# The Barracuda Project
## Building an efficient reliable platform for underwater operations

*Daniel Bryant, Tamsen Drew, Nick Gurtler, Chris LaFlash, Nick Rapp, Karl Schulze*

**Amador Valley High School Robotics**
**2426 Crestline Rd.**
**Pleasanton, CA 94566**
**(925) 485-7890 Email: karl.s@attbi.com**

## ABSTRACT

The AVHS Barracuda is a fully autonomous underwater vehicle platform, developed for the Fifth International Autonomous Underwater Vehicle Competition   The vehicle has a mass of 30 kg and fits within a 52x32x14¾" box, small enough to place in the back seats of most cars. Four engines provide propulsion: two laterally-mounted to control speed and heading, and two vertically-mounted to control pitch and depth. The vehicle includes a full suite of external sensory systems, including an active sonar altimeter, a 3-axis compass with 50 degrees of pitch/roll range, a side-scanning active sonar system with a 90 degree scan range, a water pressure sensor, and a camera. Additionally, the system has a suite of self-diagnostic modules, including five water alarms, several internal temperature sensors, and a software integrity checker to verify the data coming from the various systems.

Autonomous control is provided by a single-board computer running Linux. The computer is the center of the vehicle, individually linked to each sub system and is in charge of all autonomous control. Furthermore, the system is designed to be as robust as possible, as it can function even in the absence of many of the external sensors. The challenge of any engineering project is to piece together efficient and reliable pre-existing subsystems, but to nevertheless produce something novel; Barracuda rises to that challenge.

## Problem Statement

The Association for Unmanned Vehicle Systems International and the U.S. Office of Naval Research run annual competitions that challenge teams to build autonomous underwater vehicles. Each year, a mission is presented; each mission is designed to be an analogue to real challenges facing the developers of autonomous systems. The AVHS Barracuda was built for the fifth competition in this series, which imposed new constraints and provided the opportunity for a broader range of sensory systems to be brought to bear upon the problem.

The arena for the 2002 competition consists of a large ellipse with a flat bottom at a depth of 16 feet, coupled with a parabolic indentation at the center. The indentation reaches a depth of 38 feet (from the surface), takes up a circle with a 160 foot diameter, and is radially symmetric. The side walls of the ellipse are sharply curved, providing an almost flat surface. The material of the arena is highly sonar-reflective, simplifying the task of active sonar driven navigation.

Placed inside the arena are several mission targets. The vehicle is launched near one end of the short axis of the ellipse and has to first navigate through a floating inverted-U shaped gate. This challenge insures that the vehicle has the minimal required functionality to perform the remainder of the mission. After passing the gate, the vehicle must drive over a series of boxes placed at various depths in the arena. Each box includes a visible custom bar-code, and the challenge is to correlate bar-code identifications with the depths of the boxes.

The boxes are located in two rings in the arena: one ring of eight boxes arranged in a forty foot diameter circle at the center of the arena (within the parabolic indentation) and the other ring of eight boxes arranged in an ellipse 10 feet from the edge of the arena. Both sets are arranged with equal angles of separation between boxes. There is also one final box located in the center of the arena.

Also at the center of the arena is an acoustic pinger, which emits a 27 kHz pulse at regular intervals. The pinger provides the opportunity to simplify navigation, as knowing the direction toward the pinger simplifies the task of driving in a circle to locate the boxes.

## Introduction

Manta Ray, the vehicle developed for the 2001 competition, was a powerful and reliable system for underwater missions; however, it was not without its drawbacks. Of primary concern was the large amount of power consumed during operation, requiring a significant portion of the allowable weight (100 kg) to be devoted to batteries. Further, the large weight of the vehicle meant that transportation and deployment required either disassembly or special equipment, impeding the test-revision process. With these issues in mind, Barracuda was designed to retain the flexibility and reliability of the Manta Ray platform, but with significantly less weight and power consumption. (Table 1)

| | Barracuda | Manta Ray |
|---|---|---|
| **Weight** | 30 kg | 98 kg |
| **Power Consumption** | 220 W | 450 W |
| **Dimensions** | 52x32x14¾” | 72x32x24” |

