RoboBoat 2024: VantTec Technical Design Report VTec S-III

Max Pacheco, Alexa Arreola, Abiel Fernández, Diego Govea, Eduardo Hernández,
Ernesto Urrea, Christian Villarreal, Alejandra Coeto, Mauricio Degollado,
Demian Marin, Oscar Cárdenas, Felipe García, Jocelyn Velarde, Sofia Cantú
David Soni, Juan Hernández, Edgar Mayorga, Rodrigo Monterroso,
Elisa Borjas, David Pimentel, and Herman Castañeda
Monterrey, Mexico
vanttec@servicios.tec.mx

***Abstract*—In this report, the design of VantTec's autonomous boat VTEC S-III and the overall strategy for RoboBoat's 2024 competition is described. It relies on scoring as many points as possible, while developing all the mechanisms and peripherals needed, and testing the updated solution of previous tasks first on simulation, and then on water. The engineering decisions are based on previous research works from old members, and the teachings that experiments provide.**

***Index Terms*—Unmanned surface vehicle, robotics, autonomous boat, GNC system, computer vision, artificial intelligence.**

Fig. 1. VTec S-III USV.

I. Competition Goals

The main goal for the team this year is to get back into the game, and to start competing at RoboBoat with improvement on performance by the team from last competition. Because of the amount of research and development accumulated in the vehicle's dynamics and control models, the team decided to keep using the same vessel from previous years; that is also because of VantTec's limited funding for the project, as there are now 4 other projects in the team besides this boat: a car for touring on campus, a drone for pizza delivery, RoboSub's submarine, and a 42-drone swarm for university shows.

A. Course Approach

To achieve the main goal, the boat needs to be able to perform all the tasks on the competition, and that's why the team focused on the design, development, and manufacture of mechanisms and the electronics iterations needed to solve this. On the other hand, in order to be able to use the most recent and newest technologies, the Ubuntu version of the Jetson TX2 used as the main computer was updated as well, from Ubuntu 16 to 18, which wasn't enough for ROS 2 support, so a Docker container was implemented too.

As the *Navigation Channel* is a mandatory task in the qualifying round, it was a priority to have that task solved. The good news is that the boat was able to solve that task on previous years; however, the bad news were that all of those algorithms were outdated because of the ROS 2 upgrade implemented on the computer; so, the first and easiest task to program and simulate was this one, based on the pre-existing version of the workspace. The next one was the *Speed Challenge*, as the logics from the first one were very similar, and the adaptive sliding mode controller (ASMC) directly deals with the tangent velocity as the reference to control. The difference is that the *Navigation Channel* calculates an infinite straight line that intersects the center of the starting gate and stops until it detects the new gate ahead; on the other hand, the *Speed Challenge*

stops generating new waypoints for the vehicle to follow just when it detects the yellow buoy ahead, and then it generates the waypoints needed to go around it with the best radius so that it doesn't travel too far in order to gain time, but also, not so small that it could crash with it.

Another very similar task which relates with the previous two in a sense that it's also mostly navigation and obstacle detection is *Follow the Path*. The beginning of the task is the same as the previous two, as the boat needs to go through the first gate, but it changes in the frequency that the gates are detected, so the generated waypoints for the navigation are closer than in the previous tasks, as obstacles may be closer in this one. The logics are the same as the mandatory task, except that the ZED stereo camera is also used in this one for color detection, and the LiDAR verifies every detected buoy has been color-classified. The *Return to Home* task is basically half of the *Navigation Channel* with just a waypoint iteration five meters in front of the gate. Another very similar task which relates with the previous two in a sense that it's also mostly navigation and obstacle detection is *Follow the Path*. The beginning of the task is the same as the previous two, as the boat needs to go through the first gate, but it changes in the frequency that the gates are detected, so the generated waypoints for the navigation are closer than in the previous tasks, as obstacles may be closer in this one. The logics are the same as the mandatory task, except that the ZED stereo camera is also used in this one for color detection, and the LiDAR verifies every detected buoy has been color-classified. The *Return to Home* task is basically half of the *Navigation Channel* with just a waypoint iteration five meters in front of the gate.

