TITLE
DESIGN AND DEVELOPMENT OF ROBOBOAT

By

ABDUL WAHAB
MUHAMMAD TALHA SAQIB
KHIZAR NASEER
MALIK FAYZ

BSc. Electrical Engineering

Department of Electrical Engineering and Technology

University of Gujrat

Session 2013-17

01,14,29,45 BSc. Electrical Engineering 2013-17

DESIGN AND DEVELOPMENT OF

ROBOBOAT

Group Members:

Abdul Wahab 13083122-001

Muhammad Talha Saqib 13083122-014

Khizar Naseer 13083122-029

Malik Fayz 13083122-045

B.Sc. Electrical Engineering

Batch (2013-17)

University Of Gujrat

Hafiz Hayat campus

DECLARATION

We students of B.Sc. Electrical Engineering, University of Gujrat, Pakistan, hereby declare that the data in this report is our own work. It do not contain any material previously published material by another party of any degree or diploma at any other university or college except where due acknowledgement is mention.

Abdul Wahab (13083122-001) _____

Muhammad Talha Saqib (13083122-014) _____

Khizar Naseer (13083122-029) _____

Malik Fayz (13083122-045) _____

I certify that these students of B.Sc. Electrical Engineering, University of Gujrat, Pakistan, worked under my supervision and the above stated declaration is true to the best of my knowledge.

Name (Dr. Syed Muhammad Wasif)
Assistant Professor, Department of Electrical Engineering,
University of Gujrat, Punjab, Pakistan.
Email: syed.wasif@uog.edu.pk

Dated: _____

Abstract

The aim of this project is to design and develop a small robotic water surface vehicle which can navigate in shallow waters i.e. on the surface of rivers, lakes, and coastal sea waters to perform monitoring and observational tasks autonomously. A series of digital sensors such as digital compass, GPS module and ultrasonic sensors are equipped on the boat. These sensors will play a vital role in navigation process through a series of set points without the involvement of any human operator. The GPS and the digital compass working simultaneously will provide the current location and current heading of the boat. The control system will perform some calculations based on the numerical equations to calculate the required heading towards destination. PID controllers will steer the boat towards destination heading.

Table of Contents

TABLE OF CONTENTS III

TABLE OF FIGURES V

TABLE OF TABLES VI

CHAPTER 1 1

1.1 PROBLEM STATEMENT 1

1.2 OBJECTIVES 1

CHAPTER 2 3

2.1 STABILITY 3

2.2 AUTONOMOUS NAVIGATION 5

2.3 OBSTACLE DETECTION & AVOIDANCE 6

2.4 WIRELESS COMMUNICATION 7

CHAPTER 3 8

3.1 PHYSICAL STRUCTURE 9

3.1.1 Hulls 9

3.1.2 Deck 11

3.1.3 The Steering and Propulsion Assembly 11

3.2 NAVIGATIONAL ALGORITHM 13

3.2.1 First Approach 15

3.2.2 Final Approach 16

3.3 OBSTACLE DETECTION AND AVOIDANCE 19

3.3.1 Detectable Area 22

3.4 PID CONTROL ALGORITHM 22

3.5 COMPONENTS 24

3.5.1 Digital Compass 24

3.5.2 GPS Module 25
3.5.3 Ultrasonic Sensors 26
3.5.4 Brushless DC Motor (BLDC) 27
3.5.5 Servo Motor 27
CHAPTER 4 29
4.1 DIGITAL COMPASS 29
4.1.1 True North and Declination Angle 29
4.1.2 Difference in Position and the Relative Angle 30
4.2 GPS MODULE 31
4.2.1 Hot and Cold Start 31
4.3 ULTRASONIC SENSORS 33
4.4 BLDC MOTOR 33
4.5 COMPASS CONTROLLED SERVO 34
4.6 NAVIGATION 34
4.7 OBSTACLE AVOIDANCE 36
CHAPTER 5 37
5.1 CONCLUSION 37
5.2 DISCUSSION 38
5.3 FUTURE WORK 38
APPENDIX A. 41

Table of Figures

Figure 3.1: Block diagram 9

Figure 3.2: Dimensions of the roboboat 10

Figure 3.3: Template pieces of one hull 10

Figure 3.4: Steering and propulsion assembly 12

Figure 3.5: Side-view of roboboat 13

Figure 3.6: Navigation flow diagram 14

Figure 3.7: Obstacle avoiding flow diagram 20

Figure 3.8: Placement of ultrasonic sensors on the boat. 21

Figure 3.9: PID flow diagram 23

Figure 3.10: PID MATLAB results 23

Figure 3.11: Digital compass 24

Figure 3.12: GPS module 25

Figure 3.13: Ultrasonic sensor 26

Figure 3.14: BLDC motor and ESC 27

Figure 3.15: Servo motor 28

Figure 4.1: Compass output 30

Figure 4.2: In the case of reading from the ultrasonic sensors as shown above, the obstacle is nearest to the sensor 4. This will trigger the conditional in the code to check on both sides of the sensor 4. Since the output from sensor 3 is greater the sensor 5 so the boat will turn towards sensor 3. 32

Figure 4.3: Graphical representation of ultrasonic sensor. [11]. 33

Figure 4.4: Navigation path with waypoints 35

Figure 4.5: Boat during navigation testing 36

Table of Tables

Table 3.1: Obstacle avoidance control standards 21

Table 3.2: Specification of GY-271.....24

Table 3.3: Pin configuration of GY-271 25

Table 3.4: GPS specifications 25

Table 3.5: GPS pin configuration 26

Table 3.6: Ultrasonic pin configuration 26

Table 3.7: Power supply pin configuration of ESC 27

Table 3.8: Control pin configuration of ESC 27

Table 3.9: Pin configuration of the servo motor.....28

Table 4.1: Change in the output of GPS when moved a distance from a reference point (74.5310, 32.4925).....31

Acknowledgements

We'd like to thank Dr. S. M. Wasif, our supervisor for helping us with this project.

Abdul Wahab

_____

Muhammad Talha Saqib

_____

Khizar Naseer

_____

Malik Fayz

_____

Abbreviations

Abbreviation	Meaning
BLDC Motor	Brushless Direct Current motor
USVs	Unmanned Surface Vehicles

Chapter 1

INTRODUCTION

As the technology advances steadily, the need for autonomous machinery and autonomous vehicles grows dire. Scientists are looking at the remotest of the places in search of new discoveries. On the other hand Military engineers are searching for modern ways to keep tabs on the enemy's movements. Military espionage is a very important part of modern tactical warfare, and every alternative, that requires less human life endangerment, is being prioritized. In wake of such urgent need to progress, unmanned vehicles such as Air Drones, Robo-Boats and Remote controlled boats are gaining immense popularity in defense field.

1.1 Problem Statement

After years of fighting terrorism, Pakistan Army and PAF have evolved a great deal but due to the lack of naval warfare experience, Pakistan Navy is still lagging behind its other counterparts on land and air. Pakistan initiated it's highly active Unmanned Aerial Vehicle (UAV) development program in mid of 2000s, but there is still a lot of work that is needed to be done in the fields of Unmanned Surface Vehicles (USV). This project will begin with a robotic boat which will be able to float, navigate and avoid obstacles in its way.

1.2 Objectives

The main Objectives of this project are listed as:

- To design and implement a physically feasible structure of the boat, that can support its weight and the weight of the equipment on board.
- To create a steering system, that will help the boat to navigate.
- To construct a navigational algorithm for the boat, complete with:
    - Speed control.
    - Current location.
    - Current direction.
    - Starting point.
    - Way points.
    - Destination.
- To formulate an obstacle avoiding algorithm for the boat.

Chapter 2

BACKGROUND AND LITERATURE SURVEY

In this chapter pervious different approaches for the design and development of USVs will be discussed. This will help to develop a better understanding of the current boundaries of this field.

As the name implies, the Unmanned Surface Vehicles (USVs) operate without the need of a human operator. With the availability of large bandwidths and longer ranges on the wireless communication, and with the advancements in the GPS technology, the growth in the USV field has been exponential.

First of all, let's list the stages in the development of RoboBoat.

1. Stability (Physical Design)
2. Autonomous Navigation
3. Obstacle Detection and Avoidance
4. Wireless Communication

Each of these steps will now be discussed in detail.

2.1 Stability

The most essential part of the RoboBoat design is its physical structure. A good physical design helps the boat to achieve all of the other objectives easily.

In 1993 MIT developed the first USV. It was named ARTEMIS. This surface craft helped to test the navigational boundaries of this project. This boat was modelled after a fishing trawler. It was almost a failure in the stability department due to its

small size. [1]. This proved a very important point for the researchers of the time that RoboBoat are not like regular boat. Their smaller size makes it a difficult task to achieve stability with the same model that was very stable in a normal sized boat.

Equipped with this knowledge engineers at MIT developed another boat. This time the boat was modelled after a kayak. [2]. It was noted that the stability of the boat increased significantly.

A new USV was developed in 1996 and it was named as Autonomous Coastal Exploration System (ACES). The objective of this craft was to conduct surveys to collect data for scientific experimentation. [3]. It completed its first survey in December 1997. In 1998 the ACES was upgraded to help it perform better in water, and was renamed as AutoCat. AutoCat was modelled after a catamaran. [4]. This system had excellent stability on the water surface. This was a turning point in the history of unmanned surface vehicles.

All this experimentation was going on in MIT, but after the success of AutoCat, newer programs were inspired out of MIT.

In 2011, a team of students from the University of Rhodes Island, took part in a competition. The boat was of the catamaran type. [5]. This boat was named as RAMboat, which was obviously a reference to the ramming competition. The boat offered a payload of 250lbs. The electronics were encased in a water tight aluminum encasement. The case also served as a heat sink for the electronic components.

2.2 Autonomous Navigation

The RAMboat used two 18lb Sevylor trolling motors for the propulsion. A 50A motor controller was used with four 14.8V LiPo batteries.

This boat was equipped with two FitPC single board computers. One was commissioned with the task of mission control while the other computer was equipped for image processing.

The boat had 4 main sensors:

1. LIDAR sensor
2. IR sensor
3. Microsoft HD 5000 webcam
4. GPS

The project proved to be of great success and helped URI win the competition.

