

Virginia Tech

Autonomous Underwater Vehicle Team

2011 – 2012



Team Captain:

Madison Farruggia

Members:

John Allen, Shawn Furrow, David Gayman, Ben Guzzardi, Chelsea Mancuso, Erin Mazza, Logan Sturm

Abstract:

The Virginia Tech Autonomous Underwater Vehicle Team currently works on a two year design plan; meaning that the first year is spent doing a complete design overhaul. The second year, which we will have just completed for competition, focuses more on minor design changes, and making the vehicle more robust in all areas. While some modifications were made to the mechanical and electrical areas of the vehicle, we mainly focused on tweaking the software, vision and controls components of the vehicle. We changed the thruster orientation, made the overall vehicle longer, and added a sonar system to the vehicle, to name a few key changes. Because of the two year process, we were able to add all three mechanisms to the vehicle as well, which the team hasn't ever accomplished since it was founded ten years ago. We are hoping that because of our extensive planning, and two year vehicle program, Mako will stand a better chance at competition and ultimately make it into finals!

Website:

<http://www.auvt.org.vt.edu/>

Introduction

This year the VT AUV Team has undergone a variety of structural changes, including a website redesign and new leadership structure. Since last competition, our main focus has been to take the existing design we had and make it better. Because we run our team based around a two year design cycle, we were able to shift resources and, for the first time in the history of our team, add a working sonar system and all three mechanisms. The rest of our report will detail each of the six subsystems [Mechanical, Electrical, Sonar, Software, Vision and Controls] and their accomplishments.

Mechanical Design

For the 2011-2012 year the team focused on making refinements to the previous year's design instead of performing a complete overhaul. The mechanical team worked closely with the electronics and controls teams to determine what systems needed to be improved from last year's design. The three main areas that improvements were made to were the electronics rack, the thruster configuration, and the mechanisms.

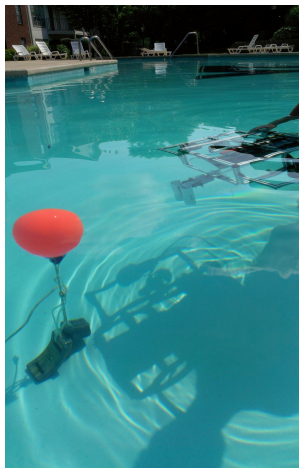


Figure 1. Mako during outdoor testing involving buoy identification

E-rack

The electronics rack is mounted inside of the sub and supports all of the electrical components. The e-rack consists of three acrylic shelves which are supported by two vertical pieces. Due to the addition of a sonar board and several other electrical components it was necessary to increase the size of the e-rack. The mechanical team worked closely with the electrical team to determine what their needs were for the rack. Using these parameters it was determined that the best way to accommodate the new hardware was to lengthen the rack. The decision to increase the length of the rack had several effects on the design. First, the length of the hull and the frame of the submarine had to be increased to accommodate the longer rack. Second, the rack itself had to undergo several changes. The shelves on the existing rack were supported by two pieces of acrylic. These pieces had already proved to be fragile and were damaged in several locations. Increasing the length of the rack would apply additional load to these pieces we determined that acrylic would not be suitable for this task. After examining several different materials it was determined that aluminum would be the best choice for these supports. Due to the conductivity of aluminum these pieces were coated with a nonconductive paint in order to decrease the risk of an accidental short circuit.

Hull and Frame

The hull of the vessel is made out of 1/4in acrylic tubing and is sealed on one end by a 1/4in plate of acrylic. The other

end of the vessel is sealed by the use of an aluminum end-cap. The end-cap from the previous year was used after modifying it to accommodate additional connectors for both sonar and mechanisms. The frame of the vessel is made out of 20mm extruded aluminum bars. These bars allow us to easily modify the frame and add or remove components. With the current iteration of the frame the modularity of the old frame was kept while adding two additional “wings” to mount the forward thrusters on.

Thrusters

The thruster configuration was changed from the previous angled thruster design to a more standard two forward and two vertical thrusters. There were two main reasons for this change, both relating to the control software. When using angled thrusters it was difficult to stabilize the sub. In addition to this the sub was never able to successfully strafe which was one of the primary goals of the angled thruster configuration. The current thruster configuration sacrifices the ability to strafe for more stability and the ability to use a simpler control program. To compensate for the lack of strafing ability the thrusters were mounted on “wings” on the side of the vessel. This allows the thrusters to generate a greater moment and gives us the ability to turn faster.