Moving on to *Docking*, the vehicle's approach is to stand still in front of the task until the *Perception*'s algorithms based on the camera detect the correct bay, and using the LiDAR to avoid crashing with each bay's walls, the necessary waypoints are generated to stop the vehicle on a certain coordinate. Right after that, the boat returns to the start of the task, and begins with the *Duck Wash*. The same peripherals are used for this one, except that VantTec's custom STM32-embedded PCB also enables a servomotor and a relay-activated 12V DC water pump to shoot the duck, by first performing a similar approach to the *Docking Task*, but maintaining a little bit more of distance to bay to give the camera the room to keep detecting the obstacle and also to avoid getting splashed.

For the remaining two tasks (*Collection and Delivery Octagons*, the approach was to use the same system to both collect and deliver the items, while minimizing its effects on the dynamic properties of the boat. An arm mechanism was designed to contain and transport the items on these tasks, and the tasks' logic was to first identify the *Delivery Octagon* to deliver the preloaded racquetballs, by identifying with the ZED camera the *Beaver Nest*, generating the waypoints to travel right next to the octagon's border, and then opening the servomotor holding the doors of the closed mechanism (detailed in the *Design Strategy*) to open in order for them to drop in its designated area. Next, it travels to the *Collection Octagon* to pick up the other racquetballs, using another perception algorithm to first get to the octagon guiding itself from the black triangles of the task, and then detecting the red balls floating on the water, picking them up, travelling back to the other octagon to repeat the delivery, and finally, doing the same with the rubber ducks.

The strategic vision of the team is to solve the most tasks in the least amount of time, while saving as much power as possible. The first part of it can be achieved by aiming towards robust systems, which come with a lot of testing and design planning, which has been implemented and worked on the team, and the latter one is mainly because the boat has to power the Jetson computer, the IMU, the LiDAR, the camera, the thrusters, the STM32 PCB, a NEMA motor driver for lifting the Octagon Tasks' arm, two servomotors, and a water pump. That's why since the *Task Ideas* document from the beginning of the season came out, the team's efforts have been focused on developing the mechanisms for this years' new tasks: the arm mechanism and the water blaster. As new members were incorporated into the group this semester, and after a short period of learning, the team started working in parallel on the migration of the workspace to ROS 2, the design and manufacture of the mechanisms, the training and programming of the perception algorithms for labeling and object detection, and the

electronic iterations needed for the correct powering and functioning of the mechanisms' peripherals. Careful planning beforehand came useful, as each sub-team had to work independently, but knew the requirements that it had to deliver, for the other areas to work properly and have an easy integration process. This gives the team a chance to have reliable solutions for the newest tasks, and the time to work on the verification of the previous ones. Of course, developing these new mechanisms for the competition while also upgrading the workspace version was a high risk, as it basically meant that the team had no real compatible solutions for any of the tasks, but doing so didn't really take much of a toll, since other projects (vehicles) on VantTec had implemented ROS 2 for their current solutions, and those worked as a template, along the official tutorials, for the new workspace; also, that upgrade meant better compatibility for current software tools like for the perception algorithms and for communication, as last year, a Python TCP connection was required between scripts to go over the Xbee's Python3-only support limitations. Taking this into consideration, and counting on the time the team has to prepare and validate the pending challenges and objectives, the team decided to carry on with the enhancements in software, as the controllers and dynamic models' mathematical equations remain the same and no further research is needed for its implementation on ROS 2.

II. Design Strategy

The same vessel from previous years was used on this competition as well, for three reasons: first, because of the limited members that were part of the project at the beginning of the semester; second, because of the team's limited funding towards the project on the development of a new boat; and finally, because the limited time that the team had to prepare for this year's competition would not be enough to perform the experiments for the right modelling of a new boat and its controller gains' syntonization would not deliver a system as robust as the one there currently exists.

A. Software

All the missions take advantage of the Object-Oriented Programming that come with the languages used in the ROS system, so a main change in the missions for this year was also the use of one class for the general-purpose vehicle's performance, using inherited children classes just for the small changes between missions, and a segmentation based on the general functionalities of each one, so checking out the needs for each task, it was noted that every one (on the team's solution) needed the necessary control nodes for navigation, and the perception nodes from the LiDAR either for obstacle detection, or for aiding the camera, whose attributes were part of the parent class. The vehicle's design, then compliments the main navigation requirements for these tasks' solutions, by making use of the tools provided by the software's features, resulting in a simple but robust system, as it makes an efficient use on the limited resources from the computer. As mentioned before, the software migrated to ROS 2, so the memory from the Jetson was even more limited, which meant that everything had to remain as efficient as possible.

