DYNAMIC LEVELING CONTROL OF A WIRELESS SELF-BALANCING ROV USING FUZZY LOGIC CONTROLLER

MOHAMMAD AFIF BIN AYOB

A thesis submitted in
fulfilment of the requirement for the award of the
Masterof Electrical Engineering

Faculty of Electrical and Electronic Engineering
UniversitiTun Hussein Onn Malaysia

JANUARY 2013

ABSTRACT

A remotely operated vehicle (ROV) is essentially an underwater mobile robot that is controlled and powered by an operator outside of the robot working environment. Like any other marine vehicle, ROV has to be designed to float in the water where its mass is supported by the buoyancy forces due to the displacement of water by its hull. Vertically positioning a mini ROV in centimetres resolution underwater and maintaining that state requires a distinctive technique partly because of the pressure and buoyancy exerted by the water towards the hull and partly because of the random waves produced by the water itself. That being said, the aim of the project is to design and develop a wireless self-balancing buoyancy system of a mini ROV using fuzzy logic controller. A liquid level sensor has been implemented to provide feedback to the Arduino microcontroller. A user-friendly graphical user interface (GUI) has been developed for real-time data monitoring as well as controlling the vertical position of the ROV. At the end of the project, the implemented fuzzy control system shows enhanced and better performance when compared with one without a controller, a proportional-derivative (PD) controller, and a proportional-integral-derivative (PID) controller.

ABSTRAK

Sebuahkenderaan yang dikendalikandarijauh (ROV) padaasasnyaadalah robot mudahalih yang beroperasi di dalam air yang dikawaldandikuasaiolehpengendalidi luarpersekitarankerja robot.Sepertimanakenderaanmarinyang lain, ROV direkauntukterapung di dalam air dimanajisimnyadisokongolehdayakeapungan yang disebabkanolehanjakan air olehstrukturkenderaan. Menyelaraskedudukan ROV minisecaramenegakdalamresolusisentimeterdi dalamair danmengekalkankeadaanitumemerlukanteknikterse ndirisebahagiannyakeranatekanan dankeapungan yang dikenakanoleh air kearahbadankenderaandansebahagiannyakeranagelombangrawak yang dihasilkanoleh air itusendiri.Disebabkanhalini, tujuanprojekiniadalahuntukmerekabentukdanmembangunkankeseimbangansendirike apungansistemsebuah ROV mini menggunakanpengawallogikkabur.Sebuahpenderiacecairtelahdigunakanuntukmembe rimaklumbalaskepadamikropengawal Arduino.Satuantaramukamesrapenggunagrafik (GUI) telahdibangunkanuntukmemantaudata secaramasasebenarsertamengawalkedudukanmenegakdaripada ROV.Padaakhirprojek, sistemkawalankabur yang dilaksanakanmenunjukkanprestasiyang dipertingkatkandanlebihbaikapabiladibandingkandengansatutanpapengawal, pengawalberkadar-derivatif (PD), danpengawalberkadar-integrasi-derivatif (PID).

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AUV	Autonomous Underwater Vehicle
COG	Center of Gravity
DC	Direct Current
FAMM	Fuzzy Associative Memory Matrix
GND	Ground
GUI	Graphical User Interface
I/O	Input/Output
ICSP	In-Circuit Serial Programming
LED	light emitting diode
LQ-PID	Least Squares Prediction Fuzzy Compensated PID
LQR	Linear Quadratic Regular
LSM	Least Squares Method
MISO	Master In Slave Out
MOSI	Master Out Slave In
NM-FPID	Normal Fuzzy Compensated PID
PD	Proportional Derivative
PID	Proportional-Integral-Derivative
PVC	Polyvinyl Chloride
PWM	Pulse Width Modulation
RF	Radio Frequency
ROV	Remotely Operated Vehicle
RST	Reset
SCK	Serial Clock
USB	Universal Serial Bus
VBS	Variable Buoyancy Control System

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

A remotely operated vehicle (ROV) is essentially an underwater mobile robot that is controlled and powered by an operator outside of the robot working environment via an umbilical cable or a remote control. A ROV differs from autonomous underwater vehicle (AUV) in a way that ROV always take command from its operator and takes no action autonomously. The boundless functionality of modern ROVs have brought great impact to the society from operations in both offshore and inshore by commercial, government, military and academic users [1].

Like any other marine vehicle, ROV has to be designed to float in the water where its mass is supported by the positive buoyancy forces due to the displacement of water by its hull. The provision of special tankage is required for the transition from surface to submerge and to sustain the balance between mass and buoyancy while submerges. The changes in buoyancy occur corresponding to the deepness the vehicle travels [2] consequently making the state of equilibrium in depth tends to be unstable with the vessel at rest. Since changes in buoyancy occur with depth, vehicle structures, including the hull, lose displacement as they compress [3] and thus will affect its vehicle stability. Subsequently, a special control device would have to be provided for the vessel to stay at a particular level.

1.2 Problem statement

ROV control presents several complications because of the nonlinearities model uncertainties and the influence of the external surroundings. Vertically positioning a mini ROV in centimeters resolution underwater and maintaining that state requires a distinctive technique. The reason is because of the pressure and buoyance force exerted by the water towards the vessel and also because of the random waves produced by the water itself. The study and design of a self-leveling control system for a mini ROV is significantly important because of numerous applications that can take benefits from it. Such examples include subsea installations, inspecting the hazardous inside of nuclear power plants, object location and recovery, and repairing complex deep water production systems.

1.3 Aim and objectives

The aim of the project is to design and develop a wireless self-leveling buoyancy system of a mini ROV by using fuzzy logic controller that has the ability for precise depth control. The objectives of this project on the other hand are as follows:

(i) To improve a remotely controlled mini ROV by using a microcontroller with feedback sensor.
(ii) To develop a self-balancing buoyancy system for a mini ROV based on a liquid level sensor.
(iii) To design the fuzzy controller for a self-leveling buoyancy system of a mini ROV.
(iv) To develop a wireless communication between a mini ROV and a computer.
(v) To test the performance of the developed system in order to fulfill the requirement.

1.4 Scopes and limitations

The scopes and limitations of the project are given below:

(i) The control system used in this research was fuzzy logic control.
(ii) An Arduino Uno was applied for embedding the program.

(iii) All communications with the mini ROV have been conducted wirelessly within the X-CTU and MATLAB graphical user interface (GUI).

(iv) The wireless system module was implemented by using XBee S2 1mW Wire Antenna with ZigBee 2.4GHz radio frequency (RF) protocol.

(v) The size of the mini ROV was less than $1m^2$ area and weighed less than 2kg.

(vi) The mini ROV used non-submersible direct current (DC) controlled water pump to vertically level its position.

(vii) The self-balancing system is based on the attached liquid level sensor provided feedback to the fuzzy logic controller.

(viii) The testing of the mini ROV was done in calm and shallow water (<1m).

1.5 Report outline

The project in this report is divided into five chapters. The first chapter which is the project introduction represents the overview of the project that include the declaration of the problems, the objectives of the project, the limitations of the study, and the contributions to the research area.

Chapter 2 covers the review of each critical points of current knowledge including essential findings as well as theoretical and methodological contributions to a particular topic.

Chapter 3 explains the systematic study of methods that have been applied within the project that consists of project activities, system architecture, hardware design and software design.

Chapter 4 discusses and compares the data of results and analysis that has been obtained from the testing of hardware and software throughout the project development. The testing is divided into two parts; interface testing and system testing. The interface testing discusses all peripherals that are interfaced to the microcontroller while the system testing focuses on the high-level software operation.

Chapter 5 is the final chapter that provides the conclusion of the system design and development. Additionally, it discusses the promising direction of recommendation in future work based on the expected results presented in Chapter 4.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In order to design and develop a dynamic leveling control of a wireless mini ROV, extensive research on buoyancy and depth control of the vehicle need to be fulfilled. This chapter will discuss previous studies that have been accomplished by other researchers in the same area. Comparison of the studies is given at the end of the discussion.

2.2 Related work

Folcher and Rendas [4] address the identification and the control of the vertical motion of the ROV Phantom 500. They split the matter in two decoupled problems: the propeller motion and the diving motion. The ROV can be controlled from the surface either manually by using joysticks or automatically by a surface computer. It is well equipped with a 3 axis compass, a depth (pressure) sensor, an altimeter, a sonar profiler, a video camera, and incremental encoders for the thrusters.

Root locus techniques are applied in designing each of the controllers in order to recognize the dynamic models. This method achieves acceptable transient control objectives by fixing the root location of the closed loop poles. In conjunction with higher level controllers, these body motion controllers are used for the guidance of the vehicle (where an algorithm for guidance along seabed boundaries using vision is presented).

The controller generates the control signal Δ(n), as a function of the reference signal $\Omega_r(n)$, and of the measured output $\Omega(n)$. Equation 2.1 shows the controller in z-plane, where K(z) is the transfer function of the controller.

$$\Delta z = K(z)\left(\Omega_r(z) - \Omega(z)\right) \tag{2.1}$$

The closed-loop block diagram of the plant is illustrated in Figure 2.1. In this figure, $G(z)$ models the discrete time dynamics of the sample data system $G_c(s)$ in series with a zero order hold. A major downside of the current system is that it relies on simplified decoupled models, and thus valid only under specific operating points (where some velocities and accelerations can be neglected).

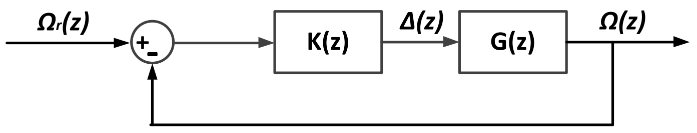


Figure 2.1: ROV plant closed-loop system

Tang and Luojun [5] developed a nonlinear depth control method for ROV. The method establishes the vertical motion model and uses least squares method (LSM) fitting multi-order output response of the control system. The fuzzy compensation is used to achieve overshoot suppression combined with proportional-integral-derivative (PID) controller. The LSM process the multi-order data fitting and generates the nonlinear predictor, where the predictor error is normalized as the input to the fuzzy controller.

