lVRX Competition Technical Guide

Virtual RobotX 2023 Competition *www.RobotX.org*

Table of Contents

1. Introduction

The purpose of this document is to provide Virtual RobotX (VRX) teams with the information necessary to successfully prepare for and participate in the VRX competition. It covers the following topics: the competition goals and approach, the robotic platform, propulsion and sensor configuration, generic information about how tasks work, runs and environmental envelopes, the application programming interface (API), and instructions for submitting your code for the competition.

For details of the individual tasks in the competition, see the VRX 2023 Task Descriptions.

2. VRX Goals and Approach

The VRX competition is a means of supporting engineering development for the Maritime RobotX Challenge and maritime autonomy in general. The tasks and scoring reflect this intent by emphasizing the foundational capabilities that lead to better autonomous performance. Testing autonomy algorithms in a relevant simulation environment is an efficient and cost-effective approach when compared to the challenges of testing system performance in a physical environment.

The simulation-based competition also rewards robust, repeatable performance by scoring each task over multiple trials where the environmental conditions (e.g., sea state, wind magnitude and direction, lighting, etc.) are varied between trials.

2.1. Submission and Evaluation

Evaluation of the VRX tasks will be performed in a specified *evaluation environment* composed of computational resources consistent with the System Requirements for Running VRX. Team submissions, consisting of configuration and software, will be executed by the VRX technical team to generate task scoring.

The details of the evaluation environment will be provided to teams, along with working examples and tutorials to allow teams to test their submissions thoroughly within an equivalent evaluation environment. This should ensure that the performance of each team's submitted solutions will be equivalent to performance during development.

2.2. Competition Scoring

VRX scoring is inspired by the low-point system used in sailboat racing. Teams will receive a task rank for each task. The VRX competition score is the total of the task rank for all tasks with lowest total points ranked first. The final overall ranking is assigned based on this score, again with lowest total points ranked first. For entries that are classified as "did not start," "did not finish" or "disqualified" for a specific task, the task rank shall be equal to the number of teams competing in that task.

Table 1: Low-Point Scoring

Task Rank	Task Points
First	1
Second	2
Third	3
…	…

Task Ties: Ties in task rank are not usually possible. However, if a tie does occur, points for the place for which the teams have tied and for the place(s) immediately below shall be averaged. For example, if there was a tie for ranks fourth through eighth, all tied participants would receive a rank of 6.

Competition Ties: If there is a competition score tie between two or more teams, each team's task points shall be listed in order of best to worst, and at the first point(s) where there is a difference the tie shall be broken in favor of the team(s) with the best score(s). If a tie remains between two or more teams, they shall be ranked in order of their task points in the last task. Any remaining ties shall be broken by using the tied teams' task points in the next-to-last task and so on until all ties are broken.

To illustrate the scoring strategy, consider the following example:

Table 2: Competition Scoring Example

Task	Team A Points	Team B Points	Team C Points
1	1	2	3
2	2	1	3
3	1	3	3
4	3	2	1
5	2	1	3
Total	8	8	13

In this example, Teams A and B would be tied for first place in the competition and Team C would be in third place. The task points for A and B would be listed in order for each team. For both teams the ordering is 1, 1, 2, 2, 3 To break this tie the scores would be compared for Task 5; Team B has 1 and Team A has 2, so the tie is broken in favor of Team B. The final results would be Team B, first place; Team A, second place; and Team C, third place.

2.3. Competition Phases

The VRX competition consists of three sequential phases.

- Phase 1: Hello World
- Phase 2: Dress Rehearsal
- Phase 3: VRX Challenge

The first phase will take place in September of 2023. The second and third phases will follow in October and November of 2023. Final due dates are posted on the VRX website.

2.3.1. Phase 1: Hello World

This simple check encourages teams to start early and is a means to identify technical issues with the simulation environment. The goal of this phase is for teams to demonstrate that they have set up the VRX simulation environment locally, on their own computers, and to provide a means for teams to demonstrate prototype solutions to the VRX tasks.

- Preparation:
  - Teams access the VRX code, documentation and tutorials to support setting up their local development environment.
  - Teams review the competition documents: Task Descriptions, Technical Guide, etc., available on the VRX website.
  - For technical support, teams are encouraged to submit to the VRX issue tracker.
- Submission:
  - Each team submits a video demonstrating the team's ability to run their own autonomy software within the VRX simulation environment. The purpose of the video is to document team status and progress towards completing the VRX challenge tasks. While not required, it is expected that many teams will be able to demonstrate prototype solutions to a subset of the VRX tasks.
  - It is expected that the VRX simulation and the team's solutions (control and autonomy software) run locally on the team's computers.
  - Teams are encouraged to demonstrate successful solutions to as many of the VRX Tasks as possible.
- Evaluation:
  - Teams must submit a video to be eligible to participate in future phases.
  - Videos will be shared with the community unless teams request that their videos remain private.
- Next Steps:
  - Teams review instructions for final submissions (see the VRX Wiki) and become familiar with the evaluation environment infrastructure.
  - Teams continue to develop solutions to the VRX tasks and begin testing solutions within the evaluation environment.

2.3.2. Phase 2: Dress Rehearsal

To complete this phase successfully teams should have prototype solutions for the VRX tasks and should be able to submit their solutions for automatic evaluation within the evaluation environment as described in the VRX Technical Guide. The task scores awarded in this phase are for practice only and will not count toward final rankings.

- Preparation:
  - Teams review instructions to create and test solutions within the VRX evaluation environment.
  - Teams are expected to test their solutions in their own evaluation environment, including running the automated scoring as preparation for the dress rehearsal submission.
  - Teams review detailed documentation of the submission process and tutorials on how to submit and score their solutions.
  - Technical support continues to be managed primarily through the VRX issue tracker.
- Submission:
  - Each team submits a solution as described on the VRX Wiki.
- Evaluation:
  - To be eligible for Phase 3, teams must submit their solution for automatic evaluation as described in the VRX Technical Guide.