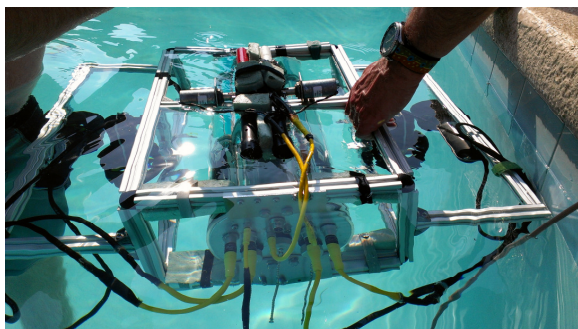


Figure 2. Back view of the vehicle- in which the new thruster configuration can be seen

Mechanisms

Torpedoes

The torpedo launchers were kept the same as the previous year with some slight modifications. The torpedoes are propelled by using a compressed springs and are released by actuating solenoids. The solenoids work reliably underwater and require little power since they are only briefly switched on for the launch. This year's team tuned the torpedo launcher by adjusting the length of the tubes and the spring compression distance to get the best performance.

Drop Mechanism

The drop mechanism operates on a similar principle to the torpedo launcher. Two markers are held inside of tubes by solenoids. The markers are dropped by actuating the solenoids. The solenoids are held in the closed position by springs in order to ensure that the markers do not accidentally fall out. The drop mechanism was moved to the center of the submarine to prevent pitching from occurring when the markers were dropped.

Grab Mechanism

The grab mechanism was newly added to the sub this year since this was the first year that we were able to mount sonar. After discussing a number of ideas the final design that we decided on was one with two L-shaped arms that would rotate to lift the wreath. These arms face each other when they are not in use to prevent the arms from

accidentally catching on obstacles. To pick up the wreath the arms are first lowered into the center of the wreath and then rotated outward to catch the middle bar connecting the wreath. Upon catching the wreath the sub would use the arms to lift the wreath and then rotate them back to their original position to drop it. The arms are rotated using servos and have a rubbery material on the contact surface to prevent the wreath from slipping while it is being lifted. The final design requires few moving parts giving it fewer possible points of failure and making it easier to maintain.

Electronics Design

The electronics subsystem for Mako is comprised of two major sections: the power system and the sensors/instrumentation system.

Power System

The power system of the vehicle was designed to provide stable and sufficient power for all electronics and actuator components. Our power flow design incorporates both familiar and new implementations compared to last year's design.

In order to better manage Mako's power consumption, the power system was split into two isolated subsystems which we call the electronics power and actuator power. This split power system also allows for all moving parts on the vehicle to be powered off via the kill switch without having to kill power to the computer and possibly corrupt the system.

The electronics power consists of the computer and all of the vehicle's

sensors and the actuator power consists of the thrusters and the vehicle's mechanisms. The entire power system uses four Thunder Power 14.8V, 5000mAh, LiPo batteries with the electronics power using three of these batteries and the actuator power using one. The reason for this is because the computer and all of the sensors use a lot more power than the thrusters and mechanisms.

Sensors/Instrumentation System

This section briefly describes what each electronic part does and how all of the electronics integrate to create a complete working system.

Servo Controllers

One of our servo controllers is a Pololu Mini-Maestro Servo Controller. Its primary duty is to organize data in and out of the computer acting as a mediator between it and some of the sensors. All sensors except for the MicroStrain (described below) are connected to the servo controller first. The servo controller is then connected to the computer via USB. The controller allows for easy interfacing between sensors and the computer. When connected to the computer, the Maestro comes up as two different serial ports. One port is used for talking to the sensors the other is used to talk to the thrusters and our second serial servo controller.

The second servo controller is a Pololu Micro Serial Servo Controller. It acts as a separate interface for the actuators which include the thrusters and the mechanisms. The mechanisms include the torpedo launcher, drop mechanism, and grab mechanism.