Table 1

To accomplish this goal, several major changes were applied to the Manta Ray design. First off, the number of sealed containers for electronic components was reduced from three to one, which saved on both weight and cost of plastic and production. The primary container was also greatly reduced in size, further reducing the weight. Rather than using DC trolling motors for the vertical thrusters, which were overpowered, a bilge pump system was installed, greatly reducing both weight and power consumption. A more efficient computer was purchased, which also reduced power draw and allowed for a smaller power supply, which produced less heat and required less weight and space. The various power saving changes allowed for a significant reduction in battery size; to make the weight savings even greater, the design was switched from gel-cell security batteries to nickel metal hydride (NiMH), which have a much higher power density per unit weight.

## Hardware Overview

Barracuda is designed as a composition of independent functional subsystems (Figure 1), simplifying the replacement of any given part, should it break. The mechanical components serve mainly to keep water away from the electrical components, but also to drive the vehicle and ensure proper stability in the water. The electrical components provide the central computer with the sensory information it needs to complete the mission, and insures that the proper signals are sent to the engines.

### Mechanical Components

The core of the vehicle is an acrylic tube, 6 in diameter with 3/16 inch walls, secured to an aluminum frame with hose clamps. The main tube contains the majority of the electronics for the system, including the computer, as well as all the batteries. Mounted to the frame are four engines and other external sensors, all of which connect to the main tube via Brad Harrison waterproof connectors. The tube is sealed with two end caps using compression o-rings that face the inside of the tube, with latches to insure proper placement. Attached to the top of the vehicle are two metal handles, allowing for easy deployment into the water. Furthermore all aluminum components have been anodized for corrosion resistance as well as aesthetics.

### Main Engines

The main engines are Motor Guide ET22 Trolling motors. Mounted laterally to the frame, they provide control of forward and reverse thrust, but can also be run at different speeds to allow for heading control. The engines are mounted to the underside of the frame using two U-bolts per engine, providing a secure mounting, but still allowing for easy adjustment of position and pitch of the engines. Each engine spins in the same direction for forward motion (as they use identical propellers), but by adjusting the pitch of the motors, it is possible to counteract the resulting roll torque. Both engines have been modified to include an internal water alarm and to also use a waterproof connector for connection to the main tube. Additionally, a welded aluminum shroud is attached to each engine, in order to allow divers to work in close proximity to the vehicle with minimal risk of injury due to spinning propellers.

### Vertical Engines

The vertical engines are Rule 1100 bilge pumps, using CPU fans as props. These

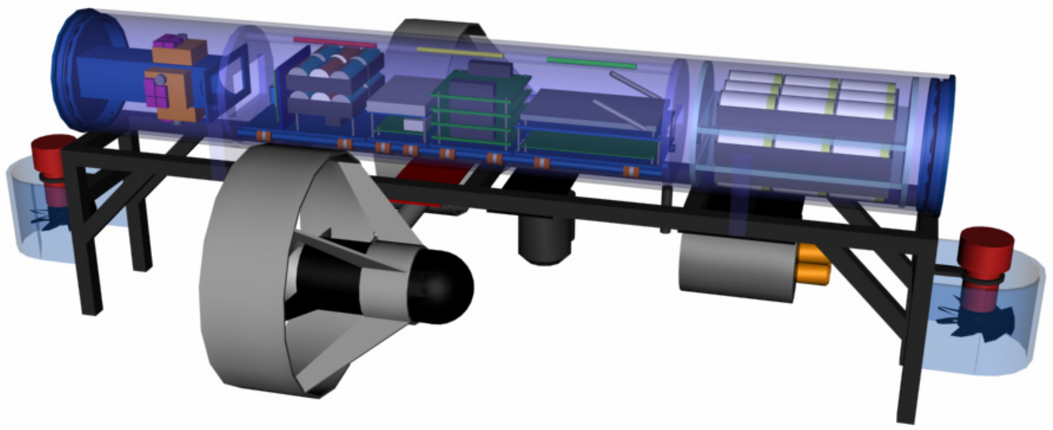

Figure 1 – 3D Rendering of Vehicle

were chosen as they provide a convenient pre-sealed motor that operates on very little power (approximately 2 A max draw). Each bilge pump has been modified to accept a waterproof connector and to contain a water alarm. The bilge pumps have been sealed for submerged operation and are mounted to the front and back of the frame. For safety, acrylic shrouds are placed around the engines.