- After the submission deadline, the VRX technical team will execute each team's solution for all of the VRX tasks. Execution will be performed in the same evaluation environment used for the final stage of VRX (see Phase 3: VRX Challenge, below).
- A scoring report is generated for each team and for each task. The scoring process is detailed in the competition documents as well as the Task Tutorials.
- The VRX technical team will publish a "leaderboard" summarizing scores from the Dress Rehearsal.
- We will also release detailed log files for all scored runs.

- Next Steps:
  - Teams continue to develop and test their solutions. Teams will be supplied with software, instructions and computational specifications to do their own automatic evaluation, emulating the final VRX scoring.

2.3.3. Phase 3: VRX Challenge

In this final phase of the competition the task scoring will count toward the team's final rankings.

- Preparation:
  - Teams continue to develop and test their autonomous solutions based on results from the Dress Rehearsal.
  - Real-time technical support is provided during the last days before final submission. Early submissions are encouraged to ensure time for troubleshooting.
- Submission:
  - Following the instructions on the VRX Wiki, each team submits their software for automatic evaluation and scoring within the evaluation environment.
- Evaluation:
  - After the submission deadline, the VRX technical team will execute each team's solution for all the VRX tasks. The simulated environmental conditions (e.g., wind, waves, buoy locations, etc.) will be different from the Dress Rehearsal, but within the same environmental envelope described in section seven of this guide, below.
  - Final task performance rankings will be determined based on automated scoring plugins documented below.
  - As in phase 2, we will release detailed log files for all scored runs.
- Next Steps
  - Post-processing in preparation for announcement of results, including finalizing scoring and creating highlight videos.

  - Winners and prizes will be announced 2-3 weeks after the VRX Challenge submission deadline.

3. Robotic Platform

The VRX competition will be executed using the Gazebo simulation environment. All teams must use the simulated version of the Wave Adaptive Modular Vessel (WAM-V) surface craft distributed with the VRX software. The WAM-V as supplied by the VRX software is standard for all teams. No modification of the standard platform model (URDF description, etc.) is allowed during competition.

The simulated WAM-V includes models of rigid body dynamics, hydrodynamics, external forcing (waves and wind) and propulsion. The rigid body dynamics are captured via the Gazebo physics engine. The hydrodynamics, external forcing and propulsion forces are generated by VRX plugins with fixed parameters. No modification of the standard VRX model parameters is allowed during competition.

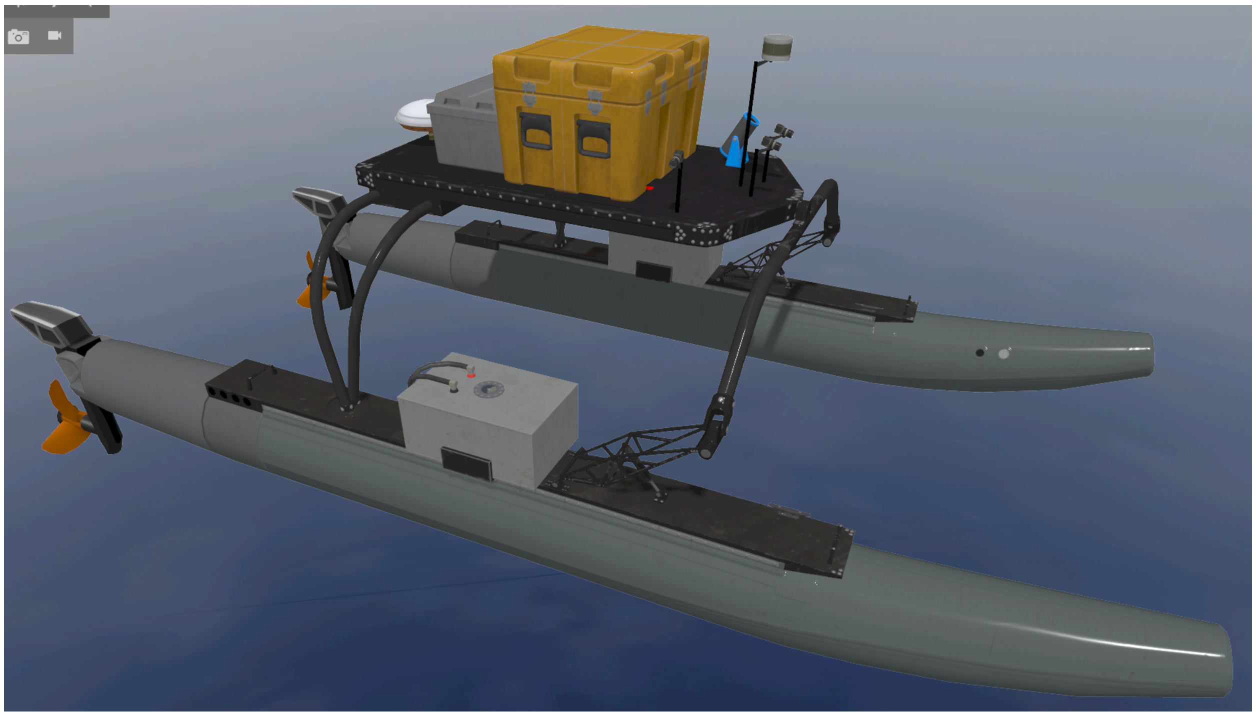


Figure 1: Simulated WAM-V model

4. Propulsion configuration

As part of the competition, teams may customize the propulsion system with which their WAM-V is equipped. Vehicle propulsion is provided by a set of identical thrusters. The details of the thruster model are explained on the Theory of Operation page of the VRX Wiki. The maximum number of allowed thrusters is four. All thrusters have the ability to dynamically rotate and thus change the thrust direction. The pose (position and attitude relative to the vessel) of each thruster is configurable via a YAML file.

In order to preserve realism, not all possible thruster positions will be accepted. The valid set of allowed positions are represented in Figure 2 as five red bounding boxes. In order for a thruster configuration to be considered valid, the origin of each thruster must be contained within one of these bounding boxes and each bounding box must contain no more than one thruster.