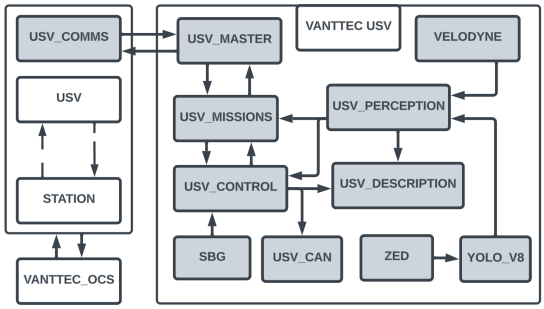

Fig. 2. VTec S-III's 2024 Software Architecture.

Fig. 3. VTec S-III USV on the new Gazebo Sim

Moving on, the Gazebo sim was also updated, and now integrated into the main workspace, against

previous years, when there was a repository for all the vehicle's onboard packages, and another one just for simulation. This makes possible an easier cloning process of the repository, and reduces problems that could be caused by wrong management on version control, since it is all handled just in one place. This enhances the simulation process, which causes faster development and validation on the mission solutions simulation and perception programming and testing for the obstacle detection and avoidance algorithms.

There were also very important changes in the controller, just for the pivoting of the vehicle, needed for the *Docking* challenge when getting out of the bay, for the *Duck Wash* challenges for angle correction on the water-shooting orientation, and for the *Follow the Path* task's ending. This involved a small tweaking in the controller by making an exception for pivoting, since the controller doesn't naturally tell any thruster to go backwards; which leads to another tweak in the controller for backwards movement. This was only implemented as a measure of precaution for the *Octagon* challenges, in order to maneuver the boat correctly to combine this backwards movement with the pivoting to place the ball collector's container right on top of each item.

The ROS 2 migration involved a change in the software architecture, as now the *usv avoidance* package has been integrated with the *usv control*, and also the *description* package was appended, which is useful for simulations or for viewing data in general.

B. Ball collector

VantTec's main goal for this competition is to be able to solve all the tasks, so a mechanism for the *Collection and Delivery Octagons*' solution was needed.

The proposed design is a NEMA 23 stepper motor-controlled arm that pivots inside of the inferior hull of the boat and is made out of PVC to reduce expenses and weight. It extends to both sides of the hull, so that the containing mechanism of the ball collector is better balanced. The arm-lifting mechanism uses a gear reduction with a belt for higher torque, and a driver for toggling its functionality (for power saving), controlling its direction, and its steps. It is necessary to correctly control angle of the arm, because in the end of it, there's a servomotor that opens and closes some doors that make possible the collection and drop of the tasks' items. To achieve the control of the arm, an MPU9250 is integrated right inside of it via I2C, which means that the accelerometer's inclination is the same as the arm's; this way, it doesn't matter which position the arm begins in, it is always possible to know and control its inclination, despite the position of the boat due to pitch movement caused by the waves.

There's a laser cut acrylic container with 3D printed parts sub-mechanism in the end of the arm, which has doors on the bottom, that are used for grabbing and dropping the items while open, and transporting them while closed. The doors have a gear arrangement that makes it so that they open and close symmetrically (for better control on dropping accuracy), and it is designed in a way that the superior part of it is exposed but because of its walls and some ball bearings which keep the whole mechanism suspended with the doors always facing the ground plane, the items will not fall while being handled.

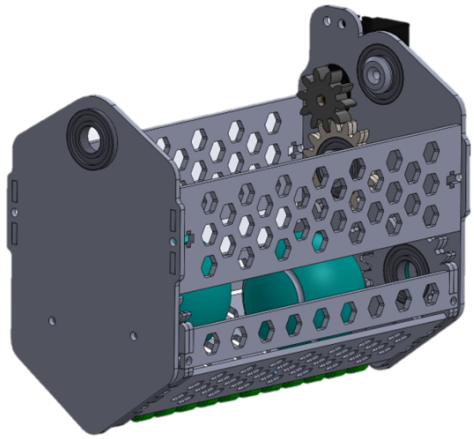

Fig. 4. Ball Collector's container mechanism.

