

Autonomous Navigation of WAM-V

Team Name: SUTD Seals

Team members: Tushar Mohan, Ziyu Chen, Swayam Narain, Wei Liang Chew, Nishard Ghose, Xueyan Oh, Jaron Lee, Cai Ruihe and Hezu Ma.

ABSTRACT

This paper describes our design process in producing a sea-worthy craft that meets the competition requirements and navigate the obstacles in the course. The issues we faced were with interfacing the proprietary off-the-shelf propulsion system with our system, the circuitry for the fail-safe system, and the software for communication and control. Also described here is the approach we took to program visual recognition into our software to navigate the course obstacles.

INTRODUCTION

To produce a sea-worthy craft capable of navigating the obstacles in the course, we first set out to establish means for actuation of the craft, and wired computer remote control. This first involved interfacing an electronics system with an off-the-shelf propulsion system, and fitting them onto the provided frame for the craft. We tried to keep the mechanics on the boat as simple as possible to minimize risk. The code for the integrating the sensor systems with electronic control was being developed concurrently, and was put together to form a working craft. We then put the craft out to sea for testing to calibrate stabilization parameters, and began coding the autonomous functions required to manoeuver through the course.

PROPULSION CONTROL

The propulsion system for the WAM-V platform was purchased from Torqeedo. The system we worked with was designed for manual throttle control by hand (ref to Figure 1).



Figure 1: Torqeedo motor speed controller

In order to control the speed of the boat, we first considered replacing the entire controller with our own, to provide us with maximum control. This proved to be too difficult and time-consuming for us, particularly due to the need to program for a RS485 library for communication with the proprietary Torqeedo protocol. Hence, we changed tack and proceeded to hack the hardware of the throttle controller. After some examination, we found that the throttle speed was determined by a sensor which detected the angular position of a small neodymium magnet attached to the control handle for a human being to manipulate. As the magnet rotates, the sensor in the motor speed controller will

detect its angle and send a signal to the PIC microcontroller. We decided to exploit this, and engineer a mechanism to manipulate the rotational position of a magnet using a servo motor, which would then be simple to control with an Arduino board. For the purposes of testing, remotely sending signals to the Arduino board would also be straightforward.

Further examination found that the magnetic sensor in the motor speed controller was very sensitive, and that it was tricky to position the magnet correctly relative to the sensor, which had a low error tolerance. If the magnet was not positioned exactly, the controller would cease transmitting to the propulsion. In order to get around this, a novel mechanism to fix the servo motor and magnet on the correct position has been designed and developed.

Vision System

This is the task about finding and identifying objects in real time sequence image or video. To us Humans, Recognition of various objects in images is simple and straight forward, despite the fact that the image of the objects may vary somewhat in different viewpoint, in many different sizes / scale or even when they are translated or rotated. It is even possible to have to objects recognised when it is partially obstructed from view. Many approaches to the task have been implemented over multiple decades with genetic algorithm, which require no prior knowledge of a given dataset and can

develop recognition procedures without human intervention.

The approaches are for example edge matching, Divide-and-Conquer search and Greyscale matching. ROS is chosen to be our vision system platform with the most popular OpenCV library.

ROS (Robot Operating System) provides libraries and tools to help software developers create robot applications. It provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management, and more. Therefore, a small and powerful firmware camera which can provide high quality image and large observation area has been integrated by Camera 1394 driver. It avoids the driver compatibility issues we faced earlier with operation system like Linux (Ubuntu 14.04) but connect the device directly on kernel level.

This is an approach that has not been well realised among the group of firmware camera users.



The Guppy PRO camera which is the derivative of the successful Guppy family has an excellent price/performance ratio.

- IEEE 1394b
- Several models, VGA to 5 Megapixels
- Sony CCD sensors, ON Semi CMOS sensor
- Optocoupled 12-pin I/Os
- Ultra-compact housing

OpenCV is released under a BSD license and it is easy for academic use for a diversity programming language supported such as C++, C, Python and Java and it supports Linux (Ubuntu 14.04) with all the most common sensors developed o the boat project. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing.

First Prototype:

Since the position of the magnet needed to be fixed very precisely, we decided to implement a mechanism which allowed us the flexibility to fine tune the position of the servo motor.

For our first iteration, we designed a mechanism to be 3D printed and which gave us the ability to calibrate the vertical and horizontal position of the magnet and servo motor by tuning two M4 screws and nuts (Figure 2).

