UNDERWATER AUTOMATED VEHICLE

By

SHAHMI BIN SULAIMAN

FINAL PROJECT REPORT

Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

UNDERWATER AUTOMATED VEHICLE

By

Shahmi bin Sulaiman

13785

A project dissertation submitted to the Department of Electrical & Electronics Engineering Universiti Teknologi PETRONAS

In partial fulfillment of the requirement for the

BACHELOR OF ENGINEERING (Hons)

(ELECTRICAL & ELECTRONICS ENGINEERING)

Approved by,

_____

(AP Dr. Mohd Noh bin Karsiti)

Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

MAY 2014

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own expect as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

(Shahmi bin Sulaiman)

ABSTRACT

Design and fabricate an Automated Underwater Vehicle (AUV) which consisted of four vertical thurster, two horizontal thruster and gyroscope together with accelerometer for self – leveling to counter unbalance orientation while carrying loads. Also consisted with pressure sensor to control depth positioning without deviation, and underwater GPS based navigation system in the restricted electromagnetic signal medium. The scope of study for this project involved combination of mechanical and electrical knowledge, where the mechanical chassis was designed using Autodesk Inventor 2014 software before fabrication process to ensure the design has required quality. Then, the fabricated design were equipped with electronics part, consisting of programmed microcontroller, motor drivers, and other electronic sensors for feedback to achieve the desired output.

ACKNOWLEDGEMENT

First of all, I would like to extend my gratitude to Allah Lord Almighty for giving me health, time, ideas and opportunity to complete this challenging project. Without His help, I would never be able to complete or solve any found problems. Next, I am forever indebted with supports that I received from my beloved parent, Sulaiman bin Hussin and Maznah bte Abdullah – in term of financial support, and words of encouragement – to keep me stay motivated doing this project. They were always at my side whenever I felt demotivated to aid myself. I am also indebted with knowledge and ideas provided by my beloved Project Supervisor, Dr.Mohd Noh bin Karsiti. This project itself was very technical and requires knowledge and expertise gathered from him to guide me throughout this whole period. Not to forget to all my dearest friends who willingly sacrifice their valuable time to discuss and share their thought and knowledge to complete this project.

TABLE OF CONTENTS

APPENDICES

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

PROJECT BACKGROUND

1.1. Background Study

The Automated Underwater Vehicle (AUV) is widely used in various application such as for oil and gas industry, scientific research, and even for military. For oil and gas industry, AUV is used as a surveillance system on the seafloor before the development of subsea infrastructure [1]. It also has been used to inspect underwater pipelines and to create sonar mapping as well as swathe bathymetry [2]. For science applications, AUV is used to monitor deep-sea ecological system as well as data gathering equipment to measure concentrations of various element and the presence of microscopic life [1] . Other than that, AUV also is used to neutralize underwater mines and as a surveillance system for military/defense purposes [1]. In order to deliver a successful result for those operations, AUV must be equipped with the gyroscope, accelerometer, GPS module, magnetometer and pressure sensors to achieve a smooth movement, and good levelling on the specified depth. In this project, a mini scale AUV will be constructed to perform analysis and thus a solution to achieve the objectives.

1.2. Problem Statement

The increasing need of underwater operations has pushed the development of AUV to be more stable and able to perform complex task that requires more precision. This can be achieved by providing a good levelling on the AUV at certain depth. For instance, the orientation of the AUV should be always level at the allocated depth [3], despite the uneven mass distribution due to the load it carried. This uneven orientation of the AUV can disturb the precision of the underwater operation which can result to the failure of the operation.

Apart from that, the buoyancy factor also effect the positioning of AUV underwater. The positive buoyancy may cause the AUV to float away from its predetermined depth. The same goes to the negative buoyancy where the weight will pull the AUV to the underwater floor [3]. To deliver complex underwater operation, the AUV must be able to remain at the specified depth without deviations.

To navigate AUV from position A to position B autonomously underwater is a challenge since water medium restrict propagation of any electromagnetic signal. Other than that, acoustic communication equipment which usually used as an alternative for underwater navigation system is too expensive to be used for this Final Year Project. A reliable method need to be proposed in this project to ensure the AUV able to navigate to target location.

1.3. Objectives

The objectives of the project are:

1. To develop a proper control algorithm to achieve good leveling while carrying load
2. To design suitable AUV coordination and elevation control
3. Program and implement microcontroller based control strategy for motion control
4. To design and fabricate a model of AUV that can perform basic shallow underwater operation

1.4. Scope Of Study

The scope of study of this project includes:

I. Understanding on the principle of operation of the AUV and its applications in the wide range of industries.
II. Identifying mechanical and electronics design requirement on the AUV to ensure its reliability to be operated as intended as well as water tight.
III. Exploring 3D modelling software (Autodesk Inventor 2014) to model AUV.
IV. Exploring Arduino microcontroller and its language preferences.
V. Understanding and identifying critical parameters (i.e. angle of rotation, pressure, and etc.) that needs to be left constant or to be manipulated during the programming to achieve a better result.
VI. Implementation of control system based on sensor's feedback on the AUV system.

1.5. Relevancy of Project

The outcome of this project is to develop a complete set of AUV which has the capability to overcome technical difficulties and problems identified during initial of this study. Despite current AUV advance technology available in the industry able to cater these found issues, it is important for Electrical and Electronics Engineering student to grasp basic engineering knowledge and operational technicalities for AUV's sustainable development in the future. AUV technological development nowadays has been pushed to the more challenging and demanding environment. Therefore, this project is a good initiative and relevance at current time.

1.6. Project Feasibility

To develop a complete set of AUV, expertise in mechanical and electrical knowledge is crucial. Mechanical knowledge is required to do task such as 3D design modelling, machine and tools handling for fabrication. Meanwhile, electrical knowledge is critical to handle task such as to do electrical wiring, circuit troubleshooting, and Arduino programming. Taking advantage as a senior technical member of Petrobot robotic society who possessed both knowledge and expertise, this project works smoothly.

Other than that, this project requires lots of time to obtain all parts, and to design and fabricate the whole mechanical and electrical system. Therefore, to ensure the final product is ready before the end of this Final Year Project 2 course, the project activities has begun since the beginning of Final Year Project 1 course.

CHAPTER 2

LITERATURE REVIEW

2.1. Navigation System

GPS system combined with the digital compass is one of the best navigation system to navigate AUV underwater. A group of researcher from Universidad Austral de Chile, and Universitat Politecnica de Valencia has applied this concept on their designed AUV. The AUV memorized the heading course in degrees using digital compass and its initial and the final coordinate using GPS system on the surface of the water. The AUV then submerged in the water and moves to the next coordinate within some interval of time. Then, the AUV will resurface to do corrections based on the recent coordinate and course before resubmerge and continue heads to the targeted coordinate. These processes will be repeated until it arrived at the target coordinate [3].

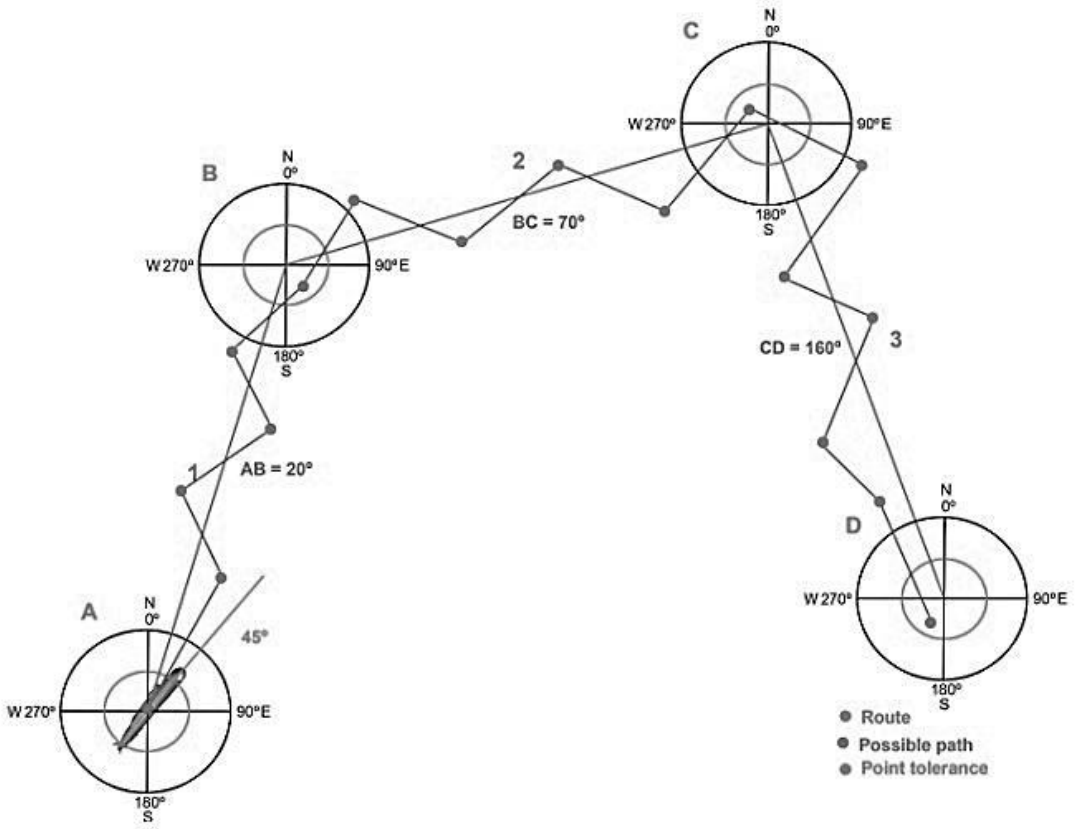

Figure 1: AUV route example

To further assist navigation system, the researchers from Virginia Polytechnic Institute and State University and Brigham Young University has proposed of using sonar system to be equipped on the AUV. The sonar system helps to determine distance between AUV to other object in the area of limited visibility [4,5].

2.2. Tilt Angle Measurement

Accelerometer is an electronic devices that are very sensitive and accurate over time to measure acceleration due to the gravity or body movement as it contain sensitive elements that are free to move internally. Acceleration is the product of vectorial components of X, Y and Z axis. These components can be mathematically translated into tilt angle X and Y axis by using this equations[6]:

$$A_x = \arctan\left(\frac{X}{\sqrt{Y^2 + Z^2}}\right)$$

$$A_y = \arctan\left(\frac{Y}{\sqrt{X^2 + Z^2}}\right)$$

Although it able to measure X and Y axis of tilt angle, accelerometers are very sensitive to vibrations and it does not recognized the source of the acceleration – either due to the gravity's effect or due to the external force applied[7].

The gyroscope in the other hand, is an electronic devices that measure the rate of rotation(deg/sec) and the rotation angle(deg) while it is far less susceptible to any weaknesses as accelerometer. However, gyroscope suffer minimal amount of error on each measurement and the effect is cumulative due to integration of the previous measurement. Later on, this integral amount of error over time will introduce a drift in the measurement [7,8].

Therefore, the integration of both accelerometer and gyroscope into the system will produce a perfect angular measurement as both of the device will compensate the weaknesses at each other[7].

