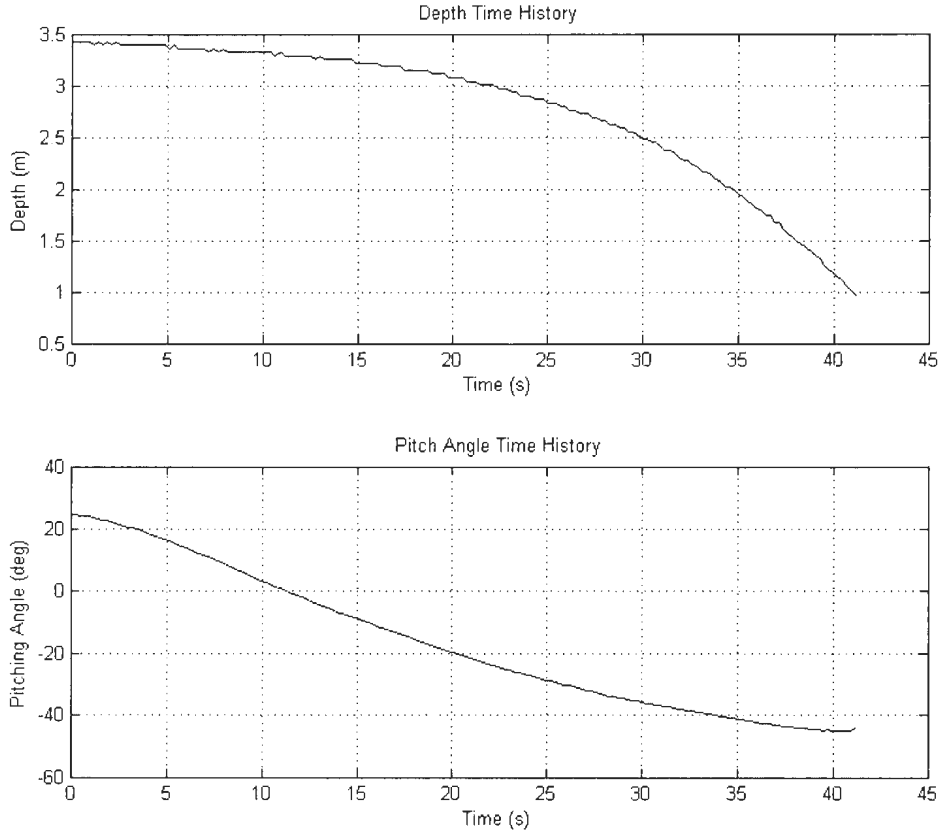


Figure B.6: Buoyancy Engine 0.5 cm, Battery Pack 8 mm, Rising

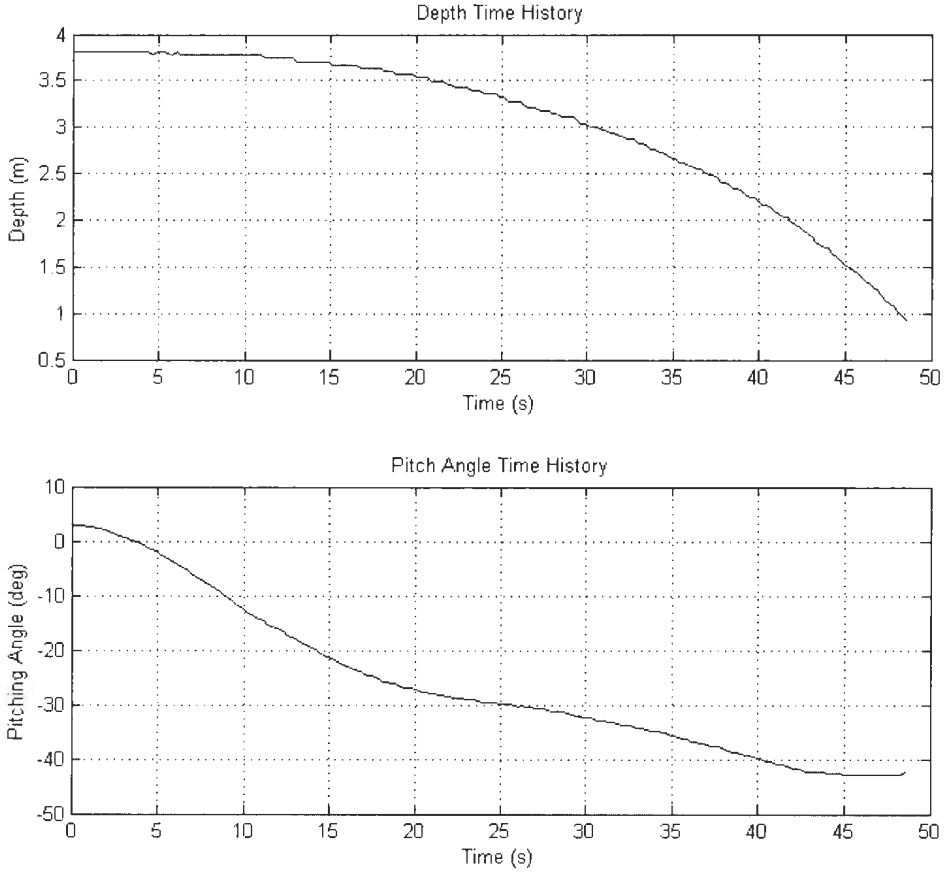


Figure B.7: Buoyancy Engine 0.5 cm, Battery Pack 16 mm, Rising

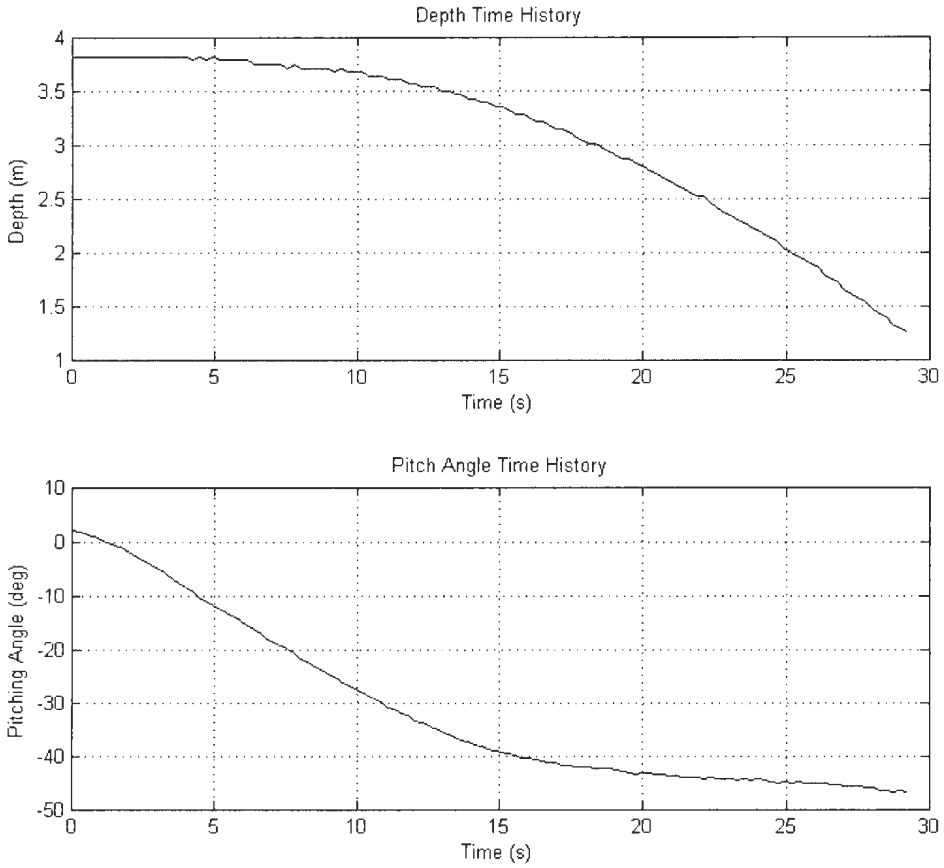


Figure B.8: Buoyancy Engine 0.5 cm, Battery Pack -8 mm, Rising

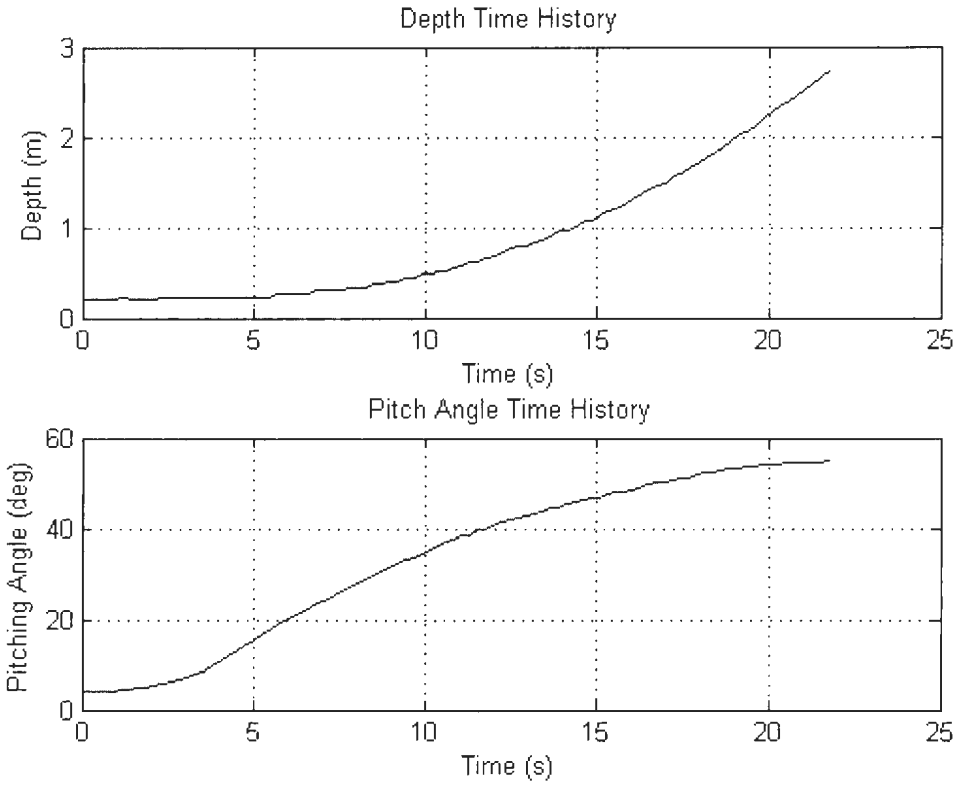


Figure B.9: Buoyancy Engine -1 cm, Battery Pack 0 mm, Diving

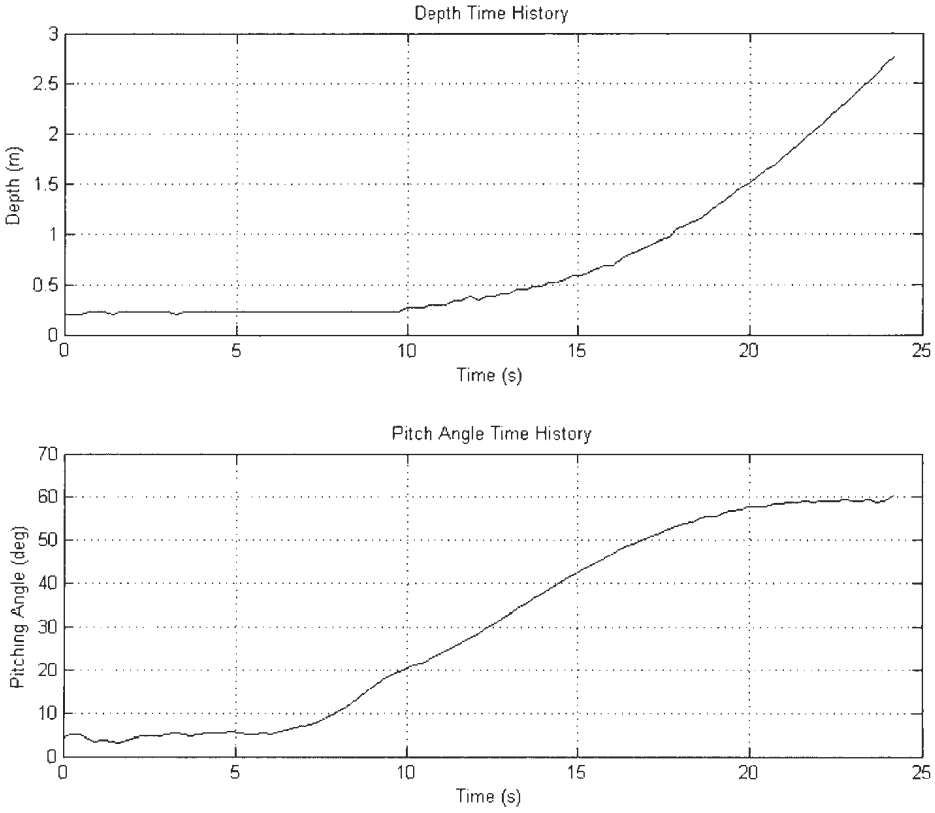


Figure B.10: Buoyancy Engine -1 cm, Battery Pack 8 mm, Diving

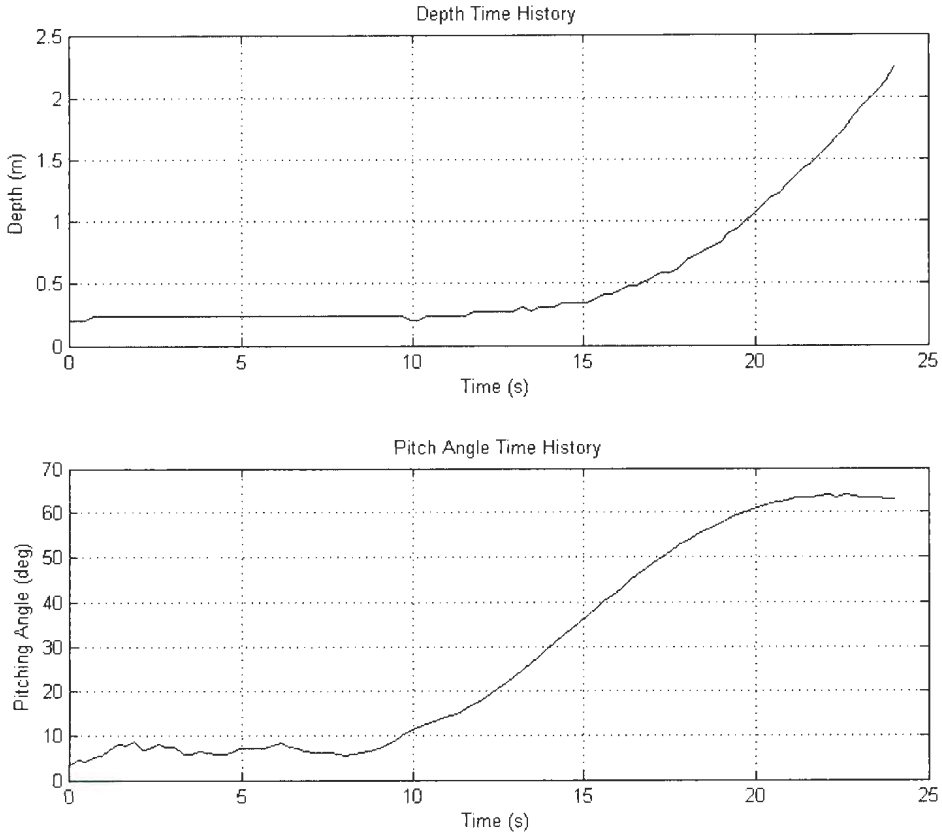


Figure B.11: Buoyancy Engine -1 cm, Battery Pack 16 mm, Diving

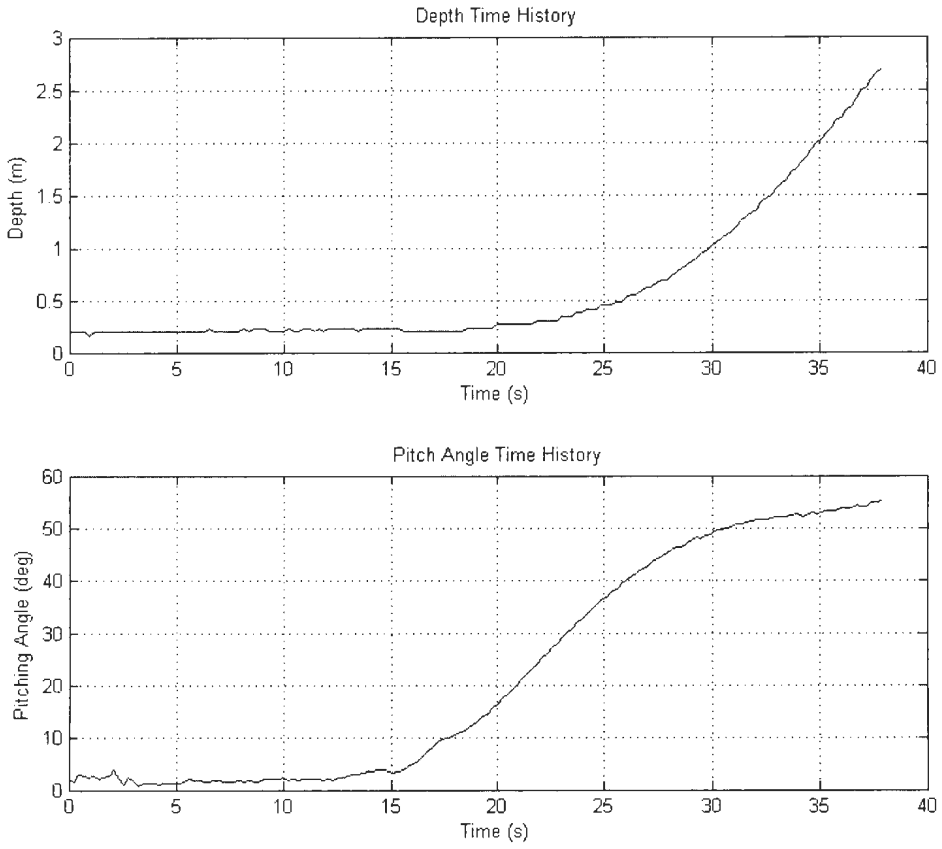


Figure B.12: Buoyancy Engine -1 *cm*, Battery Pack -8 *mm*, Diving

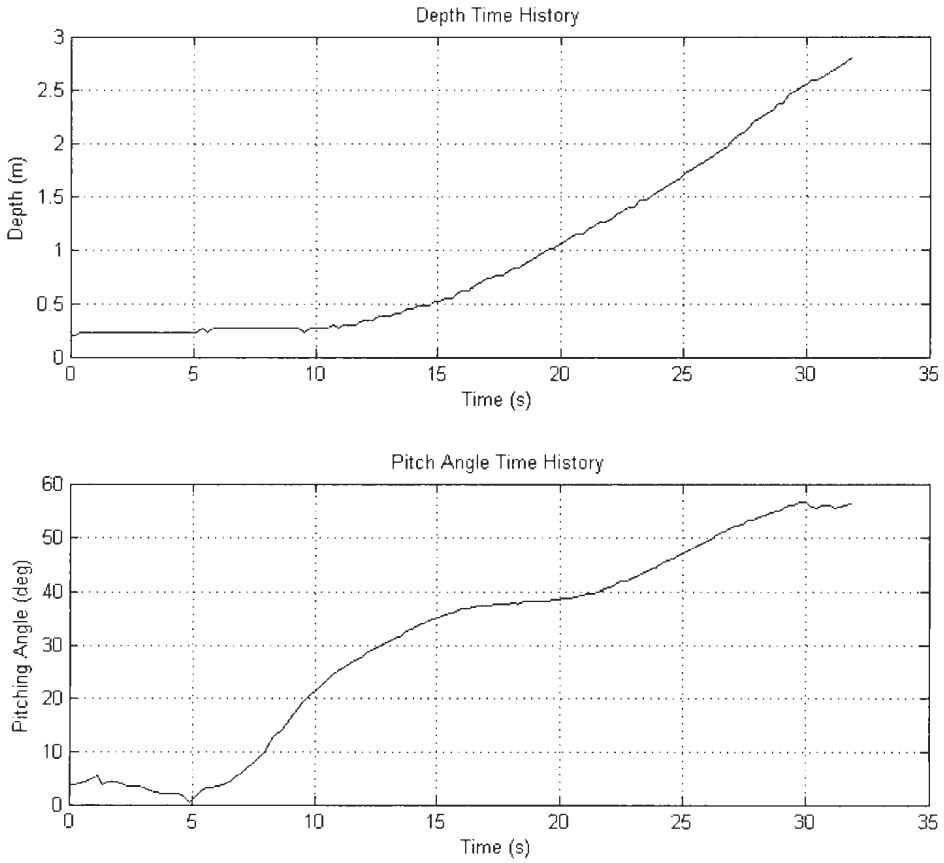


Figure B.13: Buoyancy Engine -0.5 cm, Battery Pack 0 mm, Diving

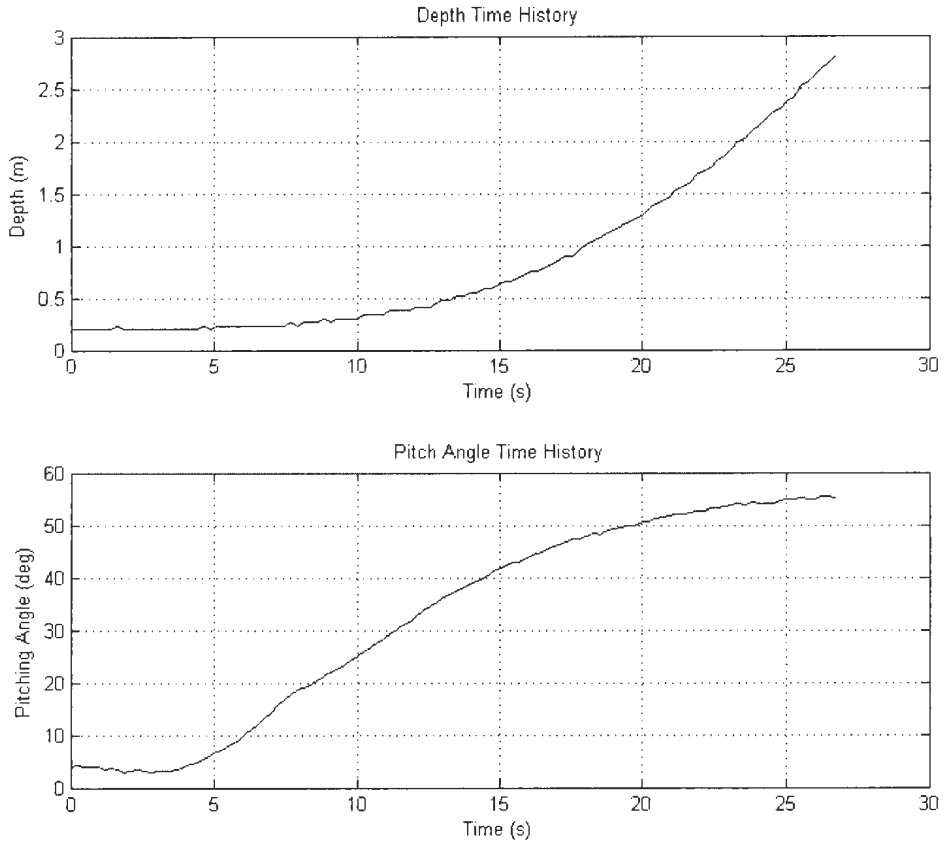


Figure B.14: Buoyancy Engine $-0.5\ cm$, Battery Pack 8 mm, Diving

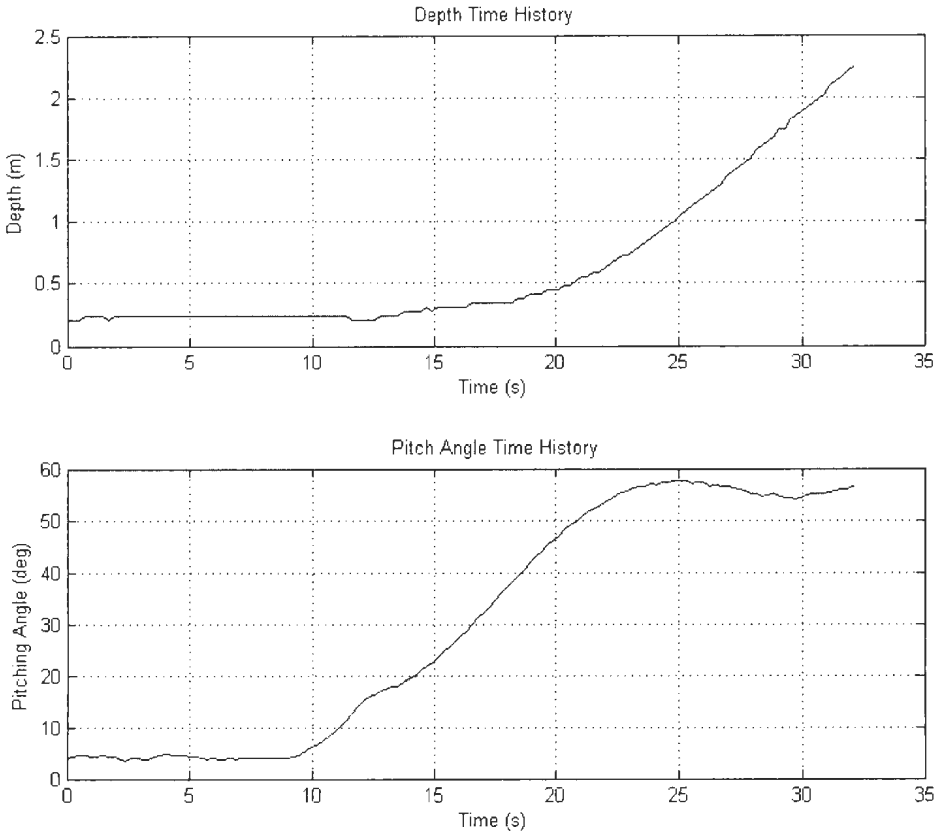


Figure B.15: Buoyancy Engine $-0.5\ cm$, Battery Pack 16 mm, Diving

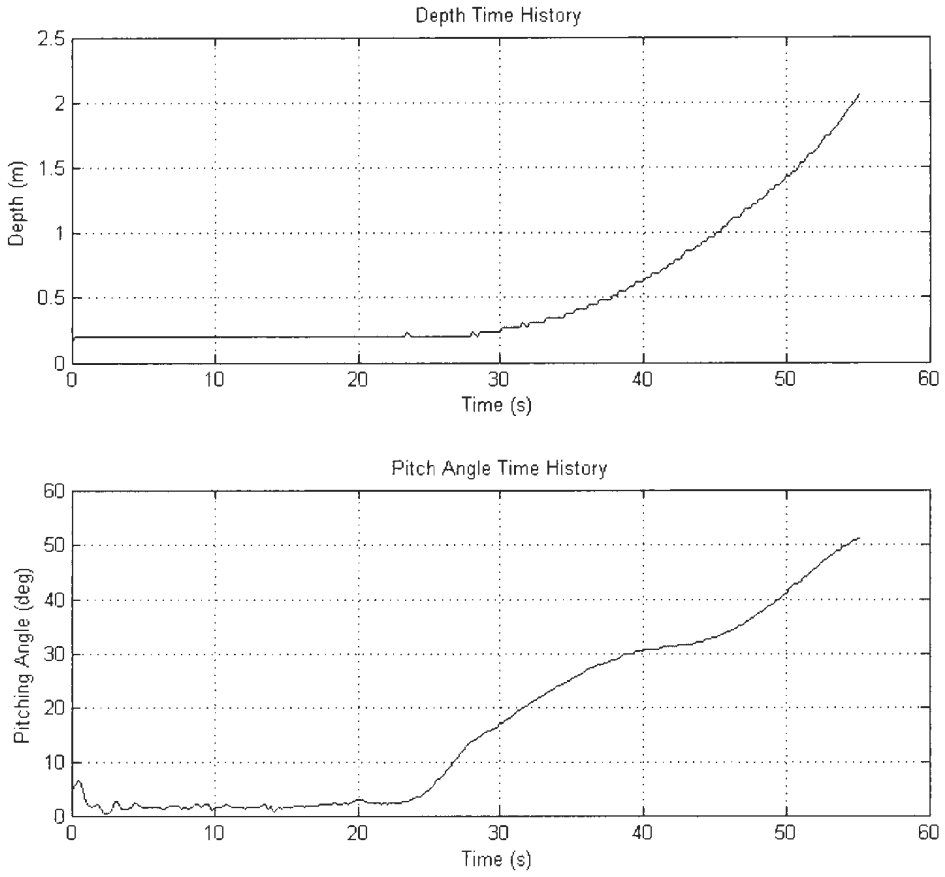


Figure B.16: Buoyancy Engine $-0.5\ cm$, Battery Pack $-8\ mm$, Diving

Appendix C

Specification Sheets

US DIGITAL E8P OEM Optical Kit Encoder

RoHS COMPLIANT

Description

The E8P optical incremental kit encoder is designed for high volume, low cost, mid-resolution OEM motion control applications. The E8P is small enough for a NEMA Size 11 stepper motor.The E8P uses a single 5V supply and offers two TTL quadrature outputs. Single-ended or differential output options are available. A single chip reflective encoder module incorporates an LED, monolithic detector and molded lenses. The phased array technology accepts far wider mechanical tolerance and misalignment than traditional aperture type encoders. The E8P uses an innovative, patent pending, push-on codewheel that provides extremely secure and accurate, yet easy installation without setscrews.

The E8P provides mounting holes for two #4-40, length .250" screws or two M2.5x.45mm, length 6mm screws on a 0.75" diameter bolt circle. When mounting holes are not available, an option with a transfer adhesive pre-applied to the base is available. A centering tool is provided to center the base to the motor shaft during installation. The codewheel pushes on by hand using a spacer tool to set the gap in one step. The cover snaps on to complete the assembly in seconds.