In 2012 a research paper was published by T. Young and M. Jenkins from the York University, Canada. In their paper they brought this fact to the attention that there was little to no significant work being done in the field of unmanned surface vehicles. [6]. They proposed that an RC motorboat could be repurposed to be a USV. The further proposed the addition of a GPS, a digital compass and digital cameras for vision to add sensing capabilities to the boat. This boat was able to communicate back with the control station. It was a successful project.

In 2013 a RoboBoat was designed for the participation in a competition. It was named the PropaGator. The PropaGator was a completely autonomous vehicle. It was conceived for the sole purpose of participating in the Association for Unmanned Surface Vehicles Systems International's 2013 RoboBoat competition.

[7]. It was developed at the University of Florida by Dr. E. Schwartz and his team. The Robotic Operating System (ROS) was used for the software base of the project.

The design for this boat was inspired by US navy warships. It was modelled after a US warship named HSV-2 Swift. It is basically a Catamaran which indicates stable voyaging capabilities. The boat was divided into multiple different layers. It had 4 trolling motors for propulsion and 5 batteries for the power supply.

One of the special feature of this boat was that it had a small landing pad on its back for a quadcopter. Designers also kept in mind the safety of the users. Trolling motor guards were installed on the trolling motors so that finger sized objects couldn't reach the propellers from the sides.

2.3 Obstacle Detection & Avoidance

In 2014, Georgia Institute of Technology, Atlanta, developed a RoboBoat for 2014 AUVSI RoboBoat Competition. The boat was modelled as a trimaran. It was equipped with a combination of sensors including a GPS module, a LIDAR, sonar sensors, underwater light sensors and stereo cameras. [8]. The competition called the boat to complete a number of given tasks. Each of these sensors helped the boat to pass through these missions.

At the end of this project, the boat was to be able to do the following number of tasks autonomously:

1. Obstacle Avoidance
2. Automated Docking
3. Acoustic Beacon Positioning
4. Under Water Light Identification

The Obstacle detection and avoidance was done with the help of LIDAR system.

2.4 Wireless Communication

In 2012, a RoboBoat was designed by the department of Computer Science and Engineering in York University, Canada. The boat was able to communicate wirelessly with the base station. The boat was equipped with an 802.11g wireless card for wireless communication.

The 2012 RAMboat's wireless communication was provided by a 1 watt bridge that connected the boat to the base station.

The 2013 PropoGator used a wireless router to communicate with the base station. A quad copter was also able to communicate with the boat's system and coordinate a perfect landing on the back of the boat.

The C&C technologies is an engineering firm that has develop a new and a better model for the USV. With the majority of its hull submerged underneath the surface of water, only a small bit of it remains above the surface, for the communication with the control station.

Chapter 3

METHODS

The first task was to design the physical structure of the RoboBoat. To achieve this, instead of designing a new boat, team followed a pre designed model, which had been previously used to implement the RoboBoat project. This model was based on a Catamaran boat, and was selected due to its ability to move in a straight line.

Next a steering mechanism had to be created for the boat. For this purpose a rudder based system was used that was controlled by a servomotor with the help of pushrods and rudder horns. The rudder was actually a PVC pipe that rotates on a fixed aluminum pivot that was screwed to a base plate.

The construction of navigational algorithm was a very important part of this project. A GPS module, a digital compass and a microcontroller board was used for this. The GPS module provided the current location of the boat while the digital compass provided the current heading of the boat. This data was fed to the microcontroller board, which applied distance formula and trigonometric formulas to obtain the distance and the correct heading to the destination. Then the microcontroller board used the PID control algorithm, with the calculated data used as the measured value and the set points as the reference value, to steer the boat with the help of the servo motor.

The obstacle avoiding algorithm was implemented using an array of ultrasonic sensors. These sensors measured and provided the data in case of any obstacle

approaching the boat. This data was utilized to set limiting values for detection and avoidance of the obstacles. If the obstacle got any closer than the prescribed safe distance, then the boat wasa steered out of the harm's way.

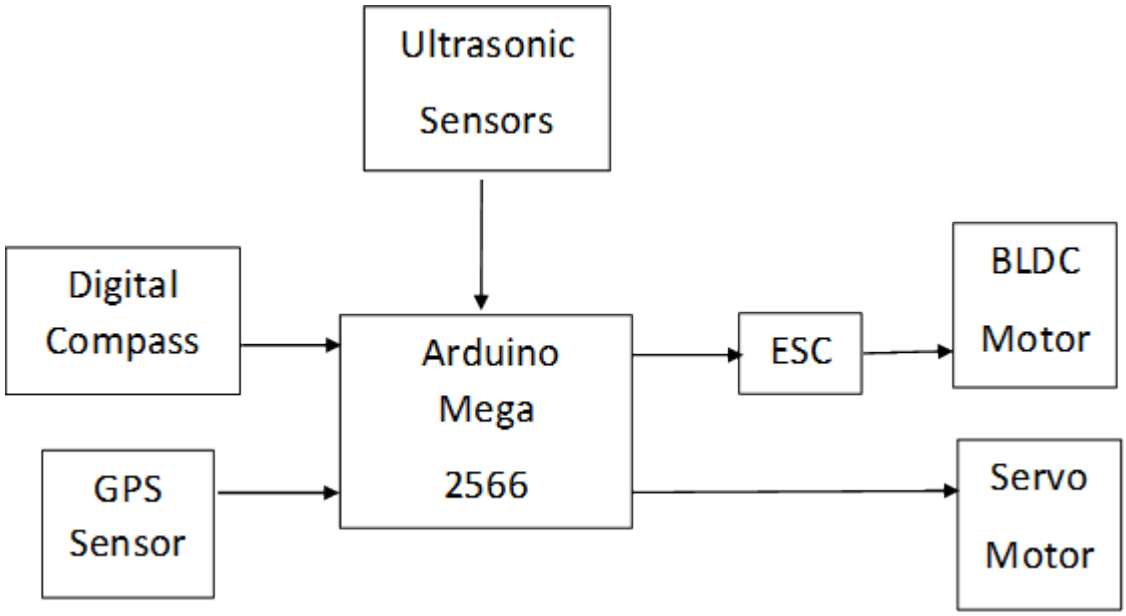

Figure 3.1: Block diagram

3.1 Physical Structure

A pre-designed Catamaran was used for this project. A catamaran is boat with 2 hulls and deck that connects the hulls to each other.

3.1.1 Hulls

Each of the hull was divided into 14 parts in a shape that can be called triangular by a long shot but with rounded edges. Each block was 4 inches thick and was carved with a help of pre-designed templates. The extruded polystyrene (XPS) boards with 2 inch thickness was used so the hull had to be divided each into 28 parts, with every part 2 inches thick. The templates can be downloaded from the website. [9].

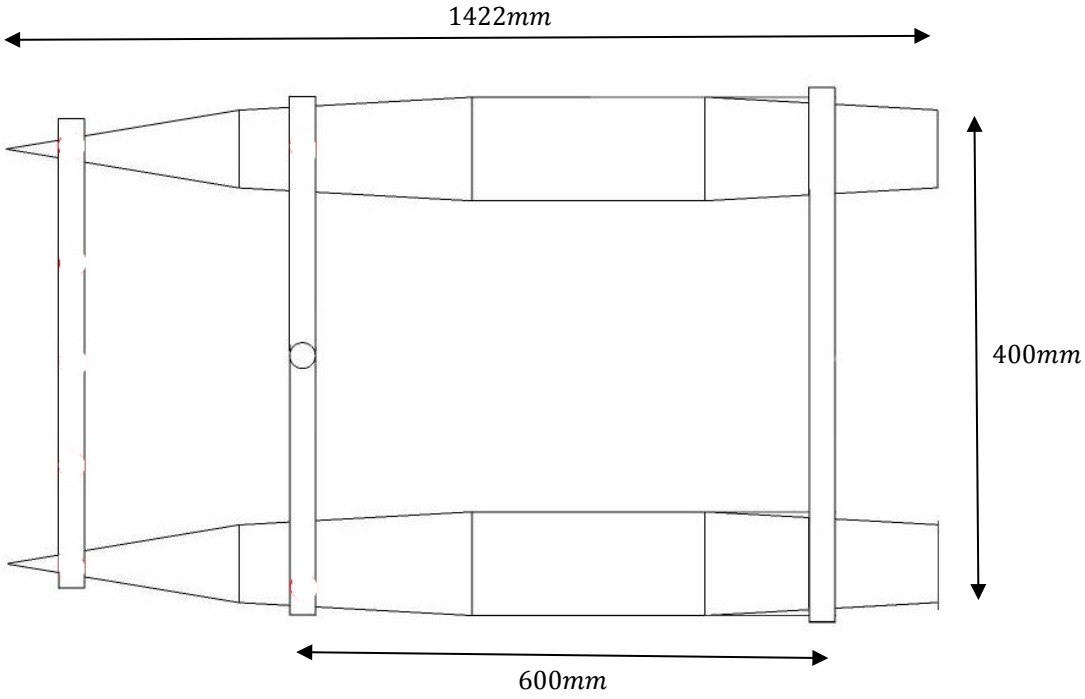

Figure 3.2: Dimensions of the roboboat

Figure 3.3: Template pieces of one hull

First of all templates were printed on paper and then the design was transferred onto the XPS boards. Then with the help of a paper cutter, the block were roughly carved out of the boards. These rough blocks were later smoothed with the help of a sand paper. The cutting and finishing of a single block took about an hour.

After all the parts were finished, they were glued together in pairs and were left to dry for 24 hours. When the pairs had dried off, they were smoothed again with the sand papers and then the whole hulls were glued for the final time. After completion, the hulls were levelled again with the sand papers. The whole process took many weeks to complete since the team could only get together on the weekends to perform this extraneous physical exertion.

The hull were then wrapped with fiberglass fabric and epoxy resin was coated on them, to harden the fiberglass and to make it water proof.

3.1.2 Deck

To make the deck was fairly easy as compared to the making of the hulls. The deck is $600mm$ long and $400mm$ wide with the thickness of $20mm$. To mount the deck onto the hulls, holes of $6mm$ diameter were drilled into the board. Foam anchors were inserted into the hulls to support the screws used to connect the deck and the hulls.

M5 screws with ‘eye’ were used to secure the deck.

3.1.3 The Steering and Propulsion Assembly

There were two methods, under consideration, for the propulsion assembly.

First method was to use underwater propulsion assembly. For this to work there were two alternatives. One, a water proof motor could be used with a water propeller connected to it, underwater. Two, a normal motor could be used with a gear box to transfer it rotation to the propeller underwater.