MOSFET Switches

The MOSFET switches (Figure 3) control the operation of both the torpedo launcher and the drop mechanism. There are four switches placed on the E-Rack. Since the drop mechanism and torpedo launcher are both actuated with two solenoids each, two switches are designated for the drop mechanism and the other two are for the torpedo launcher. If the mechanism is to activate, a signal is sent to the switch via the servo controller. This signal sends the voltage from the actuator power system through the designated MOSFET switch activates the solenoid which shoots the torpedo and/or drops the ball from the drop mechanism.

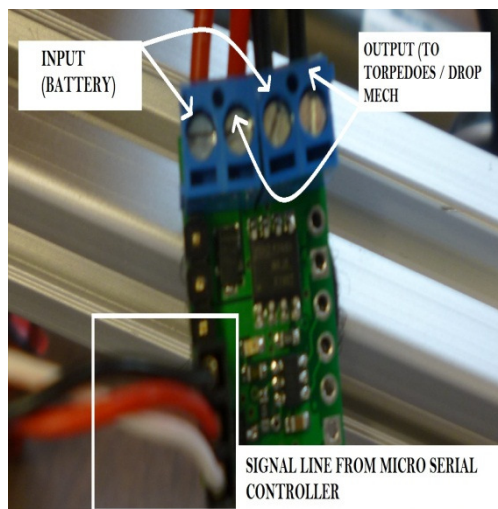


Figure 3: MOSFET switch boards

Depth Sensor

To measure depth, the vehicle will use an MSP-300 Pressure Transducer. This item detects external pressure from the environment and sends a proportional voltage to the Maestro servo controller.

Attitude and Reference Heading System

To measure roll, pitch, & yaw movement, the vehicle uses an attitude

and heading reference system (AHRS) known as the MicroStrain 3DM-GX1. It contains three angular rate gyros with three orthogonal DC accelerometers, three orthogonal magnetometers, a 16 bit A/D converter, and an embedded microcontroller. The MicroStrain can provide orientation, acceleration, and rotation information to determine heading of the vehicle. It has an AHRS mode, an IMU mode, and a vertical gyro mode. Essentially, this part contains a gyroscope and compass all-in-one.

Software Design

What our team refers to as software is all of the code on the vehicle except the computer vision algorithms. For specific details on Mako's vision algorithms, see the Vision section, and for a Module of descriptions, please see Appendix A. The purpose of Mako's software can be summarized under three primary requirements:

- To control the vehicle's movements (yaw, roll, surge, and heave)
- To control special actuators (grab mechanism, drop mechanism, and torpedoes)
- To autonomously navigate through the RoboSub 2012 competition course

This year, our team decided to completely scratch last year's code and start over. We chose to do this in order to make the code modular and to avoid having to synchronize different copies of the same data from different sections of the code. Below is a list that describes our coding environment.

- **OS:** Ubuntu 11.10
- **Source Control:** Mercurial (<http://auvc.googlecode.com/>)

- **Languages:** C++, XML, MATLAB Simulink, OpenCV
- **Build System:** qmake, make
- **Compiler:** g++ 4.6.1
- **API:** Qt 4.7.2
- **Message Passing Interface:** Lightweight Communications and Marshalling (LCM)
- **User-interface:** GUI in C++ using Qt

In order to make our code modular, we decided to break down the code into what we call modules. A module takes input data, performs its computation, and produces an output. The system consists of several modules communicating with each other to complete a goal. Refer to Table 1 in Appendix A for a description of each module in our system.

Message Passing and its Advantages

In order to completely isolate each module from the other modules, they communicate with each other through message passing. To facilitate data communications between each module, the Lightweight Communications and Marshalling (LCM) framework is adapted [1]. The message passing model is adapted for the following three reasons:

1. To promote low coupling between software units
2. To save and replay logged data from pool tests
3. To ensure type-checking in user-defined types.

A module passes a message with data to another module by publishing to a specific channel. Other modules receive messages by subscribing to only the channels they care about. Once a

module subscribes to their desired channels, it will receive any messages published to those channels by other modules. Since data is published and subscribed to on channels a module does not need direct knowledge of the existence of the data source or data destination. A module only cares about data it produces and/or consumes.

As messages are the primary mode of communication, an aggregation feature can capture all system messages. This mechanism allows post-analysis of acquired data through message replay. Since processes do not care about the origin of the data, messages can be replayed.