2.3. Depth Control

A conventional method to maintain underwater vehicle's altitude underwater is by using pressure sensor as reference [9,10]. Pressure sensor operates by converting analogue input(pressure) to electrical signal. The depth of AUV can be calculated using a simple mathematical equation;

$$P = \rho g h;$$

Where ρ is 1000 Kg/m^3 of density of water, g is 9.81 m/s^{-2} of gravity, h is the unknown water depth, and P is the pressure detected by sensor. The equation can be manipulated to find out AUV depth [11];

$$\text{AUV depth, } h = \frac{P}{(1000)(9.81)}$$

2.4. Buoyancy

When an object is immersed partially or wholly in a fluid, the object experienced an upward force known as buoyancy force and it is equal to the weight of water it displaces. This principle is known as Archimedes Principle. The water displaced is due to the effect of gravity which it force the object downwards and the volume of water displaced is equivalent to the volume of the object immersed[12].

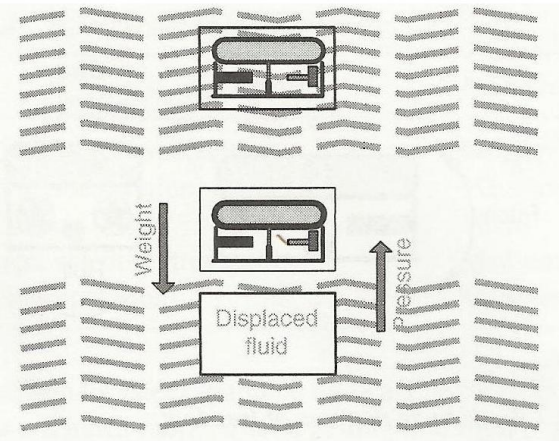

Figure2: Hydrostatic Equilibrium of AUV

In designing AUV, the type of buoyancy must be carefully selected. AUV with negative buoyancy will pull the body downward. Meanwhile, positive buoyancy will

push the AUV to the surface. Neutral buoyancy in other hand is in between positive and negative buoyancy, where it remain static underwater unless there are external forces that push it downward or pull it to the surface. However in real application, the AUV is suggested to have a very little positive buoyancy close to neutral buoyancy to ease the retrieving process later on.
To ensure the AUV have good buoyancy factor, the following concept of calculation should be followed[13]:

Nett weight = (Weight of AUV due to buoyancy) – (Weight of AUV due to gravity) = 0 Kg

Where,

Weight of AUV due to buoyancy = (AUV Total Volume, m^3) x 1000 Kg/m^3

2.5. Stability

There are 'imaginary centers' which Centre of Buoyancy(COB) and Centre of Gravity(COG) are positioned on any object underwater. The distance between COG and COB is called Buoyancy Gradient. These two parameters are very important to determine AUV's stability underwater. The increase of Buoyancy Gradient will increase stability, but in return will sacrifice manueverity. Meanwhile, reducing Buoyancy Gradient or positioned COG and COB at the same spot will increase manueverity but it will become unstable and easy to turn over without a proper control. Thus, to ensure the AUV's stability, the COB should always positioned above the COG[14].

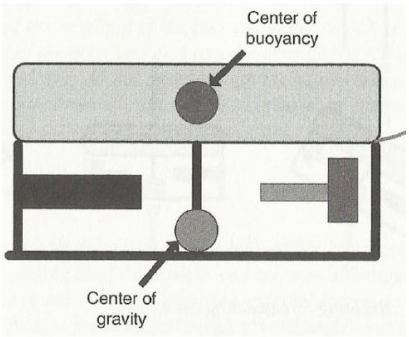

Figure 3: Stability of AUV

CHAPTER 3

METHODOLOGY

3.1. Research Methodology

Figure 4: Research Methodology

3.2. Project Activities

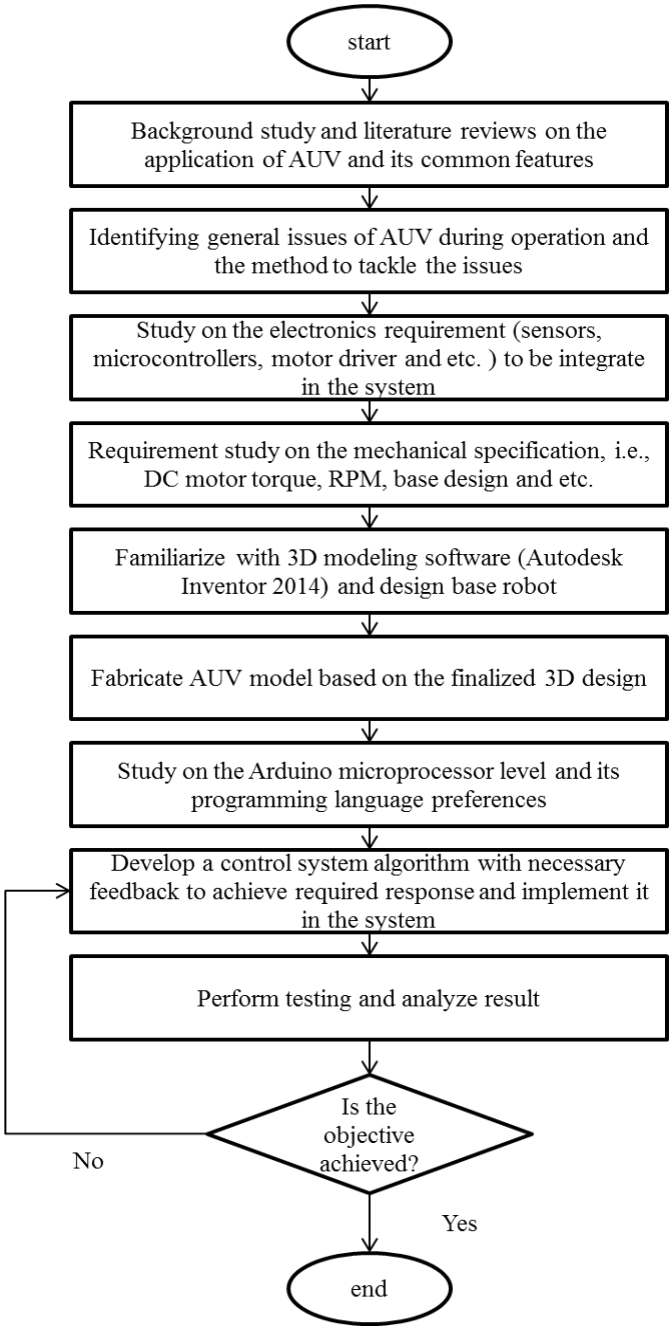

Figure 5: Flow Chart of Project Activities

3.3. Hardwares, Softwares & Tools Required

Tools & softwares that will be used throughout the project are:

Hardware(Electronics)	Software	Tools
• Arduino 2560 microcontroller • DC motor driver • Submersible brush DC motor • Gyroscope • Accelerometer • Pressure sensor • GPS module • Magnetometer • 12V Battery	• Microsoft Office • Autodesk Inventor 2014 • Arduino IDE	• Multimeter • Wire Cutter & stripper • Allen Key • Soldering Tools • Heat Glue Gun • Hex Saw • Rivetter • Clamper • Screw driver • Epoxy – consumable • Aluminium – consumable

Table 1: List of electronic hardwares, softwares and tools required

CHAPTER 4

RESULT AND DISCUSSIONS

4.1. AUV Design And Fabrication

The AUV's base design was created using Autodesk Inventor 2014 and the designs were selected based on the criteria below:

- Simple frame design
- Robust design
- Design effectiveness(hydrodynamic and slightly positive buoyancy)
- Screw-less
- Easy to fabricate
- Easily available material
- Design with lowest cost
- Good aesthetic value
- Better buoyancy gradient

Below is the selected design modeled in the software. The approximate dimension of this design is 400mm x 400mm x 270mm. The main material of this body is made of 0.5in x 0.5in aluminum for robustness while the square type shape of aluminum provides more flat surfaces which ease any joint attachment. The aluminum also is lightweight material since it is hollow and relatively cheap. For the motor holder, it is made from 6mm thickness Perspex. This Perspex offers lightweight and easy to fabricate using precision cutting machine. The motors were positioned at each edge of the frame to allow the AUV to maintain its vertical leveling in response to the gyroscope and accelerometer reading. Another two motors were placed at the centered side of the AUV as main forward and backward thruster. It was purposely positioned at the centre of AUV's length to ease the AUV to rotate 360 degrees or to turn right and left by means of speed differential.

To maintain neutral buoyancy, ballast system was included in the design. It was positioned underneath Centre of Buoyancy to achieve underwater stability. Besides, the ballast system was designed to increase distance between Centre of Gravity and Centre of Buoyancy such that it will improve underwater stability. However, by doing

this it will compromise AUV manueverity by increasing underwater drag. Since this AUV works at low speed, the effect of drag induced underwater was neglected.

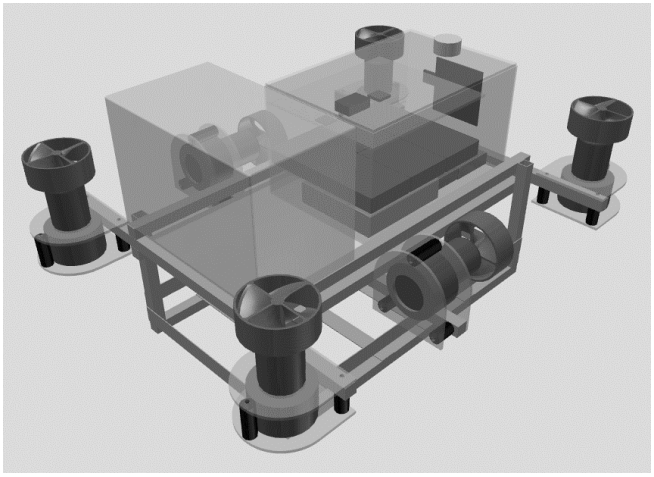

Figure 6 : Perspective view of AUV

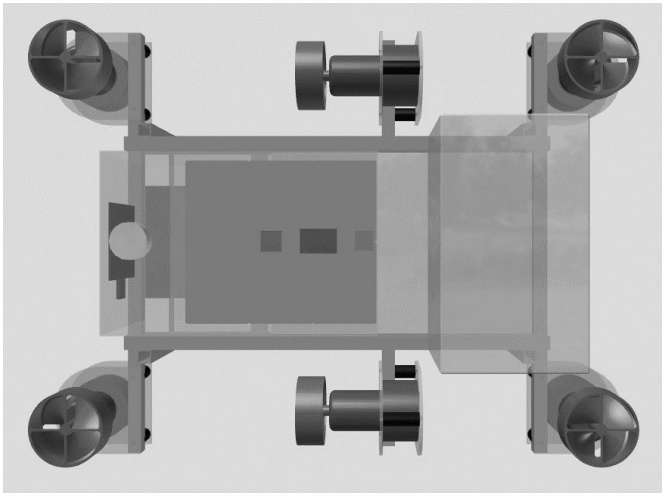

Figure 7: Top view of AUV

After design selection, the design was fabricated according to the dimension and specification modeled in the Autodesk Inventor. The fabricated design is shown in the figures below.