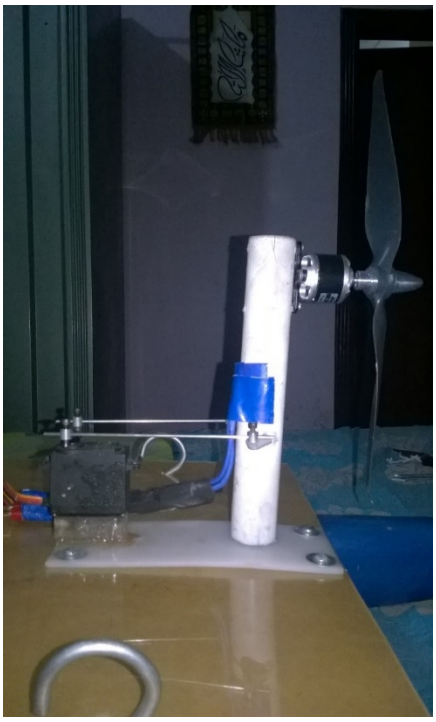

Figure 3.4: Steering and propulsion assembly

Second method was to use a propulsion assembly that was above the surface of the water. The normal brushless dc motor (BLDC) and an air propeller was sufficient for this to work. The advantage of this assembly was that the propeller would not get caught in the garbage or plants or rock that might be under the surface of the water, unseen by us. In this project, the second method has been implemented.

To develop a steering mechanism, a servo motor was used. The propulsion motor (BLDC) was connected to a polypropylene random (PPR) tube of $25mm$ diameter

and $170mm$ length. This tube sits on an aluminum pivot of $20mm$ outer diameter and $50mm$ of length. The pivot was fixed to the baseplate while the PPR tube was able to freely revolve around it.

Rudder horns were connected to the side of the PPR tube and the motion of the servomotor was transferred to the PPR tube with the help of two push rods of $100mm$ length each. The whole assembly was fitted on a baseplate made of polyvinyl chloride (PVC) sheets, $3mm$ thick.

A $10\ inch$ propeller is used for the propulsion of the boat. The complete physical structure of the boat is shown below:

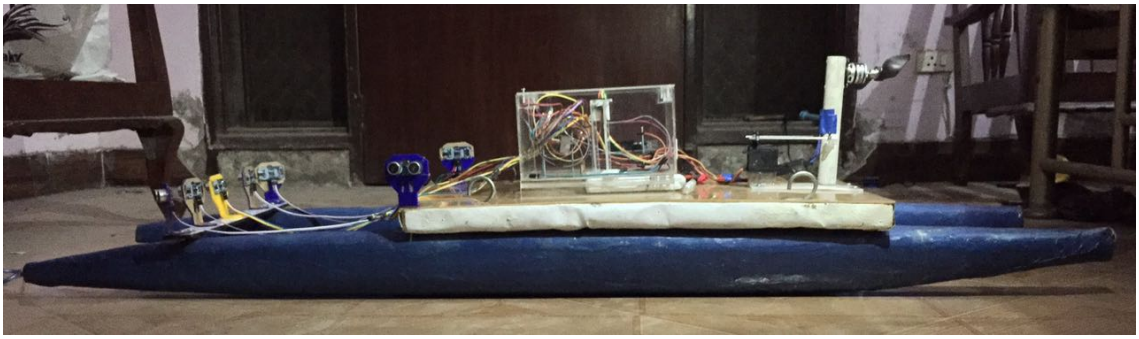

Figure 3.5: Side-view of roboboat

3.2 Navigational Algorithm

The navigational algorithm has two main parts; propulsion and steering. For the purpose of propulsion all that was needed was that the boat's control system to understand was to stop when the destination was reached or to slow down while turning so to reduce the turning radius. This was achieved fairly easily by the help of way points. And since the boat is a catamaran it didn't need for the boat to turn very often.

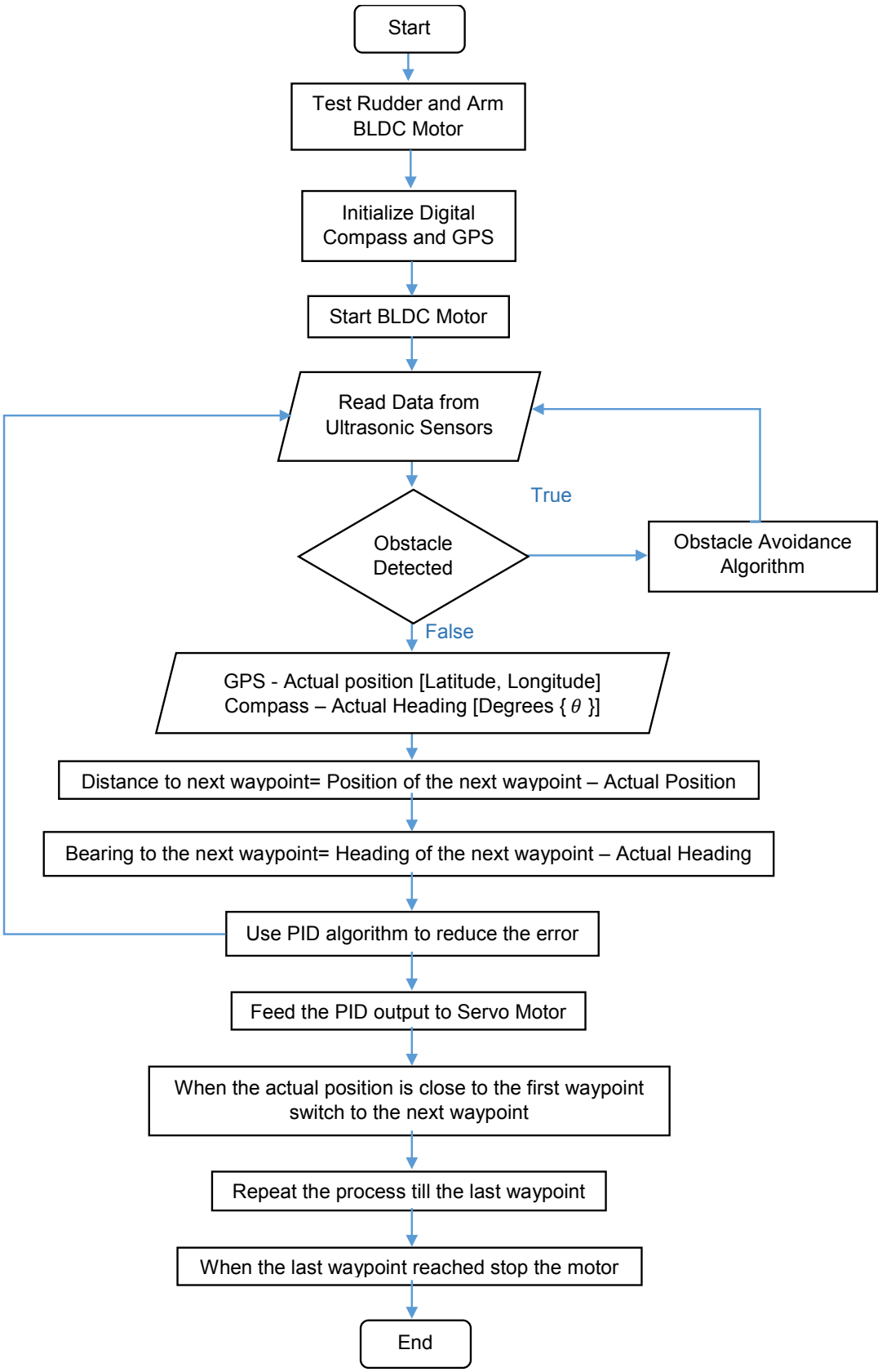

Figure 3.6: Navigation flow diagram

Secondly, the boat needed to navigate onto the correct path to reach its destination. This was done with the help of a GPS and a digital compass.

3.2.1 First Approach

The GPS and the digital compass provide the current location and the current heading of the boat. To calculate the distance of the boat from the next way point the distance formula was used which is given as:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.1}$$

This formula is obtained by using the Pythagoras' theorem. Here (x_1,y_1) represent the current coordinates of the boat where as (x_2,y_2) are the set-point coordinates. 'd' is the displacement here.

Also the bearing for the boat can be calculated easily by a simple trigonometric formula below:

$$\theta = \tan^{-1}\frac{(y_2 - y_1)}{(x_2 - x_1)} \tag{3.2}$$

Here θ is the counter clockwise angle w.r.t. positive x-axis. This was where the first problem arose. The digital compass gave its output in degrees and that angle

was clockwise w.r.t. the North which was the positive y-axis. To bring the two outputs to the same reference point, $90^{\circ}$was added to the θ to shift it forward from positive x-axis to positive y-axis and the answer was subtracted form 360° to reverse it direction from counter clockwise to clockwise. This is shown in the following formula:

$$\theta' = 360^{\circ} - (\theta + 90^{\circ}) \tag{3.3}$$

This equation synchronized the calculated bearing and compass output to the same reference point.

Now that the control had the distance and the bearing to the next set-point it should have been able to navigate our boat easily. But there was again a slight complication here.

Since 'd' represents the shortest distance from point 1 to point 2 which would obviously be a straight line. The output of this equation would be in a 2-dimentional plane. But the Earth is a spherical ball of dirt that is very much in a 3-dimentional space, so it rendered any distance equations that result in a straight line useless for accurate distance measurement. This problem and it solution is discussed in the next topic.

3.2.2 Final Approach

After testing the codes made in accordance with the previous algorithm, it was discovered that the used formula was calculating the distance inaccurately. After conducting extensive tests, it was discovered that the error in the calculation was proportional to the distance between two points.

$$error \propto distance\ between\ two\ points \tag{3.4}$$

This means that the error in the calculation increases as the distance increases. This is in compliance with the structure of the Earth. Since the surface of the Earth seems flat over short distances thus the error in the calculations was negligible. But as the distance increased, the curvature of the Earth's surface also came into play and had to be accounted for in the final calculations.

