Nautical Autonomous System with Task Integration

Authors: Jeremy Borgman, Terry Max Christy, Zackary Knoll, Steven Blass
Advisors: Dr. Gary Dempsey & Mr. Nick Schmidt

Department of Electrical Engineering
Bradley University

Abstract

The goal of this project is to design and implement an autonomous mobile boat that can navigate a nautical buoy channel. To accomplish this task, the robot needs to be able to detect and differentiate the buoy colors, be able to navigate the channel effectively, and avoid the obstacles. To accomplish this task a BeagleBoard development board using a TI ARM Cortex-A8 processor has been selected as the main processing unit for the system. To make the robot mobile, two main thrust motors and four side thrusters have been built into the nautical platform of the robot. This chassis is a catamaran style boat with a flat deck to support the BeagleBoard and Remote Acquisition for Interfacing navigation Systems (BRAINS) box and the Motor Interfacing Module (MIM) box. The thrusters are then controlled via control signals sent from an Atmega series microcontroller in the BRAINS box to the MIM box using a serial communication protocol. There is a top mounted web cam that is used to acquire the vision input to the system. The images from the camera are then processed to determine an optimal path through the buoy channel using high level algorithms based in object oriented C++. Using this optimized desired path, a proportional digital controller is used to correct the current trajectory to match the desired path.

I. Project Summary

The members of Team NASTI (Nautical Autonomous System with Task Integration) chose to design an autonomous system based around a catamaran platform as the basis for a senior project. The platform will be designed so it can autonomously navigate a channel of water buoys. The robotic platform will compete in the AUVSI (Association for Unmanned Vehicle Systems International) Foundation and ONR's(Office of Naval Research) 5th International RoboBoat Competition. Our project will center on completion of the main navigational task of sailing through the buoy channel and avoiding the obstacles using image processing. The buoy channel can be seen in Figure 1 [1].

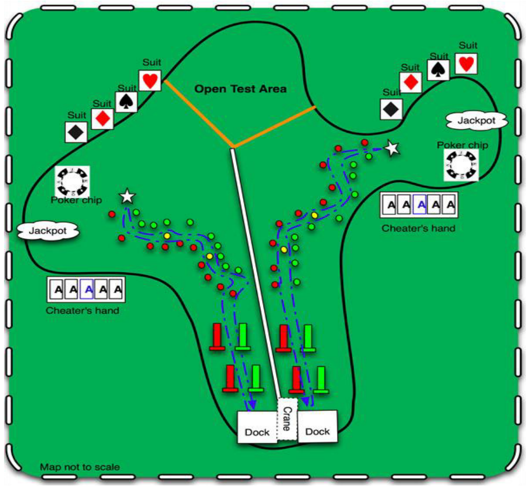

Figure 1. Competition Layout

II. Boat Construction

The basic form factor for the boat utilized by team NASTI was a catamaran style vessel. A pair of pontoons connected via two beams with a platform deck mounted on top of the beams served as the basic shell for the boat. The overall size of the boat was driven by a few key factors. The distance between the pontoons was set by the propeller diameter and the added safety shrouds of the pair of main drive motors. The final beam was then set by the individual widths of each pontoon section and the distance between the pontoons. The width of each pontoon was determined by how much lifting force was needed and the physical size of the smaller side thrusters that were purchased for each pontoon. The small holes just before the taper of the bow and aft sections of each pontoon are the intake/outlet ports of the small side thrusters. The side thrusters were added to improve the boat's maneuverability both rotationally and laterally. A more in depth discussion about the main motors and side thrusters will be discussed in later sections. The finished size of the each pontoon was 44 inches long by 5 inches wide. This provided approximately 6.5 pounds of lifting force per inch of displacement for each pontoon. The height of the pontoon sections was 7.75 inches at the time of final completion. This gives the boat an ideal lifting force of about 90 pounds at 7 inches of water displacement. The framework of each pontoon is composed of plywood bulkheads sandwiched between a plywood top and bottom. Closed cell foam was added to increase glue surface area without sacrificing weight. The frame was then encased in polystyrene plastic to provide a watertight vessel. The addition of a deck and main drive motor supports rounded out the boat that was to be used for remote and autonomous control.