The single-ended output version has a 4-pin high retention polarized connector and is designed to drive cables up to six feet long. For longer cable lengths, the differential output version (6-pin connector) is recommended to maximize noise immunity. The internal 26C31 differential line driver can source and sink 20 mA at TTL levels. The recommended receiver is industry standard 26C32. Maximum noise immunity is achieved when the differential receiver is terminated with a 110 Ω resistor in series with a .0047 µF capacitor placed across each differential pair. The capacitor simply conserves power. Otherwise power consumption would increase by approximately 20 mA per pair, or 40 mA for 2 pairs.

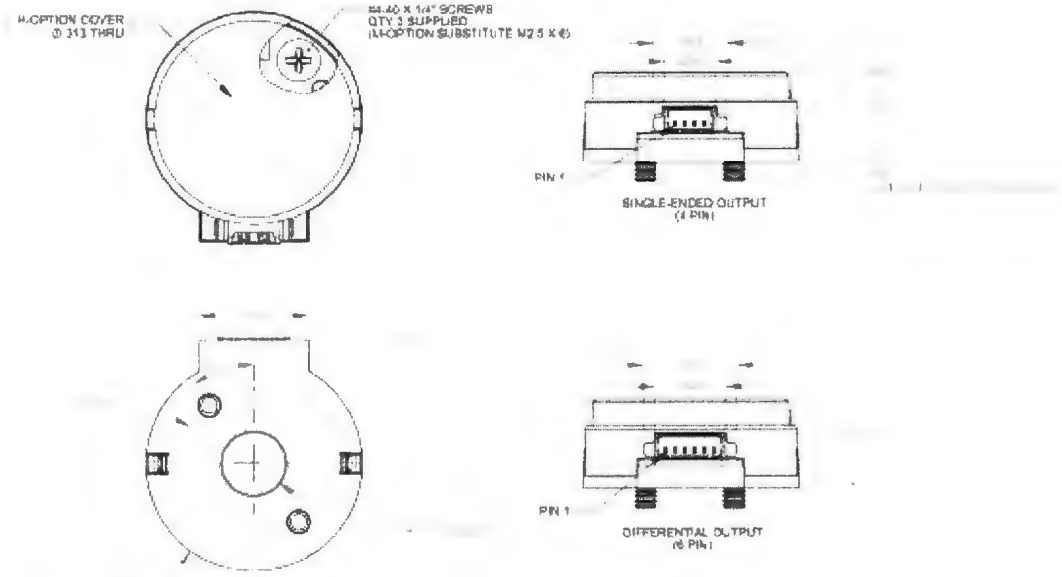
Features

- Subminiature size, easy installation
- Single-ended or differential output option
- A and B quadrature TTL outputs
- Fits shaft diameters from 0.118" (3mm) to 0.276" (7mm)
- Accepts +/- 0.020" axial shaft play
- Off-axis mounting tolerance of 0.010"
- Count frequency from DC to 60 kHz
- 180 to 512 cycles per rev (CPR)
- 720 to 2048 quadrature states per rev.
- Single +5V supply

Mechanical Drawing

US DIGITAL
1400 NE 136th Avenue
Vancouver, Washington 98684, USA
info@usdigital.com
www.usdigital.com
Local: 360.260.2468
Toll-free: 800.736.0194

RoHS COMPLIANT



Environmental

Parameter	Min.	Max.	Units
Vibration (5Hz to 2kHz)	-	20	G
Relative Humidity	-	90	%
Storage Temperature	-40	100	C
Operating Temperature	-20	100	C
ESD (Human Body Model JESD22-A114-A Class 2)		3	kV
ESD (Machine Model JESD22-A115-A Class B)		300	V

Mechanical

Parameter	Value	Units
Moment of Inertia	1.81×10^{-5}	oz-in-s^2
Required Shaft Length		
With D-Cover option	0.385" to 0.400"	in.
With H-Cover option	>=0.385"	in.
Mounting Screw Torque (-D, -M option)	2-3	in.-lbs
Shaft Axial Play	± .020	in.

Parameter	Value	Units
Off-axis Mounting Tolerance	± .010	in.
Shaft to Mounting Surface Perpendicularity	90 ± 1	deg.
Acceleration	250000 max.	rad/sec²
Maximum RPM (1) e.x. CPR = 720, max. rpm = 5000 e.x. CPR = 180, max. rpm = 20000	minimum value of (3600000/CPR) and (60000)	rpm

(1) 60000 rpm is the maximum rpm due to mechanical considerations. The maximum rpm due to the module's 60kHz maximum count frequency is (3600000/CPR).

Base Options

Base Option	Bolt Circle	Screws Included
D	0.75"	#4-40, length .250" pan head phillips (qty: 2)
M	0.75"	M2.5x.45mm, length 6mm pan head phillips (qty: 2)