To overcome this problem the team researched some formulas for calculating distance accurately over the surface of the Earth. Here is the formula that provided acceptable results with negligible amount of error:

$$x = \{sin\,(\frac{\Delta\varphi}{2})\}^{2} + cos\,\varphi_1.cos\,\varphi_2.\{sin\,(\frac{\Delta\lambda}{2})\}^{2} \qquad (3.5)$$

$$y = 2.tan^{-1}\,(\frac{\sqrt{x}}{\sqrt{1-x}}) \qquad (3.6)$$

$$d = R.y \qquad (3.7)$$

This formula is called the *"Haversine formula"* [10]. This is used to calculate the circular distance instead of the distance in a straight line. This formula is commonly used in spherical trigonometry.

Where φ is latitude, λ is longitude, R is earth's radius (mean radius = 6,371km); note that coordinates need to be in radians to pass to trig functions. Here 'd' is the calculated distance.

And the angle to the destination can be found by using the tangent inverse formula which is given as:

$$\theta = tan^{-1}\left(\frac{sin\,\Delta\lambda.cos\,\varphi_2}{cos\,\varphi_1.sin\,\varphi_2 - sin\,\varphi_1.cos\,\varphi_2.cos\,\Delta\lambda}\right) \qquad (3.8)$$

This formula is known as *"Forward Azimuth"* formula [10].

After conducting tests of the code made by using these formulas as the base and comparing the results with the Google maps, it was discovered that θ is referenced

w.r.t. the North and increased in the clockwise direction. So there was no need angle synchronization in this formula.

When this data was collected, the PID control algorithm was used to steer the boat towards the destination point. The PID algorithm works on the principal of error removal with the help of feedback. The destination is called the ‘reference value’ i.e. the value that is to be achieved and the current position is known as ‘measured value’. The calculated error is the difference between the reference value and the measured value. The error is fed into the PID control equation to calculate the optimized output.

3.3 Obstacle Detection and Avoidance

Obstacle detection is the most important part of the boat’s working as the boat must not collide with any obstacle during its operation. This part of the code is prioritized over all other parts.

To achieve this a system had to be defined, which would determine how many sensors would be needed for the Obstacle detection, and their placement on the boat.

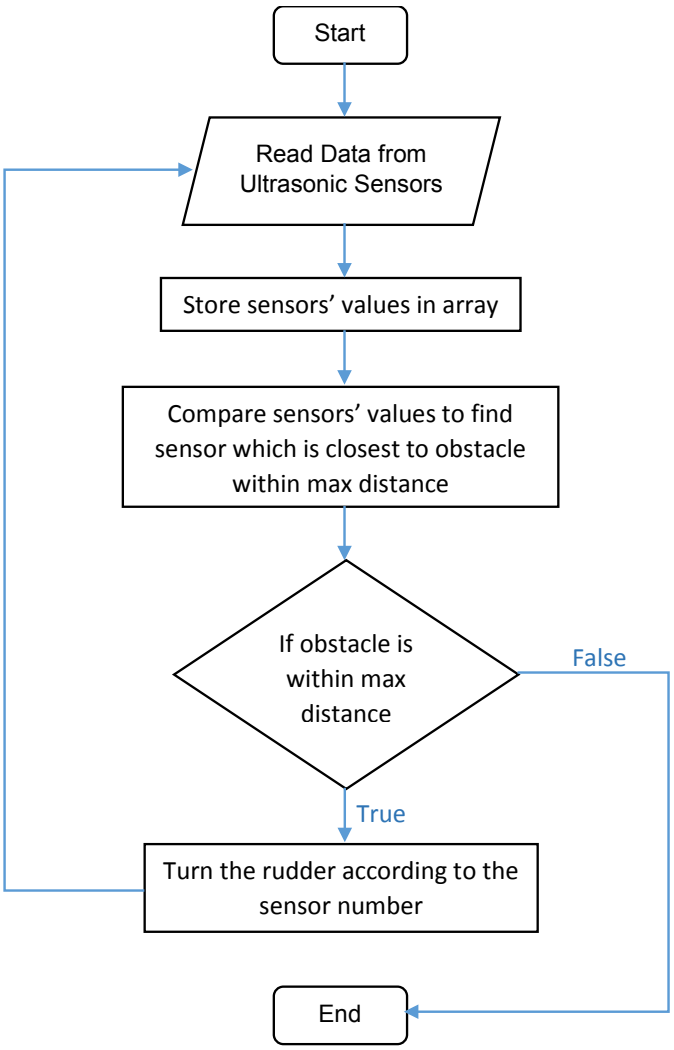

Figure 3.7: Obstacle avoiding flow diagram

After much discussion on the methodology, it was decided to use 7 ultrasonic sensors for the obstacle detection system on the boat. The placement of the sensors is shown below.

The red dots show the positions of the ultrasonic sensors on boat. And the red arrows show the direction in which they are pointed.

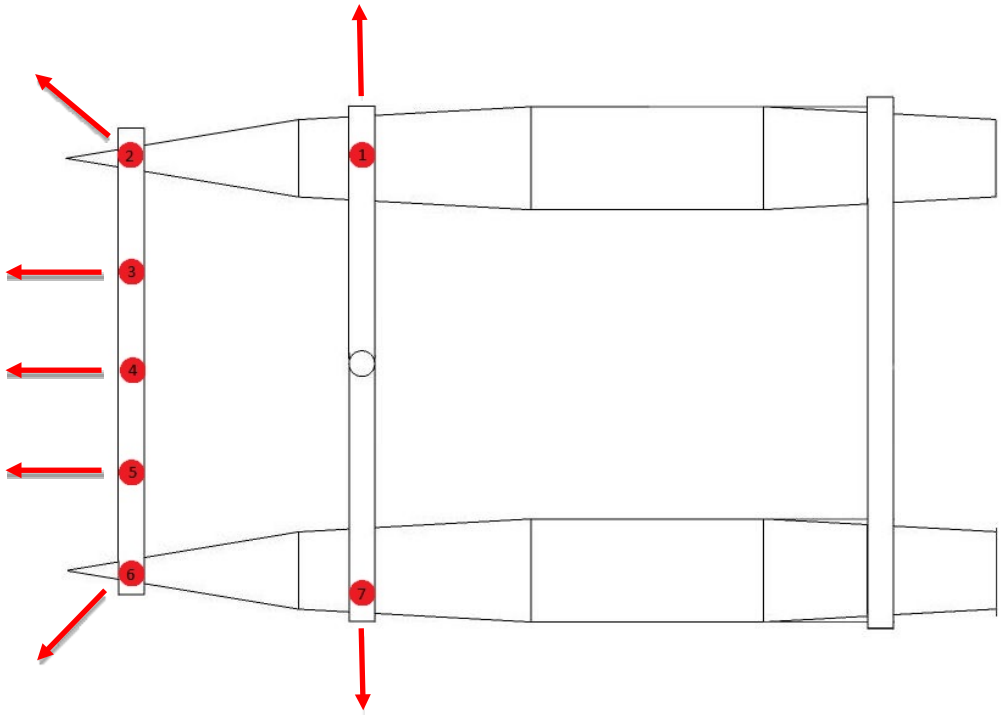

Figure 3.8: Placement of ultrasonic sensors on the boat.

After the obstacle had been detected, the next step was to avoid that obstacle. For this to work effectively, first of all the boat needed to determine which of the sensor was closest to the obstacle. After that a table of responses was devised for the boat in case of obstacle detection by each sensor separately. The table is given below.

Table 3.1: Obstacle avoidance control standards

	Sensor1	Sensor2	Sensor3	Sensor4	Sensor5	Sensor6	Sensor7
Response (angle)	15	30	45	±60	-45	-30	-15

These responses would be added to the default position of the servo motor. Thus providing the servomotor with a total movement range of 120 degrees. When the obstacle was detected by either of the sensors on farthest from the center i.e.

sensors 1 or 7, then the boat only needs to turn a small degree because the obstacle was already on the side of the boat. On the other hand if the obstacle was detected by the center most sensor i.e. sensor 4, then the boat need to turn to a maximum degrees of angle allowed by the propulsion assembly.

3.3.1 Detectable Area

After a successful run, the team were interested in determining the area for which the sensor can detect an object. After a few tests, it became clear that the sensor detect objects that are directly in front of it and not in a sector as was anticipated. Thus the objects that are not right in front of the sensor, will not be detected by the sensor.

This was an important find. This would help in designing a combination of Ultrasonic sensors that will accurately detect and measure the distance from obstacles around the boat.

3.4 PID Control Algorithm

To test the PID control algorithm, first of all a code was made for it in the MATLAB to see its error removing properties. A very simple code was made with just the PID equation and the related variables. By adjusting the values of kp, ki and kd almost all of the oscillations and overshoot was removed. The code for this can be found in the Appendix A.

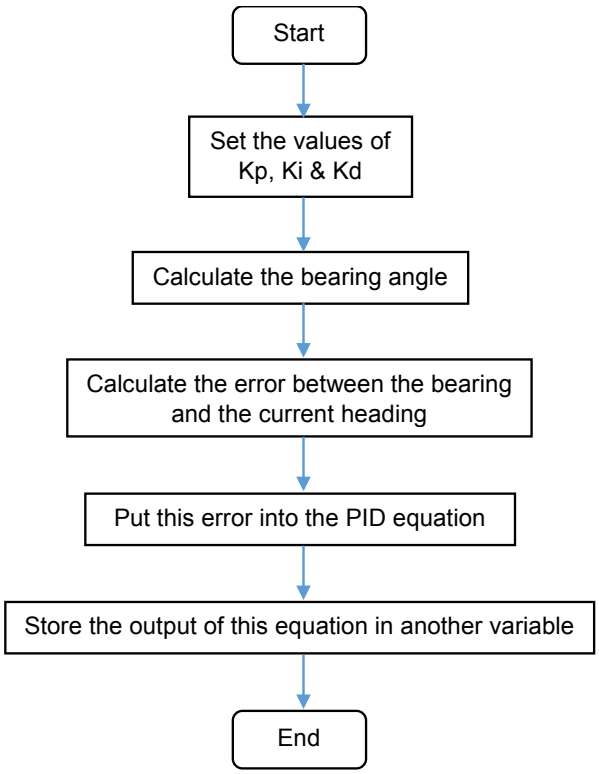

Figure 3.9: PID flow diagram

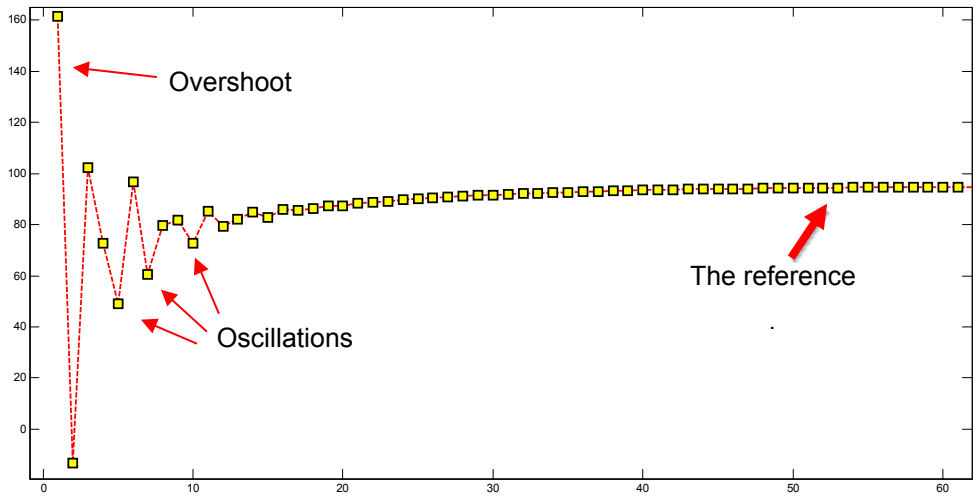

Figure 3.10: PID MATLAB results

There are two methods to calculate the values for the constants kp, ki and kd for BLDC motor. The first is to make a mathematical model of the motor and then calculating the values from that mathematical model. The second method is by 'Hit

and Trail', The value of the constants was gradually increased and note the values at which our motor shows least overshoot and oscillations.