A header definition is required for each LCM data structure. The LCM framework automatically encodes and decodes these data structures guaranteeing that data sent and received are of the correct type.

Vision

Design Decisions

The software architecture necessitates that the computer vision module input one image per time step, perform internal processing and return information about the objects detected. One camera feed, sourced from either the forward- or downward-facing camera, is processed at any given time. The primary data resulting from vision processing is the coordinates of a selected object in the relative reference frame.

The tasks of determining which objects exist in the image, selecting the appropriate object, and estimating its position with respect to the vehicle are fully encapsulated by the computer

vision module. Design goals for the module include reliability and scalability. These are accomplished by providing an architecture which is readily scalable to accommodate multiple processing techniques for each visual task. Simple, reliable algorithms have been integrated into the code structure with the purpose of achieving reliable operation before expanding efforts with more advanced approaches.

In previous years Simulink has been used for the vision module. This year the OpenCV 2.3 library is utilized in a custom C++ object-oriented framework. The decision to move from Simulink to OpenCV was made because of the variety of algorithms available as well as proven real-time performance.

Dataflow Architecture

A second design iteration of the software architecture in light of requirements for two-way communication between Dashboard and Computer Vision includes a standard known as Dataflow 1.0. Sets of inputs, outputs, internal structures, and communication channels are partially standardized by this model. In addition to the clearly specified data objects, flexibility to handle other types is supported. Figure N shows the standard module interface. Custom classes wrap sets of LCM types, which retain their type-checking capability. This permits simplified data handling and recall for applications during both run-time and unit testing procedures.

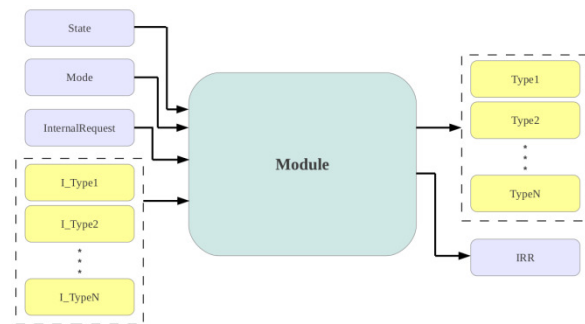


Figure 4: Dataflow 1.0 standard module

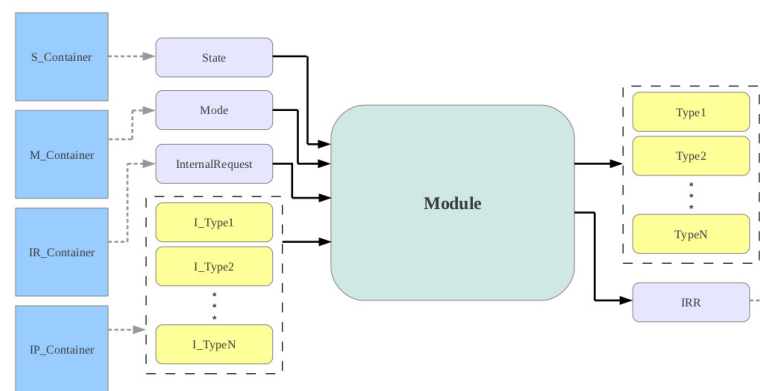


Figure 5: Unit testing data encapsulation scheme

Internal Architecture

The current obstacle course task is determined externally and reported to the Computer Vision module. Each task is handled by a dedicated class. To achieve reliable operation, multiple algorithms operate on a given image and their results are compared. Permitting the collaboration of multiple approaches is an internal class that standardizes the low-level data collection requirements, and intelligently collapses the acquired results into the object location in the relative reference frame.

Within this architecture each image encounters three stages of processing,

shown in Figure N. A preprocessing step filters unwanted artifacts and flattens colors. Color processing fully classifies colors detected in the image, taking into account dynamic lighting effects. Finally shape-based processing is performed by a set of algorithms ranging from Hough-based techniques to feature recognition.