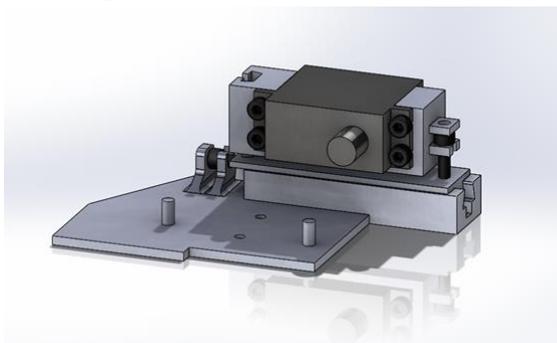


Figure 2: The first version of mechanism

This mechanism consists of 3 separate parts. A “Base” plate fixes the assembly to the motor speed controller and provides a dovetail slot for the “Horizontal Sliding Block” part. The “Horizontal Sliding Block” can slide horizontally along the dovetail slot and is fixed and tuned by the M4

screw. It also provides the guide shaft for the “Vertical Sliding Block at its top.

The “Vertical Sliding Block” slides vertically on the guide bar and is tuned by a M4 screw. It also provides the base for the servo motor. The magnet is fixed on the end of the servo motor’s shaft.

For the installation of the mechanism onto the speed motor controller, two M3 screws will run through the base of the attachment and into holes already at the bottom on the controller. Then the vertical and horizontal position would be tuned until the controller can sense the magnet properly. All parts were made using a Makerbot Replicator 2 3D printer for prototyping.

Testing the prototype for this mechanism, we found that the design did not work with the materials we were prepared to use. This resulted in trouble calibrating the vertical position, which had the possibility of changing position if the boat shakes too violently. Tuning was also troublesome, and not as smooth as we had hoped.

Second Prototype:

We decided to try a simpler design which we hoped would be more reliable and avoid the same shortcomings as our first iteration. The second iteration of our design consists of 2 acrylic components laser cut from 5mm transparent acrylic board. Multiple M3 and M4 screws would be used to assemble and fix the components onto the controller. (Figure 3)

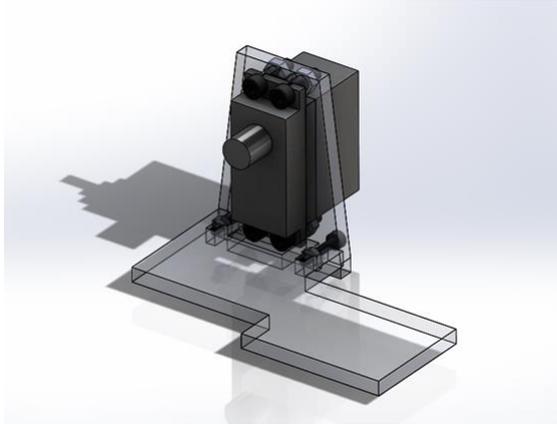


Figure 3: The second version of mechanism

The “Base” acrylic component would fix the whole mechanism onto the base of the motor controller, and has 2 pre-cut T-slot for 2 M3 screws and nuts for attaching the “Motor Base” acrylic component. The “Motor Base” component has 4 pre-cut holes for M4 screws used to fix the servo motor which actuates the magnet. In order to install the servo motor and the magnet to the correct position, we first fix the servo motor on the “Motor Base” part. Then we hand-tune the correct position of the magnet. Another person then attaches the “Base” in position to connect to the “Motor Base” and the controller, without altering the positions of either part. We then mark the position of the screw holes for the screws that would connect the “Motor Base” to the “Base”, as well as the position of the screw holes for the screws to connect the “Base” and the motor speed controller. By relying on hand-tuning in this way, we would be able to adjust the position and orientation of the magnet and servo motor rapidly, and hopefully still be accurate enough for the sensor.

This prototype and stood well to testing, and we found it to be perfectly functional, so we proceeded to implement this for the control of both propulsion motors for the craft.

FAIL SAFE CIRCUIT

According to safety requirements, we had to design both physical and digital fail-safe switches to cut power supply to the boat in case of emergency. The current drawn by the torqeedo motor was too high for the economical switches we had to handle and we decided to make use of a double relay switch system in series in order for the circuit to be broken digitally or physically. Table 3.1 shows the main items we have purchased to construct the circuit.