T	n/a	none - .005" thick transfer adhesive with peel away backing mount.

The included centering tool and spacer tool are used to center the base to the motor shaft and to set the codewheel gap.

Single-ended Electrical

Specifications	Min.	Typ.	Max.	Units	Notes
Supply Current	-	21	27	mA	
Supply Voltage	4.5	-	5.5	V	
High Level Output	2.4	-	-	V	I_{OH} = -1 mA
Low Level Output	-	-	0.4	V	I_{OL} = 6 mA
Rise Time	-	500	-	ns	Cl = 25 pF, Rl = 2.7 kΩ
Fall Time	-	100	-	ns	

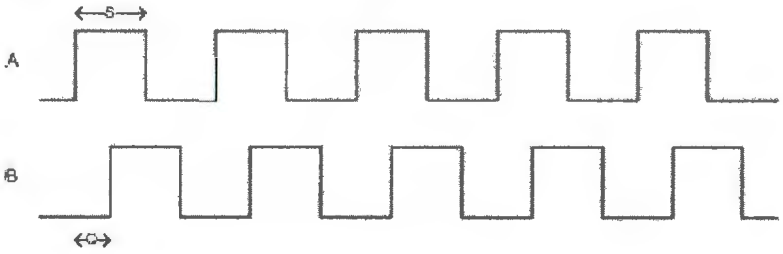
Differential Electrical

Specifications	Min.	Typ.	Max.	Units	Notes
Supply Current	-	22	30	mA	
Supply Voltage	4.5	-	5.5	V	
High Level Output	2.4	3.4	-	V	I_{OH} = -20 mA
Low Level Output	-	0.2	0.4	V	I_{OL} = 20 mA
Rise Time	-	500	-	ns	

US DIGITAL
1400 NE 136th Avenue
Vancouver, Washington 98684, USA
info@usdigital.com
www.usdigital.com
Local: 360.260.2468
Toll-free: 800.736.0194

Specifications	Min.	Typ.	Max.	Units	Notes
Fall Time	-	100	-	ns	

Phase Relationship



Parameter	Typ.	Units
Symmetry, S	180 ± 16	electrical degrees
Quadrature Delay, Q	90 ± 12	electrical degrees

A leads B for clockwise shaft rotation, and B leads A for counterclockwise rotation viewed from the cover/label side of the encoder.

Pin-outs

4-pin Single-ended		6-pin Differential	
Pin	Description	Pin	Description
1	+5VDC power	1	Ground
2	A channel	2	A channel
3	Ground	3	A- channel
4	B channel	4	+5VDC power
		5	B channel
		6	B- channel

Assembly Instructions

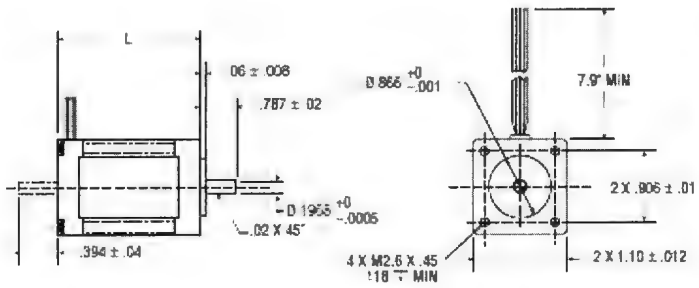
US DIGITAL
1400 NE 136th Avenue
Vancouver, Washington 98684, USA
Info@usdigital.com
www.usdigital.com
Local: 360.260.2468
Toll-free: 800.736.0194

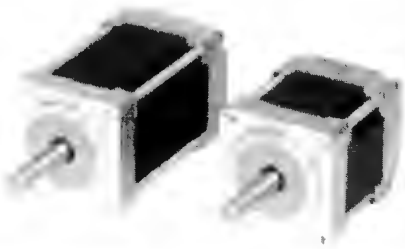
Applied Motion Products

404 Westridge Dr. • Watsonville, CA 95076
831/761-6555 • 800/525-1609 • FAX 831/761-6544
www.appliedmotionproducts.com

Hybrid Step Motors

SIZE HT 11





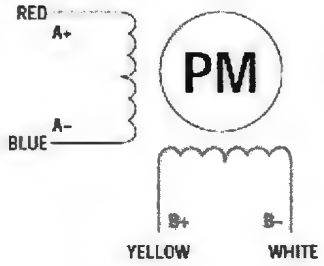
Part #	Motor Connection 1 = series 2 = parallel 3 = unipolar	Motor Length (inches)	Minimum Holding Torque (oz-in)	Leads	Step Angle	Volts	Amps	Ohms	mH	Rotor Inertia (oz-in²/G-CM²)	Motor Weight (Lbs.)
HT11-012	2	1.32	7.0	4	1.8	1.4	1.0	1.4	1.4	.044/8	.26
HT11-013	2	1.87	15	4	1.8	2.0	1.0	2.0	2.6	.098/18	.39

OTHER LENGTHS AND WINDINGS AVAILABLE UPON REQUEST

- Part numbers listed are for single shaft. To order double shaft add 'D' to the end.

Switching sequence for CW rotation facing mounting end.