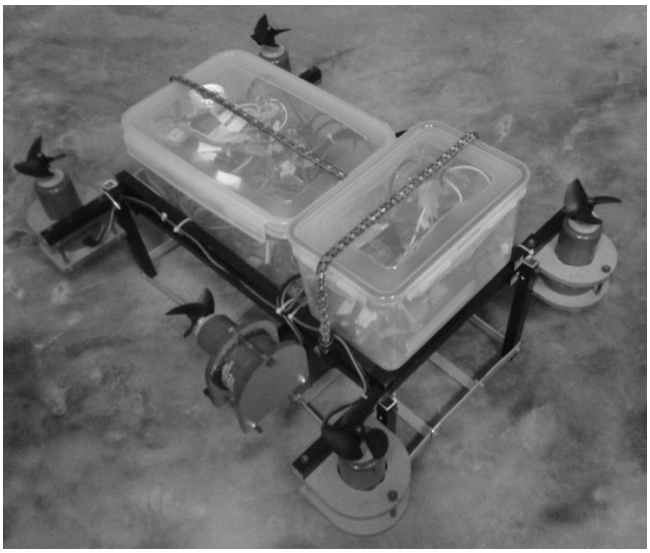

Figure 8: fabricated AUV

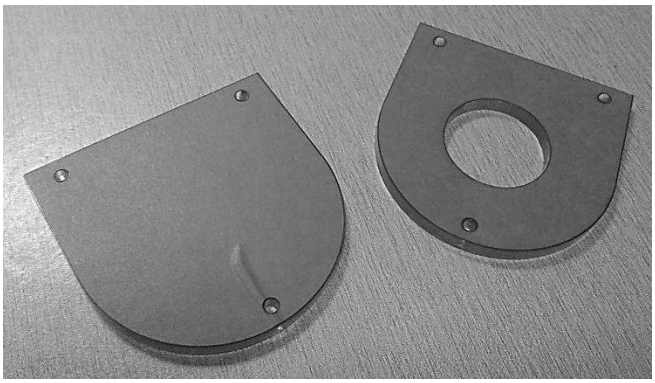

Figure 9: Perspex motor holder

To isolate electronic devices from contact with water, a good waterproof container was selected. The container's dimension is 150 x 225 x 128mm enough to accommodate motor drivers, Arduino microcontroller, and other electronic sensors. It is made of plastics and the lid is covered with a layer of silicon on its edge to prevent water or air to enter once it is sealed. An additional air tight container with dimension 200 x 135 x 115mm was added to safeguard the sensitive circuits connected to the microcontroller from issue such as intermittent failure. It is used to accommodate batteries, switches and other components that requires high maintenance.

Figure 10: Waterproof container

All external DC motor operates based on the signal it received from the Arduino microcontroller placed inside of the waterproof container. To do that, a small hole was created on the waterproof container to connect external DC motors and microcontroller. Upon successfully established the connection, the hole must be properly sealed to protect electronics hardware inside the container. Therefore, hardened epoxy was applied at the created hole as a protection layer. Several test were conducted to identify and eliminate the presence of water leakage.

Figure 11: Hardened epoxy applied at the enclosure

4.2. Obtaining Neutral Buoyancy

To obtain neutral buoyancy, the difference of buoyancy force and weight of AUV due to gravity should be zero as studied in the literature review.

Nett weight = (Weight of AUV due to buoyancy) – (Weight of AUV due to gravity) = 0 Kg

Where,

Weight of AUV due to buoyancy = (AUV Total Volume, m^3) x 1000 Kg/m^3

*Weight of AUV due gravity = **5.1Kg***

To obtain AUV's total volume, the volume of AUV's parts has been measured separately and summed. Below are volumes of AUV's parts:

ITEM	**VOLUME**
Airtight Container 1	$6.916 \times 10^{-3}\ m^3$
Airtight Container 2	$3.105 \times 10^{-3}\ m^3$
DC Motor (x6)	$0.8773 \times 10^{-3}\ m^3$
DC Motor Holder Top (x6)	$8.55 \times 10^{-5}\ m^3$
DC Motor Holder Bottom (x6)	$1.028 \times 10^{-4} m^3$
Screw (x20)	$4.241 \times 10^{-5}\ m^3$
Stretchable Rope (x2)	$4.222 \times 10^{-5}\ m^3$
Aluminium Frame	$5.38 \times 10^{-4}\ m^3$
L – Connector	$1.6 \times 10^{-5}\ m^3$
Bar Connector	$1.8 \times 10^{-6}\ m^3$
Others(wire bundle, cable tie and etc.)	$4.76 \times 10^{-4}\ m^3$
TOTAL	**$0.012176\ m^3$**

Table 2: Volume of critical items

*Weight of AUV due to buoyancy = (0.012176 m^3) x (1000Kg/m^3) = **12.18 Kg***

*Nett weight = (12.18 Kg) – (5.1 Kg) = **7.08 Kg** ← Positive buoyancy*

Since the '*Nett Weight*' should be zero to achieved neutral buoyancy, additional object weighing 7Kg was added to the AUV. Therefore, two bricks each weighing of 3.50Kg were added as a load and placed at the bottom of the AUV.

Figure 12: Bricks as a load

4.3.Thruster And Propeller

Six units of 1100 GPH Bilge Pump motor were used as thruster(two units) and balancer(four units) in this project. This DC brush motor has submersible capability and it has outstanding amount of RPM and torque which were the main reason it was selected. Besides, this motor operates at 12V, with 3.3A current. Huge modifications were needed on this motor before using it as it was designed for other purpose. The front white plastic casing was removed using hex saw as well as the blue strainer attached on the white casing.

Figure 13: 1100 GPH Bilge Pump before modification

Figure 14: 1100 GPH Bilge Pump after modification

For the propeller, the three blade(Left hand) boat propeller with dimension 40x57mm was selected. This propeller will provide a forward thrust when it rotates counter clockwise and the 57mm propeller's diameter will increase the force produced to move the body forward. A proper coupling was made using Polyethylene according to the dimensions of the propeller. To connect the custom made coupler with the propeller, the hardened epoxy was applied on the internal surface of the propeller as well as on the surface of the coupler. The result is shown in the figure below,

Figure 15:Propeller mounted on the coupler

4.4. Microcontroller Unit

Arduino Mega 2560 microcontroller was used in this project to control any input or output from sensors, motors, and other electronics devices. Arduino Mega 2560 is based on the Atmega 2560 processor and can be programmed using Arduino IDE software. The main consideration when choosing this microcontroller was due to its

outstanding amount of input/output pins that it offered. It has 54 digital input/output in which 15 pins are allocated as PWM outputs, 16 pins of analog input and 4 UARTs. This microcontroller also can provide a 3.3V output to power up any low powered device and 5V for standard operating powered device. The outstanding amount of pins offered more space for any electronic devices to communicate with this microcontroller and thus reserved for future development. The operating voltage for this microcontroller is 5V and it has 40mA output current for each input/output pins. Other than that, it has 256KB of flash memory as well as 16MHz clock speed.

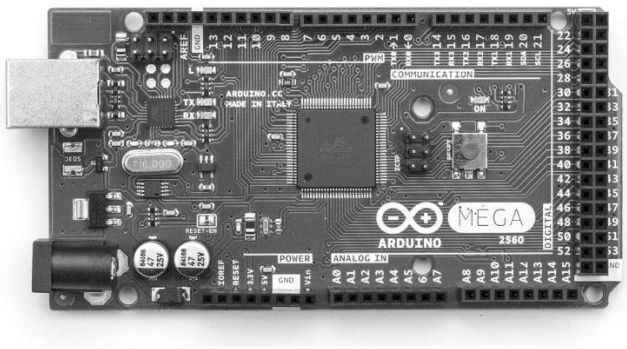

Figure 16: Arduino Mega 2560 microcontroller

In this project, the I^2C bus was used extensively for short distance communication between microcontroller and multiple slave devices, i.e., gyroscope, accelerometer, pressure sensors and etc. The I^2C device consisted of bidirectional SDA(data) and CLK(clock) generated by master device. The data will be send/received when the signal from the clock is LOW and sampled when the clock is HIGH. Even though I^2C bus able to communicate with multiple slave just by using two ports, only 8 data bits from each slaves is allowed to send for each clock turn, thus making all connected sensors activated in sequence. To avoid data complication, each connected slave hold unique address differ from each other.

Most of the I^2C devices operates on low voltages(3.3V) and requires 10K Ohm of pull-up resistor for data in/out reliabilility. However, Arduino Mega 2560 provides 5V of SDA and SCL pins which are not suitable for any required sensors. Connecting 5V of Arduino's SDA/SCL to 3.3V powered I^2C device could damage the sensors. Therefore, a logic shifter device was required. This bidirectional logic shifter act as a 5V to 3.3V converter and it has four bidirectional outputs. The 3.3V outputs then are used to connect GPS's TX/RX and SDA/SCL lines. Since this bidirectional logic

shifter comes with the internal 10K Ohm pull-up resistor, any additional external 10K Ohm pull-up resistor thus was not needed.

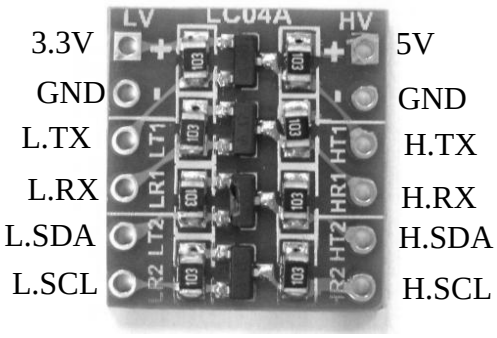

Figure 17: Bidirectional Logic Shifter

4.5. Gyroscope And Accelerometer

IMU Digital Combo Board – 6 Degrees of Freedom(3 DOF for Accelerometer, 3 DOF for Gyroscope) ITG3200/ADXL345 was selected in this project as its main electronics gyroscope and accelerometer sensor. The operating voltage for this sensor is 3.3V, and 6.5mA current which is compatible to be used together with Arduino Mega 2560 microcontroller. The ITG – 3200 gyroscope has approximately 2000 degree per second range of angular velocity that it can read and this breakout is convenient to use since it also utilized I^2C digital bus.

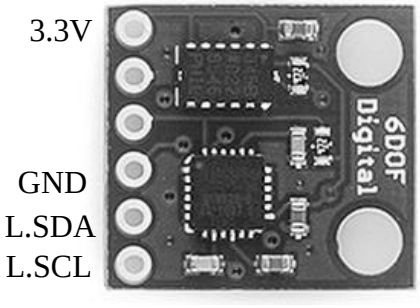

Figure 18: IMU Digital Combo Board – 6 Degrees of Freedom ITG3200/ADXL345

The accelerometer was used to determine AUV's acceleration and tilt, while the gyroscope was used to determine AUV's angular velocity and orientation. The integration of both accelerometer and gyroscope into one breakout board give us a complete understanding of the AUV's orientation, position and many more. This sensor works perfectly fine to measure any changes in 'Pitch' and 'Roll' angle of the AUV. However, we realized that the value of 'Yaw' was drifting over time. The

accelerometer was not able to measure this type of motion while the gyroscope able to measure it without the sense of direction, therefore a drift was produced over time. The amount of drift was depended on the initial sensor calibration during the code upload since the Arduino's library works to suppress the Yaw values at that particular of time. Thus, this breakout need to be remain still during the code upload until the calibration process completed.

