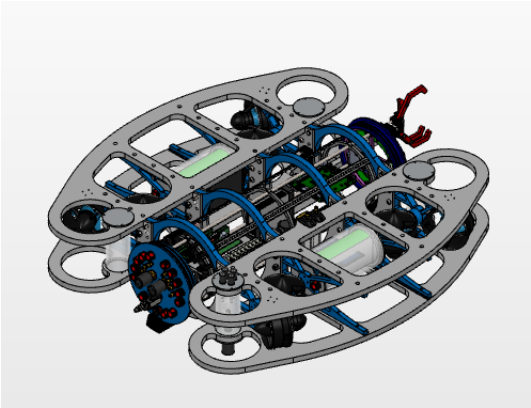
Danger 'Zona^2 Design Overview

Dr. Urs Utzinger, *Advisor*; Shirley Govea, *President, Mechanical Lead*; Jonathan Mitchell, *Treasurer, Software*; Kevin Forbes, *Software Lead*; Caros Lagos Pino, *Software Temporary Member*; Joseph Ornellas, *Mechanical Co-Lead*; (Barry) Bunwan, *Electrical Lead, Public Relations*; Barry Bunwan, *Electrical*; Nicholas Carter, *Electrical*; Erica Rao, *Mechanical*; Chris Canfield, *Mechanical*; Danielle Racelis, *Mechanical*.

***Abstract*—Our autonomous underwater vehicle maintained several of the practical systems of our previous years robot making improvements where necessary. The main electrical housing compartment was modified to allow easier access to implement adjustments as well as quicker maintenance of parts. Removing the housing for the doppler velocity log permitted reduction of the weight and height profile to the vehicle.**



I. Introduction

For the 2017 RoboSub challenge, *20 Years Under the Sea*, the Autonomous Underwater Vehicle team at the University of Arizona (AUVUA) presents Danger 'Zona 2 (DZ2), an upgrade from its 2015 predecessor, Danger 'Zona (DZ). The new vehicle contains similar electronics, but with a redesigned frame, updated primary hull, and a software overhaul. The vehicle has been designed for easier access to the electronics by allowing the primary hull to be detached from the frame via four quick-release clamps. The system voltage has been increased for higher thrust while reusing DZ's BlueRobotics T100 and T200 thrusters. The software has migrated from a custom Java solution to Robot Operating System, which abtracts away complex interprocess communication and provides a variety of open-source packages for stereo vision, mapping, logging, simulation, and more. The overhaul of DZ will provide the team a robust testbed for research over the upcoming year.

M. Shell is with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332 USA e-mail: (see http://www.michaelshell.org/contact.html).

J. Doe and J. Doe are with Anonymous University.

Manuscript received April 19, 2005; revised January 11, 2007.

II. Mechanical Design

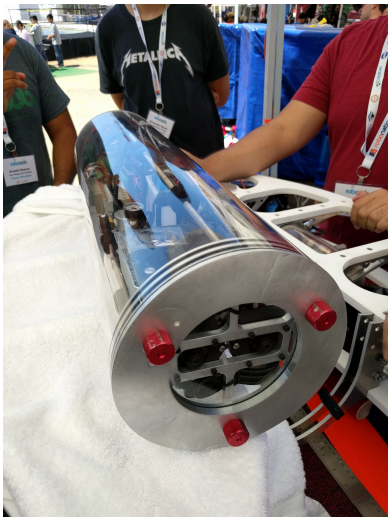
This year DZ2's mechanical design is the same design used in the previuos year. The team learned from the last two vehicles that HDPE is easily machined, neutrally buoyant, and cheap. However, it tends to warp when thin and long, so the team took material thickness and geometry into heavier consideration when creating the design. Cast acrylic is used for the tube which allows a better visibility than what previous materials provided.

A. Frame

The frame design focuses on accessibility; eight cotterpins and three bolts are the only barrier to entry of the electronics, which can be accessed away from the frame. Frame improvements include retainers to prevent the tube from sliding in its mounts during high power maneuvers.

The frame is constructed with two materials: HDPE for the horizontal 'wings' and aluminum 6061 for the vertical supports, clamps, and endcaps. HDPE (a slippery plastic used in cutting boards) was relied upon heavily for DZ and simplifies the design by having a neutrally buoyant density and being easily machinable. Aluminum is used for the vertical supports and clamps to minimize the footprint of the vehicle while maintaining rigidity along joints.