## Main Batteries

The main battery pack is located in the front end of the main tube, and is composed of 24 D-Cell 9.8 A-hr nickel metal hydride (NiMH) batteries. The batteries are arranged in two parallel arrays of 12 batteries each, providing a total of 224 W-hr at 14.4 V. We use a Tri-M PC/104 power supply and charger, which allows us to properly charge the batteries and to reduce the 14.4V output to the appropriate voltages for the computer.

## Connector Endplate

The connector endplate provides the interface between the main tube and the external components. The plate is made of aluminum and acts as the sealing surface for the rear end of the tube. Attached to the plate is a rectangular shaped housing, where the pins from the waterproof connectors are routed to an Elcon Products Rack Connector. Mounted along the outside of the housing are Novak RC speed controllers (Novak Super Roosters), located here to provide the speed controllers with a ready heat sink to the water as well as to isolate their noisy switching activity from the rest of the systems. Additionally, the pressure sensor is mounted directly into the plate, with the pressure transducer directly exposed to the water.

## Data Rack

The data rack sits in the main tube, resting between the connector endplate and the main batteries. Elcon Products Rack Connectors connect the rack to both the

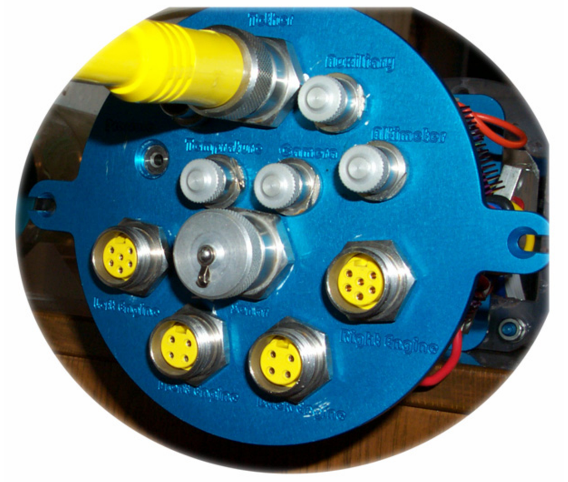

Figure 2 – Connector Endplate

connector endplate and the main batteries, allowing the three main components to simply snap together. This facilitates servicing of all three components, as well as replacement of the main battery pack. At the far end of the rack, near the connector endplate is a custom terminal block, which mates the ribbon cables for various ports to the appropriate pins on the connector endplate. The use of ribbon cables for each general port helps to reduce confusion and simplifies wiring.

Also on the data rack is an internal array of batteries, 12 NiMH C batteries, arranged in series to give 54 W-hr at 14.4 V. This internal battery pack serves as an isolated power supply for the computer and allows for hot-swapping of the main battery pack without shutting down the computer. Mounted in the center are the PC/104+ computer stack and a Precision Navigation TCM2-50 3-axis compass.

## Computer System

The computer is an EEPD PIII mobile chipset PC/104+ computer using a Sensoray Model 311 Framegrabber and an Advantech serial expander. The computer runs at 700 MHz, with 256 MB RAM, providing ample processing power for the mission control code and hardware drivers. In addition to the primary Ethernet adaptor on the computer, we have a USB wireless card and an additional Ethernet adaptor that is used for communication with the side scanning sonar. The system runs a Red Hat distribution of Linux and uses an IBM Travelstar 12 GB hard drive for persistent storage. On average, the computer system consumes only about 15 W of power.

## Side-Scanning Sonar

Mounted in a separate module, so as to minimize electromagnetic interference, the side-scanning sonar provides the vehicle with an active sonar image for one side. The controller is a Rabbit 2000 microprocessor, which uses an Ethernet interface to communicate with the main computer and uses serial to drive an Interphase controller board. This board allows us to drive dual phased array transducers. Each transducer features eight individual ceramics, which are combined in order to gather up to 90 degrees of depth information at approximately 5 Hz.