Figure 6: Vision processing stages

Test Program

This year the vision team developed a software tool, known as Vision TestBed (VTB), which facilitates offline testing of vision algorithms. Image streams can readily be sourced from cameras, video and image files. To facilitate code development this test bench links to the classes implemented in the code base for in-situ testing and modification. The internal structure was developed in conjunction with the custom unit-testing classes used selectively by other software modules. This allows the data acquisition procedure to exist as a mere extension of structures implemented across the entire code base. Parts of the VTB useful for real-time training have also been implemented into Dashboard.

Sonar

This is the first year we decided to build the passive SONAR system in order to complete the Laurel Wreath and Emperor's Palace task. The only components that we had from previous

teams' work were our Benthos AQ2000 hydrophones.

Our system is designed to detect the time differences of arrival of the ping between each hydrophone. In order to do this, the hydrophones are mounted in a triangular array on the frame of Mako.

Our system determines if the ping is detected by allowing only the given frequencies in the competition rules to pass through the filtering stage. Next, the signal is amplified in order for it to be large enough to use as a digital input into the SONAR software system. We fabricated a custom designed PCB with the entire subsystem. This allows for the amplification applied to the signal to be variable in order to account for a weaker input signal due to the distance of the pinger with respect to the hydrophones. This is an important design decision because our hydrophones have significant attenuation in the given frequency range of the competition.

Each hydrophone has an analog channel connected to a digital input pin on a PIC24HJ128GP202 microcontroller. The arrival of the signal at one of the digital input pins causes an interrupt handler to start measuring the time until the signal arrives at each of the other two inputs.

Once the third hydrophone detects the ping, the interrupt handler signals the main loop of the software in order to begin computation. The algorithm uses the measured time differences to determine the heading of the source of the ping. Once the algorithm has executed, the system remains idle until there is silence before enabling the hydrophone interrupts again. This prevents the input of the PIC to

continually be triggered before the algorithm has completed the calculations from the previous ping.

The algorithm had to be developed in parallel with the analog section, therefore the software needed to be tested independently. To facilitate the rapid development of the SONAR software algorithm, we developed a test harness to allow the algorithm code to run on a desktop computer. The test harness walks a simulated ping through a 2-D space and computes the difference between the actual angle to the source of the ping and the angle computed by the time difference of arrival (TDOA) algorithm. The error in the heading calculation at each point is rendered as a gray map, with lighter shades indicating greater error, as seen in Figure 1 of Appendix A. This allows us to rapidly observe the effect of changes to the code, which accelerates the development of the algorithm.

Controls

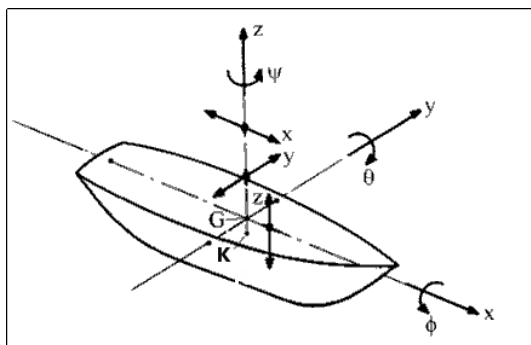


Diagram of typical degrees of freedom

Design Decisions

Building on the software from previous years a Simulink model was structured to provide vehicle control. The Simulink library offers discrete-time PID loops with advanced tuning options, rapid code verification capabilities, and has

been shown to incur an acceptable overhead in computation time. The controls code was re-factored with simplicity in mind, in light of a new thruster configuration.

Internal Architecture

The control system is implemented in a dedicated software module. The module listens for the instructed state from Director and values from all sensors are available for query. Control is provided with four thrusters aligned with the vehicle's surge and heave axes. This setup allows for active control over four decoupled degrees of freedom – surge, heave, yaw, and roll. One PID controller is operational for each DOF and assumes an operational mode. The combination of the controller modes synthesizes the controls module state, which can be altered externally over the message-passing system.

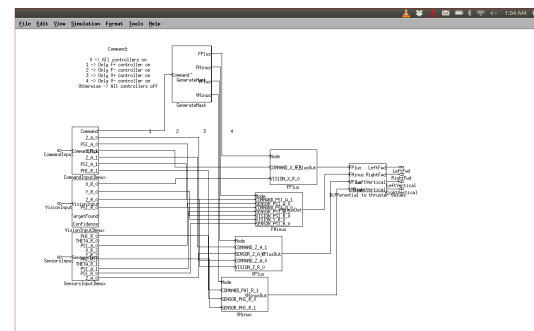


Figure 7: Simulink controls model

Testing and Simulation

The Simulink model is linked into a custom C++ class which interfaces with the software system. In addition to the primary model a Simulink-based test bench was built for static tests during development. In conjunction with the Dataflow structure implemented in the software, a set of unit tests are implemented at the Lcm level. These

are executed at build time and verify correct operation of the controls module.