Item	Image	Qty
Relay Switch 24V coil V, 30V contact Nom, 80 Contact Current Max		2
Relay Switch TE CONNECTIVITY / POTTER & BRUMFIELD - RTE24005 - Product Range:TE - Power PCB Relay RT2 Series		1
24 AWG cable BELDEN - 9533 - CABLE 9533, 3CORE, PER M		10 m
Push Button Switch,foot/palm,red,la tching		2
Matrix board		1

12V Deep Cycle Lead Acid Battery, 7 Ah		1
----------------------------------------	-----------------------------------------------------------------------------------	---

Table 1: List of components

A 5V relay switch is first connected to a power source from an Arduino board. The switch is soldered permanently onto a matrix board along with connecting wires.

The board will be mounted on the top of the boat in the vicinity of the other components, and probably waterproofed. This is the first method of breaking the circuit where we can cut off this source by toggling a switch on a RC controller. This relay switch is NO (Normally Open) which means cutting off the 5V supply will break the entire circuit. Note that a NC (Normally Closed) configuration was not used as we do not want the circuit to be closed in the

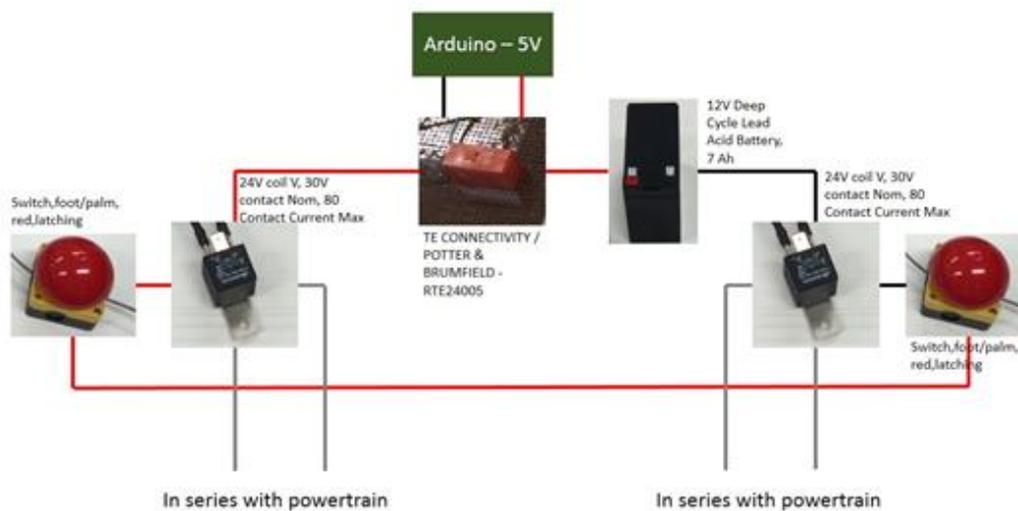


Figure 4: Schematic of fail-safe circuit

case that the Arduino board is no longer powering. Closing the 5V relay switch allows current to be drawn from a 12V lead-acid battery and into 2 other 12V relays.

The two 12V relays are also NO and controls the power-train directly. These secondary relay switches have to be used as the 5V ones are not able to withstand high current in the power-train. The second method of breaking the circuit is a physical one by pressing any of the two red push-buttons which are connected in series with the 12V supply and the two

12V relay switches. These two buttons were connected to the relay switches by 6mm push-on terminals and each of them was attached to one side of the boat by tying them to the support columns using cable ties for ease of access. Pushing any of the red push-buttons will break the circuit of the 12V supply and open the relay switches directly involved in the power-train. This will cut the main supply from the battery to both of the torqeedo motors.

The system is designed such that in the case of power failure of any device, the

boat will automatically cease to function until the problem is resolved. Hence, providing a reliable fail-safe feature to stop the boat in the case of any malfunction.



Figure 5: Actual setup of fail-safe circuit

SOFTWARE ARCHITECTURE

The software is written in Java and is responsible for using the sensors to control the boat and communicate with the base station. We use the principles of Object Oriented Programming coupled with strong modularity and thread safety to ensure a robust, fault tolerant control system. The base station communication, sensor aggregation and controller actuation happen on independent threads optimized by the use of robust java services such as Executor Service.

Sensor Integration:

The sensors used in the boat include a GPS module for location awareness and a compass module for heading information. These are integrated with the main software through an open source daemon service. The sensor information is accessed by independent threads through the GPS daemon service using socket programming and a specified API. The

daemon responds to requests via JSON which is then parsed and made available to the other parts of the software such as the control module and the heartbeat module.

The software retains only the last known bearing and location from the sensors in its memory, and makes it available for reading only in a thread safe manner. This architecture ensures the integrity of sensor data at the same time minimizes the memory overhead required to store the data. Since the control algorithm only requires the current position and bearing to send actuation commands to the propellers, we only need to store the current location and current bearing. This greatly simplifies the architecture in making the sensor information thread safe.