STEP	RED	BLUE	YELLOW	WHITE
0	+	–	+	–
1	–	+	+	–
2	–	+	–	+
3	+	–	–	+
4	+	–	+	–



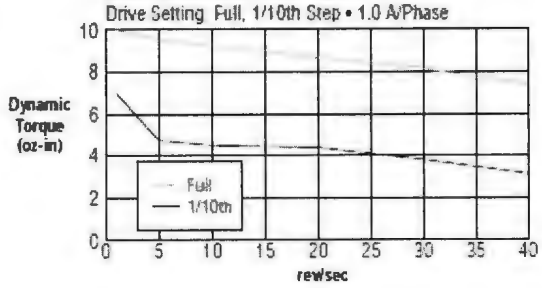
Applied Motion Products
404 Westridge Dr. • Watsonville, CA 95076
831/761-6555 • 800/525-1609 • FAX 831/761-6544
www.appliedmotionproducts.com

Dynamic Torque Data

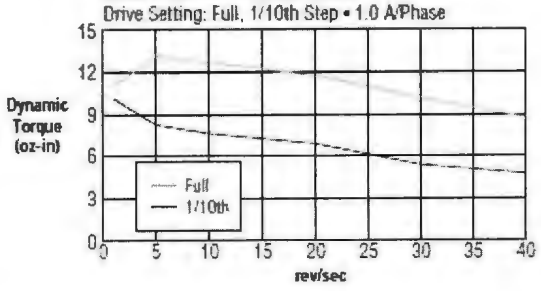
Size HT11 Motor

Typical Speed/Torque Performance*

HT11-012 MOTOR
Motor Connection: 4 Lead Bipolar
Drive Setting: Full, 1/10th Step • 1.0 A/Phase



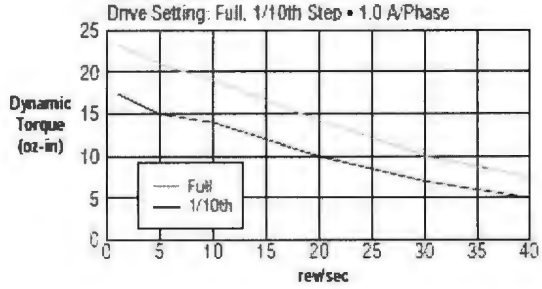
HT11-013 MOTOR
Motor Connection: 4 Lead Bipolar
Drive Setting: Full, 1/10th Step • 1.0 A/Phase



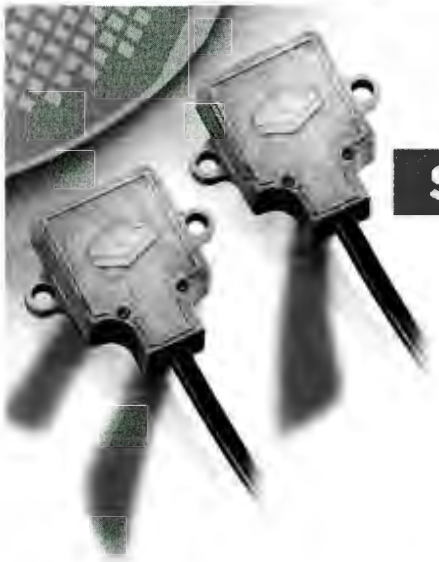
Size 14 Motor

Typical Speed/Torque Performance*

5014-842 MOTOR
Motor Connection: Parallel
Drive Setting: Full, 1/10th Step • 1.0 A/Phase



*Full steps/sec = Rev/sec x 200. Half steps/sec = Rev/sec x 400.



SCA121T Series

Stand Alone Inclinometer

Dual Axis Analog Output

FEATURES
- Silicon 3D MEMS sensor
- 0,1 ° accuracy
- Resolution < 0,001 °
- Operating temperature range -40...+85 °C
- Long term stability < 0,02 °
- Shock resistance >20 000 g
- Sensing element -3 dB @ 18 Hz
- Main dimensions: 30x30x13 mm size, single or dual axis
- Voltage output
- RoHS compatible

BENEFITS
- Excellent long term stability
- Sensing element controlled frequency response
- Outstanding shock durability
- Harsh environment robustness

APPLICATIONS
- Platform tilt measurement
- Equipment and instrument condition monitoring
- Inclination based position measurement
- Rotational orientation measurement

For customised product please contact VTI Technologies

ELECTRICAL CHARACTERISTICS

Parameter	Condition	Min.	Typ	Max.	Units	
Supply voltage	Unregulated or	7	16	35	V	Applies to: -D03, -D07
	regulated ratiometric	4.75	5	5.25	V	Applies to: -D05
Current consumption			5		mA	
Output load	Resistive	10			kΩ	
	Capacitive			20	nF	

PERFORMANCE CHARACTERISTICS

Parameter	Condition	SCA121T-D03	SCA121T-D07	SCA121T-D05	Units
Measuring range [1]		±90	±30	±90	°
Supply voltage		7 ...35	7 ...35	5 ±0.25	V
Measuring axis	(see "Directions")	X-Y	X-Y	X-Y	
Offset [2, 5]	Output at 0 °	2.5	2.5	2.5	V
Offset zero point error [5]	Max. deviation	1	1	1	°
Offset temperature error	0...70 °C	±0.2	±0.2	±0.2	°
	-25...85 °C	±0.6	±0.6	±0.6	°
Sensitivity		2	4	2	V/g
		35	70	35	mV/°(@offset pos.)
Sensitivity temperature error [5]	0...70 °C	-0.8 ...0.3	-0.8 ...0.3	-0.8 ...0.3	%
	-25...85 °C	-1.5 ...0.5	-1.5 ...0.5	-1.5 ...0.5	%
Nonlinearity	Sinus output	N/A	0.1	N/A	°
Frequency response -3 dB [3]		18	18	18	Hz
Cross-axis sensitivity [4]		3	3	3	%

Typical values unless otherwise specified.

Note 1 The measuring range is limited by the sensitivity and offset.
Note 2 Offset specified as Output @ 0 °.
Note 3 The frequency response is determined by the sensing element's internal gas damping. The output has true DC (0 Hz) response.
Note 4 The cross-axis sensitivity determines how much inclination, perpendicular to the measuring axis, couples to the output.
Note 5 For optimum zero point accuracy, mounting angle of the part can be adjusted.

MEASURING DIRECTIONS

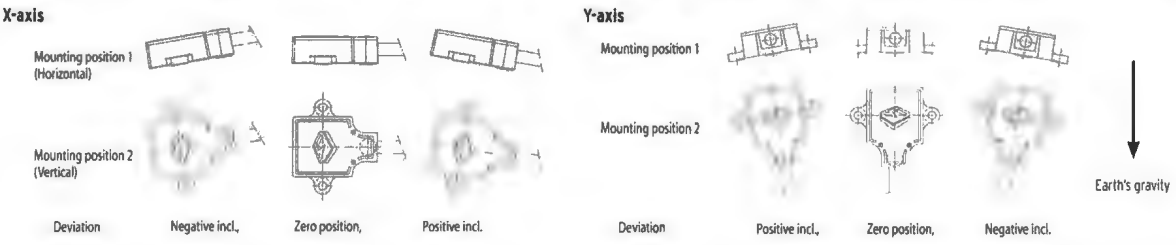


Figure 1. Positions

Notes:
- It is important that the part is parallel to the mounting plane, and that the output equals the zero value when sensor is in zero position.
- Zero position: Please note the picture above which provides information on how the output of the accelerometer behaves in different circumstances when assembled. Please also note that you can rotate the part around the measuring plane for optimum mounting location.

VTI TECHNOLOGIES

ELECTRICAL CONNECTION

SCA121T series

Wire color	Name	Function
Blue	GND	Ground
Red	V_{cc}	Power supply
Yellow	Out X	X-axis output
Green	Out Y	Y-axis output
White		Not connected

MECHANICAL SPECIFICATION

Cable length:	-D03, -D07 30 cm
	-D05 110 cm
Total weight:	Approx. 60 grams
Protection class:	IP66
Housing:	Zinc casting with passivation

MOUNTING

The sensor module is to be mounted on a flat and smooth surface with 2 screws, dimension M4. Mounting torque 5 ±1 Nm.

SENSOR DIMENSIONS

Dimensions in mm.

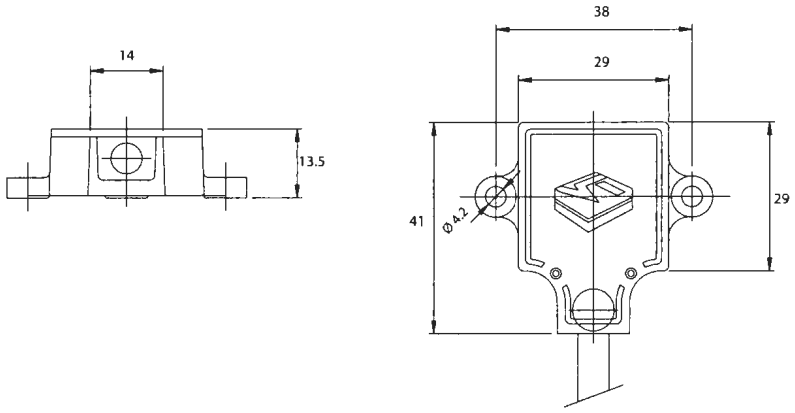


Figure 2.

VOLTAGE TO ANGLE CONVERSION

$$\text{Inclination angle} = \arcsin\left(\frac{\text{Vout} - \text{Offset}}{\text{Sensitivity}}\right)$$

where:
Vout = analog output [V]
Offset = 2.5 V, output at 0° inclination position
Sensitivity = sensitivity of device [V/g]

VTI Technologies Oy
Myllynkivenkuja 6
P.O. Box 27
FI-01621 Vantaa
Finland
Tel. +358 9 879 181
Fax +358 9 879 18791
sales@vti.fi

VTI Technologies Oy
Frankfurt Branch
Rennbahnstr. 72-74
D-60528 Frankfurt am Main
Germany
Tel. +49 69 6786 880
Fax +49 69 6786 8829
sales.de@vti.fi

VTI Technologies, Inc.
One Park Lane Blvd.
Suite 804 - East Tower
Dearborn, MI 48126
USA
Tel. +1 313 425 0850
Fax +1 313 425 0860
sales@vtitechnologies.com