3.5 Components

In this section, the components used in the design of this RoboBoat will be described briefly.

3.5.1 Digital Compass

The digital compass that was used in this project is GY-271 model.

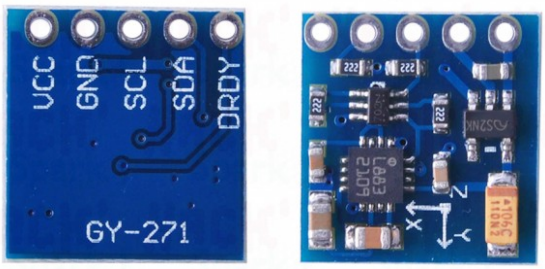

Figure 3.11: Digital compass

This compass is designed to sense low magnetic field and give a digital output. The small size of this compass allows it to be used for the small robotic projects. The sensor takes input in the form of magnetic field and gives output as voltage magnitudes for three axis. This voltage was used to calculate the magnitude of magnetic field in different directions.

The specifications of the sensor are given below:

Table 3.2: Specification of GY-271

Power	$3V - 5V\ DC$
Measuring Range	$\pm$ 1.3 – 8 $Gauss$
Dimensions	$14.8 \times 13.5 \times 3.5mm$

The pin configuration of the module is given below:

Table 3.3: Pin configuration of GY-271

VCC	$3V - 5V\ DC$
GND	ground
SCL	analog input (A5)
SDA	analog input (A4)
DRDY	not connected

3.5.2 GPS Module

The GPS module that was used for this project is NEO-6M model.

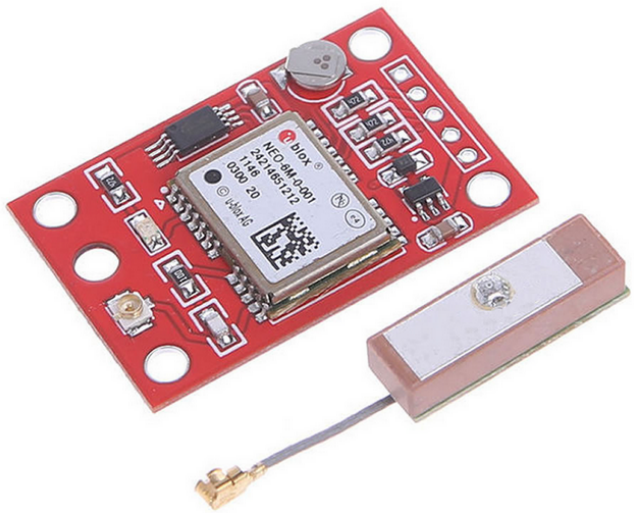

Figure 3.12: GPS module

This sensor is used to get coordinates of the current position of the boat in the form of a latitudes and longitudes. These readings were later utilized to calculate the distance and angle to the next way-point. It has a built-in antenna that acts as a receiver. The specifications of the module are given below:

Table 3.4: GPS specifications

Supply	$2.7V - 3.6V\ DC$
Configuration pins	3

The module has 4 pins. The pin configuration is given below:

Table 3.5: GPS pin configuration

VCC	$5V\ DC$
GND	Ground
TX	RX (designated on the Arduino)
RX	TX (designated on the Arduino)

3.5.3 Ultrasonic Sensors

The Ultrasonic sensors that was used in this project are HC-SR04 model.

Figure 3.13: Ultrasonic sensor

This sensor can detect and measure the distance from any object accurately within the range of $2cm - 400cm$. These sensors have an accuracy of almost $3mm$. These sensors have three essential parts:

a) An ultrasonic transmitter

b) A receiver

c) A control circuit

The Ultrasonic Sensors have 4 pins. The pin configuration is given below:

Table 3.6: Ultrasonic pin configuration

VCC	$3V - 5V\ DC$
GND	Ground
TRIG	Trigger pin (designated in the code)
ECHO	Echo pin (designated in the code)

3.5.4 Brushless DC Motor (BLDC)

The BLDC that was used in this project is Emax XA2212 model.

Figure 3.14: BLDC motor and ESC

A Brushless DC (or BLDC) is a synchronous motor that is powered by a DC supply through an inverter that converts the DC into AC. This AC was then supplied to each phase of the motor by a closed loop control system. The BLDC motor has 3 phase wires. The motor was connected to an ESC which has 5 pins. 2 of these pins for battery supply while 3 pins are for speed control of the motor. The pin configuration is given below:

Table 3.7: Power supply pin configuration of ESC

RED pin	+ive terminal of battery
BLACK pin	-ive terminal of battery

Table 3.8: Control pin configuration of ESC

YELLOW pin	PWM pin (designated in the code)
ORANGE pin	$+5V$ (not connected)
BROWN pin	ground

3.5.5 Servo Motor

The Servo motor used in this project Tower Pro MG945 model.

Figure 3.15: Servo motor

The servo motor was used to turn the steering assembly of the boat. It was a high-torque motor. These motors have low speeds but relatively high power. Servo motor has 3 pins. The pin configuration is given below:

Table 3.9: Pin configuration of the servo motor

YELLOW pin	Position input signal
ORANGE pin	$+5V$
BROWN pin	Ground

Chapter 4

RESULTS

In this section, the testing of each component will be discussed separately and in combination with the other components.

4.1 Digital Compass

When the package was first opened, that digital compass came in, it was discovered that it had male connectors with it. These male connector had to be soldered to the module before testing. After soldering the connectors, the sensor was connected to Arduino UNO. The first example was uploaded that came with its Arduino library to test run the sensor. This attempt was unsuccessful due to reasons still unknown to us. But after researching about the specific model of our compass and its compatible libraries, it was finally managed to obtain a reading on the 'Serial Monitor' of the Arduino IDE. After the first triumphant wave of excitement of success wore off, it was discovered that the compass was giving the wrong readings. The North that the compass was pointing was way off the original North that was measured using the path of the Sun (considering that the Sun rises in the East).

4.1.1 True North and Declination Angle

This sent the team into another dive of research to discover the source of this error. It was discovered that the magnetic field of the Earth is not a fixed entity, but moves with the passage of time.

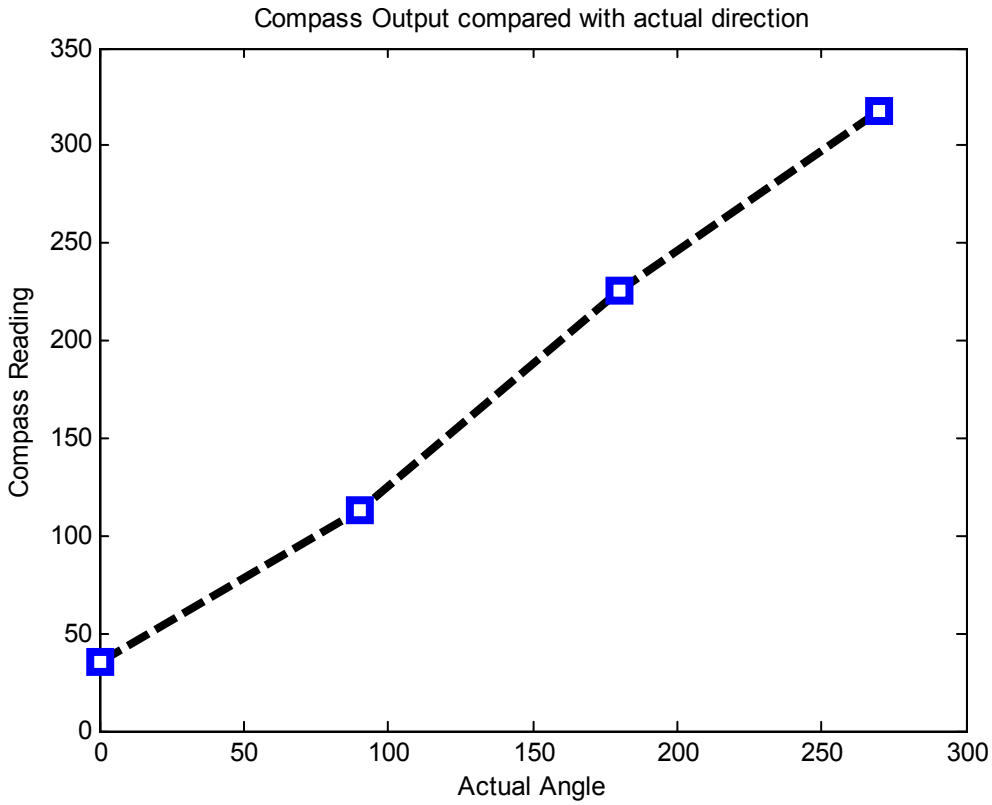

Figure 4.1: Compass output

The North that the compass was measuring was actually the magnetic North and "not" the true North. Fortunately the remedy to this problem exists. All that is needed to be done is to add a declination angle to your original reading to calculate the actual angle. This declination angle acts as an offset value which decreases the so called "zero-error" of the sensor.