III. System Overview

Shown in Figure 3 (on the next page) is a detailed overview of the autonomous boat. This is a high level visualization of the information flow from the data acquisition to the motor control system. There are seven major sub-systems each of which will be discussed below in a bottom up order, starting with the motor interfacing module and ending with the 32 bit control code written for the embedded system located in the B.R.A.I.N.S. module. Figure 2 shows the actual side thrusters and main drive motors. The small holes in the side will be known as side thrusters. The two larger motors in the aft will be referred to as main drive motors.

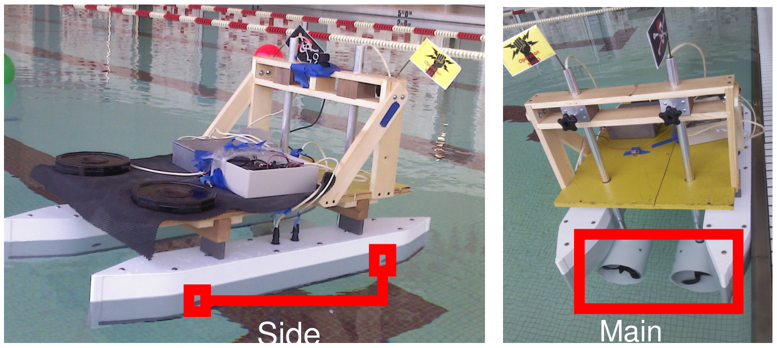

Figure 2. Boat Motors

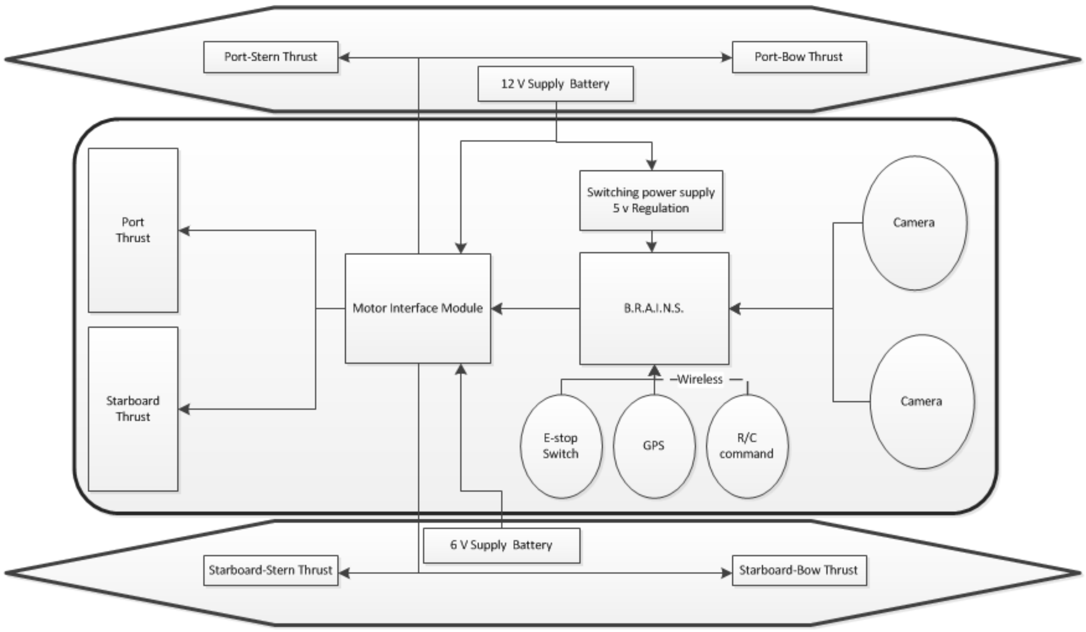

Figure 3. System Overview

IV. Motor Interfacing Module

The purpose of the motor interfacing module is to decode the output of the BRAINS box. Figure 4 shows the layout of the module.