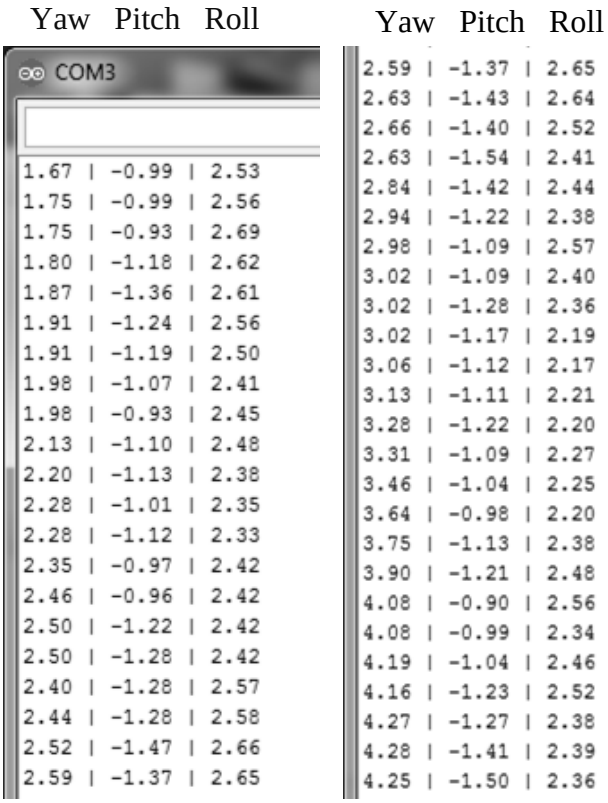

Figure 19: Returned 'Yaw', 'Pitch' and 'Roll' values

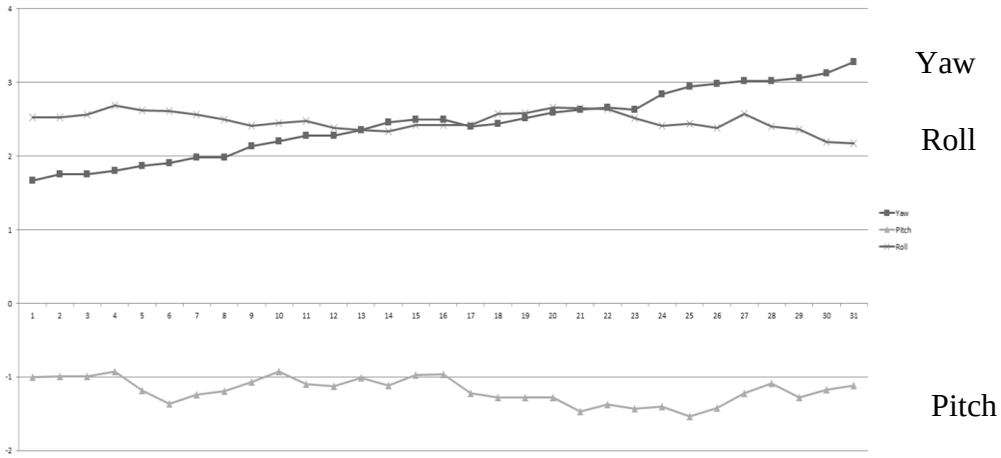

Figure 20: Increment of 'Yaw' over time versus 'Pitch' and 'Roll'

The figure above shows some simulation result of this IMU Digital Combo Board. The sensors was hold still during the calibration process yet, the value of 'Yaw' shows slight increment over time while the 'Pitch' and 'Roll' shows relatively stable at zero decimal point. To completely remove the 'Yaw' drifting, a magnetometer was required to compensate 'Yaw' angle variation. However, since the 'Yaw' angle plays minor role in this project, this issue was neglected to focus for other crucial issues. The accuracy of the returned value can be further improved by averaging the values for few iterations.

4.6. Pressure Sensor

MS5803-05BA Miniature Altimeter and Diving Module was selected in this project as underwater pressure sensor. This sensor offered a high precision reading down to 0.036 mbar, low – power consumption sensor and has measuring range up to 5 bar underwater. Additionally, this sensor also able to measure underwater temperature with 2.5 degrees Celsius of accuracy. It has flexible operating voltage, range from 1.8V to 3.6V which is compatible with the Arduino microcontroller board. Similar to Gyroscope and Accelerometer, this device used I^2C digital interface to be connected to the microcontroller.

Figure 21: MS5803-05BA Miniature Altimeter and Diving Module

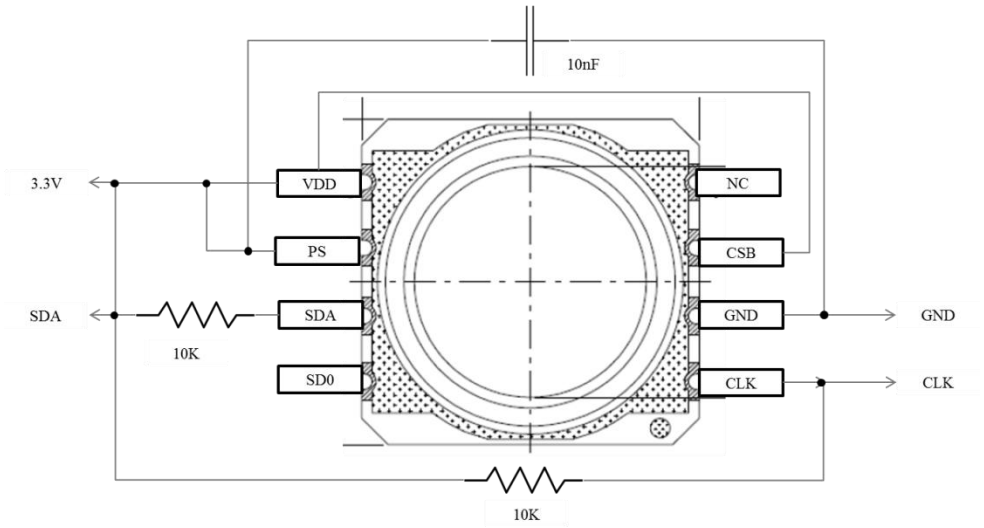

Figure 22: Wiring the MS5803-05BA

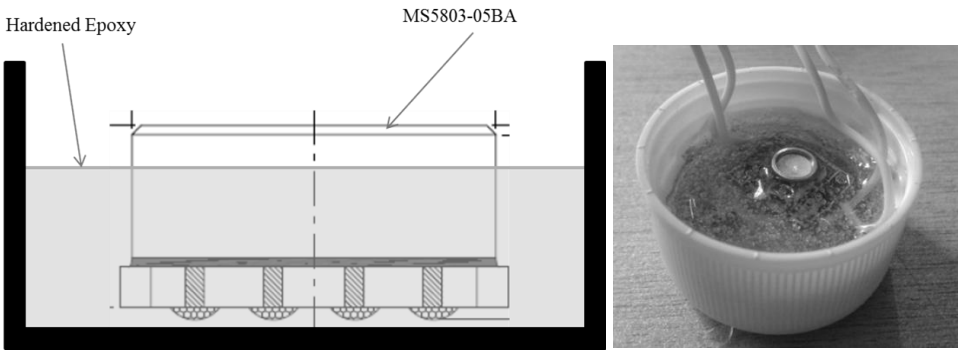

Figure 23: Waterproofing MS5803-05BA

Even though MS5803-05BA pressure sensor able to operates underwater, the soldering joint need to be protected from contact with water to avoid short circuit. Therefore, a hardened epoxy was required to secure the connection.

The depth of AUV in water can be measured by means of pressure difference between pressure measured on the surface and pressure measured in the water. Later on, the difference in pressure is converted into meters as shown in the codes below.

```
float diff = pressure1 - pressure2;
float dbar = diff / 100.0;
float meters = dbar * 1.019716;
```

Based on the stated equations, the waterproofed sensors was submerged into the water and the readings were recorded.

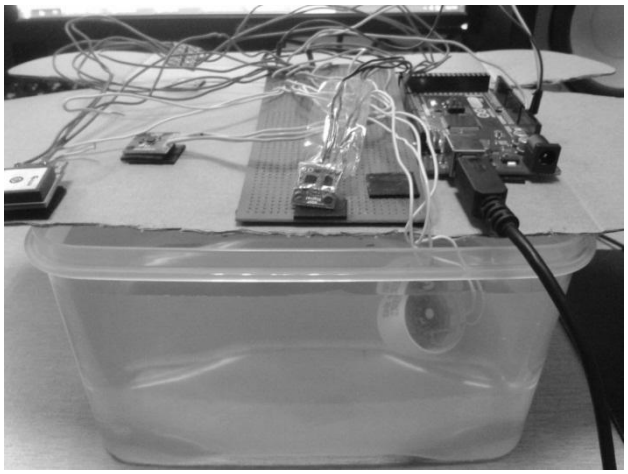

Figure 24:Submerging MS5803-05BA underwater

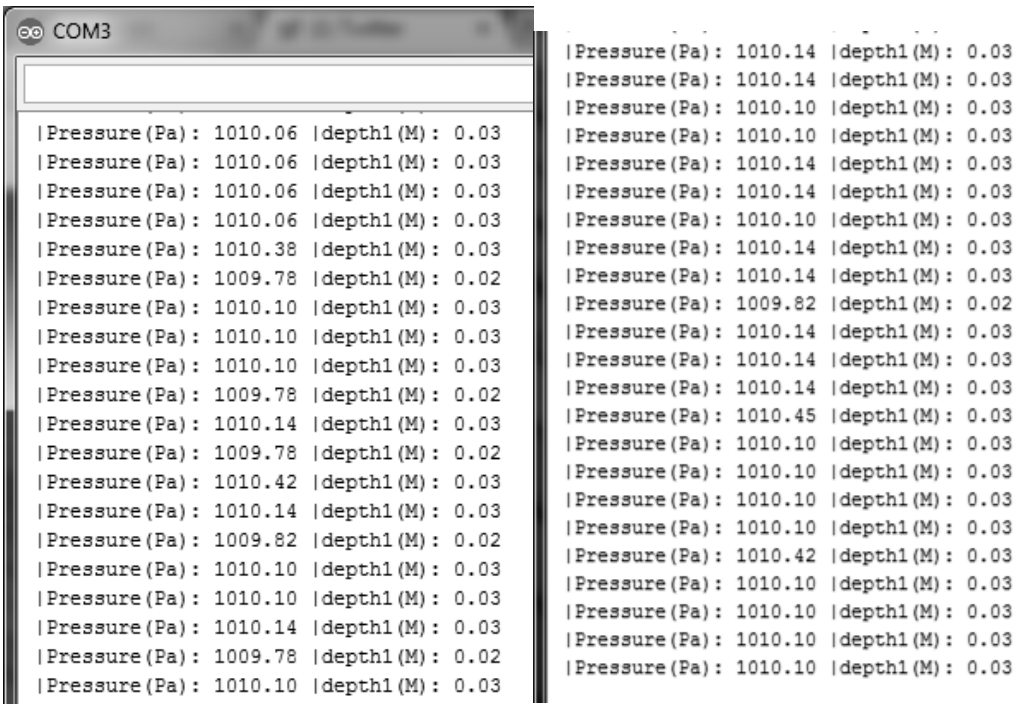

Figure 25: Simulation result for MS5803-05BA

The result of this experiment shows an outstanding precision of this sensors. Any slight changes of pressure underwater can be measured and translated into meters. The result of this experiment was further confirmed by measuring the depth using ruler and compared it with the result shown in the Serial Monitor.

4.7. Global Positioning System

The SkyNav SKM53 GPS Module was selected for the purpose of navigation system via satellites. It is known for its ultra-high tracking sensitivity(-165dBm) suitable to

use at urban or foliage area where GPS coverage is hard to reach. This low powered consumptions device(5V) uses National Marine Electronics Association, NMEA protocols with 9600 bps as default baud rate. This GPS module is relatively cheaper compare to other model in the market due to the slow navigation data update rate, which is only 1Hz.