Teams can visit the VRX Wiki for a tutorial on customizing the WAM-V. The same tutorial describes the format of the YAML file used for propulsion configuration, and provides working examples of YAML configuration files with typical propulsion configurations.

During competition events, teams will need to provide a valid YAML file named `thruster_config.yaml` with their propulsion configuration.

Table 3: Dimensions of Bounding Boxes

Name	Position and Orientation of Center of Bounding Box (x y z R P Y) (Relative to the Origin of the WAM-V)	Size (x y z)
Front Left	1.25 1 0 0 0 0	1 1 1.2
Front Right	1.25 -1 0 0 0 0	1 1 1.2
Center	0.25 0 0 0 0 0	2.5 1 1.2
Back Left	-2.25 1 0 0 0 0	1 1 1.2
Back Right	-2.25 -1 0 0 0 0	1 1 1.2

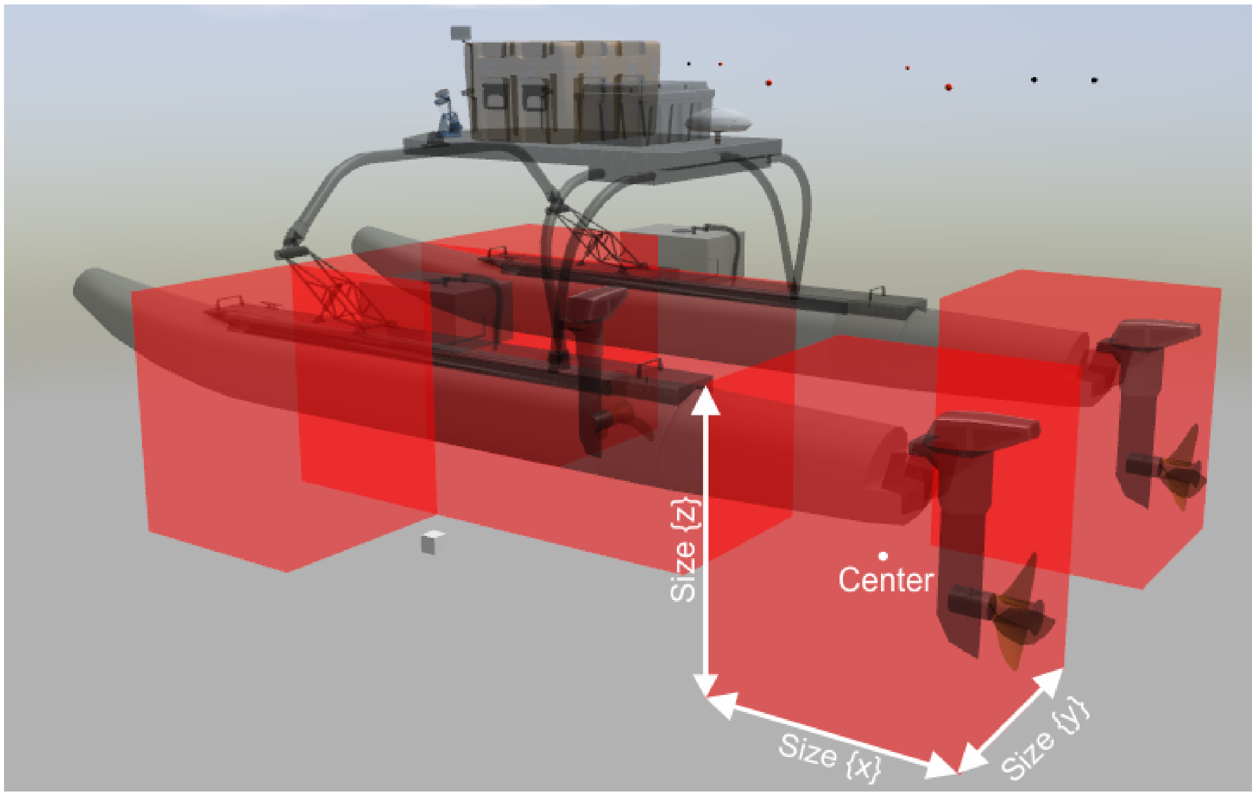


Figure 2: Allowed thruster regions in WAM-V

C

5. Component configuration

As with the propulsion system, teams should customize the other components with which their WAM-Vs are equipped by choosing the type and placement of those components. Standard components include navigation sensors (GPS and IMU), perception sensors (cameras, lidars and acoustic receiver), and the ball shooter. In addition to configuring component types, teams may also customize their placement.

Table 4: Maximum number of sensors allowed per sensor type

Sensor type	Maximum number of instances
Camera	3
Lidar	2
GPS/IMU	1
Acoustic Receiver	1
Ball shooter	1

The available component types and their performance characteristics have been chosen to reflect commonly used components from the RobotX physical competition. These characteristics do not represent any specific sensor choice but are meant to be representative of typical hardware options.

Note: While the sensor performance specifications (update rates, noise values, etc.) are detailed below, the exact values of these specifications may change before the final release of this document.

5.1. Navigation Sensor

A single standard navigation sensor, representing a GPS-aided IMU, will be used for VRX. This single sensor is simulated through the use of Gazebo's NavSatSensor and ImuSensor (see gz-sensors) which generate GPS information (position and velocity) and IMU information (attitude, attitude rate and accelerations). While these two measurements are presented separately, the characteristics of the measurements are consistent with a sensor that estimates a complete navigation solution, e.g., a GPS-aided IMU.

Table 5: Characteristics of VRX Navigation Sensor

GPS-Aided IMU		
GPS	Update Rate	20 Hz
	Horizontal Position Noise[1]	0.85 m
	Vertical Position Noise	2.0 m
	Velocity Noise	0.1 m/s
IMU	Update Rate	100 Hz
	Acceleration Offset/Bias	+/- 0.002 g
	Acceleration Noise	0.275 g
	Attitude Rate Noise	0.08 degrees/s
	Heading Noise	0.8 degrees

[1] Unless otherwise noted, noise values are specified as one standard deviation and represent a Gaussian distribution.