## Altimeter

Local depth information is collected using a Marine Electronics Model 5001 500 KHz altimeter. The ceramics and electronics are mounted in a custom PVC housing and connected to the vehicle using a waterproof connector. Power is supplied at 12 V and communication is handled with a direct serial protocol with the controller board. The board includes numerous potentiometers to allow setting of gain values, in addition to being fully programmable via the serial link. Samples are gathered at approximately 30 Hz by the controller board, which drives the ceramics to produce a narrow beam of only 4 degrees, and return depths resolved to the mm. The vehicle uses the altimeter to take very precise measurements directly beneath its center, which is useful for determining its depth column, as well as the depth column of the boxes.

## Camera

The vehicle uses an RF Concepts Submergible Bullet Camera. A waterproof connector is used to connect the sealed housing for the camera to the vehicle. The CCD provides a 320x240 image and an array of LEDs provides for 0.1 lux operation. Color NTSC output is routed to the capture card, where the vehicle processes frames at the rate requested by the optical analysis algorithms. Higher quality digital industrial cameras are available, but the NTSC feed provides for an easily debugged signal, a cheaper camera, and a simple method for retrieving the image, using a readily available and Linux-supported frame grabber.

## Speed Controller Board

Mounted to the connector plate, the speed controller board is a custom microcontroller board, using a PIC microcontroller. The board receives information from a variety of internal and external sensors and routes them to 10 bit analog to digital converters, with appropriate filtering and isolation components where necessary. Additionally, the board generates PWM servo signals in order to interface to the Novak speed controllers. The computer communicates with the board using a simple protocol over a serial connection.

## Tether and Communications

A communications link to the main computer is required in order to facilitate testing of the vehicle. This is provided via a wireless 802.11b implementation, when on the surface, and by using a tether, when doing dive testing. The tether includes a 24V power line as well as a 100MBit Full Duplex Ethernet connection. The power line serves to power the computer, as well as to charge the main batteries. This allows for long test-runs without having to recharge the internal computer battery. Further, the tether, when plugged in, enables usage of a remote hardware kill.

## Software Overview

Running on the computer is a Red Hat distribution of Linux, optimized to include the services relevant to the task of the mission. Communication with the vehicle is handled using telnet, FTP, Samba, and a custom protocol (written over UDP) for communication with the sub control code. All control processes execute in parallel, sharing information through shared memory mapping. To improve on last year's shared memory design, the single shared memory chunk was broken into several structures, allowing for more efficient sharing of the information over the network.

### System Control Process

The primary control process manages all control over the vehicle. It handles the work created to establish the shared memory space and handles all network communication using the custom UDP protocol. All other processes on the vehicle are launched using this process, allowing it to manage their execution. Should a mission-critical process fail, the master process detects this condition and ensures that it is restarted. Additionally, the master process provides a mechanism for processes to provide log messages to be sent to the clients, for the purposes of debugging, as well as mission logging in order to analyze a mission after the fact.

### Shared Memory Structures

All processes communicate through shared memory, which is broken into several structures. The first two structures manage processes and client connections, providing the backbone for the master process. Additionally, there are two structures that contain configuration info, one that contains configuration specific to a given run, and one that contains global configuration info for the whole system. The next two structures are real-time-value structures that contain information on the mission-specific and general system values, for display in the client. In addition, the driver processes communicate through the real-time values interface. Finally, we have a structure that maintains error state information, allowing any process or client to see what modules are reporting errors, so as to know what data is valid. The entire shared memory subsystem, coupled with the system control process, makes for a flexible and robust environment in which the various control processes can operate.

## Driver Processes

All driver processes communicate to the software system through the real-time values shared-memory structure. Each process implements the protocol to access a specific piece of hardware on the vehicle and provides a transparent interface to the hardware subsystem.