4.1.2 Difference in Position and the Relative Angle

This was not the only problem with the module though, as it became apparent after a few more test runs. The module had a very serious measuring error.

When moved from 0° forward, the output varied very slowly and when it had reached the half rotation, which should actually be 180°, it was still far behind it. And after the value had been achieved, the reading varied very rapidly compared to the compass movement.

The solution of this problem still evades the team.

4.2 GPS module

Like the compass, GPS had to be soldered to male connectors before working. After soldering the connectors, GPS was connected to Arduino UNO and the first basic example that came with the library called 'TinyGPS' was uploaded. As expected it failed to deliver any significant results on the first try. So the team began to read about the working of the GPS module and how it operated. It was discovered that the GPS module should be able to receive data from at least 4 out of the 24 GPS satellites to calculate the 4 parameters i.e. Latitude, Longitude, Altitude and time. For the project only two out of four of these parameters were required i.e. Latitude and Longitude.

Table 4.1: Change in the output of GPS when moved a distance from a reference point $(74.5310,\ 32.4925)$

Distance Moved	Current Longitude	Current Latitude
$9m$	74.5311	32.4925
$11m$	74.5310	32.4926
$14m$	74.5311	32.4926

4.2.1 Hot and Cold Start

The GPS required some time to lock onto these satellites. If the GPS was in line of sight of these satellites i.e. outdoors, then the locking on took less time, on the other hand if GPS did not have a line of sight connection i.e. indoors, then the locking on took significantly longer time. This was where the problem arose. Since boat was indoors the GPS was taking ridiculously long time to lock on. In fact it was kept running for 1 hour without any success.

Further studies revealed that the GPS has two starting conditions: a) a cold start b) a hot start. A cold start is when the GPS hasn't been in use for a long time and the previous values of the positions of satellites are very inaccurate. When this condition was true, the GPS took longer time to lock on. On the other hand a hot start is when the GPS was in use very recently (about less than half an hour or so). When this condition was true then the GPS would lock on to satellites almost instantaneously (less than a minute usually).

Equipped with all of this knowledge, team once again fired up the engines of their Arduino and there it was. After waiting some time computer began to receive the NMEA sentences on the 'Serial Monitor' of the Arduino IDE.

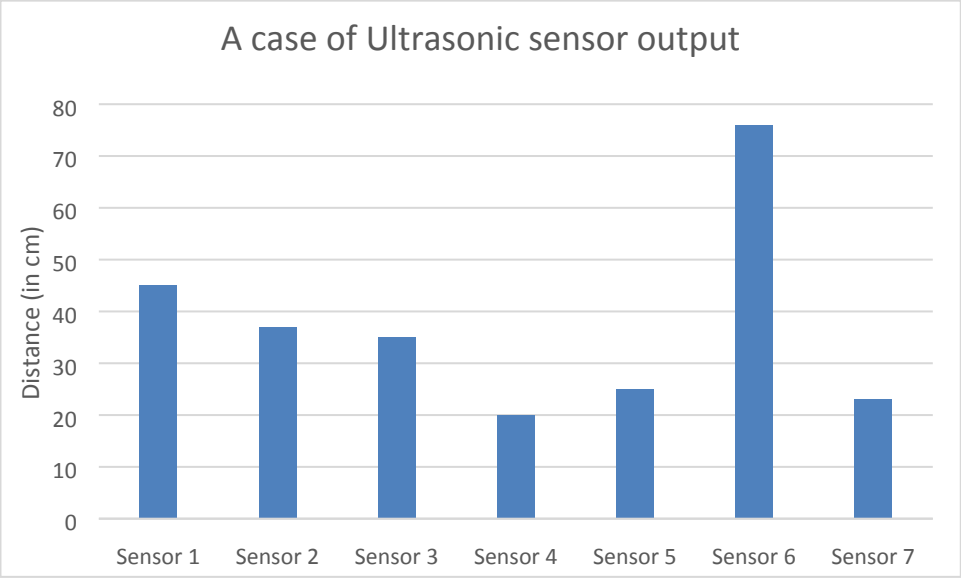

Figure 4.2: In the case of reading from the ultrasonic sensors as shown above, the obstacle is nearest to the sensor 4. This will trigger the conditional in the code to check on both sides of the sensor 4. Since the output from sensor 3 is greater the sensor 5 so the boat will turn towards sensor 3.

4.3 Ultrasonic Sensors

Thankfully the sensor modules already came soldered with the male connectors. After the sensor was connected to the Arduino UNO and uploaded with the basic code for measuring and detecting objects in its path, it ran flawlessly on the first run.

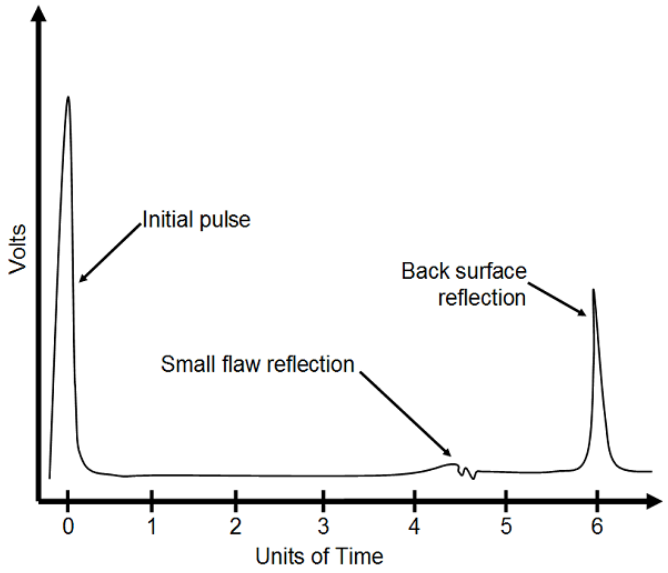

Figure 4.3: Graphical representation of ultrasonic sensor. [11].

4.4 BLDC Motor

The brushless DC motor required delicate control system that converts the input DC power into three phase AC power that was 90 degree out of phase. To run it, it was needed to connect an ESC with it in series, which acted as a controller. The ESC needed to be armed first. This could be easily done with the help of an RC controller, but without it, team had to make use of an Arduino to run it. The arming process of the ESC with the help of an Arduino is a little bit peculiar; random values had to be input to the motor ESC to find the arming input for the ESC.

The motor can be armed from the input of $19\ to\ 45$. The motor starts working when the $input = arming\ input + 5$. The motor stops when input is between $18\ to\ (arming\ input + 4)$. And the speed will not vary between the input of $0\ to\ 17$. It was observed that the BLDC, when run on high speeds, gets very hot very fast.

4.5 Compass Controlled Servo

Now the Servo motor was combined with the compass. The objective of this test was to determine the feasibility of a Servomotor when the data isn't fed to it manually. The output angle of the compass would be the input for the Servomotor. The code for this was very simple and yielded the expected results.

It was expected that when the compass was rotated, the servo motor would mimic its motion up to its moveable range. The Servo motor has the range of $0°\ to\ 180°$ so theoretically, when the compass was in this range, then the servo would follow but when the compass transcended this range then servo would remain at the maximum or minimum extreme, depending on the direction of movement of the compass.

The results of this test were in accordance with the theory.

4.6 Navigation

For the testing of navigation algorithm, a large body of water was needed, like a swimming pool or a lake. This helped to determine the turning radius of the boat. The boat has a turning radius of about $3 - 4ft$.

The turning radius of the boat isn't a fixed parameter. It varied with the magnitude of error between the bearing and current heading. When the error exceeded the maximum turning range of the rudder, the boat turned rapidly. While when the

error was less than the maximum turning range of the rudder, then the turning radius increased steadily. This can be best described by the following formula:

$$turning\ radius\ \propto magnitude\ of\ error \tag{4.1}$$

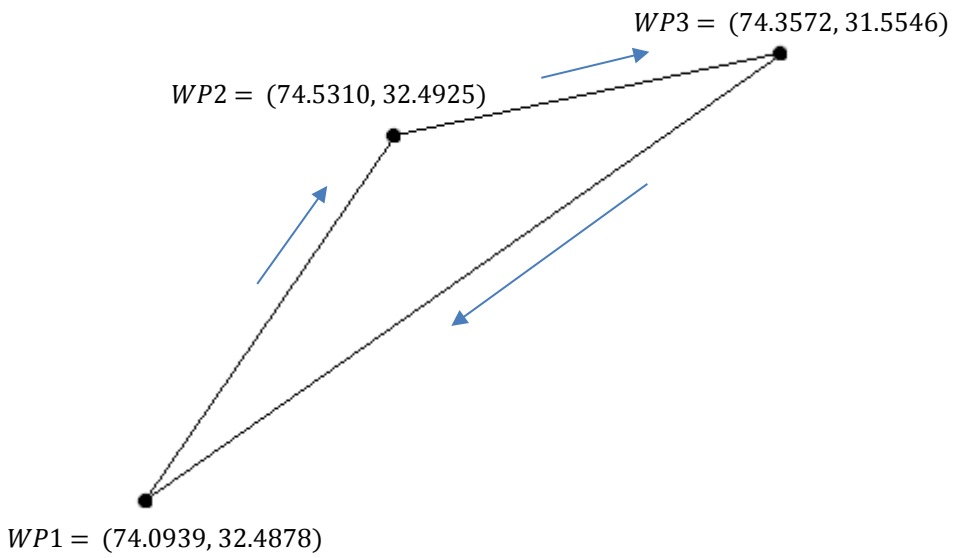

Figure 4.4: Navigation path with waypoints

During its movement through the water, the boat followed a wavy path instead of a straight line. It was theorized that once the boat was in right direction, it would follow a straight line but this theory proved to be incorrect after the testing. There might be a few reasons for this behavior. First, the boat might be turning ever so slightly because of the water current resistance. This would trigger the PID algorithm and turn the boat towards the correct path, causing the wavy path. Second, this might be just the result of the overshoot of the PID output and could be removed by adjusting the values of PID constants.

Third, the body of water that was used to test the boat was very shallow. It might be so that the boat was actually touching the ground beneath the water and turning because of that.

4.7 Obstacle Avoidance

The testing for obstacle avoiding is very simple. Seven ultrasonic sensors were used for the Obstacle detection and avoidance. The maximum distance for the boat was set to $30cm$ to avoid the unnecessary triggering of the algorithm. The testing was completely in accordance with the theoretical results.