5.2. Camera Sensor

A standard camera is simulated via the Gazebo camera plugin. Teams may provision their system with up to three cameras.

Table 6: Characteristics of VRX Camera Sensor

Camera	
Update Rate	30 Hz
Resolution	1280x720 px
Color format	R8G8B8

5.3. Lidar Sensors

Two types of lidar sensors are provided for VRX: 16 beam and 32 beam. Teams may provision their system with up to two lidars.

Table 7: Characteristics of VRX Lidar Sensors

	16 Beam	**32 Beam**
Update Rate [Hz]	10	10
Lasers (Number of beams)	16	32
Samples (Number of horizontal rotating samples)	1875	2187
Min Range [m]	0.1	0.1
Max Range [m]	130	130
Noise [m]	0.01	0.01
Min. Horizontal Angle [rad]	$-\pi$	$-\pi$
Max. Horizontal Angle [rad]	π	π
Min. Vertical Angle [rad]	$-\frac{\pi}{12}$	-0.186
Max. Vertical Angle [rad]	$\frac{\pi}{12}$	0.54

5.4. Ball Shooter

This component is used in tasks that require the WAM-V to send a payload through a target area. It is limited to 4 shots, and only one is permitted per system. Teams may specify the pitch and yaw of the barrel.

5.5. Acoustic Receiver

The inclusion of this sensor enables the system to subscribe to topics broadcast by acoustic beacons. Unlike other sensors, its location is fixed below the WAM-V, and it does not need to conform to constraints on component placement described in the section that follows.

5.6. Component Placement

The pose of each component is configurable via a YAML file. In order to preserve realism, not all possible sensor positions will be accepted. The valid set of allowed positions are represented in Figure 3 as green bounding boxes. The origin of each sensor needs to be contained within one of these bounding boxes to be considered valid. Note that the specification of the sensor is not customizable; all teams share the same sensor types.

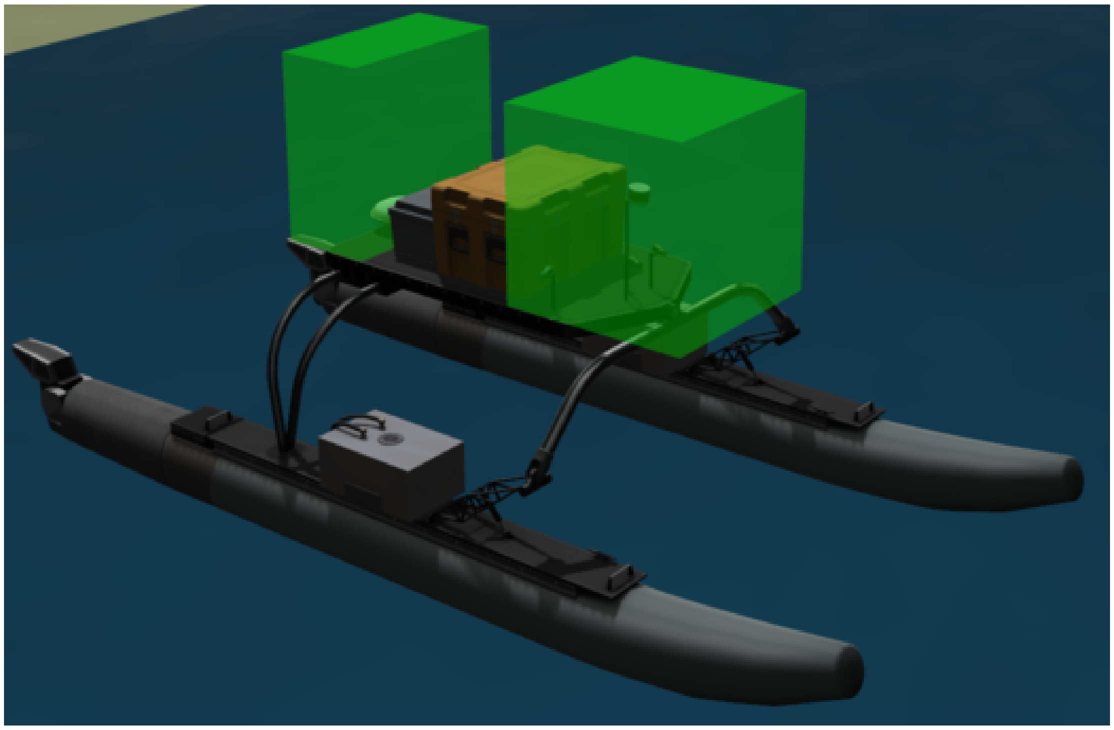


Figure 3: Allowed sensor regions in WAM-V

As with thruster placement, teams can visit the VRX Wiki for a [tutorial on customizing the WAM-V](). The tutorial also describes the format of the YAML file used for sensor configuration.

During competition events, teams will need to provide a valid YAML file, named `component_config.yaml`, with their sensor configuration.

Table 8: Allowed sensor regions in WAM-V

Placement region	**Bounding box center (x y z)**	**Bounding box size (x y z)**
Front	**0.8 0 1.8**	**1 1 1**
Back	**-0.9 0 1.8**	**0.5 1 1**

6. Task Structure and Implementation

This section describes several important features common to all competition tasks.