VTI TECHNOLOGIES

Appendix D

Dynamic C Codes

```
/* START LIBRARY DESCRIPTION *************************
TCMCcommunication.LIB
Copyright (c)
DESCRIPTION:    Communication with TCMC 303 & 323
                using Serial Port C.
                Pitch motor actuator is Motor 0 and Encoder 0;
                Roll motor is Motor 1 and Encoder 1;
REVSION HISTORY:          DD/MM/YYYY
        27/02/2012        Initial Release by Peng Wen.
END DESCRIPTION **************************************/
/** GLOBAL DECLARATIONS *******************************/
/*** BeginHeader */
#define CINBUFSIZE   255 // buffer size for serial port C inputs
#define COUTBUFSIZE  255 // buffer size for serial port C outputs
/*Motor Numbers */
#define PitchMotor 0
#define RollMotor 1
/*Axis Parameter Numbers */
#define ActualPosition 1
#define MaxSpeed 4
#define MaxAcce 5
#define MaxCurrent 6
#define StandCurrent 7
#define Microstep 140
#define EncoderPosition 209
```

```
#define PitchMotorGAIN   3.840E-5     // mm/EncoderStep
#define RollMotorGAIN   0.0075     // degree/EncoderStep
#define PitchMotorSpeed   200     // ROR, ROL max speed
#define RollMotorSpeed   40        // ROR, ROL max speed
/*** EndHeader */
/** FUNCTION PROTOTYPES & DESCRIPTIONS ********************/
/*** BeginHeader setupSerialPortC */
void setupSerialPortC(void);
/*** EndHeader */
/* START FUNCTION DESCRIPTION ****************************
setupSerialPortC                        <TCMCcommunication.LIB>
SYNTAX: void setupSerialPortC(void);
DESCRIPTION:     Sets up serial port C. Opens serial port C
                 at 9600 baud with no flow control. Allows
                 communication with TCMC303.
                Defines both CINBUFSIZE and COUTBUFSIZE
                to be 255 characters.
PARAMETER:   None.
RETURN VALUE:   None.
KEY WORDS:       open, PortC, baud
END DESCRIPTION ***************************************/
void setupSerialPortC(void) {
   serCopen(9600);  // open serial port C, baud rate 9600 bps
        serCwrFlush();  // flush serial port C transmit buffer
        serCrdFlush();  // flush serial port C input buffer
   serMode(0);       // port C set up as RS-232, 3 wire
                     //communication port
}
/*** BeginHeader msDelay */
void msDelay(unsigned int delay);
/*** EndHeader */
/* START FUNCTION DESCRIPTION ************************
msDelay                                 <TCMCcommunication.LIB>
SYNTAX: void msDelay(unsigned int delay);
DESCRIPTION: delay for # of ms
PARAMETER:   delay - # of ms
RETURN VALUE:   none.
KEY WORDS: delay, ms
END DESCRIPTION ***************************************/
```

```
nodebug
void msDelay(unsigned int delay)
{
        auto unsigned long done_time;
        done_time = MS_TIMER + delay;
   while( (long) (MS_TIMER - done_time) < 0 );
}
/*** BeginHeader serCgets */
char* serCgets(char* buffer);
/*** EndHeader */
/* START FUNCTION DESCRIPTION ************************
serCgets                          <TCMCcommunication.LIB>
SYNTAX: char* serCgets(char* buffer);
DESCRIPTION:     Reads in a string of characters from
                 serial port C into *buffer
PARAMETER:  *buffer - Pointer to a character buffer
RETURN VALUE:  Last string transmitted to Serial Port C.
KEY WORDS: string, PortC
END DESCRIPTION ************************************/
char* serCgets(char* buffer) {
        int i, temp;
   // Get the data string that was transmitted to serial port C
   i = 0;
   while((temp = serCgetc()) != -1) {
        // Copy only valid RCV'd characters to the buffer
      buffer[i++] = temp;    }
   return (buffer);
}
/*** BeginHeader serCdone */
int serCdone(void);
/*** EndHeader */
/* START FUNCTION DESCRIPTION ***********************
serCdone                          <TCMCcommunication.LIB>
SYNTAX: int serCdone(void);
DESCRIPTION:     Checks to see that serial port C is
                 finished transmitting all the data
                 sent to it by checking if the write
                 buffer is empty and the serial holding
                 and shift registers are empty.
```

```
PARAMETER:   None.
RETURN VALUE:  1 if the port has finished transmitting.
               0 if the port hasn't finished transmitting.
KEY WORDS:       transmitted, PortC
END DESCRIPTION **********************************/
int serCdone(void) {
        unsigned int bufused, wrstatus;
// Wait for memory data buffer, serial holding register
// , and shift register all to become empty
   bufused = serCwrUsed();
   wrstatus = RdPortI(SCSR)&0x0C;
   if ((!bufused) && (!wrstatus)) {
      return (1);
   } else {
        return (0);
   }
}
/*** BeginHeader InitTMCM */
void InitTMCM(void);
/*** EndHeader */
/* START FUNCTION DESCRIPTION ********************
InitTMCM                          <TMCMcommunication.LIB>
void InitTMCM(void);
DESCRIPTION:     Initialize the TCMC ICs, set
                 parameters for glider use.
PARAMETER:
RETURN VALUE:
KEY WORDS:       Initialization, TCMC
END DESCRIPTION **********************************/
void InitTMCM(void)
{
SAP(ActualPosition,0,0);
SAP(ActualPosition,1,0);
SAP(MaxSpeed,0,PitchMotorSpeed);
SAP(MaxSpeed,1,RollMotorSpeed);
SAP(MaxAcce,0,2500);
SAP(MaxAcce,1,250);
SAP(MaxCurrent,0,500);
SAP(MaxCurrent,1,800);
```

```
SAP(StandCurrent,0,200);
SAP(StandCurrent,1,800);
SAP(Microstep,0,6);
SAP(Microstep,1,6);
SAP(EncoderPosition,0,0);
SAP(EncoderPosition,1,0);
}
/*** BeginHeader ROR */
char ROR(char Motor, long Speed);
/*** EndHeader */
/* START FUNCTION DESCRIPTION ********************
ROR                                <TMCMcommunication.LIB>
SYNTAX: char ROR(char Motor, int Speed);
DESCRIPTION:      Rotate Right.
PARAMETER:   Motor - motor number, 0,1,2.
             Speed - Velocity, 0..2047
RETURN VALUE:   Status number if checksum correct.
                (100 - ok)
                0 if checksum wrong
KEY WORDS:        rotate, right
END DESCRIPTION **********************************/
char ROR(char Motor, long Speed)
{
        char TxBuffer[9];
   char RxBuffer[9];
        char i, Checksum;
        TxBuffer[0]=0x01;
        TxBuffer[1]=0x01;
        TxBuffer[2]=0x00;
        TxBuffer[3]=Motor;
        TxBuffer[4]=Speed >> 24;
        TxBuffer[5]=Speed >> 16;
        TxBuffer[6]=Speed >> 8;
        TxBuffer[7]=Speed & 0xff;
        TxBuffer[8]=0;
        for(i=0; i<8; i++)
                TxBuffer[8]+=TxBuffer[i];
//Now, send the 9 bytes stored in TxBuffer to the module
   for (i=0;i<9; i++)
```

```
        serCputc(TxBuffer[i]);
    msDelay(50);
    serCgets(RxBuffer);
    Checksum=0;
          for(i=0; i<8; i++)
                  Checksum+=RxBuffer[i];
          if(Checksum!=RxBuffer[8])
        return 0;
    else
        return(RxBuffer[2]);
}
/*** BeginHeader ROL */
char ROL(char Motor, long Speed);
/*** EndHeader */
/* START FUNCTION DESCRIPTION ********************
ROL                              <TMCMcommunication.LIB>

SYNTAX: char ROL(char Motor, int Speed);

DESCRIPTION:     Rotate Left.
PARAMETER:  Motor - motor number, 0,1,2.
            Speed - Velocity, 0..2047

RETURN VALUE:  Status number if checksum correct.
               (100 - ok)
               0 if checksum wrong

KEY WORDS:       rotate, right
END DESCRIPTION **********************************/
char ROL(char Motor, long Speed)
{
        char TxBuffer[9];
    char RxBuffer[9];
        char i, Checksum;
        TxBuffer[0]=0x01;
        TxBuffer[1]=0x02;
        TxBuffer[2]=0x00;
        TxBuffer[3]=Motor;
        TxBuffer[4]=Speed >> 24;
```

```
        TxBuffer[5]=Speed >> 16;
        TxBuffer[6]=Speed >> 8;
        TxBuffer[7]=Speed & 0xff;
        TxBuffer[8]=0;
        for(i=0; i<8; i++)
                TxBuffer[8]+=TxBuffer[i];
//Now, send the 9 bytes stored in TxBuffer to the module
   for (i=0;i<9; i++)
      serCputc(TxBuffer[i]);
   msDelay(50);
   serCgets(RxBuffer);
   Checksum=0;
        for(i=0; i<8; i++)
                Checksum+=RxBuffer[i];
        if(Checksum!=RxBuffer[8])
      return 0;
   else
      return(RxBuffer[2]);
}
/*** BeginHeader MST */
char MST(char Motor);
/*** EndHeader */
/* START FUNCTION DESCRIPTION ********************
MST                              <TMCMcommunication.LIB>
SYNTAX: char MST(char Motor);
DESCRIPTION:     Motor stop.
PARAMETER:  Motor - motor number, 0,1,2.
RETURN VALUE:  Status number if checksum correct.
               (100 - ok)
               0 if checksum wrong
KEY WORDS:       motot, stop
END DESCRIPTION ******************************/
char MST(char Motor)
{
        char TxBuffer[9];
   char RxBuffer[9];
        char i, Checksum;
        TxBuffer[0]=0x01;
        TxBuffer[1]=0x03;
```

```
        TxBuffer[2]=0x00;
        TxBuffer[3]=Motor;
        TxBuffer[4]=0x00;
        TxBuffer[5]=0x00;
        TxBuffer[6]=0x00;
        TxBuffer[7]=0x00;
        TxBuffer[8]=0;
        for(i=0; i<8; i++)
                TxBuffer[8]+=TxBuffer[i];
//Now, send the 9 bytes stored in TxBuffer to the module
   for (i=0;i<9; i++)
      serCputc(TxBuffer[i]);
   msDelay(50);
   serCgets(RxBuffer);
   Checksum=0;
        for(i=0; i<8; i++)
                Checksum+=RxBuffer[i];
        if(Checksum!=RxBuffer[8])
      return 0;
   else
      return(RxBuffer[2]);
}
/*** BeginHeader MVP */
char MVP(char Type, char Motor, long Value);
/*** EndHeader */
/* START FUNCTION DESCRIPTION ********************
MVP                                <TMCMcommunication.LIB>
SYNTAX: char MVP(char Type, char Motor, int Value);
DESCRIPTION:     Move to position.
PARAMETER:  Type - operation types, 0 - absolute,
                    1 - relative
            Motor - motor number, 0,1,2.
            Value - Position or offset.
RETURN VALUE:  Status number if checksum correct.
               (100 - ok)
               0 if checksum wrong
KEY WORDS:       move, motor, position
END DESCRIPTION ******************************/
char MVP(char Type, char Motor, long Value)
```

```
{
        char TxBuffer[9];
    char RxBuffer[9];
        char i, Checksum;
        TxBuffer[0]=0x01;
        TxBuffer[1]=0x04;
        TxBuffer[2]=Type;
        TxBuffer[3]=Motor;
        TxBuffer[4]=Value >> 24;
        TxBuffer[5]=Value >> 16;
        TxBuffer[6]=Value >> 8;
        TxBuffer[7]=Value & 0xff;
        TxBuffer[8]=0;
        for(i=0; i<8; i++)
                TxBuffer[8]+=TxBuffer[i];
//Now, send the 9 bytes stored in TxBuffer to the module
    for (i=0;i<9; i++)
       serCputc(TxBuffer[i]);
    msDelay(50);
    serCgets(RxBuffer);
    Checksum=0;
        for(i=0; i<8; i++)
                Checksum+=RxBuffer[i];
        if(Checksum!=RxBuffer[8])
       return 0;
    else
       return(RxBuffer[2]);
}
/*** BeginHeader SAP */
long SAP(char Paramter, char Motor, long Value);
/*** EndHeader */
/* START FUNCTION DESCRIPTION *****************
SAP                             <TMCMcommunication.LIB>
SYNTAX: char SAP(char Parameter, char Motor, int Value);
DESCRIPTION:     Set axis parameter.
PARAMETER:   Parameter - Parameter number
             Motor - motor number, 0,1,2.
             Value - parameter value.
RETURN VALUE:   parameter value
```

```
                0 if checksum wrong
KEY WORDS:        set, parameter
END DESCRIPTION **********************************/
long SAP(char Parameter, char Motor, long Value)
{
        char TxBuffer[9];
   char RxBuffer[9];
        char i, Checksum;
   long r4, r5, r6, r7;
        TxBuffer[0]=0x01;
        TxBuffer[1]=0x05;
        TxBuffer[2]=Parameter;
        TxBuffer[3]=Motor;
        TxBuffer[4]=Value >> 24;
        TxBuffer[5]=Value >> 16;
        TxBuffer[6]=Value >> 8;
        TxBuffer[7]=Value & 0xff;
        TxBuffer[8]=0;
        for(i=0; i<8; i++)
                TxBuffer[8]+=TxBuffer[i];
//Now, send the 9 bytes stored in TxBuffer to the module
   for (i=0;i<9; i++)
      serCputc(TxBuffer[i]);
   msDelay(50);
   serCgets(RxBuffer);
   Checksum=0;
        for(i=0; i<8; i++)
                Checksum+=RxBuffer[i];
   r4=(long)RxBuffer[4];
   r5=(long)RxBuffer[5];
   r6=(long)RxBuffer[6];
   r7=(long)RxBuffer[7];
        if(Checksum!=RxBuffer[8])
      return 0;
   else
      return((r4 << 24) | (r5 << 16) | (r6 << 8) | r7);
}
/*** BeginHeader GAP */
long GAP(char Paramter, char Motor);
```

```
/*** EndHeader */
/* START FUNCTION DESCRIPTION ***************
GAP                          <TMCMcommunication.LIB>
SYNTAX: int GAP(char Parameter, char Motor);
DESCRIPTION:     Get axis parameter.
PARAMETER:  Parameter - Parameter number
            Motor - motor number, 0,1,2.
            Value - parameter value.
RETURN VALUE:  parameter value
               0 if checksum wrong
KEY WORDS:       get, parameter
END DESCRIPTION ***************************/
long GAP(char Parameter, char Motor)
{
        char TxBuffer[9];
   char RxBuffer[9];
        char i, Checksum;
   long Value, r4, r5, r6, r7;
        TxBuffer[0]=0x01;
        TxBuffer[1]=0x06;
        TxBuffer[2]=Parameter;
        TxBuffer[3]=Motor;
        TxBuffer[4]=0x00;
        TxBuffer[5]=0x00;
        TxBuffer[6]=0x00;
        TxBuffer[7]=0x00;
        TxBuffer[8]=0;
        for(i=0; i<8; i++)
                TxBuffer[8]+=TxBuffer[i];
//Now, send the 9 bytes stored in TxBuffer to the module
   for (i=0;i<9; i++)
      serCputc(TxBuffer[i]);
   msDelay(50);
   serCgets(RxBuffer);
   Checksum=0;
        for(i=0; i<8; i++)
                Checksum+=RxBuffer[i];
   r4=(long)RxBuffer[4];
   r5=(long)RxBuffer[5];
```

```
    r6=(long)RxBuffer[6];
    r7=(long)RxBuffer[7];
          if(Checksum!=RxBuffer[8])
       return 0;
    else
       Value=(r4 << 24) | (r5 << 16) | (r6 << 8) | r7;
       return(Value);
}
/*** BeginHeader readRMAngle */
float readRMAngle(void);
/*** EndHeader */
/* START FUNCTION DESCRIPTION **************
readRMAngle                    <TMCMcommunication.LIB>
SYNTAX: float readRMAngle(void);
DESCRIPTION:     Read roll motor rotation angle,
                 determined by encoder position.
PARAMETER:  N/A
RETURN VALUE:  Current rotation angle of the roll motor, in degree
KEY WORDS:       Angle position ,roll motor
END DESCRIPTION ******************************/
float readRMAngle(void)
{
   long currentReading;
   float gain;
   currentReading = GAP(EncoderPosition,RollMotor);
   gain=RollMotorGAIN;
   return (currentReading*gain);
}
/*** BeginHeader readPMPos */
float readPMPos(void);
/*** EndHeader */
/* START FUNCTION DESCRIPTION **********************
readPMPos                      <TMCMcommunication.LIB>
SYNTAX: float readPMPos(void);
DESCRIPTION:     Read pitch motor position,
                 determined by encoder position.
PARAMETER:  N/A
RETURN VALUE:  Current position of the pitch motor , in mm
KEY WORDS:       position ,pitch motor
```

```
END DESCRIPTION *************************/
float readPMPos(void)
{
   long currentReading;
   float gain;
   currentReading = GAP(EncoderPosition,PitchMotor);
   gain=PitchMotorGAIN;
   return (currentReading*gain);
}
/*** BeginHeader MTP */
float MTP(char Motor, float Pos);
/*** EndHeader */
/* START FUNCTION DESCRIPTION ******************
MTP                              <TMCMcommunication.LIB>
SYNTAX: char MTP(char Motor, int Value);
DESCRIPTION:    Move to position, feedback by encoder.
PARAMETER:  Motor - motor number, 0,1,2.
            Pos - Target Position, in mm(motor 0)
                  or degree(motor 1).
RETURN VALUE:  Current position
KEY WORDS:      move, motor, position
END DESCRIPTION *************************/
float MTP(char Motor, float Pos)
{
   long currentReading;
   int i, spd;
   float targetReading;
   float gain;
   currentReading = GAP(EncoderPosition,Motor);
   msDelay(50);
   if (Motor==0)
   {
   gain=PitchMotorGAIN;
   spd = PitchMotorSpeed;
   }
   else if  (Motor==1)
   {
   gain=RollMotorGAIN;
   spd = RollMotorSpeed;
```

```
    }
    targetReading = Pos/gain;
    if (currentReading < targetReading) {
         ROR(Motor, spd);
        //msDelay(50);
        while (currentReading < targetReading) {
            currentReading = GAP(EncoderPosition,Motor);
            //msDelay(50);
            output();
            serBputs("\r\n");
        }
        MST(Motor);
    } else if (currentReading > targetReading) {
        ROL(Motor, spd);
        //msDelay(50);
        while (currentReading > targetReading) {
            currentReading = GAP(EncoderPosition,Motor);
            //msDelay(50);
            output();
            serBputs("\r\n");
        }
        MST(Motor);
    }
    currentReading=GAP(EncoderPosition,Motor);
          return (currentReading*gain);
}
```

```
/* START LIBRARY DESCRIPTION ****************************
INCLINOMETER.lib
Copyright (c)
DESCRIPTION:    Driver library for inclinometer Sensor.
SUPPORT LIBS:   RAM_ADC.LIB
REVSION HISTORY:          DD/MM/YYYY
        22/02/2012        Initial Release by Peng Wen.
END DESCRIPTION ****************************************/
/** GLOBAL DECLARATIONS ******************************/
/*** BeginHeader OUTX, OUTY */
   #define OUTX 3         // Output X (Pitch angle) connected
                              // to analog input channel 3
   #define OUTY 4        // Output Y (Roll angle) connected
                         // to analog input channel 4
   #define VTI_GAIN 0.5
/*** EndHeader */
/** FUNCTION PROTOTYPES & DESCRIPTIONS *****************/
/*** BeginHeader readPitch */
        float readPitch(void);
/*** EndHeader */
/* START FUNCTION DESCRIPTION ********************
readPitch                                  <INCLINOMETER.lib>
SYNTAX: float readPitch(void);
DESCRIPTION:    Determines the pitch angle of the IOT Glider
                by reading the voltage output of the
                inclinometer X output, the voltage is
               converted to degree.
PARAMETER:  None.
RETURN VALUE:  The pitch angle in degree.
KEY WORDS: pitch angle, inclinometer VTI SCA121T-05
END DESCRIPTION ****************************************/
float readPitch(void) {
   float v;
   v=(readADCVolts(OUTX) - 2.5) * VTI_GAIN;
   if (v > 1)
     {
     v = v - 2;
     }else if (v < -1){
     v = v + 2;
```

```
    }
  return (asin(v)/3.14159*180);
}
/*** BeginHeader readRoll */
       float readRoll(void);
/*** EndHeader */
/* START FUNCTION DESCRIPTION ************************
readRoll                                  <INCLINOMETER.lib>
SYNTAX: float readRoll(void);
DESCRIPTION:    Determines the Roll angle of the IOT Glider
                by reading the voltage output of the
                inclinometer Y output, the voltage is
                converted to degree.
PARAMETER:  None.
RETURN VALUE:  The roll angle in degree.
KEY WORDS: roll angle, inclinometer VTI SCA121T-05
END DESCRIPTION ***************************************/
float readRoll(void) {
  float v;
  v=(readADCVolts(OUTY) - 2.5) * VTI_GAIN;
  if (v > 1)
    {
    v = v - 2;
    }else if (v < -1){
    v = v + 2;
    }
  return (asin(v)/3.14159*180);
}
```