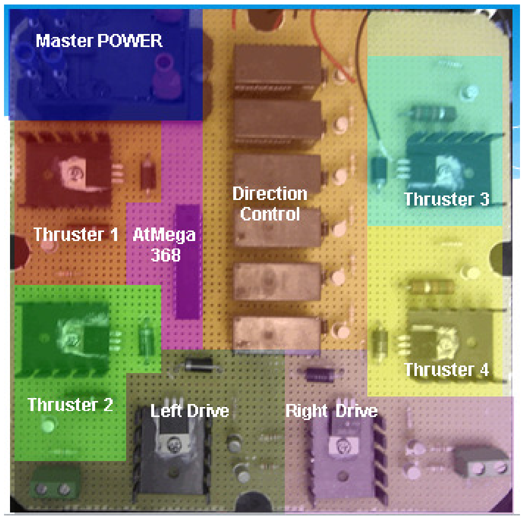

Figure 4. Motor Interfacing Module

As can be seen in Figure 3, there are four side thrusters to help pivot the boat and move it laterally. The thrusters require twelve volts and have an estimated peak current of 3A. To accommodate these requirements, a two stage NPN PNP transistor circuit was created. This can be seen on the next page in Figure 5. The microcontroller can source a maximum of 40mA and the motor requires 3A. Additionally the motor runs on 12 volts and the microcontroller runs on 5 volts. The hFE of Q1 was not large enough to increase the current to the needed 3A. For this reason, two transistors were used. A diode was added as a freewheeling diode to prevent the voltage generated by the back EMF of the motor from damaging the transistor Q2.

The circuitry for the differential drive motor is shown in Figure 6. As can be seen, the general layout is the same with different discrete components. The motors are powered off a lead acid 6v battery with an estimated peak current of 8 amps. To satisfy these conditions the circuit in Figure 6 was created. The larger base current needed for this motor lead to a different transistor for Q2 in Figure 6 than Q2 in Figure 5.

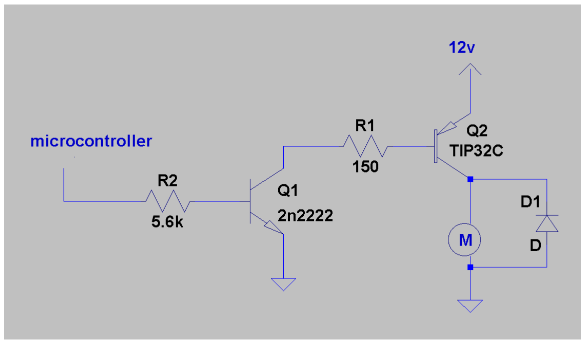

Figure 5. Thruster Circuitry

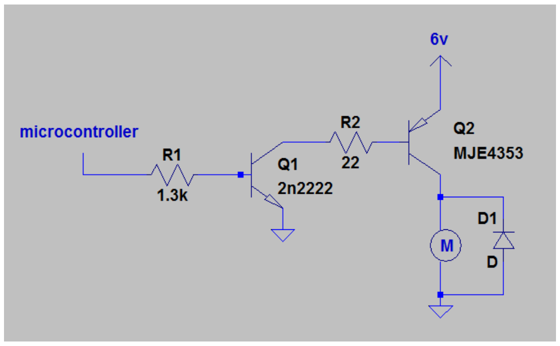

Figure 6. Differential Drive Circuitry

All of the motors needed to be run in either direction which necessitated implementing bi-directionality circuitry. The high current specifications combined with the goal for a low cost solution ruled out most of the typical methods to achieve bi-directionality. A double pull double throw relay connected to a switching transistor was ultimately utilized. This can be seen in Figure 7. When the coil of the relay becomes active, the switch moves into the open position. This flips the polarity of the motor resulting in the motor turning in a different direction.