The lead compensates to the control value is generated by the fuzzy controller by using fuzzy rules. At the end, the system uses PID control method in main control channel and fuzzy nonlinear compensation control method in the feedback forward channel to achieve the whole control system. The process for the control strategy is shown in Figure 2.2.

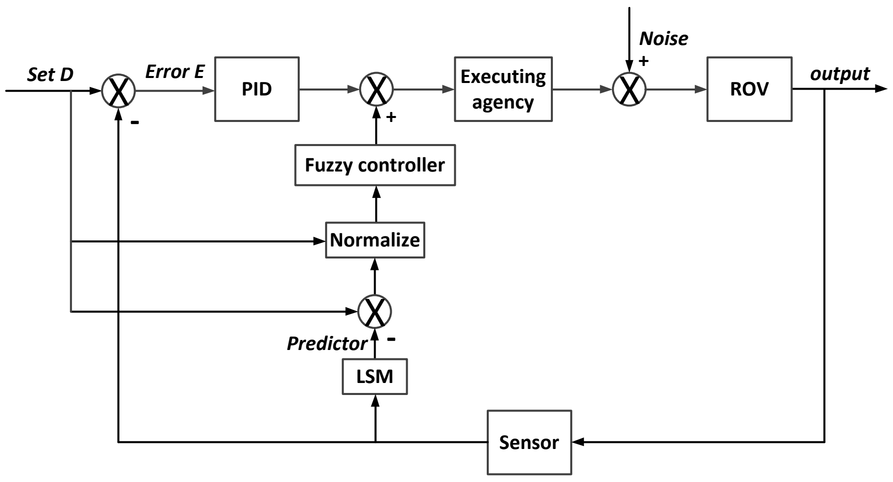


Figure 2.2: ROV control strategy

The comparison of simulation waveforms between normal fuzzy compensated PID (NM-FPID), normal PID controller, and the proposed least squares prediction fuzzy compensated PID (LQ-PID) is described in Figure 2.3. The simulation results show that the PID with prediction and fuzzy compensation control system response has a certain overshoot suppression performance and the system response speed faster than that of normal PID.

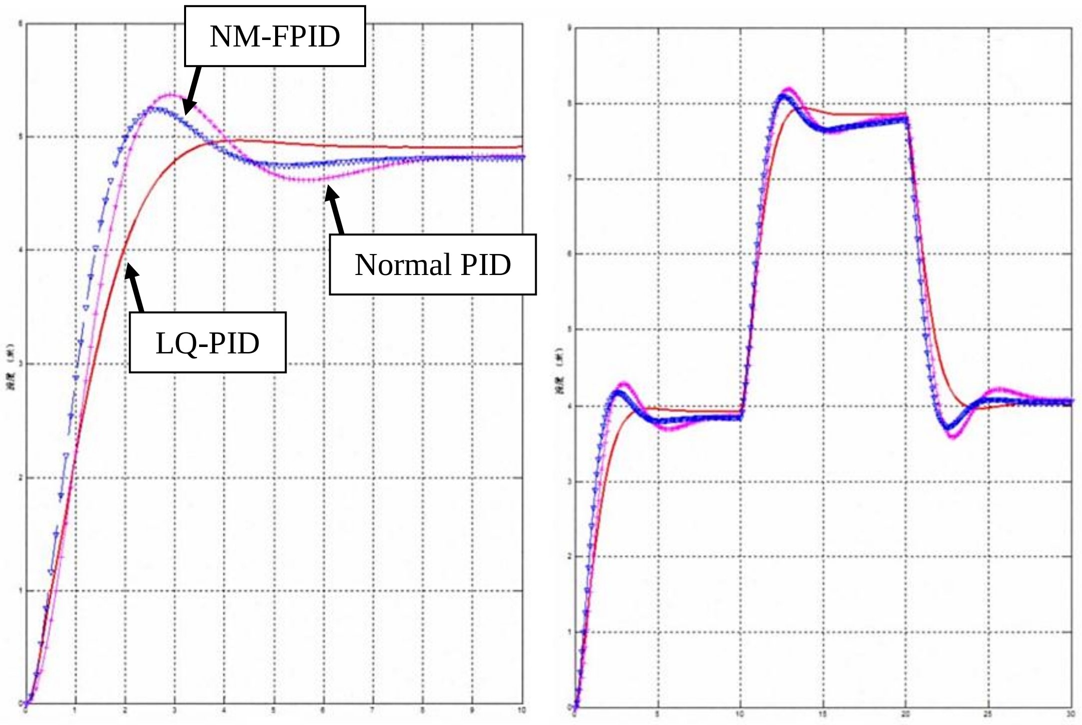


Figure 2.3: Depth control simulation

Shibuya, *et al*. [6] developed an underwater mobile robot with a buoyancy control system based on the spermaceti oil hypothesis originated back in 1970s [7]. The hypothesis insists that sperm whales melt and congeal their spermaceti oil that is located in their head and change the volume of the oil to control their own buoyancy. The anatomy of a sperm's whale head is presented in Figure 2.4. The spermaceti organ that contains capillaries is located in the head. The nasal passage that goes around the organ acts as a coolant and heating element for the spermaceti oil. To choose the best materials as a spermaceti oil substitute, the densities of four materials at both liquid and solid states were measured. Afterwards, the buoyancy differences between both states were calculated. The experiment resulted in paraffin wax as the best substance.

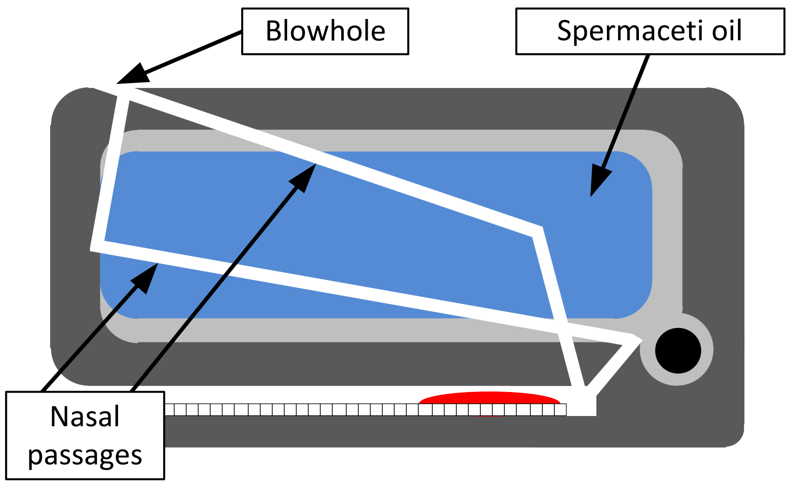


Figure 2.4: Schematic diagram of a sperm whale's head

It is noted that although the mobile robot can surface and submerge, it is not able to control depth and at least following problems must be solved to tackle this matter. First, sensors to detect depth of the robot, such as a depth-pressure sensor, must be added to the robot. Second, the heating method should be improved to shorten the time for melting paraffin wax, because the response time will affect robot's depth control ability greatly. Additionally, temperature of the paraffin wax should be controlled to regulate wax's volume and robot's buoyancy precisely.

Tangirala and Dzielski [2] developed a variable buoyancy control system (VBS) for a large AUV to launch in shallow water (<10m) and to hover without propulsion. The vehicle is equipped with two VBS tank to meet these requirements. The resulting control problem is that the control variable (pump rate) is proportional to the third derivative of the sensed variable (depth). There are significant delays and forces are nonlinear (including discontinuous). The VBS control software operates in two modes: depth control mode as in Figure 2.5 and trim control mode as in Figure 2.6.

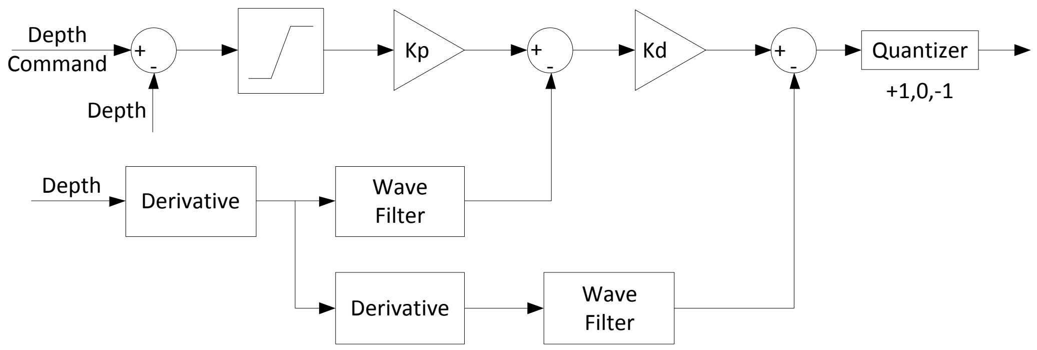


Figure 2.5: VBS depth controller

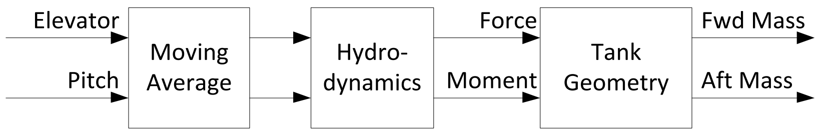


Figure 2.6: VBS trim controller

The AUV is launched from the surface using the VBS depth controller implementing a proportional derivative (PD) type of control law in conjunction with the fins. Logic and filtering are used to sequence operational modes and to reject low-frequency disturbances such as waves. While this depth control strategy is shown to be more than adequate to launch the vehicle, it cannot hovers due to the limit time cycle. Vehicle trim is achieved and maintained by the VBS trim controller that uses a model-based approach to determine the force/moment combination that trims the vehicle using a moving average of the pitch and elevator angles.