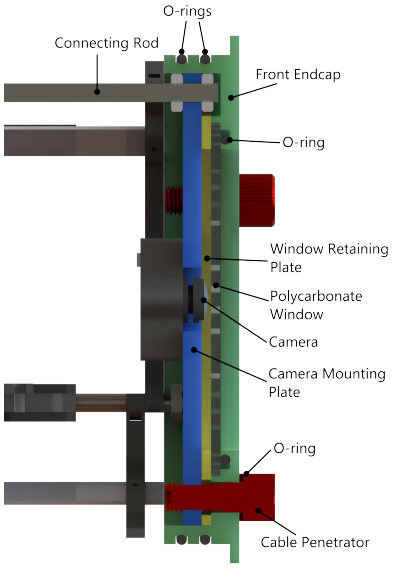
B. Housings



Electrical Housing

1) Primary Hull: The main hull is composed of a 0.25" wall, 7" OD cast acrylic tube housing three connecting rods for mounting electronics and pneumatics. The tube is mounted to the vehicle with four removable top clamps for easy access, each with a strip of neoprene to constrain the tube when in use. Felt attachments on the connecting rods ensure a smooth positioning of the tube, a feature sorely lacking in the previous vehicle that caused excessive amounts of scratching and impeding vision. Both endcaps use double o-ring bore seals.

The hull is secured with three sealed bolts at the front endcap. The bolts thread into a plate which is rigidly attached to the back endcap via three threaded connecting rods. This allows the hull to be closed without external rods or clamps.



Front Endcap

The rear endcap contains all bulkhead connections for the primary hull, including eight cable penetrators for thrusters, six auxiliary penetrators, eight push-to-connect fittings for pneumatics, two large SubConn connectors for the ethernet tether and hydrophone assembly, and charging and venting ports for the pneumatic system. A bracket spacing the connecting rods is mounted on the end and fastens into the endcap, allowing the main electronics to disconnect easily from the frame and its fixed electronics.

The front endcap contains a thin polycarbonate window with an o-ring face seal. A retaining plate ensures a robust connection. The camera mount performs the same function for the front endcap as the rear spacing bracket, while also providing threaded connections for the sealing bolts.

A thick polycarbonate sheet is used to support all electronics and pneumatics in the hull, with the exception of the LEDs. Mounting of the electronics is accomplished with 3D printed brackets specially designed for each component. This allows for easy positioning of tightly integrated parts, and the rapid prototyping means adjustments to the design are less cumbersome. The LEDs are mounted along rails attached to 3D-printed rings.

2) Battery Hulls: The battery housings are mounted on the sides of the vehicle. Cutouts on the longer aluminum stabilizers are used to captivate the tube while clevis pins retain their position, making for quick re-installation. Each compartment houses one battery, and a cable penetrator supplies power through the main bulkhead.

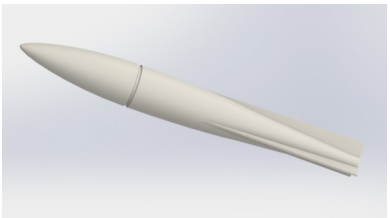
C. Actuation

This year the team made a push to include actuation beyond movement for the first time. Pneumatics are fully plumbed, and the torpedoes and markers were designed pragmatically to allow for greater margins of error leading up to the competition.

1) Thrusters: DZ2 uses four Blue Robotics T100s and four T200 thrusters for vertical and translational movements, respectively. The T200 thrusters are mounted in a vectored arrangement at 30 degrees from forward for a simplified and symmetrical thrust scheme. Frame components such as the vertical supports are designed to allow unobstructed water flow through the vehicle. The thrusters are all mounted in the same plane to minimize vehicle height. This lowers the stability of the vehicle, removing control of the robot from gravity and bringing it into the realm of the software. The thrusters are also placed on the outer corners of the vehicle. These two design choices provide the opportunity to ambiguously orient the vehicle with little cost on energy to maintain the position.

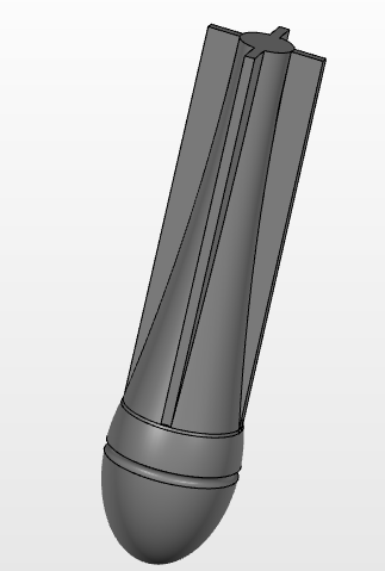
2) Torpedoes and Markers: The vehicle features pneumatically actuated torpedoes and markers for the Battle a Squid and Cultivate Pearls competition tasks. Pneumatics were chosen for their efficient use of air, simple overall design compared to other methods and low cost.