6.1. Task States

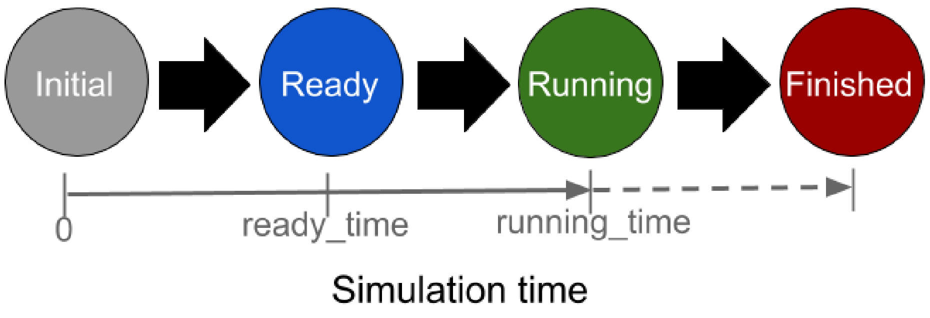


Figure 4: Task states as a function of time

A task can be in one of four different states. The task state is set by Gazebo according to the task configuration. The current state is included in the task message periodically published on the task information ROS 2 topic.

6.1.1. Initial

After Gazebo starts, the task is in the *initial* state. The robot's motion is fixed in the X (surge), Y (sway) and yaw degrees of freedom, but allowed to move in Z (heave), pitch and roll degrees of freedom. Thus, the robot is pushed up and down by the waves and wind and will change its orientation (except in yaw) but stays in the same 2D position. The purpose of this initial state is to allow for simulation startup transients to decay and for all the user's software to have sufficient time to initialize.

6.1.2. Ready

The task transitions to *ready* when simulation time reaches the value `ready_time`. In the *ready* state, the robot motion is free in all degrees of freedom and is under the participant's full control. While in the ready state no scoring is performed.

6.1.3. Running

The task transitions to *running* when simulation time reaches the value `running_time`. In the *running* state, the task officially starts. The scoring and the task timer are enabled.

6.1.4. Finished

The task transitions to *finished* when the `remaining_time` field of the task message reaches 0 or when the task is considered complete. If all task time has been consumed, but the task has not been fully solved, the field `timed_out` of the task message will be set to true. The score will not be updated in this state.

6.2. Task Info Message

During every task, the status of the task is published as a ROS 2 ParamVec message over a ROS 2 topic. The message contains a series of name-value pairs that summarize the task status. See Table 9 for a complete list of parameter names and descriptions. Please refer to the VRX API section of this document for further details.

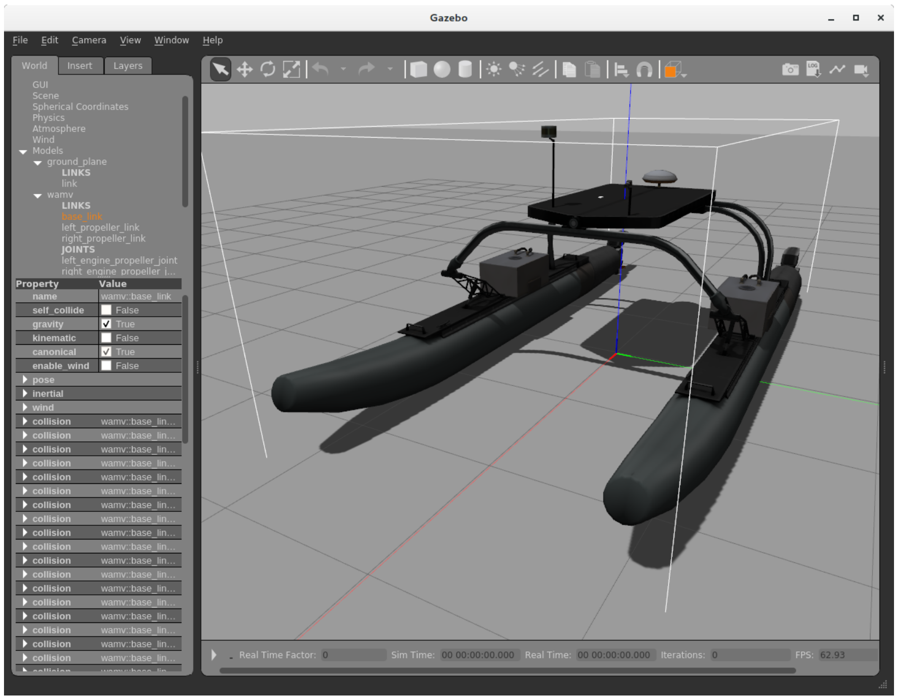
Table 9: ROS 2 Task message definition

Parameter Name	Description
`name`	Unique task name (e.g.: "stationkeeping", "wayfinding").
`state`	The current task state = {initial, ready, running, finished}. See the *Task States* section for more information.
`ready_time`	Simulation time at which the task transitions to the *ready* state.
`running_time`	Simulation time at which the task transitions to the *running* state.
`elapsed_time`	Elapsed time since the start of the task (since running_time).
`remaining_time`	Remaining task time.
`timed_out`	Whether the task has timed out or not.
`score`	Current task score.
`num_collisions`	Number of times the vehicle has collided with a course element.

Teams are expected to subscribe to this task ROS 2 topic and select their appropriate robot behavior given the current task under execution. In addition, teams need to react to the task states accordingly. The *initial* state is only used to stabilize the vehicle, allow for initial transients to decay and make sure that all the software blocks are ready. While the system is in the initial state, teams receive sensor information, but the robot control will be very limited. In the *ready* state, teams have full control of their robot and we expect them to get ready for the start of the task. Teams need to monitor the simulation time published over the clock ROS 2 topic and compare it with the `ready_time` and `running_time` to be prepared to take control of the vehicle and start the task respectively. Once the task is in the *finished* state, teams still can control the vehicle, but the score will not change.

6.3. Simulated Pose Reference for Task Evaluation

Task evaluation often involves the use of the true, simulated pose (position and attitude) of objects within the environment. For the WAM-V, the pose is evaluated at the origin of the `WAM-V` model frame, which is coincident with the `base_link` reference frame. This origin is shown in the image below as the location of the red-green-blue axes.



Property	Value
name	wamv::base_link
self_collide	False
gravity	✓ True
kinematic	False
canonical	✓ True
enable_wind	False
pose	
inertial	
wind	
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...
collision	wamv::base_lin...

```
Real Time Factor: 0    Sim Time: 00 00:00:00.000    Real Time: 00 00:00:00.000    Iterations: 0    FPS: 62.93
```

Figure 5: Illustration of origin reference frame location for WAM-V.