Figure 4.5: Boat during navigation testing

Chapter 5

DISCUSSION AND CONCLUSION

5.1 Conclusion

This project started when the need of an unmanned surface vehicle arose for the accomplishment of remote tasks such as observing and monitoring.

A physical structure was designed for this project and the whole boat was modelled after a Catamaran. The first USV that was modelled after a Catamaran was named SESAMO, it was an Italian based USVs and was used in oceanographic research. The ability of a catamaran to remain stable, carry higher amount of load and to travel in a straight line, make them a very good choice for USV development.

In this project two aspects of autonomous operation are combined i.e. point to point movement and the obstacle detection and avoidance. Since this boat was not for a competition so there is not a list of tasks that it had to perform but a list of tasks has been prepared for the boat.

After this, a navigation algorithm was designed for this boat to help the boat traverse autonomously through a collection of given set points.

An obstacle detection and avoidance algorithm was devised and given a priority over the navigation algorithm. This ensured a safe and collision free journey of the boat.

5.2 Discussion

Most of the recent work done in the field of RoboBoat is by students participating in the robotic competitions. These competitions provide a list of tasks that the boat are required to perform accurately and efficiently to win.

In contrast to that, the work done on the RoboBoat by us was to provide a basic autonomous vehicle with a navigation and an obstacle avoiding algorithm. This boat can be modified to perform a number of application, ranging from observation to data collection. For example, if the observation and monitoring tasks are required of the boat then it can be equipped with a digital camera and a storage device. If the data is required in real time then it can be transmitted through a wireless module.

Usually the naval applications of a USV are limited to the minesweeping or for security purposes. Sometimes for traditional observatory missions and monitored and executed in a supervisory control approach. This approach means that the steering control of the boat is autonomous but an operator can control the selection of way points and destination. Also the operator can monitor if the mission is going as it was supposed to go. This helps one operator to supervise a number of different USVs.

5.3 Future Work

The latest developments in the field of USV are with the advent of renewable resources like wind, solar and water energy. USVs are being devolved to use such resource for their energy intake and to reuse them for propulsion and for other uses of electrical power.

The USV hold a very critical position in the unmanned vehicle program because of it contact with the air and the sea at the same time. The US navy is one of the biggest users of the unmanned vehicles in the market. There recent interest in this field show that there are some great prospects for it in the future.

References

1. Vaneck, Thomas W., et al. "Automated bathymetry using an autonomous surface craft." *Navigation* 43.4 (1996): 407-419.
2. Goudey, Clifford A., et al. "A robotic boat for autonomous fish tracking." *Marine Technology Society. Marine Technology Society Journal* 32.1 (1998): 47.
3. Manley, Justin E. "Development of the autonomous surface craft" ACES"." *OCEANS'97. MTS/IEEE Conference Proceedings*. Vol. 2. IEEE, 1997.
4. Manley, Justin E., et al. "Evolution of the autonomous surface craft AutoCat." *Oceans 2000 MTS/IEEE Conference and Exhibition*. Vol. 1. IEEE, 2000.
5. Kollanda, Rick, Hayden Radke, and Ian Vaughn. "URI RAMboat 2012: A Flexible Architecture for the AUVSI RoboBoat Competition and Beyond."
6. Calce, A., et al. "Roboboat-building unmanned surfaced vessels from RC motorboats."
7. Gray, A., et al. "Propagator 2013: Uf autonomous surface vehicle." *AUVSI Foundation's 6th Annual RoboBoat Competition, Virginia Beach, VA* (2013).
8. O'Sullivan, Sinead, et al. "Georgia Tech 2014 RoboBoat Competition." (2014).
9. http://code.google.com/p/roboboat/downloads/list and Apress.com
10. Movable Type Scripts http://www.movable-type.co.uk/scripts/latlong.html
11. PC-Based Ultrasonic Test System Basics http://www.ni.com/white-paper/3766/en/
12. "Arduino Robotics" Copyright © 2011 by John-David Warren, Josh Adams, and Harald Molle

Appendix A.

CODES

PID Algorithm MATLAB code:

```
clear all; close all; clc;
p=500;
plotter=zeros(1,p);
drive=1;
integral=0;
output= 0
setpoint=95
kp = 1.5;
ki = .2;
kd = .5;
last = 0;

for f=1:p
    error = setpoint - output;
    integral = integral + error;
    P = error * kp
    I = integral * ki
    D = (output - last) * kd
    drive = P + I + D;


    last = output;
    output=drive;
    plotter(:,f) = drive;
end
plot(plotter,'--rs','LineWidth',2,...
                        'MarkerEdgeColor','k',...
                        'MarkerFaceColor','y',...
                        'MarkerSize',10);
```

Navigation Algorithm:

```
#include <TinyGPS.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_HMC5883_U.h>
#include <Servo.h>
//-------------------------


#define KP_HEADING 2.15
#define KI_HEADING 0.07
#define KD_HEADING 0.00001
#define RUDDER_PIN 11
#define PROPELLER_PIN 9
#define MIDDLE_RUDDER 90
#define HEADING_MIN -60
#define HEADING_MAX 60
#define INTEGRATOR_LIMIT 240
//-------------------------


Adafruit_HMC5883_Unified mag = Adafruit_HMC5883_Unified(12345);
Servo rudder;
Servo bldc;
TinyGPS gps;
void gpsdump(TinyGPS &gps);
bool feedgps();
void select_WP();
double calculate_gps_course();
double calculate_distance();


//-------------------------
int heading_previous_error;
```

```
float heading_I = 0.0;
int wp_bearing;
int wp_distance;
float current_course;
float flat, flon;
char ch;
String incomingString;
char m;
int x=0;
unsigned long t;
unsigned long y;
double wp_Lat=0;
double wp_Long=0;
float current_Latitude, current_Longitude;
//-------------------------

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial1.begin(9600);
  int_compass();
  rudder.attach(RUDDER_PIN);
  bldc.attach(PROPELLER_PIN);
  arm_bldc();
  test_rudder();
  delay(2000);
}

void loop()
{
  // put your main code here, to run repeatedly:
```

```
bool newdata = false;
unsigned long start = millis();

// Every fourth of a second we print an update
while (millis() - start < 250)
{
  if (feedgps())
    newdata = true;
}

if (newdata)
{
  start_bldc();
  if(x=0){
  x=1;
  t=millis();
  }
  Serial.println("Acquired Data");
  Serial.println("-------------");
  gpsdump(gps);
  Serial.println("-------------");
  Serial.println();

    ch=3;
    select_WP();
    //float current_Latitude, current_Longitude;
    // selecting the set point
    //distance formula
    current_Latitude=flat;
    current_Longitude=flon;
    //-----------------------
```

```
    double distance=calculate_distance();
    //----------------------

    Serial.print("Way Point Longitude= ");Serial.println(wp_Long);
    Serial.print("Way Point Latitude= ");Serial.println(wp_Lat);

    //----------------------
    double bearing=calculate_gps_course();
    //----------------------

    Serial.print("Distance=");
    Serial.print(distance);
    Serial.println("km");
    Serial.print("Bearing=");
    Serial.println(bearing);


    Serial.println("Compass Data:");
    current_course=get_heading();
    Serial.print(current_course);Serial.println("degrees");

    //---------------------------------------
    //PID servo control
    rudder_control();
    y=millis();
int m=y-t-7000;
if(m>=120000){
  stop_bldc();
  while(1);
}
  }
```

```
}
//-------------------------------------

double calculate_distance(){
  const float R = 6371.000; // km
  float p1 = current_Latitude * M_PI/180;
  float p2 = wp_Lat * M_PI/180;
  float dp = (wp_Lat-current_Latitude) * M_PI/180;
  float dl = (wp_Long-current_Longitude) * M_PI/180;

  float x = sin(dp/2) * sin(dp/2) + cos(p1) * cos(p2) * sin(dl/2) * sin(dl/2);
  float y = 2 * atan2(sqrt(x), sqrt(1-x));
  Serial.println(y);
  Serial.println(x);
  return R * y;
}
//---------------------------------------

double calculate_gps_course(){
  float p1 = current_Latitude * M_PI/180;
  float p2 = wp_Lat * M_PI/180;
  float dp = (wp_Lat-current_Latitude) * M_PI/180;
  float dl = (wp_Long-current_Longitude) * M_PI/180;
  float bearing= atan2(sin(dl)*cos(p2),cos(p1)*sin(p2)-sin(p1)*cos(p2)*cos(dl));
  float bearingDegrees=bearing*180/M_PI;
  if(bearingDegrees<0) bearingDegrees=360+bearingDegrees;
//  Serial.print("Difference b/w Latitude=");Serial.print(wp_Lat-current_Latitude);
//  Serial.print("Difference b/w Longitude=");Serial.print(wp_Long-current_Longitude);
  return bearingDegrees;
}
//---------------------------------------
```

```
void select_WP(){
          if(ch==1){
        Serial.println("Lahore");
        wp_Long=74.3572;
        wp_Lat=31.5546;
        }
      else if(ch==2){
        Serial.println("Multan");
        wp_Long=74.0939;
        wp_Lat=32.4878;
      }else if(ch==3){
        Serial.println("Sialkot");
        wp_Long=74.5310;
        wp_Lat=32.4925;
      }
}
//---------------------------------------

void gpsdump(TinyGPS &gps)
{
  long lat, lon;

  unsigned long age, date, time, chars;
  unsigned short sentences, failed;

  gps.f_get_position(&flat, &flon, &age);
  Serial.print("Lat/Long(float): "); Serial.print(flat,5); Serial.print(", "); Serial.print(flon, 5);
  Serial.print(" Fix age: "); Serial.print(age); Serial.println("ms.");

  feedgps();
```

```
  gps.stats(&chars, &sentences, &failed);
  Serial.print("Stats: characters: "); Serial.print(chars); Serial.print(" sentences: ");
Serial.print(sentences); Serial.print(" failed checksum: "); Serial.println(failed);
}
//--------------------------------------

bool feedgps()
{
  while (Serial1.available())
  {
    if (gps.encode(Serial1.read()))
      return true;
  }
  return false;
}

//-------------------------------
float get_heading() {

  // Get a new sensor event
  sensors_event_t event;
  mag.getEvent(&event);

  // Calculate heading based on magnetic intensity in X & Y axis
  float heading = atan2(event.magnetic.y, event.magnetic.x);

  // 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;
  headingDegrees = /*360-*/headingDegrees;
  //Serial.println(M_PI);
  return headingDegrees;
}

//-------------------------------------
void int_compass()
{
  Serial.println("Initializing Compass...!!!");
  // Initializing the sensor
  if(!mag.begin())
  {
    /* There was a problem detecting the HMC5883 ... check your connections */
    Serial.println("HMC5883 not detected!");
    while(1);
  }
  Serial.println("Compass Initialized");
}
//---------------------------
int compass_error(int PID_set_Point, int PID_current_Point){
  float PID_error=0;
  if(fabs(PID_set_Point-PID_current_Point)>180){
    if(PID_set_Point-PID_current_Point<-180){
      PID_error=(PID_set_Point+360)-PID_current_Point;
    }
```

```
    else{
      PID_error=(PID_set_Point-360)-PID_current_Point;
    }
  }
  else{
    PID_error=PID_set_Point-PID_current_Point;
  }
  return PID_error;
}
//------------------------------

void rudder_control(void){
  wp_bearing=calculate_gps_course();
  wp_distance=calculate_distance();
  long                                                    rudder_setpoint=MIDDLE_RUDDER-
PID_heading(compass_error(wp_bearing,current_course));
  pulse_servo_rudder((long)rudder_setpoint);
}
//--------------------------------

int PID_heading(int PID_error)
{
  static float heading_D;
  static float heading_output;
  heading_I += (float)PID_error;

  heading_I = constrain(heading_I, -INTEGRATOR_LIMIT, INTEGRATOR_LIMIT);
  //Derivation part
  heading_D = ((float)PID_error - (float)heading_previous_error);

  heading_output = 0.0;//Clearing the variable.
```

```
  heading_output =  (KP_HEADING * (float)PID_error);
  heading_output += (KI_HEADING * heading_I);
  heading_output += (KD_HEADING * heading_D);
  heading_output = constrain(heading_output, (float)HEADING_MIN,(float)HEADING_MAX);
  heading_previous_error = PID_error;
  return (int)(heading_output);
}
//-------------------------------------

void arm_bldc(){
  bldc.write(20);
  delay(15);
}
//-----------------------------------------

void start_bldc(){
  bldc.write(40);
}
//-----------------------------------------

void stop_bldc(){
  bldc.write(20);
}
//-----------------------------------------

void reset_PIDs(void)
{
  heading_previous_error = 0.0;
  heading_I = 0.0;
}
```

```
//-------------------------------------


void test_rudder(void)
{
  pulse_servo_rudder(MIDDLE_RUDDER + HEADING_MIN);
  delay(1500);
  pulse_servo_rudder(MIDDLE_RUDDER + HEADING_MAX);
  delay(1500);
  pulse_servo_rudder(MIDDLE_RUDDER);
  delay(1500);
}
//-----------------------------------------


void pulse_servo_rudder(int rudder_heading)
{
  rudder.write(rudder_heading);
  Serial.print("Rudder Heading:");Serial.print(rudder_heading);
}
//-----------------------------------------
```