### Speed PIC

The speed PIC driver process interfaces with an internal controller board powered by a PIC microcontroller. The board uses a custom serial protocol to communicate with the computer and provides an interface to several sensors, including the pressure sensor and various water alarms, as well as serving as the engine controller; the PIC has four PWM servo outputs, which drive the Novak speed controllers. In addition, it provides an interface to control status LEDs and to retrieve the status of the physical kill switch.

### Compass

This process decodes the serial stream from the compass, providing near real-time heading, pitch, and roll information. The vehicle uses a Precision Navigation TCM2-50 compass that, once configured, is extremely reliable. This is important, as the compass serves as the central component in the vehicle's navigational system.

### Power

This process interfaces with the Tri-M HEBC and Tri-M HESC104 Charger and Power Supply, providing information on the state of the vehicle's power system. Each device has a serial interface, providing simple access to voltage, current draw, and whether the batteries are currently being charged. This information helps the vehicle to estimate the power draw of the vehicle, as needed, and helps it to take steps to reduce power consumption should the need arise.

### Altimeter

The Altimeter process gathers depth-to-bottom information from a serial interface. The process also looks for changes to the configuration of the altimeter in shared memory and applies the changes to the device, allowing for real-time testing and tuning of the device. As with the compass, this interface is simple, but extremely important to navigation and collecting the box depths.

### Active Sonar

To complement the depth-to-bottom information returned by the altimeter, we have a side-scanning sonar system to give us information on our position within the competition arena, relative to the walls. This process uses a TCP/IP interface over a dedicated network port to communicate with the Rabbit microcontroller board. The board is configured to scan at two different angles, with a few samples at each, and to return an averaged distance to the wall in each direction. This information, coupled with the altimeter and compass, gives the vehicle everything it needs to navigate the arena.

### Passive Sonar

The passive sonar process communicates with a custom PIC board, designed to collect information from four Reson TC4013 Hydrophones. A Maxim Bandpass Filter and Zero Crossing Detector provide timing information to the processor, which transmits the data back to the main computer for processing. From the data, a heading and azimuth to the pinger is derived, using a simple planar wave approximation and the known speed of sound.

## Mission Processes

The mission processes contain the logic for the actual mission. Mission state is coordinated using the shared memory structure, both with flags, and by terminating/launching processes.

### Feedback Control

Used by most of the high-level navigation processes, feedback control allows the vehicle to adapt itself to changing external situations. This process implements a general Proportional-Integral-Derivative control loop, which can be run on any number of feedback inputs. Every feedback mechanism used by the system simply has to supply a configuration, an input value, and a set point; the feedback control process then produces a value from -1 to 1, indicating to the vehicle how it needs to change to reach the set point. The feedback control is used to control heading, depth, and pitch, but also for some of the high-level navigation tasks. The control loop itself includes several features, including derivative filtering (important, for sensors that have noise spikes occasionally) and mechanisms to help prevent integral windup.

### Initial Navigation

Once the mission process is started, the vehicle waits for the kill switch to be inserted. Upon insertion of the kill switch, the vehicle first dives to a cruising depth and travels straight ahead through the validation gate. After a certain amount of time has elapsed, the vehicle considers itself through the gate, turns to a pre-defined heading, and drives toward the center of the arena. The vehicle is looking for a specific depth-to-bottom value (the depth-to-bottom at which the boxes are located) and a feedback loop is run until the vehicle is over the desired depth.

## Inner Circle Boxes

At this point, the vehicle is positioned directly over the circular ring where the boxes are located. A 90 degree change of heading is requested, the vehicle dives to box-searching depth, and another feedback loop is started. This feedback loop takes the depth to bottom as input, and uses the output to adjust heading so as to maintain the vehicle's radial position relative to the center of the arena. The process continues with throttle at a fixed cruising rate, which causes the vehicle to gently oscillate its way around the circle.

## Optical Box Finder

As the above navigation process is running, an optical analysis algorithm is constantly scanning the images coming from the downward facing camera, looking for possible boxes. The process first runs horizontal and vertical contrast filters on the images, producing a black and white buffer with edges marked. The advantage of a barcode is that it has a lot of edges, so the next step is to try to locate regions of the image that are dense in edges.