Similarly, for other objects in the simulation (e.g., buoys, markers, totems, etc.) the true, simulated location is the origin of the link frame associated with the object. The image below illustrates this reference frame for the `mb_marker_buoy_white` object.

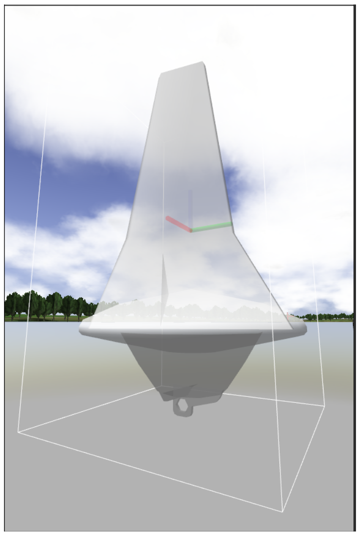


Figure 6: Illustration of origin link location for mb_marker_buoy_white object.

7. Runs and Environmental Envelopes

In the competition, each team's solution will be evaluated over multiple runs per task, where each run will use a different set of environmental conditions. Though conditions will be distinct from run to run, these distinct configurations will be identical for each team; i.e., each team will see the same set of conditions as the other teams in the competition. The subsections below describe elements of the simulation that may change between runs.

7.1. Object Location and Orientation

Relevant objects will be moved between runs to discourage training the team's controllers to handle known geometry across runs. This will include the placement of obstacles (for example, buoys or docks), as well as the starting pose of the robot itself.

7.2. Fog

Gazebo will optionally simulate fog with different densities and colors. The two images below illustrate the addition of fog to the visual scene.

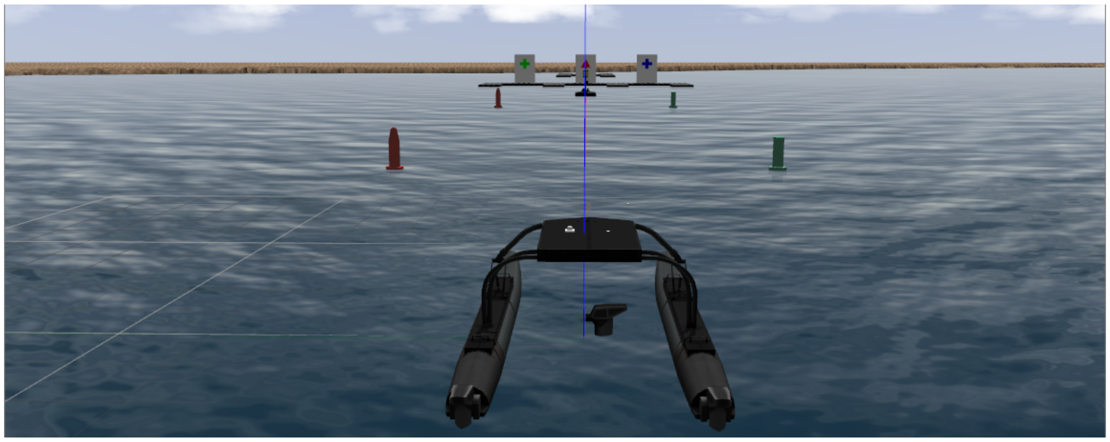


Figure 7: Visual scene with no fog

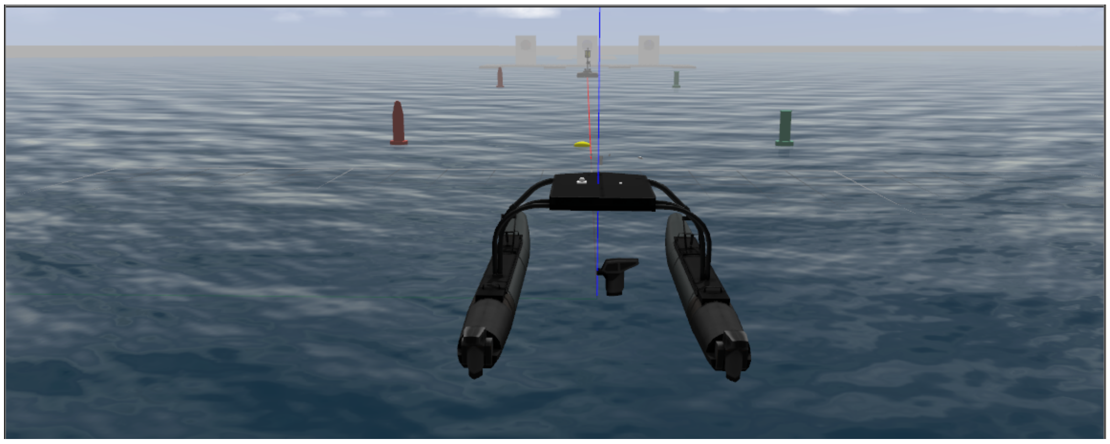


Figure 8: Visual scene with fog

7.3. Wind

Wind exerts a force on objects in the VRX environment. The total wind velocity is a combination of a constant mean velocity component and a variable wind speed (i.e., gusting). The variable component of the wind speed is modeled as a first-order linear spectrum defined by two components: the variability gain and the variability time constant. The variability gain specifies the magnitude (root-mean-square) of the variable component of the wind speed, and the variability time constant specifies how rapidly the wind speed changes with time. For details on the wind model and implementation please refer to the Theory of Operation on the VRX Wiki. For examples of how to change the wind parameters see the tutorial on Changing Simulation Parameters.

7.4. Waves

Surface waves affect the motion of objects in the simulated environment. The simulated sea state is generated using a summation of individual regular waves to create the three-dimensional water surface geometry as a function of time. The amplitudes of the individual waves are determined by sampling a Pierson-Moskowitz (P-M) ocean wave spectrum. The key parameters for specifying a particular sea state are as follows:

- Peak Period (T_p) – wave period with the highest energy.
- Gain (γ) – constant multiplier applied to the individual wave amplitudes.
- Mean Wave Direction – direction of travel of the wave component corresponding to the peak period.