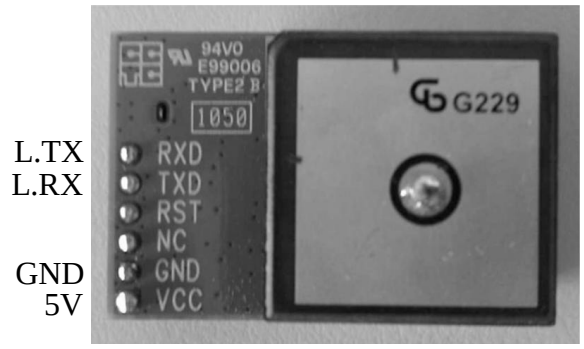

Figure 26: SkyNav SKM53 GPS Module

To communicate with SkyNav SKM53 GPS Module, the RX of the GPS module was connected to the LOW TX of the Arduino, while the TX was connected to the LOW RX of the Arduino since the GPS's TX/RX only works at 3.3V level. Upon successfully connected the GPS module, the GPS will return NMEA-0183 messages consisted of GGA, GLL, GSA, GSV, RMC, VTG, ZDA and DTM.

NMEA Sentences	Description
GGA	Global positioning system fixed data
GLL	Geographic position latitude/longitude
GSA	GNSS DOP and active satellites
GSV	GNSS satellites in view
RMC	Recommended minimum specific GNSS data
VTG	Course over ground and ground speed
ZDA	Date and Time
DTM	Datum reference

Table 3: Description of NMEA-0183 sentences

Afterwards, the acquired NMEA-0183 sentences were passed to the Arduino's "TinyGPS.h" library to be extracted and processed to obtain desired output such as latitude, longitude, geographical altitude, date, time and number of satellites locked.

The acquired GPS module's latitude and longitude on the specific point has been compared with the Google's Map latitude/longitude and latitude/longitude acquired by conventional smartphone. The result shows outstanding precision of this GPS module alongside with other GPS acquisition devices.

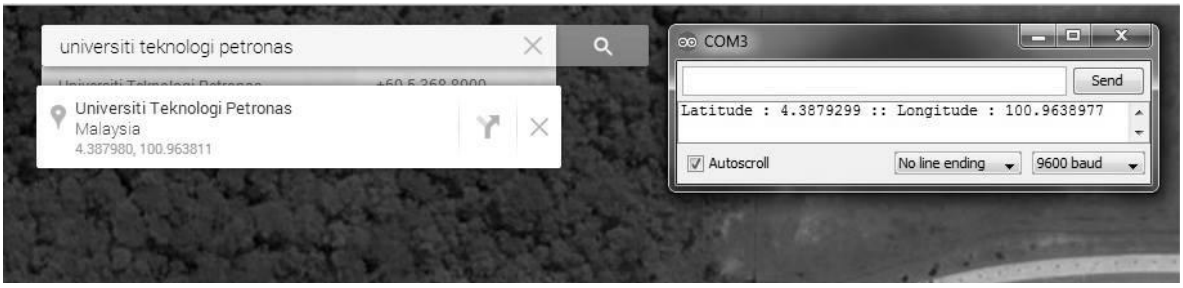

Figure 27: SkyNav SKM53 GPS Module VS Google Map's latitude/longitude

Figure 28: Smartphone's latitude/longitude

The AUV will move to the allocated coordinate according to the distance calculated between point A to B and it will stop when the AUV is positioned within considerable amount of range to the target point. To calculate the distance between two coordinates, the codes below which utilized basic trigonometric to measure hypotenus has been used:

```
float distance=sqrt((((LON)-(LON2))*((LON)-(LON2)))+(((LAT2-LAT)*(LAT2-
LAT))));

distance*=110567;
```

The experiment to test the above equation was conducted by connecting GPS module to the Arduino which was powered via mobile laptop. The experiment requires the

subject to move from a point to another point together with the GPS module at the open area between two buildings. The result of the experiment is shown on the figure below.

```
COM6
Latitude : 4.3880400 :: Longitude : 100.9641113 :: Distance : 11.57  Satellites: 9
Latitude : 4.3880400 :: Longitude : 100.9641113 :: Distance : 11.57  Satellites: 9
Latitude : 4.3880300 :: Longitude : 100.9641189 :: Distance : 12.94  Satellites: 9
Latitude : 4.3880400 :: Longitude : 100.9641189 :: Distance : 11.96  Satellites: 9
Latitude : 4.3880400 :: Longitude : 100.9641189 :: Distance : 11.96  Satellites: 9
Latitude : 4.3880500 :: Longitude : 100.9641113 :: Distance : 10.58  Satellites: 9
Latitude : 4.3880600 :: Longitude : 100.9641113 :: Distance : 9.61  Satellites: 9
Latitude : 4.3880600 :: Longitude : 100.9641036 :: Distance : 9.20  Satellites: 9
Latitude : 4.3880701 :: Longitude : 100.9641036 :: Distance : 8.22  Satellites: 9
Latitude : 4.3880801 :: Longitude : 100.9641036 :: Distance : 7.29  Satellites: 9
Latitude : 4.3880801 :: Longitude : 100.9640884 :: Distance : 6.46  Satellites: 9
Latitude : 4.3880901 :: Longitude : 100.9640884 :: Distance : 5.45  Satellites: 9
Latitude : 4.3880901 :: Longitude : 100.9640884 :: Distance : 5.45  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
Latitude : 4.3881001 :: Longitude : 100.9640808 :: Distance : 4.08  Satellites: 9
```

Figure 29:Measuring distance between two points

It was found that, even though GPS module able to lock with outstanding numbers of satellites, the captured GPS location while moving shows some data variation. The satellites does not recognized a very slow movement and it will consider the GPS module to be in the same original position. Therefore, to allow a good response from the satellites, some speed need to be put while moving from point A to B. Other than that, upon arriving at point B, the distance calculated does not return zero as it supposed due to the factors such as signal interference or bad satellites triangulation. To compensate this problem, a tolerable amount of range from the target point should be consider in the program.

4.8. Magnetometer

To measure AUV's heading angle from the original position to the final position, the Triple Axis Magnetometer Breakout - HMC5883L was selected. This 3.3V powered magnetic sensor communicates via Arduino's I^2C bus and it has one to two degrees of compass heading accuracy.

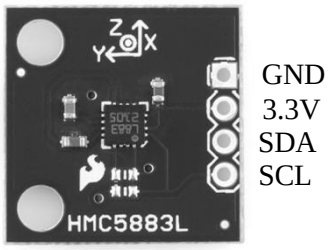

Figure 30: Triple Axis Magnetometer Breakout - HMC5883L

The principle operation of this magnetic sensor is that it is always referring to the earth's magnetic north instead of earth's true north. Earth's magnetic north is the point where the earth's magnetic field is vertically downward and the point varies over time due to the changes of earth's magnetic core. Meanwhile, the earth's true north is referring to the earth's North Pole. Due to this referral distinction, magnetic declination occurs. In order to obtain a precise heading angle, the local magnetic declination angle has been taking into account during angle calculation.

To measure heading angle with respect to the final coordinate, the below piece of codes which utilized basic trigonometric between two points has been used,

```
float angle_calc=atan2((LON1-LON2),(LAT1-LAT2));

    float declinationAngle2 = 0.2333333;
    angle_calc += declinationAngle2;

    if(angle_calc < 0){
      angle_calc = angle_calc + 360;
    }

    if(angle_calc >360){
      angle_calc= angle_calc - 360;
    }

    float angleDegrees = angle_calc;
    return angleDegrees;
```

The results of using this piece of code is shown in the figure below:

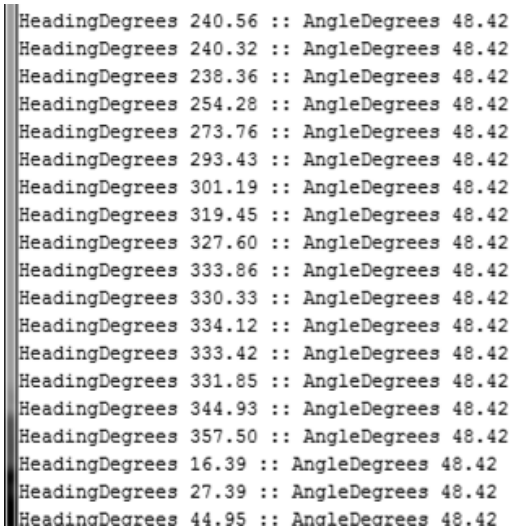

Figure 31: Heading Degrees VS Angle Degrees of HMC5883L magnetometer

Note that the 'HeadingDegrees' above was referred to the returned value by the HMC5883L magnetometer while 'AngleDegrees' was referred to the angle between point A and B. In this experiment, the initial heading degree was 240.056 degrees and upon rotating the magnetometer, the heading degrees keep increasing until it reached maximum 360 degrees and returned to zero degrees. After it reached approximately 48 degrees, the simulation will stop to indicate that it already arrived to the correct heading angle. This experiment also observed that any object-producing magnetic field, such as DC motors, and batteries could contribute to the increment of declination angle. Therefore, this magnetic sensor should be placed at the area which safe from external magnetic field.

4.9. Motor Driver

MD30A Motor Driver was selected in this project to drive single motors at once. In this project, six sets of motor driver were required to drive all AUV's motor. The motor driver helps to supply required voltages to drive the motor as well as to control the speed and rotation of the motor based on the PWM produced by the microcontroller. It uses 12V operating supply voltage, and huge 30A rated output current for continuous operations, thanks to the fan heat sink for thermal release. The huge capacity of the output current helps to compensate the load allocated on the shaft of the DC motor.

Even though the rated current of the DC motor is only about 3A, the current supplied to the motor will increase once it is submerged underwater and once the propeller is mounted on the shaft. Therefore, a motor driver that can compensate the load/current requirement is highly recommended.

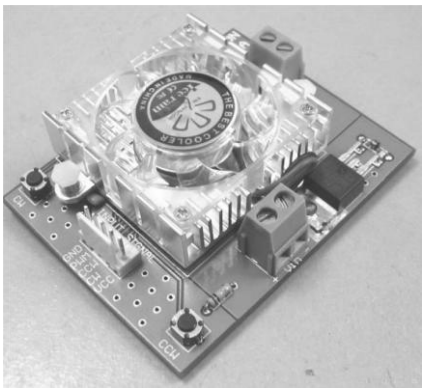

Figure 32: *MD30A Motor Driver*

To control the rotation of the motor, the below truth table has to be followed,

CCW	CW	Description
1	1	Brake to Vin
1	0	Clockwise rotation
0	1	Counter Clockwise rotation
0	0	Brake to Gnd

Table 4: Rotation truth table

4.10. Battery

For this project LiPo Rechargeable Battery 11.1V 2200mAh was selected. Three sets of 11.1V, 2200mAh, LiPo Rechargeable Battery were used in parallel to power up DC motor by supplying voltages to all motor driver as well as to power up Arduino microcontroller. Arduino microcontroller is recommended to be supplied with 7-12V for best performance.