This is done using a rectangle producing algorithm, which attempts to place areas of large markup density into boxes for further analysis (Figure 3). To perform this step, the process uses an algorithm that recursively scans the image. The first step is to load the image buffer from the camera into a 512x512 buffer. This larger block is then recursively subdivided into four smaller blocks, so that you have 256x256, 128x128, 64x64, 32x32, 16x16, 8x8, 4x4, and finally 2x2 blocks. At each of these levels, we accumulate information, starting at the pixels, and then combining them into the 2x2 blocks, and so on, building larger and larger blocks. A large amount of information can be collected, but for this algorithm it is sufficient to count the total number of pixels enclosed in the box and to then decide whether that number is high enough to count the whole block as 'marked'.

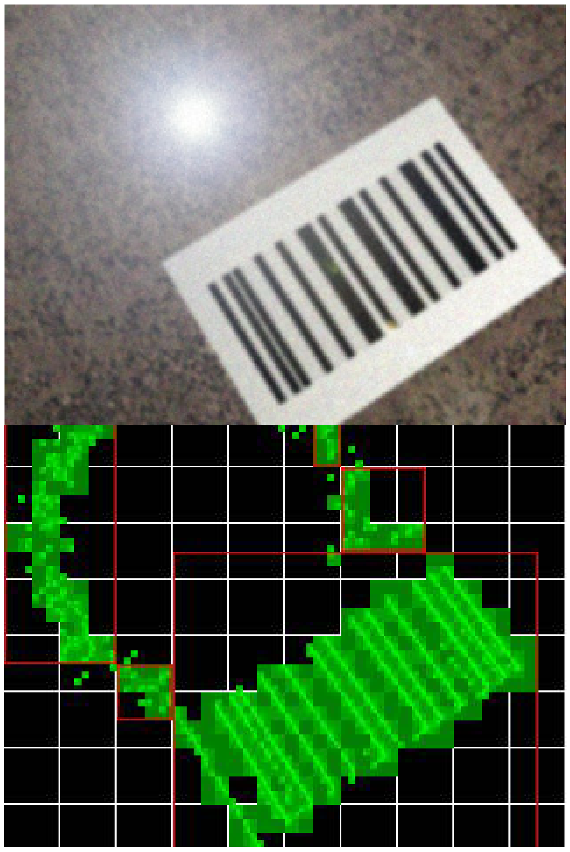
Figure 3 – Optical Box Finder

From a logical standpoint, the collection of blocks produced by this algorithm is essentially a tree structure, with each of the larger blocks containing four child nodes. However, because of the fact that the tree is completely filled, and it contains so many powers of two, it's more efficient to store it as a linear array of blocks. On top of this array structure, then, is provided an iterator class, which allows the algorithms to walk the entire tree structure of the image, including the pixels.

With this iterator, it is possible to create efficient algorithms to analyze the markup density of the image. Because of the way the tree was constructed, areas of high density will have lower level blocks marked, but they will also have the higher level blocks containing them marked. To locate areas of interest, we can then do a recursive scan, starting at some level of interest (say, 64x64) and scanning down further into blocks that are marked. Once the algorithm reaches a certain level, it can glue these blocks together to form rectangles. This gluing process is done using a flood fill, which combines each block with its immediate neighbors and continues the process until no more neighbors are found. By gluing the blocks together at a level higher than pixels, the algorithm can piece together a disjoint image, such as would be expected to be produced by a bar code.

## Bar Code Analyzer

Once the rectangles that could potentially contain boxes have been located, the vehicle has the task of figuring out which contains a valid barcode, if any. To do this, it breaks each rectangle into several lines to scan, passing each to a barcode scanning routine. The scanning routine looks at the histogram of pixel widths for marked or unmarked runs of pixels. Because of the nature of the barcode, a valid barcode should contain a bimodal distribution with one peak at approximately twice the other in pixel width.

Assuming that the histogram looks reasonable, the algorithm starts scanning the line, looking for a start frame, followed by the start bit for the barcode. If this pattern is located, it then attempts to read a barcode starting from that position. If a valid barcode is found, including a valid parity bit, the algorithm returns the barcode number. If not, it returns that it found something looking like a barcode, but was unable to analyze it. The process is repeated, reading the line in reverse, reducing the number of lines that need to be fed to it to find a match.