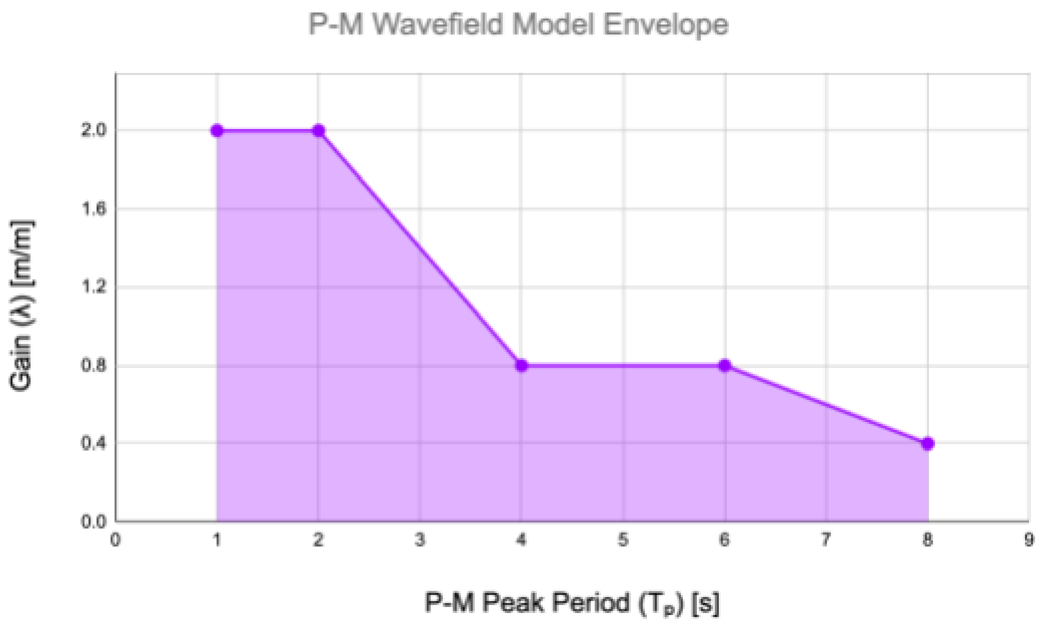


Figure 9: Envelope for sea state parameters used in VRX evaluation.

The combination of T_p and γ parameters determine the energy in the specific sea state. For the VRX competition, combinations of T_p and γ as illustrated in Figure 9 will be used for evaluation. Other aspects of the model, such as the steepness of the waves, number of samples, angular difference in direction between component waves, etc. can also optionally be adjusted if desired. For further details on the sea state model see the Theory of Operation on the VRX Wiki.

7.5. Ambient Light

The color of the ambient light. The two images below illustrate changes to the ambient lighting conditions.

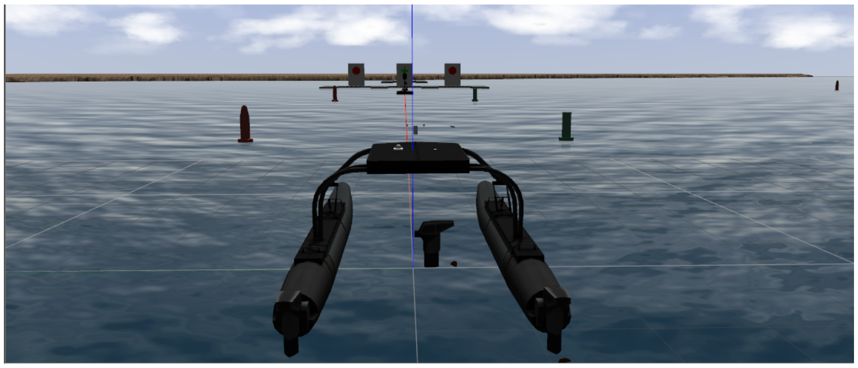


Figure 10: Visual scene with reduced ambient light

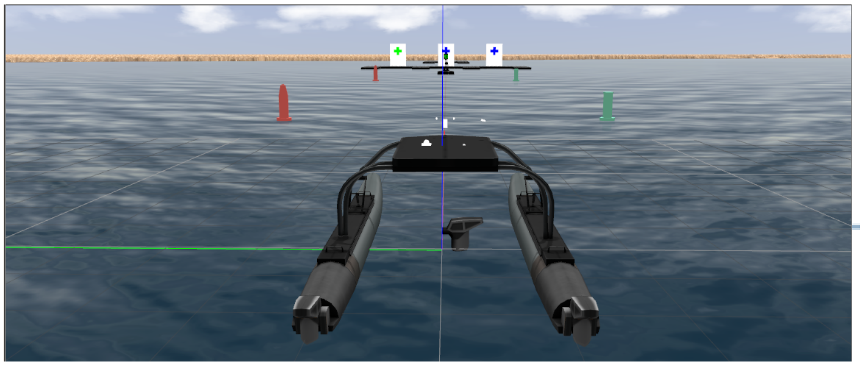


Figure 11: Visual scene with regular ambient light

7.6. Summary of Environmental Parameters

Table 10 summarizes the ranges of all the parameters that can change during runs:

Table 10: Environmental variable parameters

Gazebo Parameter	Minimum value	Maximum value
`scene::fog::color`	[0.7, 0.7, 0.7, 1]	[0.9, 0.9, 0.9, 1]
`scene::fog::density`	0	0.1
`scene::ambient`	[0.3, 0.3, 0.3, 1]	[1, 1, 1, 1]
`wamv_gazebo::wind_mean_velocity`	0	8
`wamv_gazebo::wind_variance_gain`	0	8
`wamv_gazebo::wind_variance_time`	2	20
`wamv_gazebo::wind_direction`	0	360
`wamv_gazebo::wave_period`	See Figure 9.	
`wamv_gazebo::wave_gain`		
`wamv_gazebo::wave_direction`	0	360
`wamv_gazebo::wave_angle`	0	360

The characteristics of the simulated environment will be varied during competition runs; however, teams can expect values to be within the ranges described in the table above.