The arm has 6 states: the first one is the *Rest* or *Idle* state, in which the mechanism rests on top of a platform on the back of the boat without being enabled, and isn't consuming power to avoid using the battery when not needed. The second one is the *Grabbing-Up* state, in which the doors from the container are open and the arm is on a 10° angle

(a 0° angle means a completely forward position) waiting for the boat to maneuver into a position that sets the doors right above the item to grab. The next state is the *Grabbing-Down* state, in which the doors remain open but the arm lowers to a -15° angle, which is the most suited for the container to lower and close below sea level, but not so low that it damages the servomotor, which could be prone to becoming damaged if getting wet. Next comes the *Grabbing-Close* state, in which the doors close while still at a -15° angle; then comes the *Grabbed-Item* state, in which the arm raises to 45° and the MPU makes sure that angle is kept, while the doors remain closed and the item is now being transported to the delivery area; finally, here comes the *Drop* state, in which the angle remains at 45°, and the doors open. After successfully dropping the item in the correct place, the ball collector can go back to idle.

These states are being controlled by the mission nodes, and modified in a topic that sends a message to the STM32 PCB via CAN.

Another urgent modification on the platform was to find a way to shoot water for the *Duck Wash*. The easiest solution was to buy a 12V DC water pump, which conveniently could be directly powered, because the two type of Li-Po batteries used in the boat are: a 14.8 V, and an 11.1 V battery, so using the 12 V battery was enough, as the water pump used 60 W of power (5 A), and the 12 V Zippy battery provides up to 8 A. However, this was not all of it; to enable toggling of the pump, a relay was also needed, so that water blasting could be controlled from an output pin directly from the STM32 PCB. A nozzle for better range was also designed and 3D printed, which after some calculations and testing, the best performance came with a 1.2 mm diameter cylindrical hole for the hose adapter, reaching distances of approximately 7 to 10 meters. Although the vessel is completely capable of maneuvering and pivoting thanks to the changes implemented on the controller, a servomotor was appended on the base of the shooting platform from the inferior hull. This servo has a mount on top of it, which maintains the end of the hose looking up in a fixed pitch angle, which can be modified by loosening and tightening some screws, and has the ability to yaw 90° on each direction, but is used in the blaster just for the smaller-angle corrections. With this particular device, safety precautions were taken, as to reduce the possibility of an accident like water leakage leading to electronic failures; that's why, it was decided to mount it on the back side of the superior hull of the vessel, so that it can have direct access to a water source below it, but it's also high enough for its electronics to get wet, but it still doesn't get in the way of the LiDAR's field of view, and even if there's a leakage, everything will just spill outside of the boat, protecting the electronics.

C. Electronics and Embedded Systems

To make all the needed peripherals useful, a reprogramming on the STM32 board was needed, which involved making use of one of the main reasons this was designed and manufactured: modularity. This helped the team minimize the amount of wiring needed in comparison to using general development boards, like the Arduino from previous years, for thruster control and general devices, which could also cause more accidents involving incorrect wiring, or even unsoldering or disconnections caused by the vehicle's and boards' movement causing it to loosen and making a false connection.

In the end, however, not everything was solved with it, as two buck converters were needed for voltage regulation, both for a 5 V output: one to supply the STM32's 5 V track for servo motors and a level shifter with 5 V for the control of the NEMA motor driver, and the other one to supply the relay for the pump's toggling, which also went through a MOSFET to supply the relay's coil with the necessary current. Each buck converter was mounted on a phenolic circuit board with screw terminal connectors for fast and easy connections, and they got powered from the Power Distribution PCB, which previously had only two fuses: one for each thruster; but since the new boards didn't need much current, new fuses were added for more supply channels, and the necessary connections were soldered, also using this board's modularity. Even though phenolic boards were needed because of voltage support that the STM32 PCB couldn't provide, the amount of wiring connections needed was still not that much, compared to what could have been.

D. Object Detection

Perceiving and sensing the environment surrounding the boat is not an easy task. That's why, in this iteration of the vehicle, our efforts were focused in improving already existing systems and adapt them to new requirements. For example, this is the first iteration that uses YOLO-V8 as the backbone for the buoy detection systems. Moreover, the neural networks were adapted and optimized to take full advantage of the computing resources; this involved utilizing certain conversion scripts and code snippets following a certain path to achieve certain optimizations, as described on the following figure.