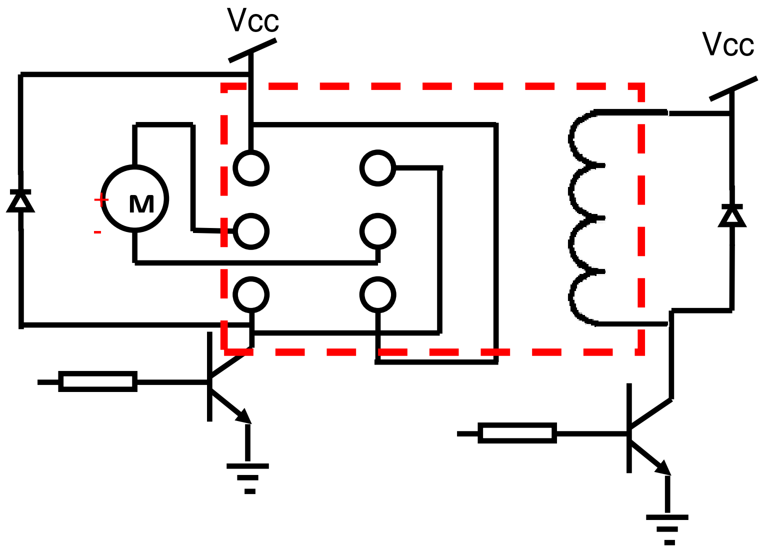

Figure 7. Bi-directional Relay Circuitry

V. Remote Control

Using the Futaba T6EX R/C transmitter (Figure 8 on next page) with the Futaba R617FS receiver and a communication protocol described below, a remote control scheme has been developed for emergency control of the boat and for use in testing. In this control scheme, the platform is allowed 3 degrees of freedom that can be controlled from 3 axes of the transmitter joysticks. To enter this emergency control mode, the right shoulder switch has to be flipped in the downward position to force the master Atmega128 in the BRAINS module to send motor control bytes based on the r/c commands rather than the BeagleBoard control signal.

In Figure 8 the yellow arrow is pointing to the control switch. The left joystick ratchets in the vertical directions. This joystick is used for throttle control of the main drive motors. The horizontal motion of the left joysticks provides a strafing thrust from the pontoon mounted side thrusters. The final degree of freedom is controlled by the right joysticks horizontal motion. This signal will rotate the boat in place using the side mount thrusters.

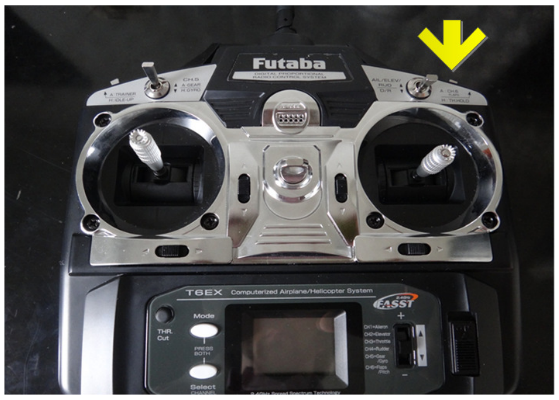
Figure 8. R/C Transmitter

VI. Communication

As can be seen in Figure 9 there are two microcontrollers, a BeagleBoard, the RC control receiver, and the interfacing circuitry that all must communicate. This section will describe the communication protocols. Figure 9 shows the protocols used to facilitate the communication between the different boards.

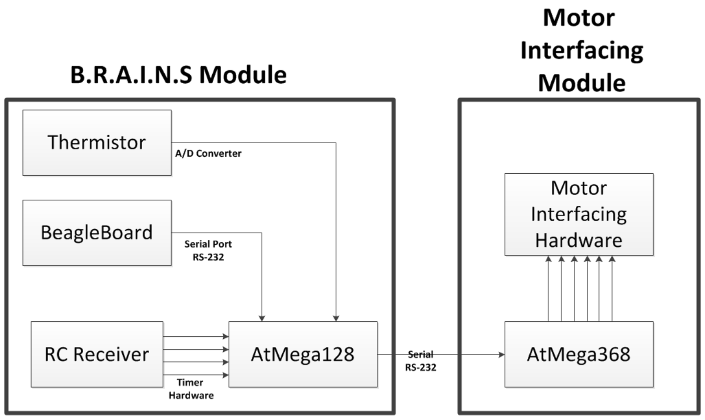

Figure 9. Communication Overview

The output of the RC receiver is a 0-5v 50 Hz servo control signal. The length of the pulse corresponds to how far the RC lever is pulled. The AtMega128 records the length of the pulses and decodes them into a signal to be sent to the Atmega368. This value is temporarily saved.

In the case of autonomous mode, the AtMega128 receives the decoded motor packet over the serial port from the BeagleBoard. This value is also temporarily saved.

A decoded motor packet consists of nine bytes. The first three bytes are *** which indicates to the AtMega368 that new motor instructions are incoming. The next six bytes correspond to each of the six motors. The first seven bits of each byte represent a pulse width modulated (PWM) value from 0-99%. The last bit corresponds to the directionality of the motor. This coding scheme allows for a minimal amount of data to be transmitted.