Figure 33: LiPo Rechargeable Battery

4.11. Circuit Schematic

All electronic devices mentioned previously were connected as figure below:

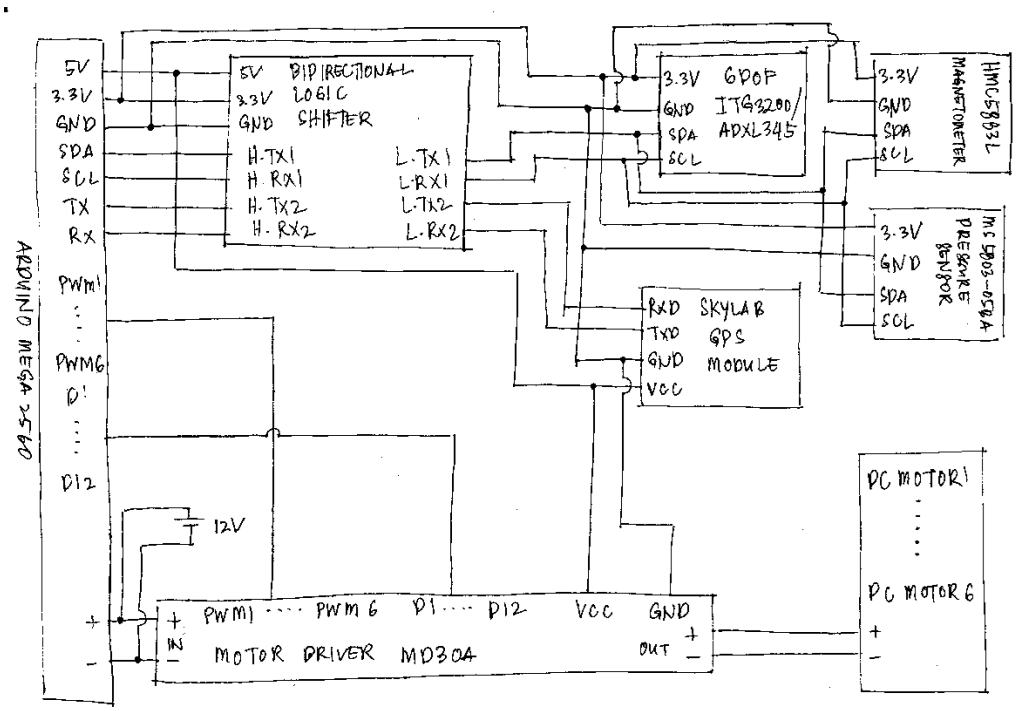

Figure 34: Circuit Schematic

4.12. Operational Flow Chart

The whole process of the operation has been written in the Arduino IDE. This flow chart represent the simplified flow of the codes written.

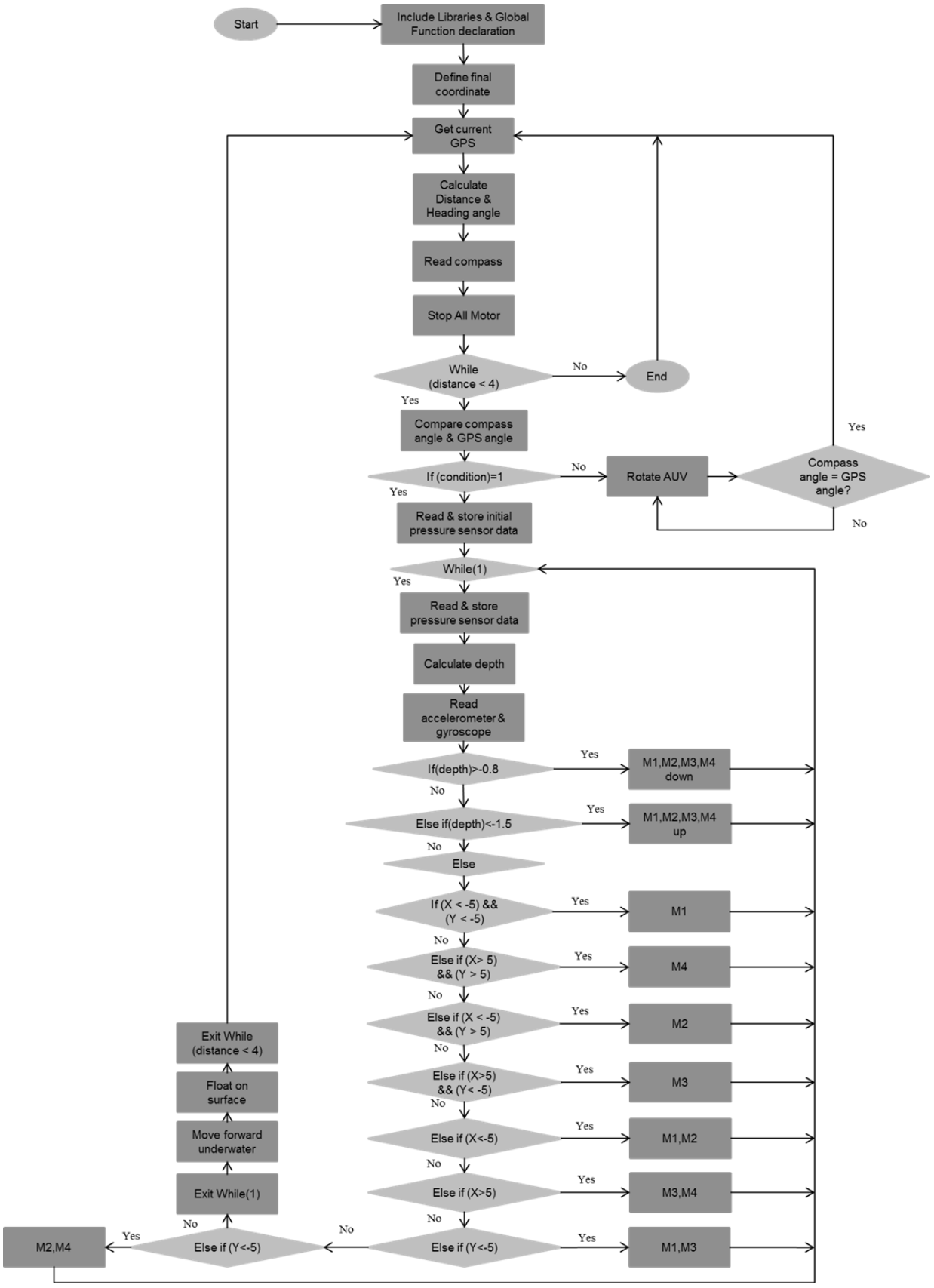

Figure 35: Arduino's Flow Program

4.13. AUV Deployment

Finally, AUV has been tested with all the circuit and program running onboard in the freshwater. Several physical enhancement and codes adjustment has been made to ensure successfulness of the project. The physical enhancement includes adding hardened epoxy layers on critical areas to avoid water goes in to the electronic compartment. Meanwhile, few adjustment to the Arduino codes involved modifying motor speed, motor ON/OFF interval, as well as stating desired depth and final coordinate. The results of these experiments were fairly satisfactory with few tolerance – AUV's final position slightly off few centimeters from desired coordinate. Overall, the result was in parallel with the objectives of this project.

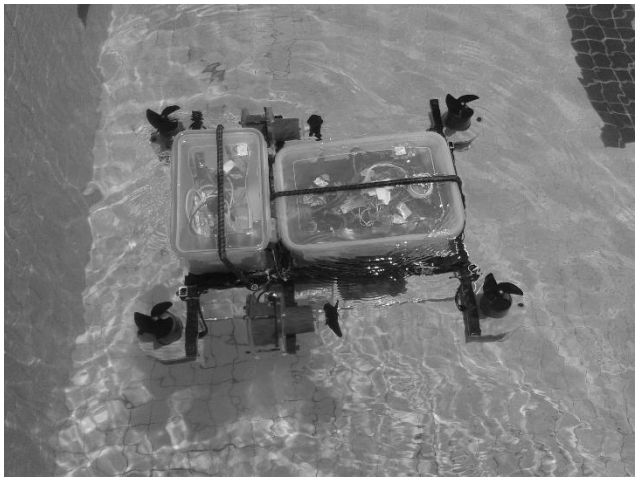

Figure 36: AUV deployment underwater

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1. Conclusion

The whole designing, fabrication, and coding process of the AUV has been achieved accordingly in parallel with the stated Gantt Chart. The main objectives of this project, which are to solve underwater orientation issue, depth leveling issue, as well as underwater navigation issues were solved and the results were fairly satisfactory. The objectives were achieved with the right use of electronic hardware and sensors, flow of codes and other mechanical aspect such as buoyancy and AUV design.

5.2. Recommendation

5.2.1. Automated Ballast System

For future development, it is recommended to provide an additional automated ballast system that operate during emergency. This is very crucial feature of any underwater vehicle as it will help to retrieve AUV in case it went missing underwater. The automated ballast system consisted of container filled with liquid that has the same density with the fluid surrounding. Upon emergency, the ballast system will automatically eject liquids it contained and fill it with air. Therefore, the body will experience positive buoyancy which will help the whole body to float on the surface of water.

5.2.2. PID Control System

A proportional-integral-derivative (PID) controller is the most suitable control system available to handle AUV's operation, yet, far more complex than the current one. The current control system relies much on the ranges to target which in return cause inaccuracy. The use of PID controller will eliminate this range dependency to target whilst focused on specific target making the result more precise.

5.2.3. Leakage Detector

One of the biggest challenge in this project is to maintain the internal circuitry from any contact with water from outside container. The internal part of container was

connected with bunch of external wire which are exposed to the water. Therefore, any unsealed gap between wires will cause catastrophic damages to the internal circuitry. Therefore, leakage sensors should present in the critical part of the AUV so that, any preventive measure can be taken if there is any leakage detected.

5.2.4. Waterproof Container

In this project, a household containers were selected and transformed into waterproof electronics container. However, this practice is highly not recommended to be use in challenging environment such as deep water exploration in sea. The current household waterproof container was made of plastic and it was not designed for underwater purpose. Increase in underwater depth will increase the amount of pressure and as a result will expose the container to leakage. Therefore, a standard waterproof container that can withstand high pressure should be use for future development.

REFERENCES

[1]. Autonomous Underwater Vehicle [Online]. Available: http://en.wikipedia.org/wiki/Autonomous_underwater_vehicle.

[2]. D. Bingham, & T. Drake, "The Application of Autonomous Underwater Vehicle (AUV) Technology in the Oil Industry – Vision and Experiences", FIG XXII International Congress Washington, D.C. USA, April 19 – 26, 2002.

[3]. J. Busquets, J. V. Busquets, D. Tudela, F. Perez, J. Busquets-Carbonell, A. Barbera, C. Rodriguez, A. J. Garcia, & J. Gilabert, "Low-cost AUV based on Arduino open source microcontroller board for oceanographic research applications in a collaborative long term deployment missions and suitable for combining with an USV as autonomous automatic recharging platform," *Autonomous Underwater Vehicles (AUV), 2012 IEEE/OES* , vol., no., pp.1,10, 24-27 Sept. 2012.

[4]. J. Luan, "*Development of a Small Sonar Altimeter and Constant Altitude Controller for a Miniature Autonomous Underwater Vehicle,"* doctoral dissertation, Virginia Polytechnic Institute and State University, 2005.

[5]. D. McArthur, "Sonar Obstacle Detection System for Underwater ROVs".

[6]. Accelerometers[Online].Available: http://www.hobbytronics.co.uk/accelerometer-info

[7]. F. Brandon, B. Ryan, H. Joseph, L. Raymond, "DRONENET: The Quad Chronicles."