Appendix E

IOT Glider Hydrodynamic Damping Coefficient Calculation MATLAB Codes

```
% Damping Coefficients Calculation Codes%
%by Peng Wen%
cdf = 1.05;
cdc = 1.1;
cdl = 1.9802;
cdw = 1.12;
%distance between aft end and body frame origin
x_bf = 1.556-0.724758;
rho=1000;
D = .1148;  %bare hull diameter
r = D/2;
L = 1.556;   %glider overall length
x_fin = -0.305358; %fin center X position
%fin center height to the hull center line
h_fin = 0.03 + D/2;
S_fin = 3.6e-3; %fin area
l_wing = 0.26315;%Wing root to tip length     [m]
x_wing_center = 0.111; %wing center X position
S_wing = 0.020412;%Per Wing area   [m^2]
```

```
l_root = 0.093617;%wing root length
l_tip = 0.062410;%wing root length
h_base = 0.02;%wing base hight
B_base = 0.210; %wing base width
tYrr = @(x)((1/2)-(1/18)*(3-(x+x_bf)/D).^2)*D.*x.*abs(x)*2;
nYrr = @(x)(0.8685*sqrt((L-x_bf-x)/D)-0.3978*((L-x_bf-x)/D)...
+0.006511*((L-x_bf-x)/D).^2...
+ 0.005086*((L-x_bf-x)/D).^3)*D.* x .* abs(x) * 2;
bYrr = @(x) r * x .* abs(x) * 2;
tMqq = @(x)((1/2)-(1/18)*(3-(x+x_bf)/D).^2)*D.*abs(x).*x.*x*2;
nMqq = @(x)(0.8685*sqrt((L-x_bf-x)/D)-0.3978*((L-x_bf-x)/D)...
+ 0.006511*((L-x_bf-x)/D).^2...
+ 0.005086*((L-x_bf-x)/D).^3)*D.* abs(x) .* x .* x * 2;
bMqq = @(x) r * abs(x) .* x .* x * 2;
Yrr = -0.5*rho*cdc*(quad(tYrr,-x_bf,-x_bf+3*D)... %tail
    +quad(nYrr,L-1.75*D-x_bf,L-x_bf)...%nose
    +quad(bYrr,-x_bf+3*D,L-1.75*D-x_bf))...%mid body
    -2*x_fin * abs(x_fin) * 0.5 * rho * S_fin * cdf;% 2 fins
Zqq = -Yrr...
+2*x_wing_center*abs(x_wing_center)*0.5*rho*S_wing*cdw;%2 wings
Nrr = -0.5*rho*cdc*(quad(tMqq,-x_bf,-x_bf+3*D)...%tail
    +quad(nMqq,L-1.75*D-x_bf,L-x_bf)...%nose
    +quad(bMqq,-x_bf+3*D,L-1.75*D-x_bf))...%mid body
    -2*abs(x_fin)^3 * 0.5 * rho * S_fin * cdf;%2 fins
Mqq = Nrr...
    -2*abs(x_wing_center)^3*0.5*rho*S_wing*cdw;%2 wings
wKpp = @(x)(l_root-(l_root-l_tip)/l_wing*(x-D/2)).*x.^3;
Kpp = -4* (abs(h_fin))^3 * 0.5 * rho * S_fin * cdf... %4 fin
    - 2*0.5*rho*cdl*quad(wKpp,D/2+h_fin,D/2+l_wing)... %2 wing
    - 2*(D/2+h_base/2)^3*0.5*rho*cdl*B_base*h_base;%wing base
```

Appendix F

Motion Simulation MATLAB Scripts for the IOT Glider

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IOT Glider 6 DOF Motion Simulator - main.m%
%main function%
%by Peng Wen%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all;
clc;
data;
init;
for ii=1:tTotalTime
    for jj=1:1/tTimeStep
        ctrsgn;
        % Time integration by RK4
        para;
        RK4M;
        output;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IOT Glider 6 DOF Motion Simulator - data.m%
%Define constant parameters%
```

```
%by Peng Wei%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IOT Glider Parameters
% ------------------------------------------------------------
rho            =  1.00E+03;
% Water Density                          [kg/m^3]
g              =  9.81;
% g                                        [m/s^2]
A_piston       =  0.0036;
% Piston Area                              [m^2]
loa            =1.556;
% overall length                           [m]
W              =  116.2485;
% Measured Vehicle Weight                  [N]
lambda_wing    =  0.9948;
%Swept angle of the wings                  [rad]
x_wing_root_leading = 0.15;
%Wing root leading edge x-position in body frame, origin of CFD
r_stbwing      =  [x_wing_root_leading-0.1187 0.157 0];
%Position of the starboard wing center [m]
r_portwing     =  [x_wing_root_leading-0.1187 -0.157 0];
%Position of the portside wing center  [m]
A_wing         =   0.020412;
%Per Wing area                  [m^2]
l_wing         =   0.3;
%Wing root to tip length          [m]
r_h = 0.0574;
%Radius of Hull cross section
A_h = pi*r_h^2;
%Cross section area of Hull
X_BodyFrame = 0.72;
%Body frame origin from nose tip (ESAM Origin)
XB             =  0.70;
% Distance from Nose tip to Initial CB (Local frame origin point)
% Center of Buoyancy w.r.t. Origin(CB)
% ------------------------------------------------------------
xB_0           =  X_BodyFrame-XB;
% Center of Buoyancy: X-dir                [m]
yB             =  0.00E+00;
```

```
% Center of Buoyancy: Y-dir                [m]
zB          =  0.00E+00;
% Center of Buoyancy: Z-dir                [m]
% Center of Gravity w.r.t. Origin at CB
% ------------------------------------------------------------------
xG_0          =  X_BodyFrame-XB;
% Center of Gravity: X-dir                 [m]
yG_0          =  0.00E+00;
% Center of Gravity: Y-dir                 [m]
zG_0          =  1.96E-03;
% Center of Gravity: Z-dir                 [m]
% Non-linear Force Coefficients
% ------------------------------------------------------------------
% Added Mass Terms
Xudot       = -1.708E-01;   [kg]
Xvdot       = 0E0;          [kg]
Xwdot       = 0E0;          [kg]
Xpdot       = 0E0;          [kgm/rad]
Xqdot       = 0E0;          [kgm/rad]
Xrdot       = 0E0;          [kgm/rad]
Yvdot       = -1.42E+01;    [kg]
Ywdot       = 0E0;          [kg]
Ypdot       = 0E0;          [kgm/rad]
Yqdot       = 0E0;          [kgm/rad]
Yrdot       =  2.011E-01;   [kgm/rad]
Zwdot       = -1.666E+01;   [kg]
Zpdot       = 0E0;          [kgm/rad]
Zqdot       = -2.612E-01;   [kgm/rad]
Zrdot       = 0E0;          [kgm/rad]
% Hydrodynamic Damping Terms
Yrr         =  -5.3140;
% Cross-flow Drag                         [kgm/rad^2]
Zqq         = 5.5957;
% Cross-flow Drag                         [kgm/rad^2]
% Non-linear Moment Coefficients
% ------------------------------------------------------------------
% Added Mass Terms
Kpdot       = -7.062E-02;    [kgm2/rad]
Kqdot       = 0E0;           [kgm2/rad]
```

```
Krdot        = 0E0;              [kgm2/rad]
Mqdot        = -2.161E+00;       [kgm2/rad]
Mrdot        = 0E0;              [kgm2/rad]
Nrdot        = -2.152E+00;       [kgm^2/rad]
% Hydrodynamic Damping Terms
Nrr          = -9.1668;
% Cross-flow Drag                          [kgm^2/rad^2]
Kpp          = -0.3547;
% Rolling Resistance                       [kgm2/rad^2]
Mqq          = -9.1981;
% Cross-flow Drag                          [kgm^2/rad^2]
% Other variables
% ------------------------------------------------------------
m_piston = 0.7;
%piston + rod mass                    [kg]
l_piston = X_BodyFrame-0.246;
%piston x position in body frame when piston in zero position
l_batt = X_BodyFrame-0.465728;
%battery pack x position in body frame when battery pack
%in zero position
m_batt = 0.5;
%battery pack mass              [kg]
max_piston = 0.01 ;
%max piston position          [m]
max_x_batt = 0.020 ;
%max battery pack position   [m]
v_piston = 0.005;
%piston speed            [m/s]
v_batt = 0;
%battery pack speed            [m/s]
theta_dot_batt = 0;
%battery pack rotating speed        [rad/s]
rG_batt = 0.02;
%battery pack CG radius w.r.t. hull central axle    [m]
% delta_max = 1.36E+01;
% Maximum Fin Angle                        [deg]
D2R = pi/180;
% type conversion: deg to rad
R2D = 180/pi;
```

```
% type conversion: rad to deg
m = W / g;
% weight/G                [kg]
% Definitions of variables for the simulation
tTimeStep = 0.01;
tTotalTime = 15;
tHorizon = tTotalTime/tTimeStep;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IOT Glider 6 DOF Motion Simulator - init.m%
% initial conditions%
%by Peng Wen%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Moments of Inertia w.r.t. Origin at CB
% ------------------------------------------------------------------------
Ixx          = 1.5E-02;
% Moment of Inertia                          [kgm^2]
Iyy          = 1.645E+00;
% Moment of Inertia                          [kgm^2]
Izz          = 1.645E+00;
% Moment of Inertia                          [kgm^2]
Ixy=0;
Ixz=0;
Iyz=0;
% State vector
% x = [u v w p q r xpos ypos zpos qt0 qt1 qt2 qt3]'
x = zeros(13, 1);
x(1) = 0.15*cos(2/180*pi);
x(2) = 0;
x(3) = 0.15*sin(2/180*pi);
% x(1) = 1.54;                    % forward velocity: 1.54 m/s
phi      =  0*D2R;              % 5 deg ==> rad
theta    =  -20*D2R;
psi      =  0*D2R;
% Converting Euler angles to Quaternions
cpsi2 = cos(psi/2);       spsi2 = sin(psi/2);
cthe2 = cos(theta/2);     sthe2 = sin(theta/2);
cphi2 = cos(phi/2);       sphi2 = sin(phi/2);
x(10) = cphi2*cthe2*cpsi2 + sphi2*sthe2*spsi2;      % qt0
x(11) = sphi2*cthe2*cpsi2 - cphi2*sthe2*spsi2;      % qt1
```