A long cruising range AUV equipped with a VBS system was developed by Zhao, *et al*. [8] and it was found that buoyancy is the resultant force between the AUV weight and the buoyancy of its displacement volume. The VBS of the long cruising range AUV was constructed with an oil tank, a rubber bladder which can regulate its displacement volume and a set of hydraulic drives system. The whole construction can bear the ambient sea water pressure when it works in 1000m depth.

The circuit diagram of the VBS for the AUV is shown in Figure 2.7. The displacement volume of the bladder is controlled by pumping the oil between the oil tank and the oil bladder with the hydraulic pump. This is to make the buoyancy of the AUV is altered without changing its weight. When the VBS computer receives the buoyancy adjustment command from the operator, it calculates the number of pump revolutions according to the ambient pressure of the AUV. It will then sends the pulse width modulation (PWM) signals to the driver of the DC motor.

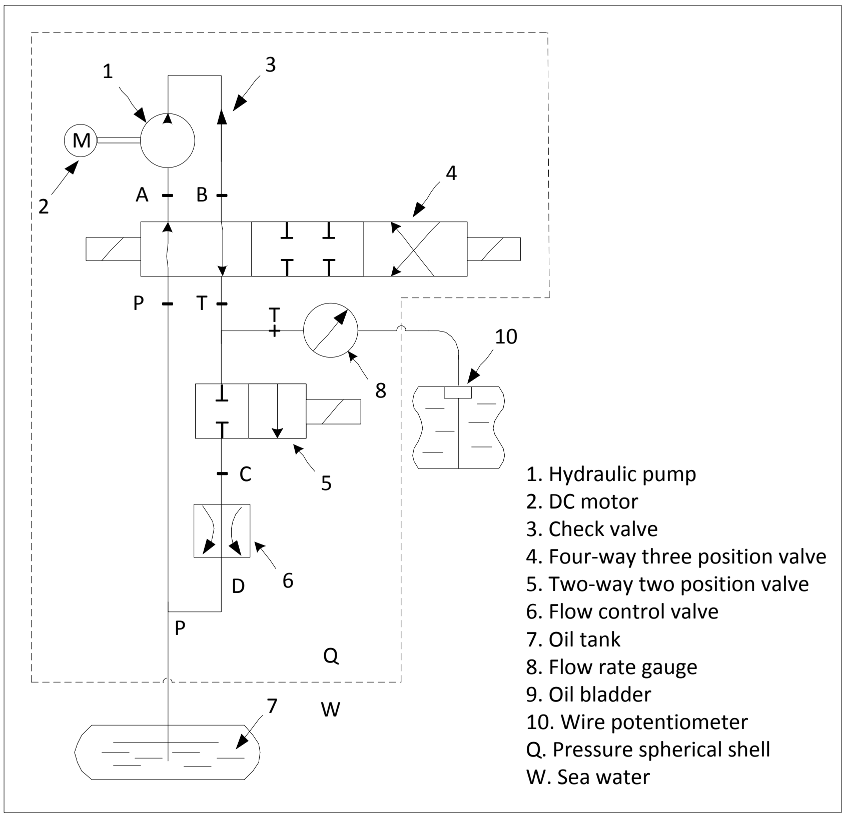


Figure 2.7: VBS circuit diagram

Wasserman, *et al*. [9] developed a dynamic buoyancy control of a tethered ROV using a variable ballast tank. The dynamic buoyancy control solution for the small-scale ROV is a pneumatic system that includes an air-filled ballast tank as depicted in Figure 2.8 with visual feedback from underwater cameras. The major parts of the system are the air source, the manifold and solenoids (used as the control system for the air), the pneumatic grippers, and the buoyancy control ballast tank. Plastic tubing is used as air lines between the major components.

The source of the air is a surface tank that is brought to the ROV through a single hose in ROV’s tether and is always pressurized. The ballast tank would be originally filled with water where a small hose would then expel the water from the tank and fill it with air to control its buoyancy.

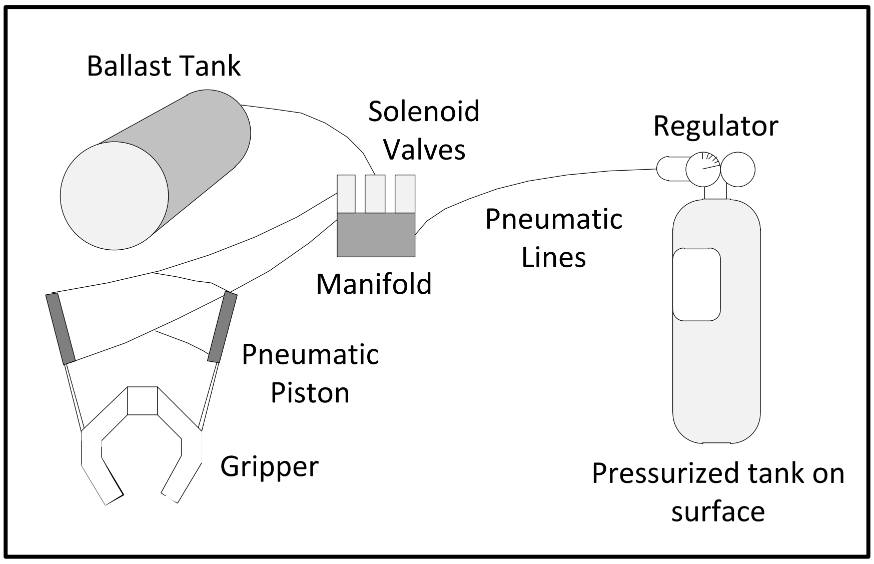


Figure 2.8: ROV pneumatic control system

2.3 Research comparison

The comparisons for each of the previous studies regarding the buoyancy and depth control for ROV are summarized in Table 2.1.

Table 2.1: Comparison between each research

No.	Research Title	Advantages	Disadvantages
1	Identification and Control of the Phantom 500 Body Motion [4]	• Achieved controllable depth using root locus techniques	• Complex • Real-life implementation not tested
2	A Novel ROV Depth Control Based on LSM Fitting Predictor and Fuzzy Compensation [5]	• Achieved vertical motion controlled by combining various method	• Complicated • Steady state output not accurate • Inaccurate depth control
3	Underwater Robot with a Buoyancy Control System Based on the Spermaceti Oil Hypothesis [6]	• Controllable pitch and roll	• Difficult to acquire information about wax densities • No mechanism to cool down paraffin wax • Inaccurate depth control
4	A Variable Buoyancy Control System for a Large AUV [3]	• Logic and filtering system to reject low frequency disturbances	• Significant delays, forces are nonlinear and highly uncertain • Inaccurate depth control
5	A Variable Buoyancy System for Long Cruising Range AUV [8]	• Energy saving in deep water	• Complicated design and operation • High energy consumption (1KW) • Inaccurate depth control
6	Dynamic Buoyancy Control of an ROV using a Variable Ballast Tank [9]	• Large surface tank can provide high pressure air for a much larger number of cycles of the grippers	• Compressed air supply above the surface makes it not portable • Inaccurate depth control

From the research comparison in Table 2.1, it can be seen that there are no studies that are perfect. Some of the studies seem to have a highly complicated system with great simulation results and the ROV are able to surface and submerge underwater. Nevertheless, they still cannot possess the ability to be positioned at a specific depth accurately. Therefore, the mini ROV in this project is developed to have the capability on precise depth control with a self-balancing buoyancy system by using a rather direct and simpler approach for prototyping purpose.

CHAPTER 3

METHODOLOGY

3.1 Introduction

The project development of this study as shown in Figure 3.1 is divided into two parts namely Part 1 and Part 2. Each part represents the work to be done for Master Project 1 and Master Project 2 respectively. Part 1 starts by first identifying the problems that exist in the current system of a ROV. Extensive literature reviews has been done on related knowledge to assist in any ways that it may. Such reviews are based on international publications, engineering-related websites, and engineering books. Detail research in hardware is needed for the mini ROV electrical and electronic development in terms of availability, performance, and technical supports. The system requirement was then determined to proceed on this project.

The next step that is in Part 2 is followed by planning on the design of the hardware and software part of the mini ROV system. The low-level software is an interfacing subroutine for associated components that consist of a personal computer, XBee wireless modules, Arduino Uno mainboard, motor control, and liquid level sensor. The interface testing had to be done to ensure the functionality of the hardware and software modules are compatible with the microcontroller. If the testing fail, the problem should be verified either it is due to hardware components or software parameters.

Subsequently, if the overall system is working, high-level software that is the application to control the mini ROV movement and depth control is developed. Upon completing the system software, the overall system needs to be tested and analyzed to verify its functionality and performance.