When the AtMega368 receives a new motor packet it first checks to see if any motor has changed direction. If a motor has indeed changed direction, the motor is turned off for 250ms. This is to prevent large current spikes from the inductive motors damaging the motor circuitry. After a directionality check has been performed the PWM for each motor is updated.

VII. BeagleBoard & Operating System

All the image processing and path planning code is executed on a BeagleBoard-XM shown in Figure 10. This runs an embedded Linux distribution stored on a SD card with the executions taking place on a TI ARM Cortex–A8 processor. It shipped with a distribution called Angstrom and many software packages and kernel modules. Many of these modules were not needed. To allow for a faster boot time and less overhead on computations, a new image was generated using the Narcissus online image builder [2]. Other distributions were benchmarked, including Ubuntu 10.10 for ARM. However, a comparison of the BogoMIPS reveals the fastest OS to be a streamlined Angstrom. BogoMIPS is an estimated measurement of instructions executed per second.

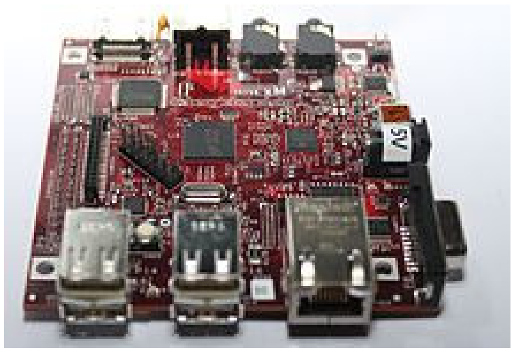
Figure 10. BeagleBoard-XM

After this new image was installed, the serial port on the BeagleBoard was configured to allow for programming without a monitor. Since the operating system image was completely minimized, no kernel modules were installed by default. The only modules that were subsequently installed were a wireless, webcam, and serial-usb driver.

As explained in the next section of this paper, our vision system is implemented using the open source image processing library OpenCV. This library has already been ported for an ARM processor. The Angstrom package manager was used to install and configure the library. As a benchmark, the library was natively compiled on the BeagleBoard but no significant improvement to the frames per second was noticed.

The final task in readying the BeagleBoard was creating a MakeFile for multi-file compiling. Since the BeagleBoard is an embedded system it is not practical to install an IDE to facilitate managing multi file projects. This is done through the use of MakeFiles. Essentially the make utility takes a MakeFile which explains all the interdependencies, links them, compiles them into objects, and then creates an executable.

VIII. High Level Program Overview

The high level control code for the autonomous system has been written in object oriented C++. Using this method, the project is able to use classes, structures, and vectors to define the information in the autonomous system. This allows high level software to be written for an embedded system.

Ultimately, the control scheme is based around a vision system. The code reads in an image from a web camera that is mounted to the top of the motor mount bar on the stern of the boat. The system uses only one camera which reduces complexity and processor computation time. Therefore, the system does not rely on stereoscopic vision, but rather uses the fact that the boat will be facing the buoy channel entrance and the reasonable assumption that the channel is mostly straight. With these assumptions in mind, the code can begin to process the images using a set of libraries known as OpenCV. These libraries allow the use of high level functions to perform image processing with an optimized implementation.

The first step in this process is to determine a horizon level. This would ideally be the location where the edge of the pond meets the ground and sky in the image. However, for the sake of simplicity and reduced computation, it is assumed that the horizon is fixed and roughly one third of the ways down the image. With this information, the algorithm determines a region of interest (ROI) to be the bottom two thirds of the image and performs all further processing on that part of the image. This both reduces the computation time on the image, and reduces the false buoys that can be found by objects that are not in the pond or pool. After this step has been performed, the algorithm can search for the red, green, and yellow buoys.

IX. Buoy Detection

After the code has determined the ROI in an image from the camera, it is stored into an OpenCV data type called an IplImage. This image is then converted from the RGB (Red, Green, and Blue) color space into the HSV (Hue, Saturation, Value) color space. This step is performed because the autonomous system is expected to perform under various lighting conditions. When the buoys are under different light, the RGB values can range wildly while the HSV values change only slightly. The saturation is the most likely to change as the buoy is a reflective surface and direct sunlight will almost completely flood a portion of the buoy in white color.