```
x(12) = cphi2*sthe2*cpsi2 + sphi2*cthe2*spsi2;      % qt2
x(13) = cphi2*cthe2*spsi2 - sphi2*sthe2*cpsi2;      % qt3
forces = zeros(6, 1);
% control variables
x_piston = 0;  %initial piston position              [m]
x_piston_0 = 0 ;
x_batt =0;  %initial battery pack position           [m]
x_batt_0 =0;  %initial battery pack positio          [m]
theta_batt = 0; %initial battery pack angular position
%deltaR = [0, 4, -4].*D2R;        % angle of rudder plane      [rad]
%deltaS = 0;                      % angle of stern plane       [rad]
U = sqrt(x(1)*x(1) + x(2)*x(2) + x(3)*x(3));
% speed       [m/s]
% Dummy variables for numerical integration(Runge-Kutta 4th order)
k1 = zeros(13, 1);
k2 = zeros(13, 1);
k3 = zeros(13, 1);
k4 = zeros(13, 1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IOT Glider 6 DOF Motion Simulator - ctrsgn.m%
% control signals%
%by Peng Wen%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%The three control variables are:
%1. position of the piston, x_piston;
%2. position of the battery pack, x_batt;
%3. rotated angle of the battery pack,theta_batt.
%The control signals are used dependent%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IOT Glider 6 DOF Motion Simulator - para.m%
% Re-evaluate parameters%
%by Peng Wen%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    B = W + x_piston * A_piston * rho * g;
    xG = (xG_0 * m + x_piston * m_piston + x_batt * m_batt) / m;
    yG = (yG_0 * m + rG_batt * sin (theta_batt) * m_batt) / m;
    zG = (zG_0 * m - rG_batt * cos (theta_batt) * m_batt) / m;
    xB = (xB_0 * W + x_piston * A_piston * rho * g * (l_piston...
```

```
+ x_piston/2 + 0.033)) / B;
Iyy = Iyy + m_piston * ((x_piston + l_piston)^2 ...
- (x_piston_0 + l_piston)^2) + m_batt * ((x_batt...
+ l_batt)^2 - (x_batt_0 + l_batt)^2);
Izz = Iyy;
Ixy = - m * xG * yG;
Ixz = - m * xG * zG;
Iyz = - m * yG * zG;
x_piston_0 = x_piston;
x_batt_0 = x_batt;
% Inertia matrix
% ---------------------------------------------------------------
m11 =  m - Xudot;
m15 =  m*zG;
m16 = -m*yG;
m22 =  m - Yvdot;
m24 = -m*zG;
m26 =  m*xG - Yrdot;
m33 =  m - Zwdot;
m34 =  m*yG;
m35 = -m*xG - Zqdot;
m42 = -m*zG;
m43 =  m*yG;
m44 =  Ixx - Kpdot;
m51 =  m*zG;
m53 = -m*xG - Zqdot;
m55 =  Iyy - Mqdot;
m61 = -m*yG;
m62 =  m*xG - Yrdot;
m66 =  Izz - Nrdot;
Inertia = zeros(6, 6);
Inertia(1, 1) = m11;  Inertia(1, 5) = m15;  Inertia(1, 6) = m16;
Inertia(2, 2) = m22;  Inertia(2, 4) = m24;  Inertia(2, 6) = m26;
Inertia(3, 3) = m33;  Inertia(3, 4) = m34;  Inertia(3, 5) = m35;
Inertia(4, 2) = m42;  Inertia(4, 3) = m43;  Inertia(4, 4) = m44;
Inertia(5, 1) = m51;  Inertia(5, 3) = m53;  Inertia(5, 5) = m55;
Inertia(6, 1) = m61;  Inertia(6, 2) = m62;  Inertia(6, 6) = m66;
Minv = inv(Inertia);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% IOT Glider 6 DOF Motion Simulator - RK4M.m%
% Numerical Integrator Using RK-4 Method%
%by Peng Wen%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
x0=x;
ME;
k1=xDot;
x=x + 0.5*tTimeStep*k1;
ME;
k2=xDot;
x=x + 0.5*tTimeStep*k2;
ME;
k3=xDot;
x=x +tTimeStep*k3;
ME;
k4=xDot;
x = x0 + tTimeStep*(k1 + 2*k2 + 2*k3 + k4)/6;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% IOT Glider 6 DOF Motion Simulator - ME.m%
% Derivative of the State Vector Evaluation%
%by Peng Wen%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Get and check state variables and control inputs
% Get state variables
% ---------------------------------------------------------------
u = x(1);         v = x(2);         w = x(3);
p = x(4);         q = x(5);         r = x(6);
qt0 = x(10);      qt1 = x(11);      qt2 = x(12);      qt3 = x(13);
% Normalization of quaternion
quat = [qt0 qt1 qt2 qt3]'/sqrt(qt0*qt0 + qt1*qt1...
+ qt2*qt2 + qt3*qt3);
qt0 = quat(1);
qt1 = quat(2);
qt2 = quat(3);
qt3 = quat(4);
U = sqrt(u*u + v*v + w*w);
% Rotation matrix:
Cb2n = ...
[1-2*(qt2*qt2+qt3*qt3) 2*(qt1*qt2-qt0*qt3) 2*(qt0*qt2+qt1*qt3);
```

```
2*(qt0*qt3+qt1*qt2)   1-2*(qt1*qt1+qt3*qt3) 2*(qt2*qt3-qt0*qt1);
2*(qt1*qt3-qt0*qt2)   2*(qt0*qt1+qt2*qt3)   1-2*(qt1*qt1+qt2*qt2)];
% Calculate AOA & drag & lift & pitch moment on wings
v_stbwing = [u v w] + cross([p q r],r_stbwing);
v_portwing = [u v w] + cross([p q r],r_portwing);
if v_stbwing(1)==0 && v_stbwing(3)==0
    alfa_stbwing = 0;
else
    alfa_stbwing = atan(sin(atan(v_stbwing(3)/v_stbwing(1)))...
    /(cos(atan(v_stbwing(3)/v_stbwing(1)))*sin(lambda_wing)));
end
if v_portwing(1)==0 && v_portwing(3)==0
    alfa_portwing = 0;
else
    alfa_portwing = atan(sin(atan(v_portwing(3)/v_portwing(1)))...
    /(cos(atan(v_portwing(3)/v_portwing(1)))*sin(lambda_wing)));
end
CD_stb = polyval([1.8551e-4 0 0.0123],alfa_stbwing/pi()*180);
CD_port = polyval([1.8551e-4 0 0.0123],alfa_portwing/pi()*180);
CL_stb = polyval([-4.922e-6 0 0.0146 0],alfa_stbwing/pi()*180);
CL_port = polyval([-4.922e-6 0 0.0146 0],alfa_portwing/pi()*180);
CM_stb = polyval([0.301 0 -0.2756 0],alfa_stbwing/pi()*180);
CM_port = polyval([0.301 0 -0.2756 0],alfa_portwing/pi()*180);
D_stb = 0.5 * rho * U^2 * A_wing * CD_stb;
D_port = 0.5 * rho * U^2 * A_wing * CD_port;
L_stb = 0.5 * rho * U^2 * A_wing * CL_stb;
L_port = 0.5 * rho * U^2 * A_wing * CL_port;
M_stb = 0.5 * rho * U^2 * A_wing * l_wing * CM_stb;
M_port = 0.5 * rho * U^2 * A_wing * l_wing * CM_port;
if v_stbwing(1)==0 && v_stbwing(3)==0
    afa_s = 0;
else
    afa_s = atan(v_stbwing(3)/v_stbwing(1));
end
if v_portwing(1)==0 && v_portwing(3)==0
    afa_p = 0;
else
    afa_p = atan(v_portwing(3)/v_portwing(1));
end
```

```
X_wing = L_stb * sin(afa_s) - D_stb * cos(afa_s) ...
    + L_port * sin(afa_p) - D_port * cos(afa_p);
Z_wing = -L_stb * cos(afa_s) - D_stb * sin(afa_s) ...
    - L_port * cos(afa_p) - D_port * sin(afa_p);
M_wing = M_stb + M_port - x_wing_root_leading * Z_wing;
% Calculate AOA & drag & lift & pitch moment on hull
% ------------------------------------------------------------------
if u == 0
    alfa_h = 0;
else
    alfa_h = sign(w) * atan(sqrt(v^2+w^2)/u);
end
if w == 0
    slip_h = 0;
else
    slip_h = -atan(v/w);
end
CD_h = polyval([13.5495 0 2.7638 0],alfa_h);
CD_h = polyval([1.3964e-6 0 0.0012752 0 0.16454],alfa_h*R2D);
CL_h = polyval([12.6763 0.0084 3.223 0],alfa_h);
CL_h = polyval([1.8219e-5 0 0.01142 0],alfa_h*R2D)/A_h...
    *0.01398^(2/3);
CM_h = polyval([-7.932 -0.0043 -0.3319 0],alfa_h);
CM_h = polyval([-9.9954e-6 0 0.022261 0],alfa_h*R2D)/A_h...
    /loa*0.01398;
D_h = 0.5 * rho * U^2 * A_h * CD_h;
L_h = 0.5 * rho * U^2 * A_h * CL_h;
M_hull = 0.5 * rho * U^2 * A_h * loa * CM_h;
Xp_h = L_h * sin(alfa_h) - D_h * cos(alfa_h);
Zp_h = - L_h * cos(alfa_h) - D_h * sin(alfa_h);
F_h = L_transform(slip_h, 0, 0, [Xp_h;0;Zp_h]);
M_h = R_transform(slip_h, 0, [0;M_hull;0]);
% Hydrostatic forces:
% ------------------------------------------------------------------
HydrostaticForces = Cb2n'*[0 0 W-B]';
Xhs = HydrostaticForces(1);
Yhs = HydrostaticForces(2);
Zhs = HydrostaticForces(3);
HydrostaticMomentW = [0 -zG yG; zG 0 -xG; -yG xG 0]*Cb2n'*[0 0 W]';
```

```
HydrostaticMomentB = [0 -zB yB; zB 0 -xB; -yB xB 0]*Cb2n'*[0 0 -B]';
Khs = HydrostaticMomentW(1) + HydrostaticMomentB(1);
Mhs = HydrostaticMomentW(2) + HydrostaticMomentB(2);
Nhs = HydrostaticMomentW(3) + HydrostaticMomentB(3);
% Added Mass cross term forces:
% ------------------------------------------------------------------
XAM = - Xvdot*u*r + Xwdot*u*q - Yvdot*v*r + Ywdot*(v*q - w*r)...
    - Ypdot*r*p + (Zrdot - Yqdot)*q*r - Yrdot*r^2 + Zwdot*w*q...
    + Zqdot*q^2 + Zpdot*p*q;
YAM = Xvdot*v*r - Ywdot*v*p + Xudot*u*r + Xwdot*(w*r-u*p)...
    + (Xpdot-Zrdot)*r*p + Xqdot*q*r + Xrdot*r^2 - Zwdot*w*p...
    - Zpdot*p^2 - Zqdot*p*q;
ZAM = -Xwdot*w*q + Ywdot*w*p - Xudot*u*q + Xvdot*(u*p-v*q)...
    + (Yqdot-Xpdot)*q*p - Xqdot*q^2 - Xrdot*q*r + Yvdot*v*p...
    + Ypdot*p^2 + Yrdot*r*p;
KAM = -Ypdot*w*p+Zpdot*v*p-Kqdot*r*p+Krdot*p*q-Xvdot*u*w...
    +Xwdot*u*v-Xqdot*u*r+Xrdot*u*q+(Zwdot-Yvdot)*v*w...
    +Ywdot*(v^2-w^2)-(Yqdot-Zrdot)*(v*r+w*q)+(Yrdot...
    +Zqdot)*(v*q-w*r)-(Mqdot-Nrdot)*q*r+Mrdot*(q^2-r^2);
MAM = Xqdot*w*q-Zqdot*u*q+Kqdot*r*q-Mrdot*p*q+(Xudot...
    -Zwdot)*u*w+Xvdot*w*v+Xwdot*(w^2-u^2)+(Xpdot...
    -Zrdot)*(p*w+u*r)-(Xrdot+Zpdot)*(p*u-w*r)...
    -Ywdot*u*v+Ypdot*v*r-Yrdot*v*p+(Kpdot-Nrdot)...
    *p*r-Krdot*(p^2-r^2);
NAM = -Xrdot*v*r+Yrdot*u*r-Krdot*q*r+Mrdot*p*r-(Xqdot...
    -Yvdot)*u*v+Xvdot*(u^2-v^2)-Xwdot*v*w-(Xpdot-Yqdot)...
    *(u*q+v*p)+(Xqdot+Ypdot)*(u*p-v*q)+Ywdot*u*w...
    -Zpdot*w*q+Zqdot*w*p-(Kpdot-Mqdot)*p*q+Kqdot*(p^2-q^2);
% Set total forces from equations of motion
X = -m*w*q + m*xG*q*q + m*v*r + m*xG*r*r - m*yG*p*q - m*zG*p*r  ...
    + Xhs...      Hydrostatics
    +XAM ...   Added mass terms
    + X wing...   Hydrodynamic, wings
    + F_h(1);
Y = m*w*p - m*u*r - (m*zG)*q*r - m*xG*p*q + m*yG*(r*r + p*p)...
    + Yhs...
    +  YAM...
    +  Yrr*r*abs(r)...  damping
    + F h(2);
```