[8]. M. Mat Ali, "*Development of Self Balancing Platform On Mobile Robot Using PID Controller*," doctoral dissertation, Universiti Tun Hussein Onn Malaysia, 2013.

[9]. J. Howse, "The ROV Pontus - A winning design," *Electrical and Computer Engineering, 2009. CCECE '09. Canadian Conference on* , vol., no., pp.346,349, 3-6 May 2009.

[10]. B. Jalving, "Depth accuracy in seabed mapping with underwater vehicles," *OCEANS '99 MTS/IEEE. Riding the Crest into the 21st Century* , vol.2, no., pp.973,978 vol.2, 1999.

[11]. T. N. Harsamizadeh Tehrani, M. Heidari, Y. Zakeri, & J. Ghaisari, "Development, depth control and stability analysis of an underwater

Remotely Operated Vehicle (ROV)," *Control and Automation (ICCA), 2010 8th IEEE International Conference on* , vol., no., pp.814,819, 9-11 June 2010.

[12]. Archimede's Principle [Online]. Available: http://en.wikipedia.org/wiki/Archimedes'_principle

[13]. Buoyancy of Rigid Structures [Online]. Available: http://www.asis.com/users/jknope/ROV/ROVBuoyancy/ROVBuoyancy.html

[14]. C. E. Charles, O. James, D. L. Russell, W. Timothy, W. L. Thomas, L. S. Peter, W. John, & M. C. Andrew, "Seaglider: A Long-Range Autonomous Underwater Vehicle for Oceanographic Research," *IEEE Journal Of Oceanic Engineering, Vol. 26, No. 4, October 2001.*

APPENDICES

APPENDIX A - Arduino Codes

1. Function And Paramaters Declaration

```
#include <Wire.h>
#include <TinyGPS.h>
#include <FreeSixIMU.h>
#include <FIMU_ADXL345.h>
#include <FIMU_ITG3200.h>
#include <MS5803_05.h>
#include <HMC5883L.h>
#include <math.h>

#define rxPin 19 // RX1/TX1
#define txPin 18

MS_5803 sensor = MS_5803(4096); //set mode depth sensor
HMC5883L compass;

unsigned long fix_age;
TinyGPS gps;
void gpsdump(TinyGPS &gps);
static bool feedgps();
void getGPS();
float lat,lon;
//long lat, lon;
float LAT, LON;

int error = 0;
float angles[2];
double temperaturel;
double pressurel;

float x2lat= 4.387035;
float x2lon= 100.974359;


int E1 = 7;
int M1 = 24;
int M11 = 26;
int E2 = 6;
int M2 = 28;
int M22 = 30;
int E3 = 5;
int M3 = 32;
int M33 = 34;
int E4 = 4;
int M4 = 36;
int M44 = 38;
int E5 = 3;
int M5 = 40;
int M55 = 42;
int E6 = 2;
int M6 = 44;
int M66 = 46;

FreeSixIMU sixDOF = FreeSixIMU();
```

2. Program Setup

```
void setup() {

  pinMode(M1, OUTPUT); pinMode(M11, OUTPUT);
  pinMode(M2, OUTPUT); pinMode(M22, OUTPUT);
  pinMode(M3, OUTPUT); pinMode(M33, OUTPUT);
  pinMode(M4, OUTPUT); pinMode(M44, OUTPUT);
  pinMode(M5, OUTPUT); pinMode(M55, OUTPUT);
  pinMode(M6, OUTPUT); pinMode(M66, OUTPUT);

  Serial.begin(9600); //115200/9600
  Serial1.begin(9600); //begin serial comm for GPS
  Serial.println("Starting the I2C interface.");
  Wire.begin();


  Serial.println("Constructing new HMC5883L");
  compass = HMC5883L(); // Construct a new HMC5883 compass.

  //Serial.println("Setting scale to +/- 1.3 Ga");
  error = compass.SetScale(0.88); // Set the scale of the compass.
  //if(error != 0) { // If there is an error, print it out.
   // Serial.println(compass.GetErrorText(error));
  //}
  //Serial.println("Setting measurement mode to continous.");
  error = compass.SetMeasurementMode(Measurement_Continuous); // Set the measurement mode to Continuous
  //if(error != 0) {// If there is an error, print it out.
    //Serial.println(compass.GetErrorText(error));
  //}
  delay(500);

  Serial.println("MS5803 CRC check.........");
  if (sensor.initializeMS_5803(false)) {
    Serial.println( "MS5803 CRC check OK." );
  }
  else {
    Serial.println( "MS5803 CRC check FAILED!" );
  }
  delay(500);

  Serial.println("IMU Initialization.........");
  sixDOF.init(); //begin the IMU
  delay(500);
}
```

3. Main Loop

```
void loop() {
  // ----------------------------GPS Setup-----------------------------------
  feedgps();
  float lat,lon;
  //long lat, lon;
  unsigned long fix_age, time, date, speed, course;
  unsigned long chars;
  unsigned short sentences, failed_checksum;

  gps.f_get_position(&lat, &lon, &fix_age);   //added f_get
  getGPS(); //get current Lat/Lon
  feedgps();
  // -------------------------------Calculate distance & angle from point A to point B --------------------
  float distance3=sqrt((((LON)-(x2lon))*((LON)-(x2lon)))+(((x2lat-LAT)*(x2lat-LAT))));
  distance3*=110567 ;
  float angleDegrees1 = angleDegrees(LAT, LON, x2lat, x2lon);
  feedgps();
  //-----------------------------Compass Setup--------------------------------
  float headingDegrees1 = CompassSetup();
  Output(headingDegrees1, angleDegrees1, distance3, LAT, LON); //call function
 // ------------------------------------------END OF COMPASS-------------------------------------
 feedgps();
 StopM5M6();
 StopAll();

 while(distance3 > 5){
 //-----------------------------------------------take action-----------------------------------
 feedgps();
 bool condition = (((angleDegrees1 - 10) < headingDegrees1) && ((angleDegrees1 + 10) > headingDegrees1));

 if(condition){ //condition == true?
  sensor.readSensor();
  temperature1 = sensor.temperature();
  pressure1 = sensor.pressure();
  float PaStart = pressure1;
```

```
while(1){

  feedgps();
  sensor.readSensor();
  temperature1 = sensor.temperature();
  pressure1 = sensor.pressure();
  ////////////////////////////////////Webbot Depth Method(1)////////////////////////////////

  double diff = pressure1 - PaStart;
  double dbar = diff / 100.0;
  double meters = dbar * 1.019716;
  ////////////////////////////////////////////////////////////////////////////////////////////
  sixDOF.getYawPitchRoll(angles);

  Serial.print(" |Pitch: ");
  Serial.print(angles[1],0);
  Serial.print(" |Roll: ");
  Serial.print(angles[2],0);
  Serial.print(" |Pressure(Pa): ");
  Serial.print(pressure1); //whole number only.
  Serial.print(" |depth(M): ");
  Serial.println(meters);//display 2 decimal places
  feedgps();

    if(meters < 0.4){  //variable dependencies on location & temperature //0.8
    AllMoveDown();
    feedgps();
      for (int value = 0 ; value <= 100; value+=5){
      StopAll();
      StopM5M6();
      delay(50);//130
    }
   }


    else if(meters > 0.6){ //1.5
     AllMoveUp();
    feedgps();
     for (int value = 0 ; value <= 100; value+=5){
     StopAll();
     StopM5M6();
     delay(50);
     }
    }

    else {
        if((angles[1] < -25) && (angles[2] < -25)) {//prev. val. 5 degree
         ActiveM1();
         feedgps();
         for (int value = 0 ; value <= 100; value+=5){
           StopAll();
           StopM5M6();
           delay(50);
         }
        }

        else if((angles[1] > 25) && (angles[2] > 25)) {
         ActiveM4();
         feedgps();
          for (int value = 0 ; value <= 100; value+=5){
           StopAll();
           StopM5M6();
           delay(50);
         }
        }
```

```
 else if((angles[1] < -25) && (angles[2] > 25)) {
   ActiveM2();
  feedgps();
    for (int value = 0 ; value <= 100; value+=5){
    StopAll();
    StopM5M6();
    delay(50);
  }
 }

 else if((angles[1] > 25) && (angles[2] < -25)) {
   ActiveM3();
  feedgps();
    for (int value = 0 ; value <= 100; value+=5){
    StopAll();
    StopM5M6();
    delay(50);
  }
 }

 else if(angles[1] < -25) {
   ActiveM1M2();
   feedgps();
    for (int value = 0 ; value <= 100; value+=5){
    StopAll();
    StopM5M6();
    delay(50);
  }
}

 else if(angles[1] > 25){
   ActiveM3M4();
   feedgps();
    for (int value = 0 ; value <= 100; value+=5){
    StopAll();
    StopM5M6();
    delay(50);
  }
}

 else if(angles[2] < -25){
  ActiveM1M3();
  feedgps();
   for (int value = 0 ; value <= 100; value+=5){
    StopAll();
    StopM5M6();
    delay(50);
  }
}

 else if(angles[2] > 25){
   ActiveM2M4();
   feedgps();
    for (int value = 0 ; value <= 100; value+=5){
    StopAll();
    StopM5M6();
    delay(50);
  }
}
```

```
      else {
        feedgps();
        for (int value = 0 ; value <= 100; value+=5){
         StopAll();
         StopM5M6();
         delay(50);
        } //Stop all diagonal motor
        break; //get out from prev While Loop
      }
    }
  }

  Serial.println("Forward.....................");
  feedgps();
  Forward();
////////////////////////////////repetitive/////////////////////////////
  sensor.readSensor(); //read depth sensor
  temperature1 = sensor.temperature();
  pressure1 = sensor.pressure();

  double diff3 = pressure1 - PaStart;
  double dbar3 = diff3 / 100.0;
  double meters3 = dbar3 * 1.019716;

  while(meters3 > 0.3){
  Serial.println("All Move UP.....................");
  Serial.print("meters3:");
  Serial.println(meters3,2);
  AllMoveUp();
  feedgps();
  for (int value = 0 ; value <= 100; value+=5){
    StopAll();
    StopM5M6();
    delay(50);
  }
```

```
    sensor.readSensor(); //read depth sensor
    temperature1 = sensor.temperature();
    pressure1 = sensor.pressure();

    double diff3 = pressure1 - PaStart;
    double dbar3 = diff3 / 100.0;
    double meters3 = dbar3 * 1.019716;
  }
  break; //exit main distance3 < 1 main while loop & recheck GPS coordinate.
 }

  else {
    while(((angleDegrees1 -10) > headingDegrees1) || ((angleDegrees1 +10) < headingDegrees1)){
    feedgps();
    Search();
    sixDOF.getYawPitchRoll(angles);
    feedgps();

    headingDegrees1 = CompassSetup();
    feedgps();
    Serial.print("HeadingDegrees ");
    Serial.print(headingDegrees1);
    Serial.print(" :: AngleDegrees ");
    Serial.println(angleDegrees1);

    for (int value = 0 ; value <= 100; value+=5){
    StopAll();
    StopM5M6();
    delay(120);
      }
    }      // After end while loop, return to the main While loop and recheck boolean condition.
}
 }
} ////////////////////////END OF PROGRAM/////////////////////////////////////////////////
```