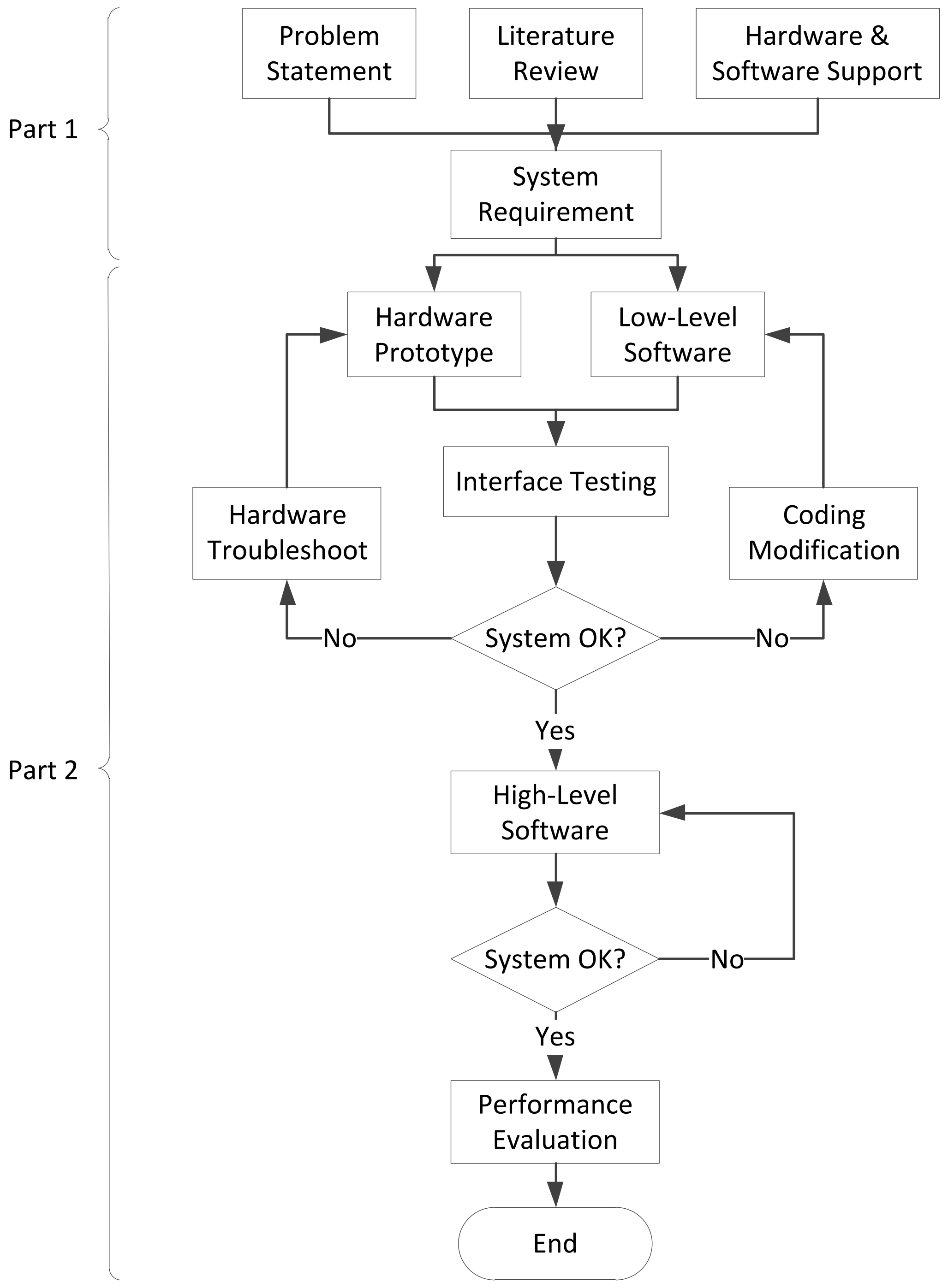


Figure 3.1: Flowchart of project activities

3.2 System architecture

Figure 3.2 shows the interface between all devices that has been implemented to the mini ROV system by using master and slave setup. Controlling the mini ROV is done in a Windows GUI by a human operator with a personal computer. The computer is serially connected to the SKXBee board while simultaneously receive data and feedback from the onboard sensor of the mini ROV. Connection between master and slave control is made by using XBee series 2 wireless module. The module which is ZigBee compliant is chosen since it provides low transmission power rate and is suitable to use in short range data transmission [10].

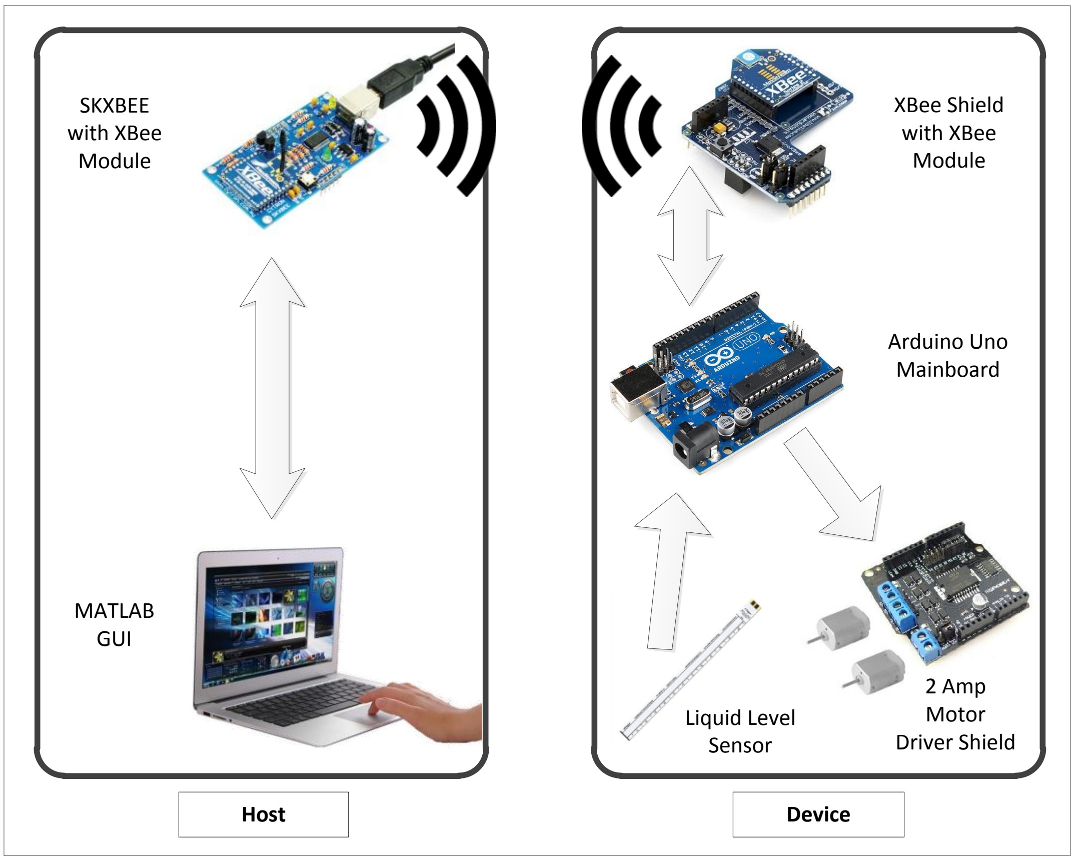


Figure 3.2: Device interfacing with Arduino Uno mainboard microcontroller

3.3 Hardware design

The hardware section gives further explanations on both mechanical and electronic parts. The mechanical part explains the prototype of the mini ROV design while the electronic part describes the interfacing between all of the electronic devices.

3.3.1 Mechanical

The mini ROV of this project was designed by using SolidWorks with every details and considerations taken into account. The prototype design is shown in Figure 3.3. The final version however had rather underwent some modifications and tweaks to cope with the overall system of this project.

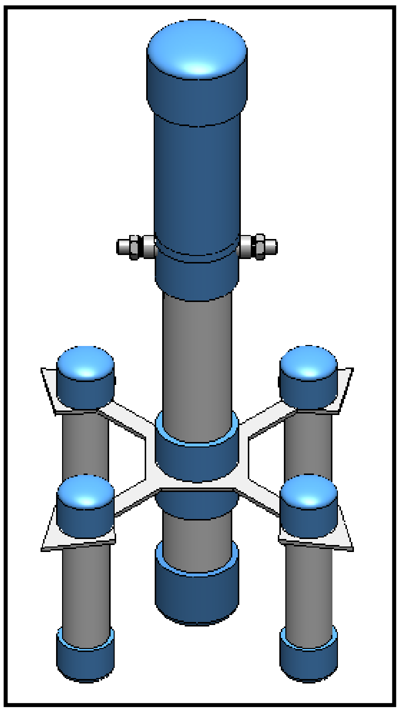


Figure 3.3: Mini ROV prototype design

The prototype design has two separated hull; the lower partition to accommodate the water intake while the upper for the electronic components. There are two check valves located just below the upper partition. Both valves function to stabilize the pressure inside the lower partition to be the same as the outside atmospheric pressure whenever water is pumped in or out of the hull.

Since fluid pressure increases with depth and that the increased pressure exerted in all directions, thus there is an unbalanced upward force on the bottom of a submerged object [11]. To overcome this problem, four small cylindrical hulls are strategically placed around the main structure to further provide stability when the mini ROV is submerged into water. These hulls also help in increasing its positive buoyancy in order to compensate the weight of the mini ROV and can be manually adjusted for prototyping purpose.

The liquid level sensor is attached to the outside of the mini ROV. Archimedes principle stated that the buoyant force on a submerged object is equal to the weight of the fluid that is displaced on by the object [12]. In other words, when a solid body is partially or completely immersed in water, the apparent loss in weight will be equal to the weight of the displaced liquid. Since the amount of the displaced liquid is directly proportional to the height of the mini ROV being submerged, hence placing the sensor on the outside of the mini ROV is suitable and a fast way to properly determine its vertical position.

3.3.2 Electronics

3.3.2.1 Arduino Uno R3 mainboard

The Arduino Uno R3 mainboard as given in Figure 3.4 is an open-source electronic board based on the ATmega328 microcontroller. It has 14 digital input/output (I/O) pins of which 6 can be used as PWM outputs, 6 analog inputs, a 16 MHz crystal oscillator, a universal serial bus (USB) connector to download a program, a power jack, and a reset button. Table 3.1 summarizes the main specification of the Arduino mainboard [13]. It contains everything needed to support the microcontroller and is highly recommended for fast prototyping of a system. The coding to develop the program for the mini ROV system in this project is presented in APPENDIX A.