Both the markers and the torpedoes are made of ABS Plastic from a 3-D printer. The material was chosen for its neutral buoyancy in seawater and for its cost. The torpedoes are housed in dual tubes made of PVC lying atop one another on top of the robot and are launched independently. The torpedoes also utilize o-rings in their design to help secure them inside the tubes and to help increase the efficiency of the pneumatics.



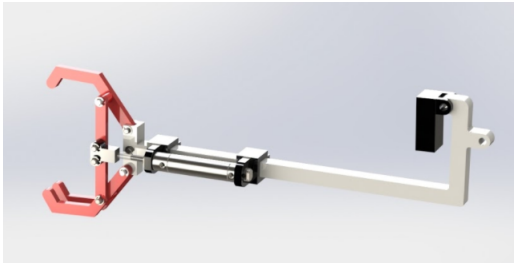
Torpedo Design

The markers are housed in two short PVC pipes vertically attached to the side of the main hull and close to the bottom-facing camera, and are launched in an identical manner to the torpedoes. Additional material is installed inside the markers to increase their density and improve their descent rate.



Marker Design

3) Claw: The claw for DZ2 is designed to be used in both the horizontal and vertical orientations with visibility from either the forward stereo or belly cameras. This is accomplished through a hinge and tilt cylinder at the very base of the claw arm and a geometry that minimizes the claw stick out in the vertical position. A pneumatic system is used to power the gripper and tilt cylinders. It utilizes HDPE plastic for the main structure and 3D printed ABS for the gripper and assorted smaller pieces.



Claw

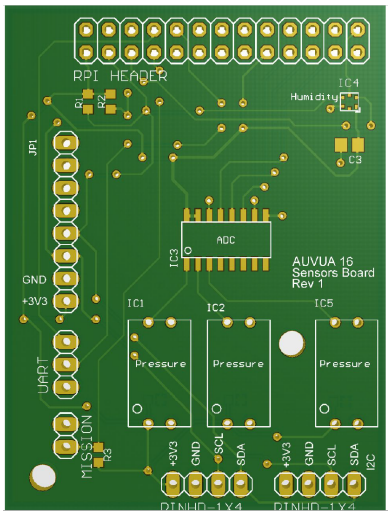
III. Electrical Design

The goal of the electrical design is to improve the modularity of electrical connections for ease of access and update the existing circuit boards. The team started this process by implementing modified, improved and revamped improvements on our existing three custom PCBs and by looking at the overall framework of the electrical design. A flowchart was created in order to clarify and organize the layout of our overhauled system.

A. Custom PCBs

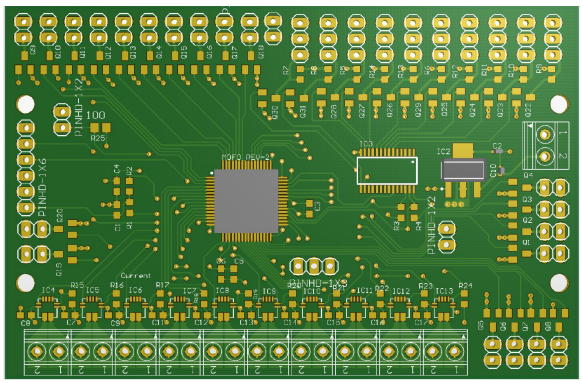
With generous donations from Altium and Advanced Circuits, the team has created five custom boards, some built off prior designs.

1) Power Distribution: A power distribution board is used to supply current from lower voltage rails and for sensitive electronics. An actuators board is responsible for controlling ESCs at a low level as well as powering pneumatic solenoids and auxiliary switches.