4. Functions

```
float CompassSetup()
{
  // Retrive the raw values from the compass (not scaled).
  MagnetometerRaw raw = compass.ReadRawAxis();
  feedgps();
  // Retrived the scaled values from the compass (scaled to the configured scale).
  MagnetometerScaled scaled = compass.ReadScaledAxis();
  feedgps();
  // Values are accessed like so:
  int MilliGauss_OnThe_XAxis = scaled.XAxis;// (or YAxis, or ZAxis)

  // Calculate heading when the magnetometer is level, then correct for signs of axis.
  float heading = atan2(scaled.YAxis, scaled.XAxis);
  feedgps();

  float declinationAngle = 0.004; //change this based on the Local Magnetic Declination in Rad
  heading += declinationAngle;

  // Correct for when signs are reversed.
  if(heading < 0){
    heading += 2*PI;
  }
  // Check for wrap due to addition of declination.
  if(heading > 2*PI){
    heading -= 2*PI;
  }
  // Convert radians to degrees for readability.
  float headingDegrees = heading * 180/M_PI; //handle by Math.h..same as PI
  return headingDegrees;
}
```

```
void getGPS()
{
  bool newdata = false;
  unsigned long start = millis();
  while (millis() - start < 1000)
      {
        if (feedgps ()){
        newdata = true;
        }
      }
        if (newdata){
        gpsdump(gps);
        }
}
static bool feedgps()
{
  while (Serial1.available())
    {
      if (gps.encode(Serial1.read()))
       return true;
    }
   return 0;
}

void gpsdump(TinyGPS &gps)
{
   //byte month, day, hour, minute, second, hundredths;
  gps.f_get_position(&lat, &lon);
  LAT = lat;
  LON = lon;
    {
      feedgps();
     }
}

void Output(float headingDegrees1, float angleDegrees1, float distance3, float LAT, float LON)
{
  Serial.print("HeadingDegrees ");
  Serial.print(headingDegrees1);
  Serial.print(" :: AngleDegrees ");
  Serial.print(angleDegrees1);
  Serial.print(" :: Distance ");
  Serial.print(distance3);
  Serial.print(" :: Latitude : ");
  Serial.print(LAT,7);
  Serial.print(" :: Longitude : ");
  Serial.print(LON,7);
  Serial.print("  Satellites: ");
  Serial.println(gps.satellites());
}

void AllMoveDown()
{
  for(int value = 0 ; value <= 255; value+=255){
      digitalWrite(M1,HIGH); digitalWrite(M11,LOW);
      analogWrite(E1,value);
      digitalWrite(M2,HIGH); digitalWrite(M22,LOW);
      analogWrite(E2,value);
      digitalWrite(M3,HIGH); digitalWrite(M33,LOW);
      analogWrite(E3,value);
      digitalWrite(M4,HIGH); digitalWrite(M44,LOW);
      analogWrite(E4,value);
      delay(200);
  }
}
```

```
void AllMoveUp()//move down reality
{
  for(int value = 0 ; value <= 255; value+=255){
      digitalWrite(M1,LOW); digitalWrite(M11,HIGH);
      analogWrite(E1,value);
      digitalWrite(M2,LOW); digitalWrite(M22,HIGH);
      analogWrite(E2,value);
      digitalWrite(M3,LOW); digitalWrite(M33,HIGH);
      analogWrite(E3,value);
      digitalWrite(M4,LOW); digitalWrite(M44,HIGH);
      analogWrite(E4,value);
      delay(250);
  }
}

void ActiveM1()
{
  for(int value = 0 ; value <= 255; value+=255){
          digitalWrite(M4,HIGH); digitalWrite(M44,LOW);
          analogWrite(E4,value);
          digitalWrite(M2,LOW); digitalWrite(M22,LOW);
          analogWrite(E2,0);
          digitalWrite(M3,LOW); digitalWrite(M33,LOW);
          analogWrite(E3,0);
          digitalWrite(M1,LOW); digitalWrite(M11,HIGH);
          analogWrite(E1,value);
          delay(100);
  }
}


void ActiveM4()
{
  for(int value = 0 ; value <= 255; value+=255){
          digitalWrite(M4,LOW); digitalWrite(M44,HIGH);
          analogWrite(E4,value);
          digitalWrite(M2,LOW); digitalWrite(M22,LOW);
          analogWrite(E2,0);
          digitalWrite(M3,LOW); digitalWrite(M33,LOW);
          analogWrite(E3,0);
          digitalWrite(M1,HIGH); digitalWrite(M11,LOW);
          analogWrite(E1,value);
          delay(100);
  }
}

void ActiveM2()
{
  for(int value = 0 ; value <= 255; value+=255){
          digitalWrite(M1,LOW); digitalWrite(M11,LOW);
          analogWrite(E1,0);
          digitalWrite(M3,HIGH); digitalWrite(M33,LOW);
          analogWrite(E3,value);
          digitalWrite(M2,LOW); digitalWrite(M22,HIGH);
          analogWrite(E2,value);
          digitalWrite(M4,LOW); digitalWrite(M44,LOW);
          analogWrite(E4,0);
          delay(100);
  }
}
```

```
void ActiveM3()
{
  for(int value = 0 ; value <= 255; value+=255){
          digitalWrite(M1,LOW); digitalWrite(M11,LOW);
          analogWrite(E1,0);
          digitalWrite(M3,LOW); digitalWrite(M33,HIGH);
          analogWrite(E3,value);
          digitalWrite(M2,HIGH); digitalWrite(M22,LOW);
          analogWrite(E2,value);
          digitalWrite(M4,LOW); digitalWrite(M44,LOW);
          analogWrite(E4,0);
          delay(100);
  }
}

void ActiveM1M2()
{
  for(int value = 0 ; value <= 255; value+=255){
          digitalWrite(M3,HIGH); digitalWrite(M33,LOW);
          analogWrite(E3,value);
          digitalWrite(M4,HIGH); digitalWrite(M44,LOW);
          analogWrite(E4,value);
          digitalWrite(M1,LOW); digitalWrite(M11,HIGH);
          analogWrite(E1,value);
          digitalWrite(M2,LOW); digitalWrite(M22,HIGH);
          analogWrite(E2,value);
          delay(100);
  }
}
```

```
void ActiveM3M4()
{
  for(int value = 0 ; value <= 255; value+=255){
           digitalWrite(M3,LOW); digitalWrite(M33,HIGH);
           analogWrite(E3,value);
           digitalWrite(M4,LOW); digitalWrite(M44,HIGH);
           analogWrite(E4,value);
           digitalWrite(M1,HIGH); digitalWrite(M11,LOW);
           analogWrite(E1,value);
           digitalWrite(M2,HIGH); digitalWrite(M22,LOW);
           analogWrite(E2,value);
           delay(100);
  }
}

void ActiveM1M3()
{
  for(int value = 0 ; value <= 255; value+=255){
           digitalWrite(M2,HIGH); digitalWrite(M22,LOW);
           analogWrite(E2,value);
           digitalWrite(M4,HIGH); digitalWrite(M44,LOW);
           analogWrite(E4,value);
           digitalWrite(M1,LOW); digitalWrite(M11,HIGH);
           analogWrite(E1,value);
           digitalWrite(M3,LOW); digitalWrite(M33,HIGH);
           analogWrite(E3,value);
           delay(100);
  }
}
```

```
void ActiveM2M4()
{
  for(int value = 0 ; value <= 255; value+=255){
            digitalWrite(M2,LOW); digitalWrite(M22,HIGH);
            analogWrite(E2,value);
            digitalWrite(M4,LOW); digitalWrite(M44,HIGH);
            analogWrite(E4,value);
            digitalWrite(M1,HIGH); digitalWrite(M11,LOW);
            analogWrite(E1,value);
            digitalWrite(M3,HIGH); digitalWrite(M33,LOW);
            analogWrite(E3,value);
            delay(100);
  }
}

void StopAll()//M1M2M3M4
{
            digitalWrite(M1,LOW); digitalWrite(M11,LOW);
            analogWrite(E1,0);
            digitalWrite(M2,LOW); digitalWrite(M22,LOW);
            analogWrite(E2,0);
            digitalWrite(M3,LOW); digitalWrite(M33,LOW);
            analogWrite(E3,0);
            digitalWrite(M4,LOW); digitalWrite(M44,LOW);
            analogWrite(E4,0);
}

void StopM5M6()
{
            digitalWrite(M5,LOW); digitalWrite(M55,LOW);
            analogWrite(E5,0);
            digitalWrite(M6,LOW); digitalWrite(M66,LOW);
            analogWrite(E6,0);
}
```

```
void Search()
{
  for(int value = 0 ; value <= 255; value+=255){
            feedgps();
            digitalWrite(M5,LOW); digitalWrite(M55,HIGH);
            analogWrite(E5,value);
            digitalWrite(M6,HIGH); digitalWrite(M66,LOW);
            analogWrite(E6,value); //value is final speed desired
            delay(150); //control motor acceleration!!
  }
}

void Forward()
{
  for(int value = 0 ; value <= 255; value+=255){
            digitalWrite(M5,HIGH); digitalWrite(M55,LOW);
            analogWrite(E5,value);
            digitalWrite(M6,HIGH); digitalWrite(M66,LOW);
            analogWrite(E6,value);
            delay(20000); //20 sec
  }
}
```

```
float angleDegrees(float LAT, float LON, float x2lat, float x2lon){
  feedgps();
  float angle_calc=atan2((x2lon-LON),(x2lat-LAT));

   float declinationAngle2 = 0.2333333; //57.29577951; degree
   angle_calc += declinationAngle2;

   if(angle_calc < 0){
     angle_calc = angle_calc + 360;
   }

   if(angle_calc >360){
     angle_calc= angle_calc - 360;
   }

   float angleDegrees = angle_calc;
   return angleDegrees;
}
```

APPENDIX B - Key Milestone

Task	Week																											
	Final Year Project 1														Final Year Project 2													
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Project title selection																												
Extended proposal submission																												
Proposal defence																												
Finalized AUV design in Autodesk Inventor																												
Gathered all electronics equipments																												
Completed frame design																												
Draft report submission																												
Final report submission																												
Accel/Gyro experiment																												
Pressure experiment																												
GPS module experiment																												
Compass experiment																												
Coupler attachment																												
Complete Wiring																												
Progress Report																												
Prototype 1																												
Pre-Sedex																												
Draft Final Report																												
Dissertation & Final Report																												

Table 5: Key Milestone of Final Year Project 1 and 2

APPENDIX C - Gantt Chart

Task	Week																												
	Final Year Project 1														Final Year Project 2														
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Project title selection																													
Background study and literature reviews on the application of AUV and its common features																													
Identifying general issues of AUV during operation and the method to tackle the issues																													
Extended proposal																													
Study on the electronics requirement (sensors, microcontrollers, motor driver and etc.) to be integrate in the system																													
Requirement study on the mechanical specification, i.e., DC motor torque, RPM, base design and etc.																													
Proposal defense																													
Familiarize with 3D modeling software (Autodesk Inventor 2014) and design base robot																													
Fabricate AUV model based on the finalized 3D design																													

Study on the Arduino microprocessor level and its programming language preferences																													
Draft report																													
Final report																													
Develop a control system, robot programming & result analysis																													
Progress Report																													
Pre - Sedex																													
Submission of Draft Final Report																													
Submission of Dissertation (SB)																													
Submission of Technical Paper																													
Project Viva																													
Submission of Dissertation (HB)																													

Table 6: Gantt Chart of Final Year Project 1 and 2