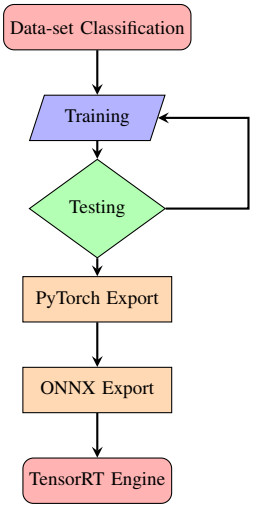

Fig. 5. Process followed to obtain an optimized neural network for buoy detection.

Some "classical" computer vision techniques were also employed in order to detect other things as necessary. The use of color masks, contour functions and shape approximations to identify from RGB frames, certain shapes with specific colors needed to be identified. Taking into account differences among shapes and specifying this differences in-code to later identify them with the algorithms already mentioned.

Finally, the well known VLP-16 LiDAR from Velodyne was employed to ensure no blind spot goes missing, this is required so that image processing is on the front but if we pass through a target, we have backup sensors capable of identifying things like buoys without pivoting on more computational heavy camera-dependant algorithms.

III. Testing Strategy

The testing strategy to reach VantTec's goals for the season, were to first know what the vessel could do in the beginning, which was thruster control via CAN, a trajectory-following validation via simulation and some previous testings. To implement the new explored solutions, in the mechanics area, several designs and prototypes were done both for the Water Blaster's nozzle, and for the Ball Collector's container as well; as for the electronics area, each mechanism's electronics were first individually validated, and then testing their integration into the whole system, being finally able to perform all of them on the vessel, with the same electronic conditions (power supplies, isolation, and complete integration) it would have on the competition. For the software missions and perception algorithms, the testing is done first via simulation, taking advantage of the dynamic model that the team has, to test control paths and the vehicle's performance. Then, each task is tested independently in real life, in the same order that they were presented in the *Course Approach*, and finally, when all of them are validated, a complete autonomous competition-like course run is performed to evaluate what the outcome would look like in the competition. The strategy on-site is to spend the first minutes testing the tasks that the vehicle performs the best at, and then working its way through to the ones with the highest rewards. The objective is to try and solve them all to have the final bonus as well.

IV. Conclusions

A new strategy for the RoboBoat 2024 competition is detailed. It shows the vehicle's main goal of attempting to solve all of the tasks, for which mechanisms were designed and manufactured to score as most points as possible. It also describes the main changes performed in the software and perception areas to contribute to this primary objective, which come as a consequence of the upgrade to ROS 2. Results from testing show that the solution provided for the mechanisms is adequate and can be implemented. Further experiments before the competition are needed to verify the Testing Strategy.

ACKNOWLEDGMENTS

This work was supported by: TechMake, RoboNation, Velodyne LiDAR, and NVIDIA. Finally, VantTec appreciates the support from the university, Tecnológico de Monterrey, and to our advisors Dr. Herman Castañeda and Dr. Alberto Muñoz.

REFERENCES

[1] A. Gonzalez, et al., "VantTec ts and Systems (IROS), pp. 2577-2582, 7-12 Oct. 2020.

[2] Musa, M.M.M., Scherer, S.A., Voss, M., et al., "UUV Simulator: A Gazebo-based package for underwater intervention and multi-robot simulation", IEEE OCEANS, 2016.

[3] Lovro, M., Velodyne Simulator, Github repository, 2021. Available: https://github.com/lmark1/velodyne_simulator

[4] Guilherme C., Librealsense, GitHub repository, 2016. Available: https://github.com/guiccbr/librealsense

[5] A. Gonzalez-Garcia and H. Castañeda, "Guidance and Control Based on Adaptive Sliding Mode Strategy for a USV Subject to Uncertainties," in IEEE Journal of Oceanic Engineering, doi: 10.1109/JOE.2021.3059210. https://github.com/morriswmz/doa-tools

[6] "STM32F405XX Datasheet " www.stm.com [Online]. Available: https://www.st.com/resource/en/datasheet/stm32f405rg.pdf. [Accessed: 05/23/2021].

[7] "What is An RTOS? " www.freertos.org [Online]. Available: https://www.freertos.org/about-RTOS.html [Accessed: 05/23/2021].