Concluding Remarks

As mentioned in the Abstract, we are currently at the tail-end of the second year of our two year cycle. We have made many modifications to the vision, controls and software code, as well as made minor modifications to the mechanical design, thruster configuration, and e-rack. We have even added all three mechanisms- and, perhaps our biggest undertaking, designed, fabricated, and tested a working sonar platform. Needless to say, this year has brought forth many challenges, but also tremendous understanding. We are taking a team of eight to competition, which [for our team] is an excellent accomplishment. We have also done a significant amount of recruiting and 'spreading the word' about our team and the competition as a whole. Overall, we are very proud of our accomplishments and the way our team has progressed over the course of the year.

Acknowledgements

We could only achieve all of the goals we set to accomplish with the help of our sponsors and excellent support system. We would like to thank the following companies and individuals for their support;

- The Ware Lab at Virginia Tech
- The SEC at Virginia Tech
- VaCAS
- Lockheed Martin
- Mathworks
- Ms. Christine Durner
- Dewey Spangler

Please excuse the excess pages. We felt the Appendix was a necessary part of our written report.

Appendix A.

Table 1: Module Descriptions

Module	Description	Input	Output
Device Manager	To provide a hub that aggregates all hardware data into one entity	None	Sensor data
Actuator	To send signals to our mechanical systems such as: activating thrusters, firing torpedoes, initiating the grab and drop mechanisms.	Mechanical device data commands from other modules	Mechanical actuation
Vehicle Controls	To provide an interface for navigating the robot; self-correcting feedback-based algorithms for stabilizing vehicle movement	Sensor data, desired values data	Thruster commands
Computer Vision	To provide intelligent sensing of our surroundings based on cameras	Task enumeration, camera data	Orientation angles to object(s) of interest
Director	To allow the robot to make intelligent decisions based on its surroundings; facilitate the rapid prototyping of mission files through dynamic loading of scripts	Mission file, target information, sensor data	Desired values data, Vision and Controls mode data, Controls commands (for direct control)
Dashboard	To provide a GUI for user testing; ability to run simulated and real-time runs; load saved data; view vision processing results	Sensor data, all relevant outputs from other modules	GUI representation of data and processing

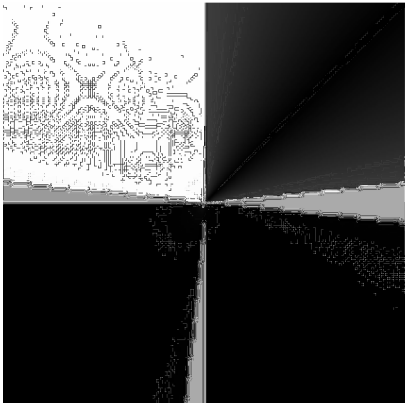
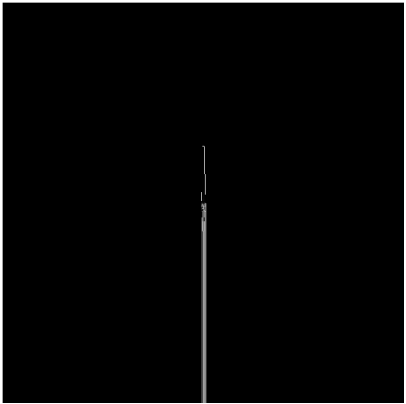
	
<p>Output of an early iteration of TDOA algorithm</p>	<p>Output of the most recent iteration of TDOA algorithm</p>

Figure 1. Sonar images depicting test harness outputs

Bibliography

[1] A. S. Huang, E. Olson, and D. Moore, "Lightweight communications and marshalling for low latency interprocess communication," Massachusetts Institute of Technology, Tech. Rep. MIT-CSAIL-TR-2009-041, 2009.