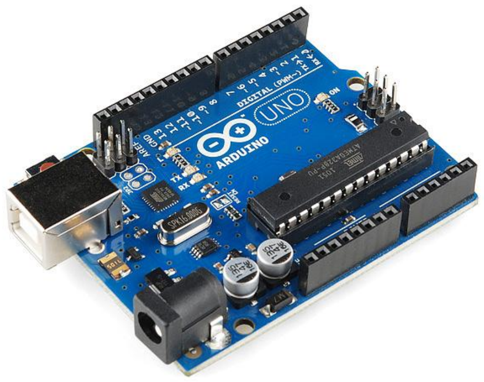


Figure 3.4: Arduino Uno R3 mainboard

Table 3.1: Arduino Uno R3 mainboard specification

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM outputs)
Analog Input Pins	6
DC Current per I/O Pin	40mA
DC Current for 3.3V Pin	50mA
Clock Speed	16MHz

3.3.2.2 Liquid level sensor

The liquid level sensor is a solid state, continuous (multi-level) fluid level sensor for measuring levels in water, non-corrosive water based liquids and dry fluids (powders) [14]. The sensor as shown in Figure 3.5 is manufactured by Milone Technologies using printed electronic technologies which employ additive direct printing processes to produce functional circuits. The sensor's envelope is compressed by hydrostatic pressure of the fluid in which it is immersed resulting in a change in resistance which corresponds to the distance from the top of the sensor to the fluid surface.

Figure 3.5: eTape liquid level sensor

The sensor provides a resistive output that is inversely proportional to the level of the liquid, in which case when the liquid level is low, the output resistance will be high, and vice versa. The sensor has its own marking printed in centimeters and inches making it easier to manually monitor the fluid level. The specification of the sensor is given in Table 3.2. The sensor can be modeled as a variable resistor (300 - 1500Ω ±10%) and the typical output of the sensor is given in Figure 3.6 [14].

Table 3.2: Liquid level sensor specification

Sensor Length	10.1″ (257mm)
Active Sensor Length	8.4″ (213mm)
Resolution	< 0.01″ (0.25mm)
Width	1.0″ (25.4mm)
Sensor Output	1500Ω empty, 300Ω full, ±10%
Temperature Range	-9°C - 65°C
Resistance Gradient	140Ω/inch (56Ω/cm), ±10%
Power Rating	0.5W (VMax = 10V)

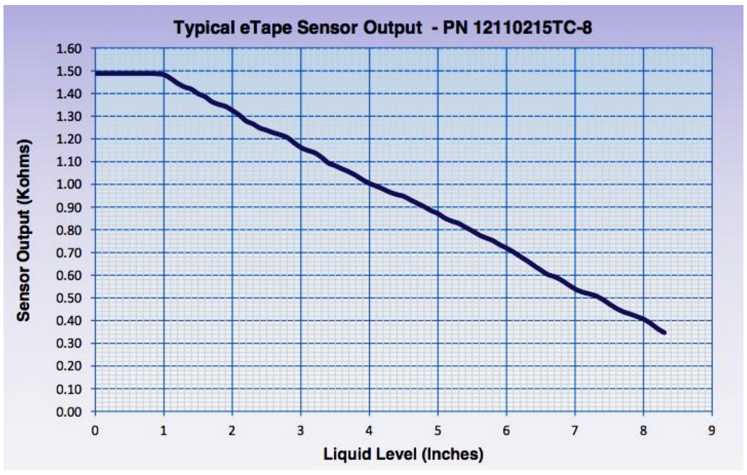


Figure 3.6: Typical output of liquid level sensor

To get the readings of the sensor when it is submerged, the following formula in Equation 3.1 is applied. The initial point at mark 15cm of the sensor has been chosen. The SensorValue is the current reading of the sensor in terms of resistance. The value 354 is the sensor reading at the initial point while the value 290 is the sensor reading 2cm above the initial point. The whole equation is multiplied by 2 since the range of the sensor calibration in this project is 2cm. The output of the equation will give the position of the sensor in centimeters resolution.

$$\text{Sensor reading (cm)} = \frac{354 - \text{SensorValue}}{354 - 290} \times 2 \qquad (3.1)$$

3.3.2.3 Device connection

The connection diagram for interfacing Arduino Uno microcontroller with the liquid level sensor, XBee shield (with XBee module), DC pumps, and 2A DC motor driver are given as in Figure 3.7. The supply voltage to power ON the whole system is a 7.4V DC connected to the Vin port of Arduino. The microcontroller shared the same pin connections for 5V, GND (ground), and RST (reset) port with the 2 amp motor driver shield which uses a L298P chip that can drives two 7 to 12V DC motors with a maximum of 2A current. Since the motor shield is specifically custom-made for the Arduino Uno mainboard and alike, it can be directly mounted on top of the board without having to do any modifications.

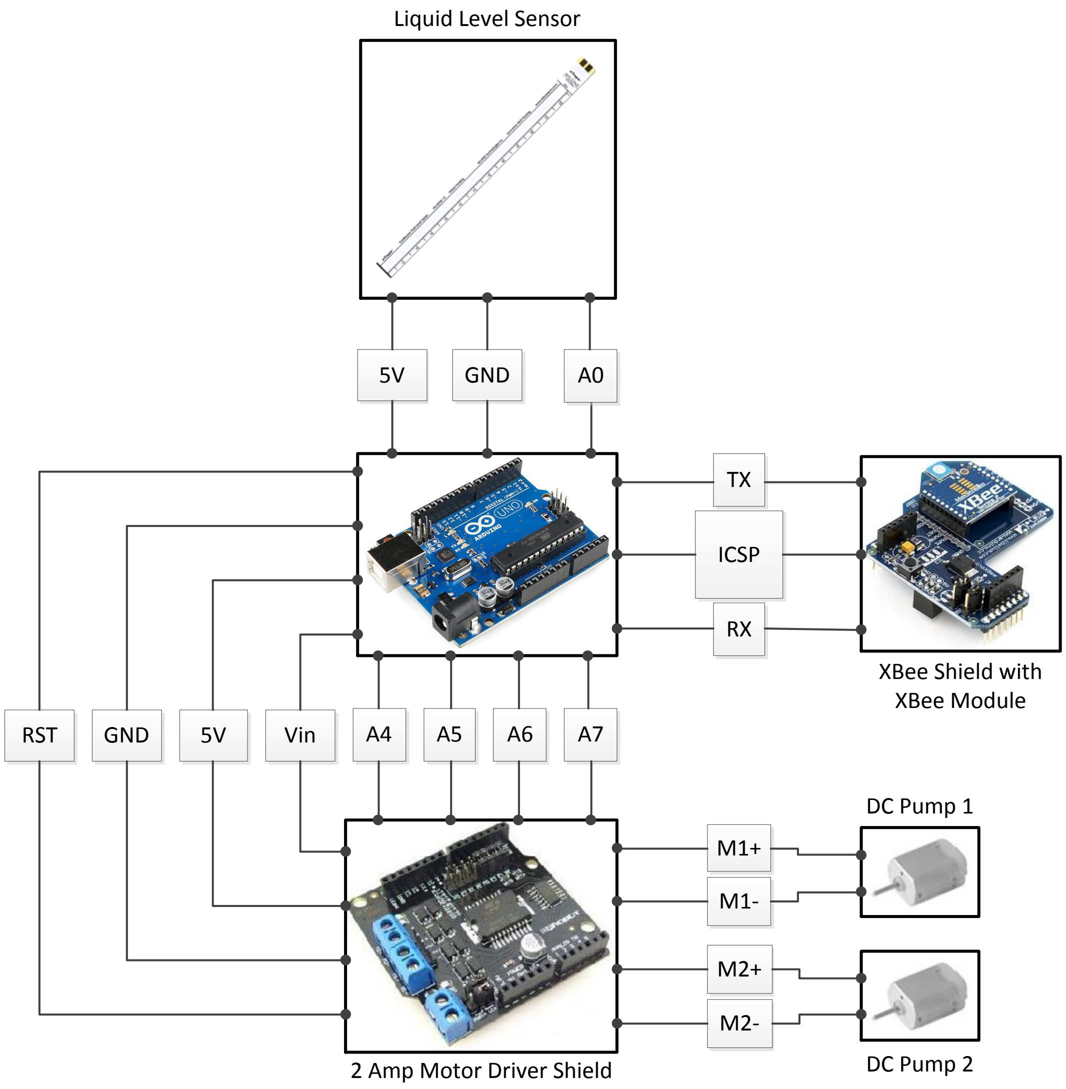


Figure 3.7: Device connection diagram to Arduino Uno microcontroller board

The micro DC pumps that are used in this system are suitable to work for both pumping water and air. They are both being supplied with a supply voltage of 4.8V whenever they are triggered by the microcontroller. Turning ON the pump is done by using PWM signal with a maximum speed of 255 while sending a signal 0 on the other hand will turn OFF the motors. The working voltage of the pump is 3 to 7.4V while the working current is 75mA. The motor driver shield is needed since the Arduino pin can only supply up to 40mA for each of the I/O pins.

The liquid level sensor uses 5V, GND, and analog A0 port from Arduino. The XBee shield that is attached with the XBee module is connected to the microcontroller via the in-circuit serial programming (ICSP) header. The XBee module works as a transceiver; therefore it transmits and receives data via the TX and RX pin. The ICSP header of the Arduino consists of a Master In Slave Out (MISO), Master Out Slave In (MOSI), Serial Clock (SCK), +5V, ground, and reset pin. Similar to the motor shield, it can be stacked right on top of the board while still having all required pins connected.