## Box Tracker Navigation

When a valid box is located by the optical algorithm, the vehicle needs a way to position itself over the box to acquire a depth to bottom. The position of the box in the image is converted into a driving distance and heading change, which is used to navigate to the box. This particular navigation algorithm doesn't use the feedback control system to handle the feedback, as it is actually generating the driving direction directly from the image.

## Center Box Navigation

Once all of the boxes in the inner ring have been identified and given a depth value, the vehicle makes its way to the center of the parabolic indentation. Located there is an additional box, so the vehicle turns on the optical analysis algorithm once close enough. The vehicle analyzes the barcode, retrieves the depth, and then sets the driving depth to the cruising depth for the outer ellipse set of boxes. Once at depth, the vehicle sets a heading for just before the first box after the gate and starts driving. After the correct amount of time has passed, the vehicle stops driving, makes a ninety degree turn, and launches the next phase of the mission.

## Outer Ellipse Boxes

The vehicle is now located with its side-scanning sonar facing the wall of the arena. Here, the vehicle wants to drive in an ellipse, rather than a circle, and must use the distance to the wall as feedback, rather than the depth to the bottom. The same optical analysis code runs in the background, interrupting the navigation when a box is located, and insuring that the depth gets recorded.

To perform the ellipse navigation, another feedback loop is required. The side scanning sonar process produces two distance values, such that the distances and the angle between them define a triangle, with the vehicle at one point, and the other two points on the wall. From this information the vehicle wants to figure out the correct heading to stay on for the ellipse path, as well as how to correct heading to insure that it stays the proper distance from the edge of the wall.

If the path were a circle, this would be relatively simple, as two feedback loops could be used. The first would take the difference between the two distances, and would try to make that difference go to zero. The second would take the average magnitude of the distances and would try to make it go to the correct value for a certain driving distance from the wall. However, the vehicle can use these simple feedback loops if it has some way to correct for the ellipse shape.

To do this, the vehicle uses its current heading, which it assumes to be close to the correct heading, to estimate the absolute spatial vectors for the two sonar returns. These vectors can then be scaled, along with the ellipse, to make it look like a circle. From the magnitudes of these vectors, the above feedback loops can then be run.

### Mission Completion

Once all the boxes in the outer ring have been identified and depth-scanned, the mission is complete. Because of the order in which the mission is completed, the vehicle will be right back near the dock where it started. The vehicle will drive forward a predetermined distance, to end up near the launch platform, and will then surface, completing the mission.

## Conclusion

Simple and reliable subsystems form the core for Barracuda. From feedback control software to Novak speed controllers, or the software process controller to a PIC microcontroller, the vehicle is a composition of subsystems. The subsystems are simple, reliable, and most not especially novel, but by combining them in the proper ways, a novel system is built. Engineering is an evolutionary process: with every iteration of the design process, more and more complex reliable subsystems can be built.

Autonomy is the natural extension of computer aided, but still mostly human-driven processes. Such transference of responsibility can only take place with the development of reliable subsystems upon which the autonomy can depend. Once so freed from their low-level obligations, the humans can begin to focus on the higher-level goals, with the assurance that the technology will function as intended. This is the goal of all engineering, and autonomous systems are at the forefront of that goal; with Barracuda, our team hopes to contribute a small piece to the worldwide toolset of subsystems, from which the future will be built.

## Acknowledgements

We would like to thank our sponsors: Amerimade Technologies, Wunderlich Malec Engineering, Rune Eliassen, Niles Machine Shop, PASS, PPIE, Pleasanton Rotary, Marine Electronics, Livermore Rotary, Clorox, Elcon Products, Sensoray, Tri-M. Our special thanks goes out to those that have donated their time in order to help us out; especially our advisors Mr. Shelley, and Mr. Hall; the engineers at Amerimade, Hank Reinstein, Rob Quandt, and Todd Thomas; all of our parents for their support, technical knowledge, food, and garage space.