```
Z = (m*zG)*p*p + (m*zG)*q*q + m*u*q - m*v*p - m*xG*r*p - m*yG*r*q  ...
    + Zhs  ...
    +ZAM  ...
    + Zqq*q*abs(q)...   damping
    +Z  wing...
    + F_h(3);
K = p*q*Ixz - (r^2 - q^2)*Iyz - p*r*Ixy - (Izz-Iyy)*q*r...
    - (m*zG)*w*p + (m*zG)*u*r + m*yG*u*q - m*yG*v*p  ...
    + Khs...
    + KAM...
    + Kpp*p*abs(p)...   damping
    +M_h(1);
M = - (Ixx-Izz)*r*p + Ixy*q*r - (p^2 -r^2)*Ixz - Iyz*q*p...
    + (m*zG)*v*r - (m*zG)*w*q - m*xG*u*q + m*xG*v*p  ...
    + Mhs...
    + MAM...
    + Mqq*q*abs(q)...    damping
    + M_wing...
    +M_h(2);
N = -(Iyy-Ixx)*p*q + Iyz*r*p - (q^2-p^2)*Ixy - r*q*Ixz...
    + m*xG*w*p- m*xG*u*r + m*yG*w*q - m*yG*v*r  ...
    + Nhs  ...
    + NAM...
    + Nrr*r*abs(r)... %damping
    + M_h(3);
forces = [X Y Z K M N]';
% Quaternion based
xDot = ...
    [Minv(1,1)*X        + Minv(1,3)*Z        + Minv(1,5)*M;
    + Minv(2,2)*Y       + Minv(2,4)*K        + Minv(2,6)*N;
    Minv(3,1)*X         + Minv(3,3)*Z        + Minv(3,5)*M;
    + Minv(4,2)*Y       + Minv(4,4)*K        + Minv(4,6)*N;
    Minv(5,1)*X         + Minv(5,3)*Z        + Minv(5,5)*M;
    + Minv(6,2)*Y       + Minv(6,4)*K        + Minv(6,6)*N;
    Cb2n*[u v w]';
    -0.5*qt1*p     - 0.5*qt2*q     - 0.5*qt3*r;
    0.5*qt0*p      - 0.5*qt3*q     + 0.5*qt2*r;
    0.5*qt3*p      + 0.5*qt0*q     - 0.5*qt1*r;
    -0.5*qt2*p     + 0.5*qt1*q     + 0.5*qt0*r];
```

```
%Steady State Solving Function%
%By Peng Wen%
%x=[theta VA alfa]
%F=[X Z M]
function F = projectmodel_modified(x)
%x=[theta VA alfa]
theta = x(1);%Pitching angle
VA=x(2);%Total velocity
alfa=x(3);%AOA
psi = 0;%Roll angle
phi = 0;%Yaw angle
x_piston=0.01;%BE position
x_batt=0;%Battery pack position
lambda_wing=  0.9948;
rho=1000;
g=9.81;
A_wing=0.020412;
x_wing_root_leading = 0.15;
l_wing =   0.3;
R2D = 180/pi;
A_h=pi*0.0574^2;
loa=1.556;
W=  116.2485;
m=W/g;
A_piston =  0.0036;
xG_0=0.72-0.677;
xB_0=0.72-0.677;
m_piston = 0.7;
m_batt = 0.5;
l_piston= 0.72-0.246;
B = W + x_piston * A_piston * rho * g;
xG = (xG_0 * m + x_piston * m_piston + x_batt * m_batt) / m;
xB = (xB_0 * W + x_piston * A_piston * rho * g * (l_piston...
    + x_piston/2 + 0.033)) / B;
yB = 0;
zB = 0;
yG = 0;
zG =   1.96E-03;
% Converting Euler angles to Quaternions
```

```
cpsi2 = cos(psi/2);      spsi2 = sin(psi/2);
cthe2 = cos(theta/2);    sthe2 = sin(theta/2);
cphi2 = cos(phi/2);      sphi2 = sin(phi/2);
qt0 = cphi2*cthe2*cpsi2 + sphi2*sthe2*spsi2;      % qt0
qt1 = sphi2*cthe2*cpsi2 - cphi2*sthe2*spsi2;      % qt1
qt2 = cphi2*sthe2*cpsi2 + sphi2*cthe2*spsi2;      % qt2
qt3 = cphi2*cthe2*spsi2 - sphi2*sthe2*cpsi2;      % qt3
u = VA * cos (alfa);
v = 0;
w = VA * sin (-alfa);
%Rotation matrix
Cb2n = ...
[1-2*(qt2*qt2 + qt3*qt3) 2*(qt1*qt2 - qt0*qt3) 2*(qt0*qt2 + qt1*qt3);
2*(qt0*qt3 + qt1*qt2)  1-2*(qt1*qt1 + qt3*qt3) 2*(qt2*qt3 - qt0*qt1);
2*(qt1*qt3 - qt0*qt2)  2*(qt0*qt1 + qt2*qt3)  1-2*(qt1*qt1 + qt2*qt2)];
%Hydrodynamics on wings
v_stbwing = [u v w];
v_portwing = [u v w];
alfa_stbwing = -atan(sin(atan(v_stbwing(3)/v_stbwing(1)))...
    /(cos(atan(v_stbwing(3)/v_stbwing(1)))*sin(lambda_wing)));
alfa_portwing = -atan(sin(atan(v_portwing(3)/v_portwing(1)))...
    /(cos(atan(v_portwing(3)/v_portwing(1)))*sin(lambda_wing)));
CD_stb = polyval([1.8551e-4 0 0.0123],alfa_stbwing/pi()*180);
CD_port = polyval([1.8551e-4 0 0.0123],alfa_portwing/pi()*180);
CL_stb = polyval([-4.922e-6 0 0.0146 0],alfa_stbwing/pi()*180);
CL_port = polyval([-4.922e-6 0 0.0146 0],alfa_portwing/pi()*180);
CM_stb = polyval([0.0000016 0 -0.0048 0],alfa_stbwing/pi()*180);
CM_port = polyval([0.0000016 0 -0.0048 0],alfa_portwing/pi()*180);
D_stb = 0.5 * rho * VA^2 * A_wing * CD_stb;
D_port = 0.5 * rho * VA^2 * A_wing * CD_port;
L_stb = 0.5 * rho * VA^2 * A_wing * CL_stb;
L_port = 0.5 * rho * VA^2 * A_wing * CL_port;
M_stb = -0.5 * rho * VA^2 * A_wing * l_wing * CM_stb;
%wing pitching moment about wing root leading edge
M_port = -0.5 * rho * VA^2 * A_wing * l_wing * CM_port;
afa_s = atan(v_stbwing(3)/v_stbwing(1));
afa_p = atan(v_portwing(3)/v_portwing(1));
X_wing = L_stb * sin(afa_s) - D_stb * cos(afa_s) ...
    + L_port * sin(afa_p) - D_port * cos(afa_p);
```

```
Z_wing = L_stb * cos(afa_s) + D_stb * sin(afa_s) ...
    + L_port * cos(afa_p) + D_port * sin(afa_p);
M_wing = M_stb + M_port - x_wing_root_leading * Z_wing;
%Hydrodynamics on hull
alfa_h = -sign(w) * atan(sqrt(v^2+w^2)/u);
CD_h = polyval([2.02E-7 0 3.01E-4 0 0.035],alfa_h*R2D)/A_h...
    *0.01398^(2/3);
CL_h = polyval([1.8219e-5 0 0.01142 0],alfa_h*R2D)/A_h...
    *0.01398^(2/3);
CM_h = polyval([-9.9954e-6 0 0.022261 0],alfa_h*R2D)/A_h...
    /loa*0.01398;
D_h = 0.5 * rho * VA^2 * A_h * CD_h;
L_h = 0.5 * rho * VA^2 * A_h * CL_h;
M_hull = 0.5 * rho * VA^2 * A_h * loa * CM_h;
Xp_h = L_h * sin(alfa_h) - D_h * cos(alfa_h);
Zp_h = L_h * cos(alfa_h) + D_h * sin(alfa_h);
%Hydrostatics
HydrostaticForces = Cb2n'*[0 0 W-B]';
Xhs = HydrostaticForces(1);
Zhs = HydrostaticForces(3);
HydrostaticMomentW = [0 -zG yG; zG 0 -xG; -yG xG 0]*Cb2n'*[0 0 W]';
HydrostaticMomentB = [0 -zB yB; zB 0 -xB; -yB xB 0]*Cb2n'*[0 0 -B]';
Mhs = HydrostaticMomentW(2) + HydrostaticMomentB(2);
F=[X_wing+Xp_h+Xhs;Z_wing+Zp_h+Zhs;M_wing+M_hull+Mhs];

%Steady State Solving Solver%
%By Peng Wen%
function solveangle()
clear all;
options=optimset('Display','iter');
[y,Fval,exitflag] =fsolve(@projectmodel_modified,...
[20/180*pi;0.2;7/180*pi],options)
thita=y(1)/pi*180
V=y(2)
alpha=y(3)/pi*180
glideangle=(y(1)+y(3))/3.14*180
VZ=-(y(2)*sin(y(1)+y(3)))
VX=y(2)*cos(y(1)+y(3))
```

Appendix G

Plan View and Mass Distribution of the IOT Glider

Table G.1: Mass and C. G. Position of the IOT Glider's Components (Masses are in kg; positions are in mm; position's reference point is the glider's nose tip)

Components	**Mass**	**X C.G. Position**	**Y C.G. Position**	**Z C.G. Position**
Black Nose	0.531	107.83	0	0
Diaphragm End Pipe	1.125	490.719	0	0
Diaphragm Mount	0.553	265.38	0	0
Diaphragm Nut	0.154	212.28	0	0
Mid-body Section	0.517	734.34	0	0
Elec End Pipe	1.017	962.98	0	0
End Cap	0.348	1183.998	0	0
Tail	0.448	1322.818	0	0
Piston	0.489	246.11	0	0
Alignment Collar	0.307	348.89	0	0
Buoyancy Engine Actuator Base	1.134	687.008	0	0
Buoyancy Engine Actuator Tube	0.34	514.498	0	0
Buoyancy Engine Actuator Rod	0.227	539.943	0	0
Roll Motor and Mount	0.157	608.823	0	31.75
Pitch Motor[1]	0.23	571.628	0	0
Pillar 1	0.016	482.728	0	23.5
Pillar 2	0.048	482.728	0	-18.42
Batteries	0.35	465.728	0	19.636
Battery Mount	0.119	485.7275	0	0
Elec Trail	1.013	913.23	0	0
PX303	0.205	1139.798	0	19.05
Wings[2]	0.1	649.6	0	0
Wing Mounts	0.146	630.9	0	0
Fins	0.08	1170.0	0	0

[1]Including Pitch Motor, Pitch Motor Mount, Ring Gear

[2]This is the aftmost position situation. The corresponding X C.G. position for the most forward position is 559.6 mm

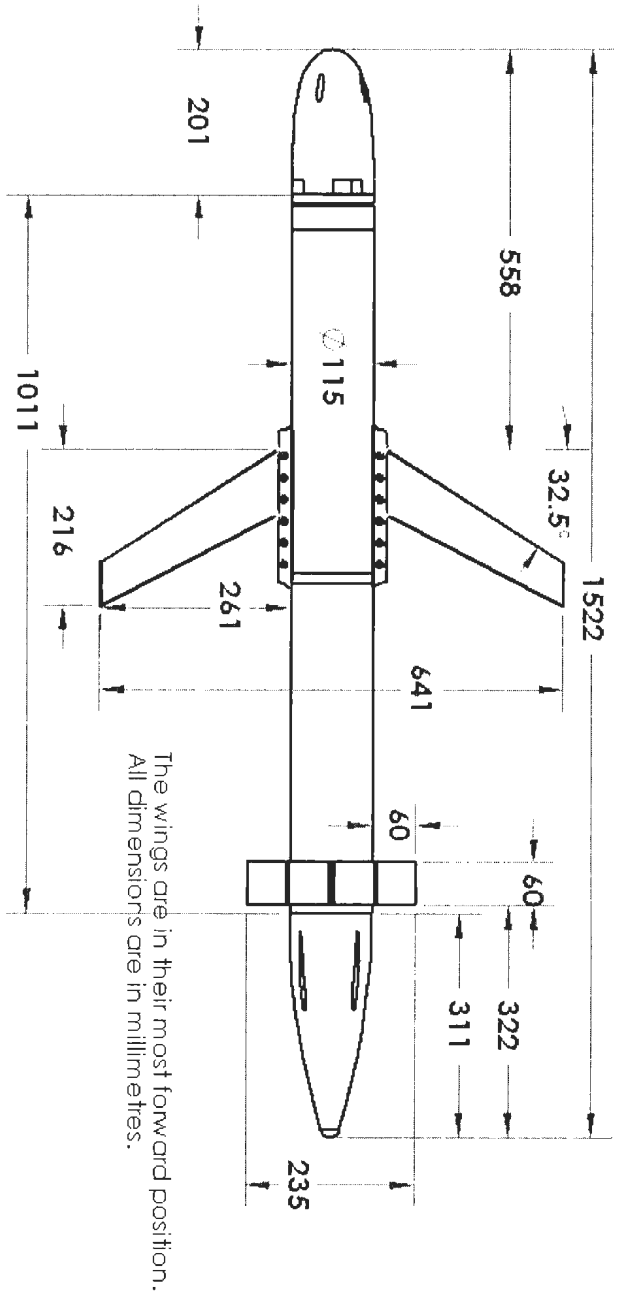


Figure G.1: The Plan View of the IOT Glider