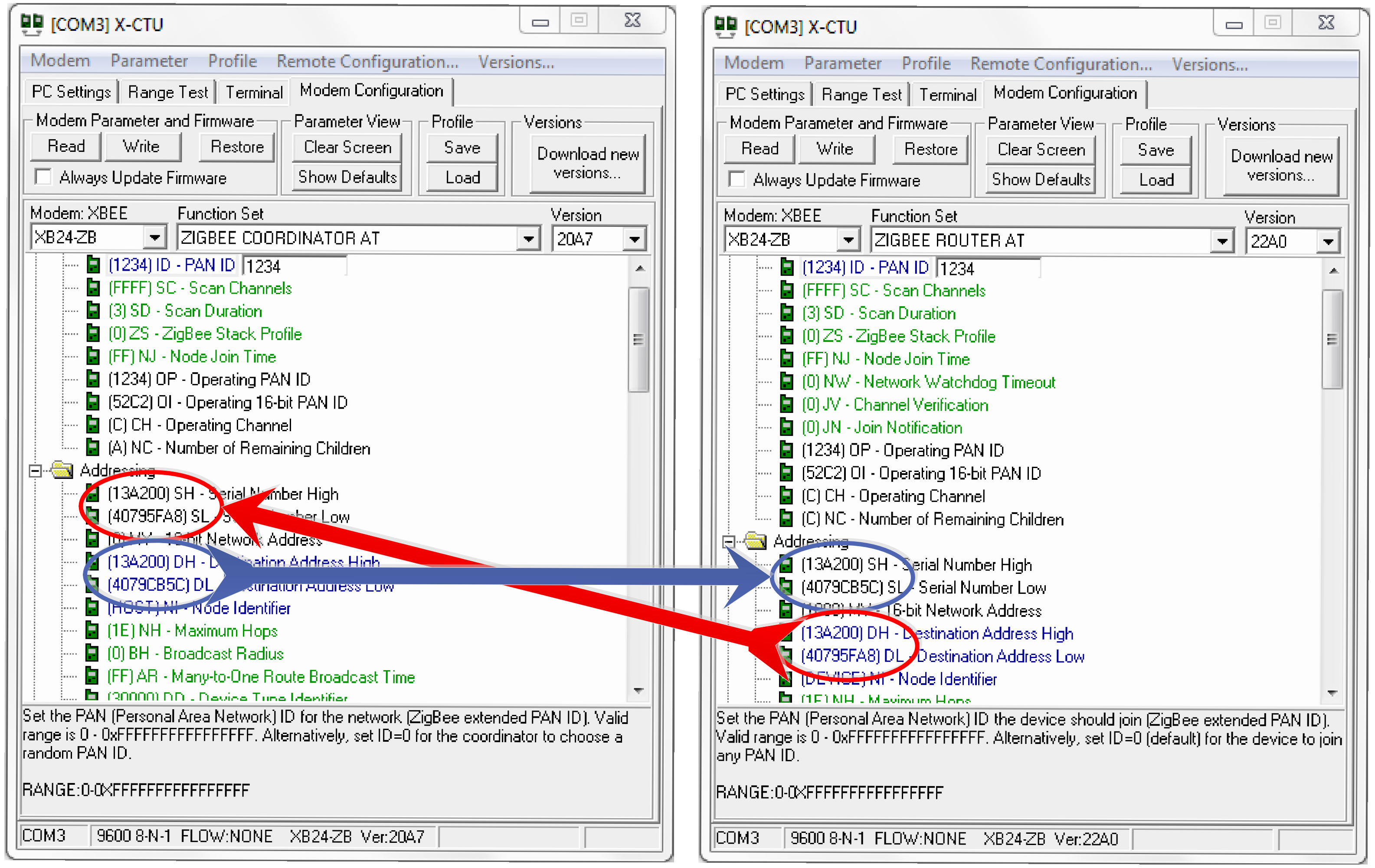
3.4 Software design

The software section gives explanations regarding the wireless communication setup between the Arduino microcontroller and computer. Other details include fuzzy logic control that is used in this project and the GUI for the control and data monitoring of the mini ROV.

3.4.1 Wireless communication

Every XBee modules have their own unique addresses. In order to get the wireless communication to work, both of the source and destination address of the XBee modules need to be paired with each other. There are two ways to do the setup; one by using a small list of programming code while another is simply by using the X-CTU software provided by Digi International. X-CTU is a free software that not only can be used to set the communication parameters on the XBee but also a monitoring platform to transmit and receive data communications.

Figure 3.8 shows the X-CTU software within the Modem Configuration tab. The window on the left is the XBee that had been set to be the coordinator while the window on the right is the XBee that had been set to be the router. Both modules need to be setup one at a time by attaching it to the SKXBEE board and link it to the computer via USB cable. In X-CTU, one of them needs to be set as a Coordinator AT while the other as a Router AT. The selection can be chosen from the Function Set dropdown menu.


[COM3] X-CTU
Modem Parameter Profile Remote Configuration... Versions...
PC Settings Range Test Terminal Modem Configuration
Modem Parameter and Firmware Read Write Restore Always Update Firmware
Parameter View Clear Screen Show Defaults
Profile Save Load
Versions Download new versions...

Modem: XBEE XB24-ZB — Function Set ZIGBEE COORDINATOR AT — Version 20A7

```
(1234) ID - PAN ID 1234
(FFFF) SC - Scan Channels
(3) SD - Scan Duration
(0) ZS - ZigBee Stack Profile
(FF) NJ - Node Join Time
(1234) OP - Operating PAN ID
(52C2) OI - Operating 16-bit PAN ID
(C) CH - Operating Channel
(A) NC - Number of Remaining Children
Addressing
(13A200) SH - Serial Number High
(40795FA8) SL - Serial Number Low
16-bit Network Address
(13A200) DH - Destination Address High
(4079CB5C) DL - Destination Address Low
(HOST) NI - Node Identifier
(1E) NH - Maximum Hops
(0) BH - Broadcast Radius
(FF) AR - Many-to-One Route Broadcast Time
(30000) DD - Device Type Identifier
```

Set the PAN (Personal Area Network) ID for the network (ZigBee extended PAN ID). Valid range is 0 - 0xFFFFFFFFFFFFFFFF. Alternatively, set ID=0 for the coordinator to choose a random PAN ID.

RANGE:0-0XFFFFFFFFFFFFFFFF

COM3 9600 8-N-1 FLOW:NONE XB24-ZB Ver:20A7

Modem: XBEE XB24-ZB — Function Set ZIGBEE ROUTER AT — Version 22A0

```
(1234) ID - PAN ID 1234
(FFFF) SC - Scan Channels
(3) SD - Scan Duration
(0) ZS - ZigBee Stack Profile
(FF) NJ - Node Join Time
(0) NW - Network Watchdog Timeout
(0) JV - Channel Verification
(0) JN - Join Notification
(1234) OP - Operating PAN ID
(52C2) OI - Operating 16-bit PAN ID
(C) CH - Operating Channel
(C) NC - Number of Remaining Children
Addressing
(13A200) SH - Serial Number High
(4079CB5C) SL - Serial Number Low
16-bit Network Address
(13A200) DH - Destination Address High
(40795FA8) DL - Destination Address Low
(DEVICE) NI - Node Identifier
(1E) NH - Maximum Hops
```

Set the PAN (Personal Area Network) ID the device should join (ZigBee extended PAN ID). Valid range is 0 - 0xFFFFFFFFFFFFFFFF. Alternatively, set ID=0 (default) for the device to join any PAN ID.

RANGE:0-0XFFFFFFFFFFFFFFFF

COM3 9600 8-N-1 FLOW:NONE XB24-ZB Ver:22A0

Figure 3.8: Pairing two XBee modules in X-CTU

The destination address high and low for each module needs to be the exact address as the source address high and low for the other module. This can be done by simply copy the value of the source address from one module and paste it to the destination address of the other module. The same method applied for both XBee modules.

3.4.2 Fuzzy logic controller

There are various controller design theories that can be used to maintain the leveling control of the mini ROV system such as fuzzy logic controller, PID controller, adaptive controller, pole placement, linear quadratic regular (LQR), and robust control. In this study, fuzzy logic controller is used to control the vertical motion of the mini ROV. The components of fuzzy logic controller are shown in Figure 3.9. It generally comprises of four principal modules; fuzzifier, knowledge base, inference engine, and defuzzifier.