Obstacle Detection and Avoidance Algorithm:

```
#include <Servo.h>


int SLIGHTLY_RIGHT= 15;

int SLOW_RIGHT= 30;

int RIGHT =45;

int STRONG_RIGHT = 60;

int STRONG_LEFT= -60;

int LEFT= -45;

int SLOW_LEFT = -30;

int SLIGHTLY_LEFT= -15;



Servo rudder;


int turn=0;

float us_sensors[7];

int max_distance=10;

int trigPin0 = 22;

int echoPin0 = 23;

int trigPin1 = 26;

int echoPin1 = 27;

int trigPin2 = 30;

int echoPin2 = 31;

int trigPin3 = 34;

int echoPin3 = 35;

int trigPin4 = 38;

int echoPin4 = 39;

int trigPin5 = 42;

int echoPin5 = 43;

int trigPin6 = 46;
```

```
int echoPin6 = 47;


int r=0;
int l=0;


void setup() {
  Serial.begin (9600);
  rudder.attach(11);
  pinMode(trigPin0, OUTPUT);
  pinMode(echoPin0, INPUT);
  pinMode(trigPin1, OUTPUT);
  pinMode(echoPin1, INPUT);
  pinMode(trigPin2, OUTPUT);
  pinMode(echoPin2, INPUT);
  pinMode(trigPin3, OUTPUT);
  pinMode(echoPin3, INPUT);
  pinMode(trigPin4, OUTPUT);
  pinMode(echoPin4, INPUT);
  pinMode(trigPin5, OUTPUT);
  pinMode(echoPin5, INPUT);
  pinMode(trigPin6, OUTPUT);
  pinMode(echoPin6, INPUT);



}


float first_sensor(){ // function is for first sensor.
  int dur_0, dist_0;
  digitalWrite (trigPin0, HIGH);
  delayMicroseconds (10);
  digitalWrite (trigPin0, LOW);
```

```
  dur_0 = pulseIn (echoPin0, HIGH);
  dist_0 = (dur_0/2) / 29.1;


      Serial.print("1st Sensor: ");
      Serial.print(dist_0);
      Serial.print("cm    ");
      if(dist_0<0){
      dist_0=fabs(dist_0)*100;
      }
      return dist_0;


}
float second_sensor(){ // function is for second sensor.
    int dur_1, dist_1;
    digitalWrite (trigPin1, HIGH);
    delayMicroseconds (10);
    digitalWrite (trigPin1, LOW);
    dur_1 = pulseIn (echoPin1, HIGH);
    dist_1 = (dur_1/2) / 29.1;


      Serial.print("2nd Sensor: ");
      Serial.print(dist_1);
      Serial.print("cm    ");
      if(dist_1<0){
      dist_1=fabs(dist_1)*100;
      }
      return dist_1;
}
float third_sensor(){ // function is for third sensor.
  int dur_2, dist_2;
  digitalWrite (trigPin2, HIGH);
```

```
  delayMicroseconds (10);
  digitalWrite (trigPin2, LOW);
  dur_2 = pulseIn (echoPin2, HIGH);
  dist_2 = (dur_2/2) / 29.1;


     Serial.print("3rd Sensor: ");
     Serial.print(dist_2);
     Serial.print("cm    ");
     if(dist_2<0){
     dist_2=fabs(dist_2)*100;
     }
     return dist_2;
}
float fourth_sensor(){ // function is for fourth sensor.
    int dur_3, dist_3;
    digitalWrite (trigPin3, HIGH);
    delayMicroseconds (10);
    digitalWrite (trigPin3, LOW);
    dur_3 = pulseIn (echoPin3, HIGH);
    dist_3 = (dur_3/2) / 29.1;


     Serial.print("4th Sensor: ");
     Serial.print(dist_3);
     Serial.print("cm    ");
     if(dist_3<0){
     dist_3=fabs(dist_3)*100;
     }
     return dist_3;
}
float fifth_sensor(){ // function is for fifth sensor.
  int dur_4, dist_4;
```

```
  digitalWrite (trigPin4, HIGH);
  delayMicroseconds (10);
  digitalWrite (trigPin4, LOW);
  dur_4 = pulseIn (echoPin4, HIGH);
  dist_4 = (dur_4/2) / 29.1;

     Serial.print("5th Sensor: ");
     Serial.print(dist_4);
     Serial.print("cm    ");
     if(dist_4<0){
     dist_4=fabs(dist_4)*100;
     }
     return dist_4;

}
float sixth_sensor(){ // function is for six sensor.
    int dur_5, dist_5;
    digitalWrite (trigPin5, HIGH);
    delayMicroseconds (10);
    digitalWrite (trigPin5, LOW);
    dur_5 = pulseIn (echoPin5, HIGH);
    dist_5 = (dur_5/2) / 29.1;

      Serial.print("6th Sensor: ");
      Serial.print(dist_5);
      Serial.print("cm    ");
      if(dist_5<0){
      dist_5=fabs(dist_5)*100;
      }
      return dist_5;
}
```

```
float seventh_sensor(){ // function is for seven sensor.
  int dur_6, dist_6;
  digitalWrite (trigPin6, HIGH);
  delayMicroseconds (10);
  digitalWrite (trigPin6, LOW);
  dur_6 = pulseIn (echoPin6, HIGH);
  dist_6 = (dur_6/2) / 29.1;

     Serial.print("7th Sensor: ");
     Serial.print(dist_6);
     Serial.print("cm    ");
     if(dist_6<0){
     dist_6=fabs(dist_6)*100;
     }
     return dist_6;

}
void obstacle_rudder_control(){
  int angle=90+turn;
  rudder.write(angle);
}
void loop() {
Serial.println("\n");

Serial.println(us_sensors[0]);Serial.println(us_sensors[1]);Serial.println(us_sensors[2]);Serial.pri
ntln(us_sensors[3]);Serial.println(us_sensors[4]);Serial.println(us_sensors[5]);Serial.println(us_s
ensors[6]);
 int sensor_number;
 int x=max_distance;
 us_sensors[0]=first_sensor();
 us_sensors[1]=second_sensor();
```

```
us_sensors[2]=third_sensor();
us_sensors[3]=fourth_sensor();
us_sensors[4]=fifth_sensor();
us_sensors[5]=sixth_sensor();
us_sensors[6]=seventh_sensor();
 // put your main code here, to run repeatedly:
 for(int i=0;i<=6;i++){
 float y=us_sensors[i];
 if(y<x){
   x=y;
   sensor_number=i;
 }
 }

 if(x<=max_distance){
 Serial.println("sensor number");
 Serial.println(sensor_number);

 switch(sensor_number){
   case 0:
   turn=SLIGHTLY_LEFT;
   break;
   case 1:
   turn=SLOW_LEFT;
   break;
   case 2:
   turn=LEFT;
   break;
   case 3:
    l=us_sensors[3]+us_sensors[4];
    r=us_sensors[3]+us_sensors[2];
```

```
    if(l<=r){
      turn=STRONG_RIGHT;
    }else{
     turn=STRONG_LEFT;
    }
   break;
   case 4:
   turn=RIGHT;
   break;
   case 5:
   turn=SLOW_RIGHT;
   break;
   case 6:
   turn= SLIGHTLY_RIGHT;
   break;
   default:
   turn=0;
   break;
  }
  obstacle_rudder_control();
//delay(1000);
  }
}
```