With this knowledge, a set of constant threshold values can be set for each of the three color channels. This is repeated for each color of buoy that the program needs to detect. Using these bounds, the areas that contain the buoy colors can be determined using an optimized search algorithm. This process creates a second binary image by threshholding the pixels that fall into the defined color regions and setting them to 255, or the color white. This leaves the buoys clearly defined in the image, but the locations and sizes still needs to be determined relative to the image.

To determine this information, another C++ library is used that is optimized to quickly search through an image and determine ‘blobs’ of color in that image. The library is called cvblobslib. The functions operate by assuming a blob is an area of connected colored pixels given certain known second order statistics and threshold values. CBlobResult, the function that returns the blob information, requires a binary or grayscale input image. To put it simply, the function searches through the image and locates areas that contain pixels that are of a value higher than the provided threshold. It then looks at the surrounding pixels and change in color level from pixel to pixel to determine the bounds of the ‘blob’ of color. After locating all the blobs the function returns a vector of blob objects.

The blob object then contains several parameters that provide information about that blob. The most important to the code is the min/max x and y locations of the blob. From this information the center and radius of the buoy can be estimated in terms of pixels. The buoys are then stored into a vector, where they are sorted from the bottom-most buoy of that color in the image to the highest. The driver for the camera that is used returns the xy axis in reference to the top left most pixel as the origin, and the positive y values count down as in Figure 11.

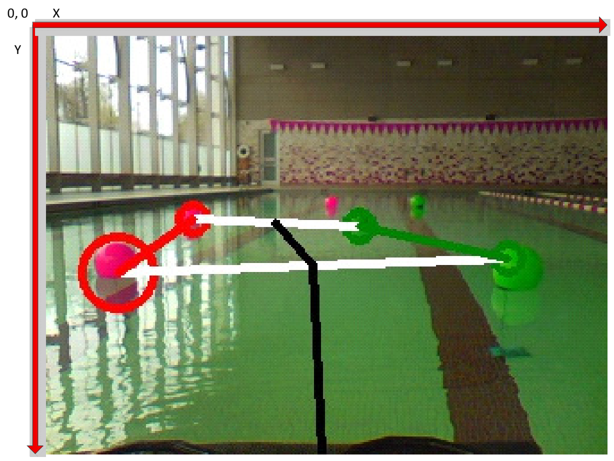

Figure 11. Sample Buoy Detection

Also in Figure 11, the vectors of red and green buoys have been drawn on the image using OpenCV drawing functions. This function uses the location of each buoy as well as the determined radius. From this image it can be seen that the algorithm sometimes considers the reflection on the water to be part of the buoy blob and occasionally only sees the bright portion. Both of these cases are ok as the relative location is still usable and considering the speed of the system it is not important for it to be extremely accurate. The figure also shows the gates (white) and path (black) which will be described below.

X. Channel Navigation

Once the buoy color and location is known, then the red and green buoys can be paired into 'gates' shown as white lines in Figure 11. Using the gate information a desired path can be determined. The algorithm is designed to move from one gate to the next, moving through the center of each one. With all of the information that is available, an overall desired path can be determined, and is shown by the black line in Figure 11. The most important section of the path is the first one at the bottom and is the only one used in the PI controller.

This section is determined by imposing a line from the bottom center of the image to the middle of the first gate. This method relies on the fact that the camera is centered and that the bottom of the image can be assumed to be level with the front of the boat. These assumptions allow the line to be the path that the boat should follow to make it through the immediate goal. Then using calculated slope and length of the line a set of control signals can be constructed to correct the current motion of the boat to match the desired path. The algorithm uses a proportional controller based on the length of the path to determine nominal speed. It then uses the slope as an input to a proportional and integrator control after being processed by a moving average filter. This method eliminates erroneous images and allows for a fast response with a medium amount of overshoot. The current algorithm uses both the side thrusters in a pivotal motion as in and the differential drive of the main thrust to turn the boat as it moves through the water as in Figure 12 and Figure 13.