Atcuators Board

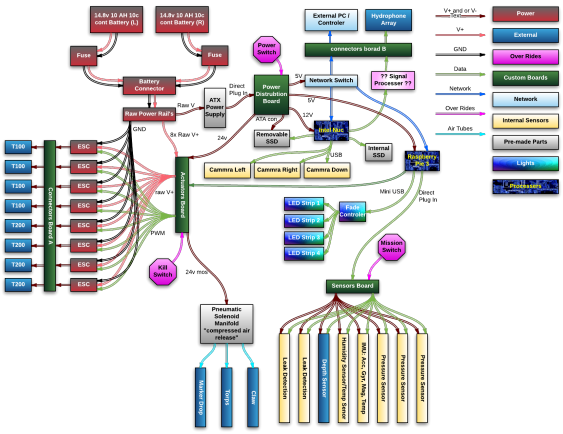
2) Sensors: A sensors board, designed as an add-on for a Raspberry Pi 2 or 3, interfaces a 9-axis IMU, depth sensor,pneumatic pressure sensors, and humidity sensor to the rest of the system.



Sensors Board

3) Quick disconnection: Quick disconnect boards were made for providing quick-disconnect functionality to all electrical interfaces in the main hull. These boards are crucial for maintenance of the electronics, which would otherwise be tethered to the rest of the vehicle.

4) Hydrophone: This year, the team is developing hydrophone board for using in pinger localization. The hydrophone board housing is at the right rear external tube. The hydrophone board design is remodelled and resize to fit in its housing.



Electrical Diagram

B. Power

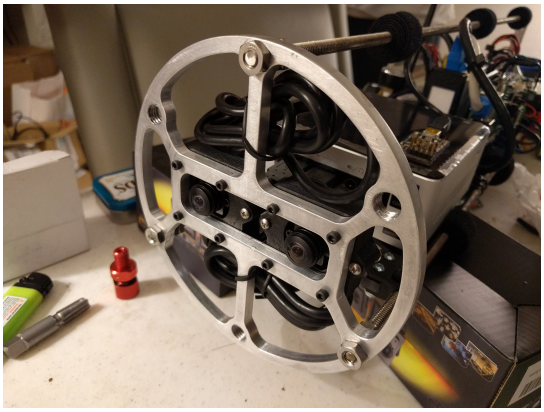
Danger 'Zona[2] has two 14.5 volt 10 Amp Power Lithium batteries, which provide vehicle runtime up to three hours. Also, power from the batteries allow the vehicle to run full speed. Additionally, the batteries were readily available and extends the runtime of the vehicle.

C. Sensors

The vehicle contains all necessary sensors for operating in an underwater environment, including an MPU-9250 9-axis inertial measurement unit and a BlueRobotics depth sensor. Additionally, a humidity sensor is used to detect critical leaks, and current sensors on various boards are used to track battery usage.

GoPro Hero 3 cameras were used for vision in the previous robot. While this significantly increased performance compared to USB cameras without exceeding the cost of COTS machine cameras, the interface was lacking and required additional HDMI capture cards and dedicated software. This year, the team opted for a simpler camera solution. Three wide-angle USB board cameras, roughly one-fifth the cost of the previous setup, are used for forward stereo vision and downward viewing. The cameras use standard interfaces for Linux, making integration with ROS straightforward.

In order to conserve space, the team opted against using a generously offered DVL from Carl Hayden Robotics. Instead, a camera-based software solution will be developed for current and future years to map the environment and localize the vehicle.



Foward Facing Cameras

IV. Software Design

A. Software layout

In terms of code the robot has two major platforms that are run on the system. First there are several embedded pic24 microprocessors that handle most of the physical control of the robot and sensor feedback. Secondly, the robot runs the core software on an Intel Nuc I7 Raspberry PI 3 both of these hardware components run Robot Operating System (ROS) allowing messages to be seamlessly sent between the two computers. The Nuc primarily deals with high load computing, this includes the agent and computer vision aspects. Whereas the PI primary handles sensor and actuator interfacing this acts as an interface layer between high level logic and heavy computing functionality on the embedded systems. The embedded systems are in charge of getting various sensor info including the inertial measurement unit (IMU) and major functionality such as the actuators board which controls the robot's thrusters, torpedoes, markers and claw on the robot.

1) Software background: In the last two years the team has been working on migrating from a custom java implementation over to an easier to work with ROS platform. The team encountered several setbacks last year in regards to migrating over to the new software platform. Due to the differences in software structure and these issues it was decided to redesign a lot of the aspects of the software. These changes are mainly in the high and mid level structures of the code which ROS is now handling to be more versatile and useable not only on the current robot but to be transportable to other robots that are built down the road.

2) Robot Operating System: At its core, ROS is a collection of libraries, binaries, and build tools for creating processing elements called nodes that can communicate via TCP sockets with little forethought. Topics are used to pass data in a publish/subscribe relationship between programs. A parameter server allows the architect to modify values between or during runs.