Base Station Communication:

Base Station communication is facilitated by using socket programming to connect to the base station's server at the specified port. Heartbeat messages are sent every second to the base station via ScheduleWithFixedDelay Executor Service. The Heartbeat messages are constructed by using the GPS information made available in a thread safe manner to the heartbeat message, which is then written on the socket. NMEA messages are constructed by parsing the JSON responses returned by the GPS module.

Navigation:

This part of the software is responsible for calculating the errors associated with current position, current heading and the required destination. There are simple

deterministic formulas that convert these into the required heading and required distance, which is then fed into the controller for the boat. To calculate the required bearing, the following formula was used:

$$\theta = \text{atan2}(\sin(\Delta\text{long}) \cdot \cos(\text{lat}_2) \times \cos(\text{lat}_1) \cdot \sin(\text{lat}_2) - \sin(\text{lat}_1) \cdot \cos(\text{lat}_2) \cdot \cos(\Delta\text{long}))$$

Whereas for the distance error, we used the haversine formula:

$$\begin{aligned} R &= \text{earth's radius (mean radius = 6,371km)} \\ \Delta\text{lat} &= \text{lat}_2 - \text{lat}_1 \\ \Delta\text{long} &= \text{long}_2 - \text{long}_1 \\ a &= \sin^2(\Delta\text{lat}/2) + \cos(\text{lat}_1) \cdot \cos(\text{lat}_2) \cdot \sin^2(\Delta\text{long}/2) \\ c &= 2 \cdot \text{atan2}(\sqrt{a}, \sqrt{1-a}) \\ d &= R \cdot c \end{aligned}$$

Arduino Communication:

Arduino communication uses the RXTx library used by arduino ide for Java. The port is opened and wrapped to obtain the output and input streams. Actuation commands to the motors are sent via the serial interface by the controller, which calculates the desired servo angles using the errors. They are sent using the following protocol:

<Left Servo Angle>, <Right Servo Angle>\n

PID CONTROL

A control system is a must have for any automated system. We had to design a control system such that it could manoeuvre from one GPS co-ordinate to another GPS co-ordinate taking the shortest path and time. Hence our system should be able to travel to a point reducing the distance to it from the boat whilst correcting its angle to the desired

location. We decided to use a PID control system on faculty recommendation and our familiarity with it. We tried to model the system theoretically and analyse the equations to find the optimal gain values. But this turned out to be a highly tedious and excruciating task as there were many variables due to our system having many devices, in addition to environmental variables once the boat is run in the water. So we tried a different approach where we came up with equations relating the motion we needed, i.e. translation and rotation, to the motor speed values. In the end we had an equation for each motor to govern its speed. Our approach was on the basis that the sum of the two speeds governs the translational motion and the difference of the two speeds governs the rotational motion. The gain values which would make our boat operate at a suitable manner were to be obtained experimentally where we would go the test site and experiment on different values.

Our control system has two primary tasks. Firstly it has to reduce the distance error, i.e. the error between the desired position and the current position. Secondly it has to reduce the angular error, i.e. the angle between the desired position and the boat's current direction. In order for our controller to work we need to obtain these two error values to calculate the appropriate throttle levels for the propellers. To obtain the distance error we used a GPS tracker which can locate our present position. The error is calculated by finding the distance

between the GPS co-ordinate of the desired position which we get externally and calculating the distance from our current position. The angle error is obtained by using a digital compass which can read the current heading and by using the GPS to calculate the angle to the desired target.

Having these values, we decided that our controller would be governed by the distance error, the rate of change of the distance error, angle error and rate of change of the angle error. In the end we implemented a PD controller as we found this to be sufficient for our task. The D controller also aids us in overcoming any sudden changes such as wind and waves. In the end coming up with the mathematical model was not the hard part but the time spent on experimentally obtaining the gain values, implementation of compass and GPS and feeding the values read by them to our code and also calculating the error values took up most of our time and effort. This was a good experience for all of us as we found out how to actually implement a control system in a complex system from start to finish.

CONCLUSION

The interfacing between the off-the-shelf propulsion systems proved unexpectedly to be the most time-consuming for us, as we considered and tried various options before settling on the simplest method (avoiding directly tapping into the proprietary electronics of the propulsion system. Much of our work was guided by the competition requirements, and

generally geared towards reduced complexity, both in hardware and software.