8. VRX API

VRX provides a ROS 2 interface to the teams for controlling all available actuators, reading sensor information and sending/receiving notifications. The use of ROS 2 as the interface between the team's software and the simulation environment does not require that the team's software internally use ROS 2. The intention of the competition is to be technology-agnostic with regard to solution architecture and implementation. However, a single standard interface is required for the feasibility of the virtual competition. Every effort will be made to offer all teams support implementing the ROS 2 interface to their software. For teams not familiar with ROS 2 we highly recommend going through the ROS 2 tutorials to get familiar with ROS 2 and, in particular, with ROS 2 topics and services.

Tables 11-14 summarize the ROS 2 API used for the competition. Note that all topic names used for propulsion and sensors are configurable via their respective YAML files.

Table 11: API for Actuators (default topics)

Actuators	
Topic Name	**Description**
`/wamv/thrusters/<thrustername>/pos`	Next angle command for the <thrustername> thruster
`/wamv/thrusters/<thrustername>/thrust`	Next power command for the <thrustername> thruster
`/wamv/shooters/ball_shooter/ fire`	A boolean message that causes the ballshooter to fire its projectile when it receives a value of "true."

Table 12: Sensor information API (default topics)

Sensor Information	
Topic Name	**Description**
/wamv/sensors/cameras/<sensorname>/camera_info	Meta information for <sensorname> camera. See CameraInfo message for details.
/wamv/sensors/cameras/<sensorname>/image_raw	Raw image data for <sensorname> camera. See Image message for details.
/wamv/sensors/gps/gps/fix	GPS position data.
/wamv/sensors/imu/imu/data	IMU data describing orientation, angular velocity and linear acceleration.
/wamv/sensors/lidars/<sensorname>/points	<sensorname> 3D lidar output
/wamv/sensors/acoustics/receiver/range_bearing	A distance (range) and two angles (bearing and elevation) indicating the relative position of the acoustic pinger from the USV, with noise.

Table 13: Task information API

Tasks[2]	
Topic Name	**Description**
/clock	Simulation time
/vrx/task/info	Task information

Table 14: Debug information API (not available during competition)

Debug[3]	
Topic Name	**Description**
/vrx/debug/wind/direction	Wind vector (units in degrees and ENU coordinate)
/vrx/debug/wind/speed	Magnitude of the wind

The interface described above is generic to the entire competition, including all tasks. The task-specific elements of the interface are described in the Virtual RobotX Task Descriptions, which details the additional ROS 2 topics and services used to support individual task execution. Finally, note that the VRX interface also

[2] Each task may also use an additional set of ROS 2 topics/services. Please consult the Virtual RobotX 2023 Task Descriptions document for additional information.

[3] All Gazebo topics to query ground truth information and other simulation aspects are available for debugging.

includes elements that are not used in either the generic or task-specific APIs; these are not needed for the competition and can be disregarded.

9. Submission and Code Execution

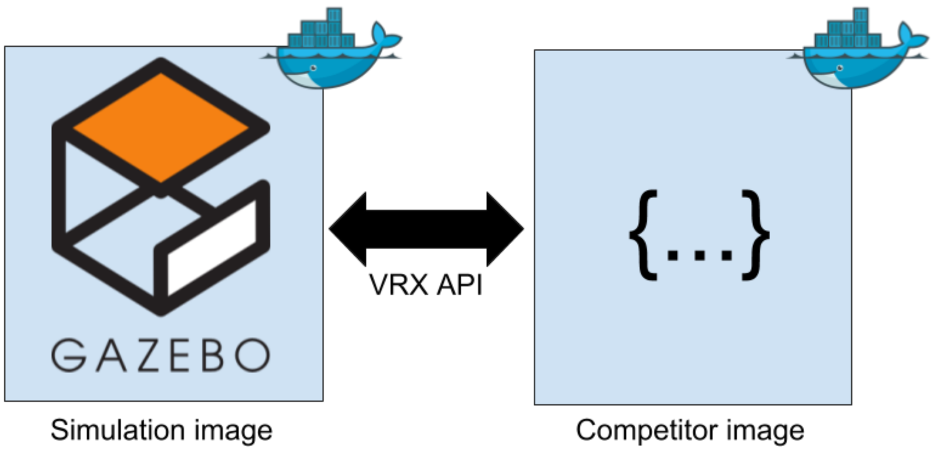


Figure 12: Architecture used to execute competitor code

We expect to receive multiple files from each competitor prior to the event associated with each phase of the competition. These files will specify different aspects of the WAM-V configuration, as well as the team's controller. Please, see the VRX Wiki for an overview of how to participate in each phase, as well as detailed instructions about how to submit a solution for a given event.

Performance for each task will be evaluated as follows:

1. Each team's customized WAM-V will be generated.
2. A Docker container running the VRX simulation image will be executed. This container will execute Gazebo with the VRX environment configured to run a particular task. Additionally, Gazebo will be configured to record a log of the execution.
3. A ROS bag (log file) will capture all task messages containing the score.
4. A team's Docker container (running the team's image) will be executed. It's expected that the entry point of this Docker instance spawns all the necessary elements of the team's code.
5. The competitor's code should interact with the simulation via the VRX API, determine the current task via the VRX API, and try to solve it.
6. When the task has been completed or has timed out, the Gazebo log file and the ROS bag will be saved and tagged appropriately.

This process will be repeated for each run of each task and for all the teams participating in the event. This architecture allows the execution of the entire competition in batch mode. Teams may run a competition themselves locally using the same set of tools that will be used during the official events. These automatic evaluation tools are available in the vrx-docker repository. We encourage all teams to use it for testing their solutions.