3) Computer Vision: For our computer vision the team is using ROSs ability to work well with openCV which was used in the previous robot with a custom java implementation. This allowed portions of the code to be transported into several

working filters and datasets that helped get a working prototype of the software up and running to make sure it would be capable of handling the tasks for the competition.

B. Agent and Mission Planner

One of the major reworks the team has done on the robot this year is a total rewrite of the Agent and Mission Planner. It has been done via implementing a two tier state machine, where the agent in effect is a control loop that runs based off the current task the robot is in. The first tier is the task level state machine which controls what task the robot is currently on or doing at a high level, or, if it is traveling to the gate or on the torpedo task. The next level is the SubTask which handles what the robot does inside for the task, what image does it look for, how does it move, does it grab something with the claw, etc

1) Task level state machine: The task level state machine holds a list of tasks which the robot can encounter. These are tasks like searching default, torpedoes, markers, or maneuvering tasks. Each of these tasks have subtasks inside of them that are required to be finished prior to being able to encounter a new task. In earlier versions of this code, the robot would be running a linear set of states, i.e. do gate, then hit the next task, etc. This design choice was made mainly due to the need to be able to effectively test the new ROS code out and make sure the software components are interacting in a predictable manner. When the software is at a mature state, the team is planning on adding a decision process based off either decision trees or markov chains which will allow for the robot to make more decisions regarding the best way to achieve the maximum points.

2) SubTask level state machine: In regards to the mission planner, this is mainly integrated into the sub tasks. The task level controls what task its on and which is next. The SubTasks control how it does each sub task. This include items such as: what is the robot looking for, how should the robot move, etc. In short, these subtasks contain the instructions that make the robot autonomous. This is achieved by looking at what the current task of the robot is in and then what does it do to interface with the environment. For example, if the robot is on the torpedo task, the robot will need to look for the target via openCV once it recognizes the base image. It then goes into a control loop to set up the first shot by checking the image against a reference image and figuring out the required movement in order for the robot to get into the ideal location to make the shot. This all can be repeated for the second shot. After this, the robot will maneuver past the task and hand control back over to the task level of the controller.

C. Custom Motion Controller

This year the team has written a custom motion controller for danger zona[2]. This is achieved through a multi-layer architecture in regards to the motion controller which was done in four levels, Embedded, PI Interface, Mid Level Controller and Agent Subtasks.

1) Embedded System level: The embedded system code on a pic24 for the actuator board sends PWM signals to the electronic speed control units (ESC). This allows control of the thrusters by sending a variable voltage to each of the thrusters determining how fast they spin, this is translated into thrust and motion for the robot. Allowing us to effectively address each thruster separately giving the team a high level of control to each thruster.

2) Interface level: On the raspberry pi there is an interface layer via I2C from the PI which has ROS installed to the actuators board, this provides the interface from ROS to the embedded hardware. Giving several advantages due to the nature of asynchronous programming throughout our design. Primarily, this acts as a data transportation layer in-regards to the motion controller.

3) Motion controller: The main purpose of this mid-level controller is to act as the main program responsible for carrying out any movement commands given to it by the robots agent. The robot handles commands in terms of six degrees of freedom over time: X-Y-Z coordinates, yaw, pitch, roll and time both in terms of absolute or relative motion. For example, move 5 feet down (Absolute motion) or move forward for 5 seconds (relative motion) this in effect gives us 13 variables the robot can work with: 6 absolute motion, 6 relative motion, and 1 time. With this, the team can build many functions and nested functions that are easy to call from the robot. For example, if the robot feeds the above movement information for X time to the PI embedded systems it allows the agent to give complex commands or queue commands to the robot.

4) Agent Subtask level: Due to the agents/mission planners structure of tasks and subtasks the robot can provide the high level movement logic, where the agent can solve for movement needed during the robot's current task. For example it could tell the robot to drive forward and X heading until it finds Y image via the computer vision which then provides feedback to the system allowing the robot to close the control loop for most tasks.

V. Conclusion

The software on this year's robot was updated from java to ROS for simplicity, a more extensive function library, and improved camera vision. A more modular, accessible, and lightweight frame was designed and manufactured by the team at our machine shop. The electronics systems are heavily based in the same platform as our previous robot with a few new custom made PCB boards. These PCBs allow for rapid integration and removal of the majority of our systems. This years robots sees actual integration of torpedoes and markers into platform. All of these changes coalesce to produce the Autonomous Underwater Vehicle of the University of Arizona for the 2017 RoboSub competition.

Acknowledgment

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.