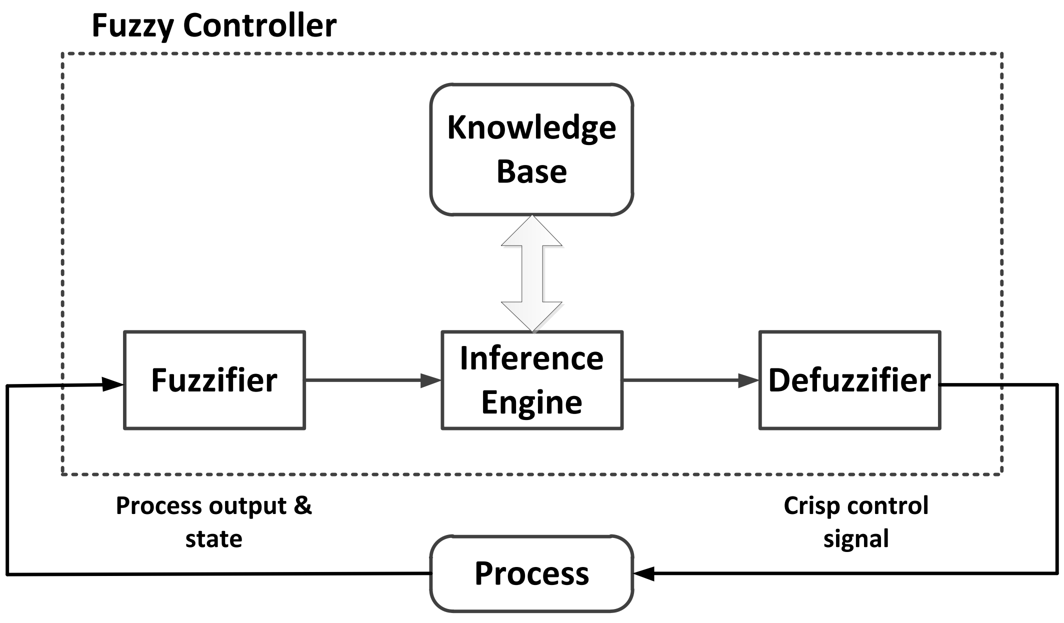


Figure 3.9: Components of fuzzy logic controller

REFERENCES

1. Marine Technology Society (2010). *ROV Applications – What ROVs Can Do*. Retrieved on April 5, 2012, from http://www.rov.org/rov_applications .cfm
2. Psarros, D., Papadimitriou, V., and Chatzakos, P., A service robot for subsea flexible risers, IEEE Robotics & Automation Magazine, 2010, Vol. 7, pp. 55-63. Retrieved July 17, 2012, from IEEE database.
3. Tangirala, S. and Dzielski, J. “A Variable Buoyancy Control System for a Large AUV,” *IEEE Journal of Oceanic Engineering*, Vol. 32, No. 4, 2007, pp. 762-771. doi: 10.1109/JOE.2007.911596.
4. Folcher, J. P. and Rendas, M. J. "Identification and Control of the Phantom 500 Body Motion," MTS/IEEE Conference and Exhibition of OCEANS 2001, Honolulu, 5-8 November 2001, pp.529-535.
5. Tang, Z. and Luojun, P. Y., “A Novel ROV Depth Control Based on LSM Fitting Predictor and Fuzzy Compensation,” 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), 2010, Chengdu, 20-22 August 2010, V2-612 - V2-614.
6. Shibuya, K, Kado, Y., Honda, S., Iwamoto, T. and Tsutsumim, K., “Underwater Robot with a Buoyancy Control System Based on the Spermaceti Oil Hypothesis,” *Proceedings of the 2006 IEEE/RSJ Interational Conference on Intelligent Robots and Systems*, Beijing, 9-15 October 2006, pp. 3012-3017.
7. Clarke, M. R. “Buoyancy Control as a Function of the Spermaceti Organ in the Sperm Whale,” *Journal of the Marine Biological Association of the United Kingdom*, Vol. 58, Issue 1, 1978, pp. 27-71. doi: http://dx.doi.org/10.1017/ S0025315400024395

8. Zhao, W., Xu, J., and Zhang, M., "A Variable Buoyancy System for Long Cruising Range AUV," *2010 International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE)*, Harbin, 24-26 August 2010, pp.585-588.

9. Wasserman, K. S., Mathieu, J. L., Wolf, M. I., Hathi, A., Fried, S. E. and Baker, A. K., "Dynamic Buoyancy Control of an ROV using Variable Ballast Tank," *Proceedings of OCEANS 2003*, Massachusetts, 22-26 September 2003, pp. 2888-2893.

10. MaxStream, (2007). *XBee Series 2 OEM RF Modules*. Retrieved May 14, 2012, from p. 4 at ftp://ftp1.digi.com/support/documentation/90000866_A .pdf

11. Nave, R. (1998), *Pressure*. Retrieved on May 24, 2012, from http://hyperphysics.phy-astr.gsu.edu/hbase/pbuoy.html

12. Carrol, B. (2009), *Archimedes Principle*. Retrieved on May 24, 2012, from http://physics.weber.edu/carroll/archimedes/principle.htm

13. Arduino, (2012), *Arduino Uno*. Retrieved on May 26, 2012, from http://arduino.cc/en/Main/ArduinoBoardUno

14. Milone Technologies, (2012), *eTape Specification*. Retrieved on May 26, 2012, from p. 1 at http://www.milonetech.com/uploads/eTape_Datasheet_ 12110215TC-8.pdf

15. Xu, M., "Adaptive Fuzzy Logic Depth Controller for Variable Buoyancy System of Autonomous Underwater Vehicles," *Proceedings of the Third IEEE Conference on Fuzzy Systems*, Florida, 26-29 June 1994, pp. 1191-1196.

16. Salim, M. A., Noordin, A., and Jahari, A. N., "A Robust of Fuzzy Logic and Proportional Derivative Control System for Monitoring Underwater Vehicles," *Second International Conference on Computer Research and Development*, Melaka, 7-10 May 2010, pp. 849-853.

VITA

The author was born in January 8, 1988, in Ipoh, Perak, Malaysia. He went to High School Batu Pahat (HSBP) in Batu Pahat, Johor for his secondary school. He pursued his degree at the Universiti Tun Hussein Onn Malaysia (UTHM), Parit Raja, Johor, and graduated with the B.Eng. (Hons) in Electronic and Electrical Engineering in 2011. Upon graduation, he attended the Graduate Studies of the UTHM and was admitted into the Master program in Electrical Engineering in 2011.

LIST OF PUBLICATIONS

1. Ayob, M. A., Hanafi, D., Johari, A., *Dynamic leveling control of a wireless self-balancing ROV using fuzzy logic controller*, in SCIRP Intelligent Control and Automation (ICA 2012), 2012. (submitted)
2. Ayob, M. A., Hanafi, D., Johari, A., *Dynamic leveling control of a wireless self-balancing ROV using fuzzy logic controller*, in 5th International Conference On Postgraduate Education (5th ICPE), UiTM, 18-19 December 2012. (participant)