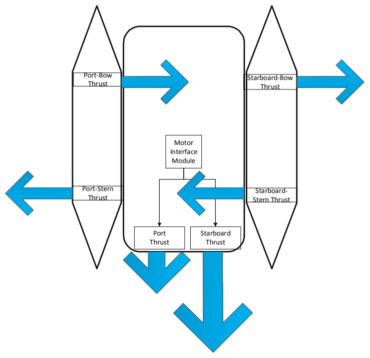

Figure 12. Left Turn

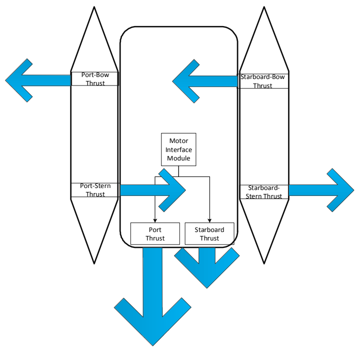

Figure 13. Right Turn

XI. Obstacle Avoidance

Scattered through the buoy channel there are a few yellow buoys which are considered obstacles by the program. The program first identifies the yellow buoys by the color as described above, then determines a threat level based on its vertical positioning in the image. If there is a yellow buoy that is closer to the boat then the first gate, it moves into the extremely hazardous mode in which the yellow buoy must be avoided at all costs. To achieve this, a line from the bottom of the image to the yellow buoy is determined and using the slope of this line as the signal to be controlled an avoidance algorithm takes over. The control scheme can be considered an avoidance gradient where the peaks of avoidance are the center of the image and the lower edges. This gradient can be described as two valleys on either side of the center. It then becomes desirable to keep the yellow buoy in these valleys. The slope of the yellow buoy line will then determine how hard the boat needs to turn. If the slope indicates that the buoy is far enough to the side of the boat, but not so far that the boat may be veering off course, then the boat can remain traveling in the direction it is heading. If the slope indicates that the boat is going to hit the buoy, or that it is moving too far off course, then the control algorithm needs to correct for this discrepancy. Once again, a proportional control is used to accomplish this task.

The main concern with this method becomes remaining on course. If the boat attempts to turn too hard away from the obstacle it may lose sight of the yellow buoy, the next gate, or all of the buoys. If it does lose sight of all of the buoys, a search algorithm will take control and the boat will rotate in place to search for a new gate to head to. The program will know if it is the correct gate to head through to advance through the channel if it follows the three R's mantra. If the only gate that can be found is in the wrong direction, it can at least travel to it and try to start over.

XII. Results

Through this report, the Nautical Autonomous System with Task Integration has been described in great detail. The system overview introduces the system, giving a quick description of each sub system and how they interact. The paper then describes the BeagleBoard and operating system set up which flows into a description of the motor interfacing module. Then the communication protocol between the systems is explained, followed by the remote control scheme and finally summed together with the image processing work.

There were problems, as there should be in any engineering endeavor. Some of which have been solved, others remain for future work. The power electronics circuits were in a confined box, and the components are very close to each other, this resulted in shorts and component failures. These have been mitigated by regulatory circuitry and isolation materials. The motors produced large voltage spikes to due to reverse EMF. This was solved by capacitors and transorbs on the supply rails. Also, the boat floated but not perfectly. The original bonding material on the pontoons was damaged by the salt water in the pool and no longer held the pontoons together. This has also been corrected as well. Aside from all of the hardware issues, there have been software problems as well.

The main problem that arises with the image processing is the field of view of the camera. When the boat is attempting to track the desired path generated by the image, it often overshoots from the inertia of the boat combined with the overshoot introduced by the control system. When this happens, one of the gate buoys may be pushed out of the field of view and the boat then does not properly generate the desired path and the navigational scheme fails. To correct this, a future team could consider adding a second camera and stitching the images together to form a wide view of the area in front of the boat. Another option would be to mount the camera to a servo, or attach a lens to the front of the camera that would widen the single camera's field of view.

Ultimately, project NASTI was able to overcome most of these barriers. Thus culminating in an autonomous system that is able to use the image processing techniques described through this paper to provide control signals to a functional electronic interfacing system and navigate a channel of red and green buoys as well as identify and avoid the yellow obstacle buoys.

Works Consulted

[1] "The Four Elements" 4th International RoboBoat Competition Final Rules. Arlington, VA: AUVSIfoundation. PDF.

[2] "Narcissus." - *Online Image Builder for the Angstrom Distribution*. Web. 04 June 2012. <http://www.angstrom-distribution.org/builder/>.

[3] Bradski, Gary Rost., and Adrian Kaehler. Learning OpenCV. Beijing: O'Reilly, 2008. Print.