[8] Robert, B., "CAN Specification, Version 2.0" [Online], 1991. Available: http://esd.cs.ucr.edu/webres/can20.pdf. [Accessed: 23-May-2021].

[9] A. Gonzalez-Garcia and H. Castañeda, "A Real-Time NMPC Guidance Law and Robust Control for an Autonomous Surface Vehicle" in IFAC Conference on Control Application in Marine Systems Robotics, and Vehicles CAMS 2021, doi: 10.1016/j.ifacol.2021.10.101.

[10] Salais R., De la Garza I., Martinez S., et al, "RoboBoat 2022: VantTec Technical Design Report VTec S-III", Available: https://www.overleaf.com/read/hmbnwmkbhjzx

[11] Tutorials — ROS 2 Documentation: Humble Documentation. (s. f.). https://docs.ros.org/en/humble/Tutorials.html

APPENDIX A: COMPONENT SPECIFICATIONS

Component	Vendor	Model/Type	Specs	Custom/Purchased	Cost USD	Year of Purchase
ASV Hull Form/Plataform	VantTec	VTec S-III	Fiberglass	Custom	NC	2018
Propulsion	Blue Robotics	T200	https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster-r2-rp/	Purchased (2)	$36	2018
Power System	Blue Robotics	Lithium-Ion Battery	http://docs.bluerobotics.com/batteries/	Purchased (2)	$350	2018
Power System	Zippy	Zippy Compact 8000 Battery	Voltage: 3S1P / 3 Cell / 11.1V Discharge: 30C Constant / 60C Burst Weight: 565g (including wire, plug & shrink wrap) Dimensions: 167x69x24mm Balance Plug: JST-XH Discharge plug: XT90	Purchased (2)	$70*	2022
Motor Controls	Blue Robotics	Basic ESC R2	https://bluerobotics.com/store/thrusters/speed-controllers/besc30-r3/ (Outdated)	Purchased	$36*	2017
CPU	NVIDIA	Jetson TX2	https://developer.nvidia.com/embedded/buy/jetson-tx2	Purchased	NC**	2018
Motor Controller PCB	VantTec	VantTec STM32 Controll Board 2022	Custom	Custom	NC	2022
Power Distribution PCB	VantTec	VantTec Power Distribution Board 2022	Custom	Custom	NC	2022
Water Pump	Tefola	12 V	Micro diaphragm pump, DC 12 V, 60 W, 0.8 Mpa, 5 L/min	Purchased	$22.95*	2023
Stepper Motor	NEMA	NEMA 23 Stepper Motor	Nema 23 CNC Stepper Motor 2.8A 178.5oz.in/1.26Nm CNC Stepping Motor	Purchased	$30.34*	2023
Stepper Driver	OUYZGIA	TB6600	TB6600 4 A 9-42 V Nema 17/23 CNC Controlador de eje único híbrido	Purchased	$11.59*	2023
MCU	STMicroelectronics	STM32F405RG	https://www.st.com/en/microcontrollers-microprocessors/stm32f405rg	Purchased	NC	2022
Teleoperation	FrSky	Taranis X9D	https://www.frsky-rc.com/product/taranis-x9d-plus-2/	Purchased	NC **	2017
Teleoperation	FrSky	X8R	https://www.frsky-rc.com/product/x8r/	Purchased	NC **	2017
IMU	SBG Systems	SBG Ellipse2-D	https://www.sbg-systems.com/products/ellipse-series	Purchased	NC **	2017
Camera	Stereolabs	ZED Camera	https://www.stereolabs.com/zed/	Purchased	$440*	2017
CAN transceiver	Waveshare	SN65HVD230	https://www.waveshare.com/sn65hvd230-can-board	Purchased	NC	2022
RF Modules	Digi	Xbee S3b	https://xbee.cl/xbee-pro-s3b-xsc-rpsma/	Purchased	NC	2021
LiDAR	Velodyne LiDAR	VLP-16	https://velodynelidar.com/vlp-16.html	Purchased	NC **	2017
Vision	Point Cloud Library, OpenCV, Yolov8					
Algorithms	VantTec Development					
Localization and Mapping						
Autonomy						
Open-Source Software	ROS 2, Python, C++, Matlab, Pytorch					

* Approximation **Sponsored