APPENDIX A

Arduino coding

```
int RiseMotorSpeed = 5; //pwm  //motor 1
int SinkMotorSpeed = 6; //pwm  //motor 2
int RiseMotorEn = 4; //enable pin
int SinkMotorEn = 7; //enable pin

int LED = 13;

float x[3], input[2], sensor, sensorValue;
float fsE[2], sectorE[2], fsDE[2], sectorDE[2], centerout[2];

float Mfout1, Mfout2;


//***************************************************************//
//Declaration of Fuzzy Set for Input 1 (Error)                   //
//***************************************************************//


float
  a1 = 0,        b1 = -1,        c1 = -0.5,
  a2 = -1,       b2 = -0.5,      c2 = 0,
  a3 = -0.5,     b3 = 0,         c3 = 0.5,
  a4 = 0,        b4 = 0.5,       c4 = 1,
  a5 = 0.5,      b5 = 1,         c5 = 0;

//F=[a1 b1 c1;a2 b2 c2;a3 b3 c3;a4 b4 c4;a5 b5 c5;];


//***************************************************************//
//Declaration of Fuzzy Set for Input 2 (DeltaError)              //
//***************************************************************//


float
  a6 = 0,          b6 = -1,         c6 = -0.375,
  a7 = -0.75,      b7 = -0.375,     c7 = 0,
  a8 = -0.375,     b8 = 0,          c8 = 0.375,
  a9 = 0,          b9 = 0.375,      c9 = 0.75,
  a10 = 0.375,     b10 = 1,         c10 = 0;

//G=[a6 b6 c6;a7 b7 c7;a8 b8 c8;a9 b9 c9;a10 b10 c10;];


//***************************************************************//
//Rules Designation (25 Rules)                                   //
//***************************************************************//


float co1[5] = {    -1.0, -0.5, -1.0, -0.5, -1.0    };
float co2[5] = {    -1.0, -0.5, -0.3, -0.5, -1.0    };
float co3[5] = {    +0.0, +0.0, +0.0, +0.0, +0.0    };
float co4[5] = {    +1.0, +0.5, +0.3, +0.5, +1.0    };
float co5[5] = {    +1.0, +0.5, +1.0, +0.5, +1.0    };





// the setup routine runs once the reset button is pressed:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  pinMode(RiseMotorEn, OUTPUT);
  pinMode(SinkMotorEn, OUTPUT);
  digitalWrite(RiseMotorEn, HIGH);
  digitalWrite(SinkMotorEn, HIGH);

  pinMode(LED, OUTPUT);
  digitalWrite(LED, HIGH);

  input[0] = 0;
  input[1] = 0;
  x[2] = 0;
}



// the loop routine runs over and over again forever:
void loop() {

  //****************************************************************//
  //Declaration of Fuzzy Set for Input 1 (Error)  (INPUT)              //
  //****************************************************************//


  for (int L=0; L<=5; L++) {    //delay 1s
  digitalWrite(LED, LOW);
  delay(200);
  digitalWrite(LED, HIGH);
  delay(200);
  }

  sensorValue = analogRead(A0);
  //sensor = ((512-sensorValue)/(512-1023)); // test with POT
  sensor = ((334-sensorValue)/(334-300))*2; // LLV. 0 = 15cm

//delay(3000);
 if (!Serial.available()); { // wait until there is signal send from computer
  input[0] = Serial.read();
 }
  if (input[0] == 45) { // read -ve input. -ve == 45
    while (!Serial.available());
    input[0] = Serial.read();
    input[0] = -(((input[0]-48)/2)*1.75);
    input[1] = input[0];
  }

  else if (input[0] > 45) { // read +ve input
    input[0] = ((input[0]-48)/2)*1.75;
    input[1] = input[0];
  }


  else {
    input[0] = input[1];
  }



  x[0] = input[0] - sensor; // error
  x[1] = x[0] - x[2];  // delta error = new error - old error
  x[2] = x[0]; // old error

    Serial.println(sensor);


  //Region a
  if (x[0] <= b1) {
    fsE[0] = 1;
    fsE[1] = 1;
    sectorE[0] = -2; //NB
    sectorE[1] = -2; //NB
  }

  //Region b
  else if (x[0] > b1 && x[0] <= b2) {
    fsE[0] = (c1-x[0])/(c1-b1);
    fsE[1] = (x[0]-a2)/(b2-a2);
    sectorE[0] = -2; //NB
    sectorE[1] = -1; //N
  }

  //Region c
  else if (x[0] > b2 && x[0] <= b3) {
    fsE[0] = (c2-x[0])/(c2-b2);
    fsE[1] = (x[0]-a3)/(b3-a3);
    sectorE[0] = -1; //N
    sectorE[1] = 0;  //Z
  }

  //Region d
  else if (x[0] > b3 && x[0] <= b4) {
    fsE[0] = (c3-x[0])/(c3-b3);
    fsE[1] = (x[0]-a4)/(b4-a4);
    sectorE[0] = 0;  //Z
    sectorE[1] = 1;  //P
  }

  //Region e
  else if (x[0] > b4 && x[0] <= b5) {
    fsE[0] = (c4-x[0])/(c4-b4);
    fsE[1] = (x[0]-a5)/(b5-a5);
    sectorE[0] = 1;  //P
    sectorE[1] = 2;  //PB
  }

  //Region f
  else if (x[0] > b5) {
  //else {
    fsE[0] = 1;
    fsE[1] = 1;
    sectorE[0] = 2; //PB
    sectorE[1] = 2;  //PB
  }


  //*****************************************************************//
  //Declaration of Fuzzy Set for Input 2 (Delta Error)                //
  //*****************************************************************//


  //Region a:
  if (x[1] <= b6) {
    fsDE[0] = 1;
    fsDE[1] = 1;
    sectorDE[0] = -2; //NB
    sectorDE[1] = -2; //NB
  }

  //Region b:
  else if (x[1] > b6 && x[1] <= b7) {
    fsDE[0] = (c6-x[1])/(c6-b6);
    fsDE[1] = (x[1]-a7)/(b7-a7);
    sectorDE[0] = -2; //NB
    sectorDE[1] = -1; //N
  }

  //Region c:
  else if (x[1] > b7 && x[1] <= b8) {
    fsDE[0] = (c7-x[1])/(c7-b7);
    fsDE[1] = (x[1]-a8)/(b8-a8);
    sectorDE[0] = -1; //N
    sectorDE[1] = -0; //Z
  }

  //Region d:
  else if (x[1] > b8 && x[1] <= b9) {
    fsDE[0] = (c8-x[1])/(c8-b8);
    fsDE[1] =( x[1]-a9)/(b9-a9);
    sectorDE[0] = 0; //Z
    sectorDE[1] = 1; //P
  }

  //Region e:
  else if (x[1] > b9 && x[1] <= b10) {
    fsDE[0] = (c9-x[1])/(c9-b9);
    fsDE[1] = (x[1]-a10)/(b10-a10);
    sectorDE[0] = 1; //P
    sectorDE[1] = 2; //PB
  }

  else if (x[1] > b10) {
```

```
//else {
  fsDE[0] = 1;
  fsDE[1] = 1;
  sectorDE[0] = 2; //NB
  sectorDE[1] = 2;
}


//*****************************************************************//
//Fuzzy Inference System (MAMDANI METHOD)                          //
//*****************************************************************//


if (fsE[0] < fsDE[0]) {    //take min
  Mfout1 = fsE[0];
}
else
  Mfout1 = fsDE[0];


if (fsE[1] < fsDE[1]) {    //take min
  Mfout2 = fsE[1];
}
else
  Mfout2 = fsDE[1];



//*****************************************************************//
//Rules Designation (25 Rules)                                     //
//*****************************************************************//

//*************************************Row-1

int k;
for (k = 0; k < 2; k++) {
  if (sectorE[k] == -2 && sectorDE[k] == -2) {
    centerout[k] = co1[0];
  }//co = central output

  else if (sectorE[k] == -2 && sectorDE[k] == -1) {
    centerout[k] = co1[1];
  }

  else if (sectorE[k] == -2 && sectorDE[k] == 0) {
    centerout[k] = co1[2];
  }

  else if (sectorE[k] == -2 && sectorDE[k] == 1) {
    centerout[k] = co1[3];
  }

  else if (sectorE[k] == -2 && sectorDE[k] == 2) {
    centerout[k] = co1[4];
  }

  //*************************************Row-2

  else if (sectorE[k] == -1 && sectorDE[k] == -2) {
    centerout[k] = co2[0];
  }

  else if (sectorE[k] == -1 && sectorDE[k] == -1) {
    centerout[k] = co2[1];
  }

  else if (sectorE[k] == -1 && sectorDE[k] == 0) {
    centerout[k] = co2[2];
  }

  else if (sectorE[k] == -1 && sectorDE[k] == 1) {
    centerout[k] = co2[3];
  }

  else if (sectorE[k] == -1 && sectorDE[k] == 2) {
    centerout[k] = co2[4];
  }

  // ************************************Row-3

  else if (sectorE[k] == 0 && sectorDE[k] == -2) {
    centerout[k] =co3[0];
  }

  else if (sectorE[k] == 0 && sectorDE[k] == -1) {
    centerout[k] = co3[1];
  }

  else if (sectorE[k] == 0 && sectorDE[k] == 0) {
    centerout[k] = co3[2];
  }

  else if (sectorE[k] == 0 && sectorDE[k] == 1) {
    centerout[k] = co3[3];
  }

  else if (sectorE[k] == 0 && sectorDE[k] == 2) {
    centerout[k] = co3[4];
  }

  //****************************************Row-4

  else if (sectorE[k] == 1 && sectorDE[k] == -2) {
    centerout[k] = co4[0];
  }

  else if (sectorE[k] == 1 && sectorDE[k] == -1) {
    centerout[k] = co4[1];
  }

  else if (sectorE[k] == 1 && sectorDE[k] == 0) {
    centerout[k] = co4[2];
  }

  else if (sectorE[k] == 1 && sectorDE[k] == 1) {
    centerout[k] = co4[3];
  }

  else if (sectorE[k] == 1 && sectorDE[k] == 2) {
    centerout[k] = co4[4];
  }

  ////**************************************Row-5

  else if (sectorE[k] == 2 && sectorDE[k] == -2) {
    centerout[k] = co5[0];
  }

  else if (sectorE[k] == 2 && sectorDE[k] == -1) {
    centerout[k] = co5[1];
  }

  else if (sectorE[k] == 2 && sectorDE[k] == 0) {
    centerout[k] = co5[2];
  }

  else if (sectorE[k] == 2 && sectorDE[k] == 1) {
    centerout[k] = co5[3];
  }

  else if (sectorE[k] == 2 && sectorDE[k] == 2) {
  //else {
    centerout[k] = co5[4];
  }

}

////*******************************************************//
////Defuzzification Using Centroid of Gravity Method
////*******************************************************//

float COG = ((Mfout1*centerout[0])+(Mfout2*centerout[1]))/(Mfout1+Mfout2);
```

```


  if (COG > 0) {
    digitalWrite(RiseMotorEn, HIGH);  // motor 1
    analogWrite(RiseMotorSpeed, 255);
    for (int r=0; r<=30; r++) {
      delay(COG*500);                 //40s = 1cm
      sensorValue = analogRead(A0);
      sensor = ((334-sensorValue)/(334-300))*2; // LLV. 0 = 15cm // LLV. 0 = 15cm
      Serial.println(sensor);
    }
    analogWrite(RiseMotorSpeed, 0);
  }

  else if (COG < 0) {
    digitalWrite(SinkMotorEn, HIGH);  // motor 2
    analogWrite(SinkMotorSpeed, 255);
    for (int s=0; s<=30; s++) {      //s<=34
      delay(-(COG*500));              //16.67s = 1cm
      sensorValue = analogRead(A0);
      sensor = ((334-sensorValue)/(334-300))*2; // LLV. 0 = 15cm // LLV. 0 = 15cm
      Serial.println(sensor);
    }
    analogWrite(SinkMotorSpeed, 0);
  }
  else if (COG == 0) {
    analogWrite(SinkMotorSpeed, 0);
    analogWrite(RiseMotorSpeed, 0);
  }
}
```

APPENDIX B

MATLAB GUI coding

```
function varargout = gui2(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui2_OpeningFcn, ...
                   'gui_OutputFcn',  @gui2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT



% --- Executes just before gui2 is made visible.
function gui2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui2 (see VARARGIN)

axis off
backgroundImage = importdata('uthm.jpg');
%select the axes
axes(handles.logo);
%place image onto the axes
image(backgroundImage);
%remove the axis tick marks
axis off
% Choose default command line output for GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

clc;
clear all;
clear s;
global s;
s = serial('COM3');

% --- Outputs from this function are returned to the command line.
function varargout = gui2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

 % --- Executes on button press in start.
function start_Callback(hObject, eventdata, handles)
% hObject    handle to start (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
%axes(handles.axes2)
```

```
global s;
TimeInterval=0.01;
loop=500;
time = now;
ori = now;
sensor = 90;
handles.plot = plot(handles.graph,time,sensor,'LineWidth',2,'Color',[1 1 1]);
sensor(1)=90;
time(1)=0;
count = 1;
fopen(s);

while ~isequal(count,loop)

    sensor(count) = fscanf(s,'%f');
    set(handles.present, 'String', sensor(count));
    time(count) = (now-ori)*100000;

    %axes(handles.axes2)
    set(handles.plot,'YData',sensor,'XData',time);
    set(handles.graph, 'YGrid','on',...
    'YColor',[0.043 0.518 0.78],...
    'XGrid','on',...
    'XColor',[0.043 0.518 0.78],...
    'Color',[0 0 0]);
    pause(TimeInterval);
    count = count +1;
end

fclose(s);
delete(s);
clear s;

function desired_Callback(hObject, eventdata, handles)
% hObject    handle to desired (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global s;
time = now;

a=get(hObject,'String');
a=char(a);
fprintf(s,'%s',a);
% --- Executes during object creation, after setting all properties.
function desired_CreateFcn(hObject, eventdata, handles)
% hObject    handle to desired (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```