

# VRX Competition and Task Descriptions

*Virtual RobotX 2022 Competition**www.RobotX.org*

## 1. Introduction

The purpose of this document is to outline the structure of the Virtual RobotX (VRX) competition for 2022, including the individual tasks which make up the competition. The details of the individual tasks, including their implementation and scoring, are under active development. This document, along with the preliminary VRX Technical Guide, are being made available as a means for competitors to provide feedback early in the design of this new competition.

## 2. VRX Goals and Approach

The VRX competition is a means of supporting engineering development for the Maritime RobotX Challenge. Participation in this competition should raise the level of vehicle performance at the Maritime RobotX Challenge. 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

Details on the process for submission of software solutions are provided in the VRX Technical Guide, available at the [VRX Competition website](#). In summary, 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:** Typically ties in task rank are not 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.

### 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 December of 2021. The second and third phases will follow in March and April of 2022. Final due dates are posted on the [VRX website](#).

### 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: Competition and Task Descriptions, Technical Guide, etc., available on the [VRX website](#) and the [RoboNation Forum](#)
  - 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.

### 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 (VRX Wiki) 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.
  - Teams are encouraged to submit standalone documentation of their solutions such as video demonstrations of their solutions. These submissions are optional.

- 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.
- 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.

### 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 for the last days/hours before final submission.
- Submission:
  - Following the instructions on the VRX Wiki, each team submits their software for automatic evaluation and scoring within the evaluation environment.
- Evaluation:
  - 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 the VRX Technical Guide.
  - Final task performance rankings will be determined based on automated scoring plugins documented below.
- Next Steps
  - Post-processing in preparation for announcement of results, including finalizing scoring and creating of highlight videos.
  - Winners and prizes will be announced 2-3 weeks after the VRX Challenge submissions.

## 4. Descriptions of Tasks

Each of the VRX tasks is scored with respect to the performance metrics described below. For each task, participating teams are ranked in order of the individual task score. The overall competition score is determined by the sum of the rank ordering for the individual tasks; the lowest score indicates the best performance.

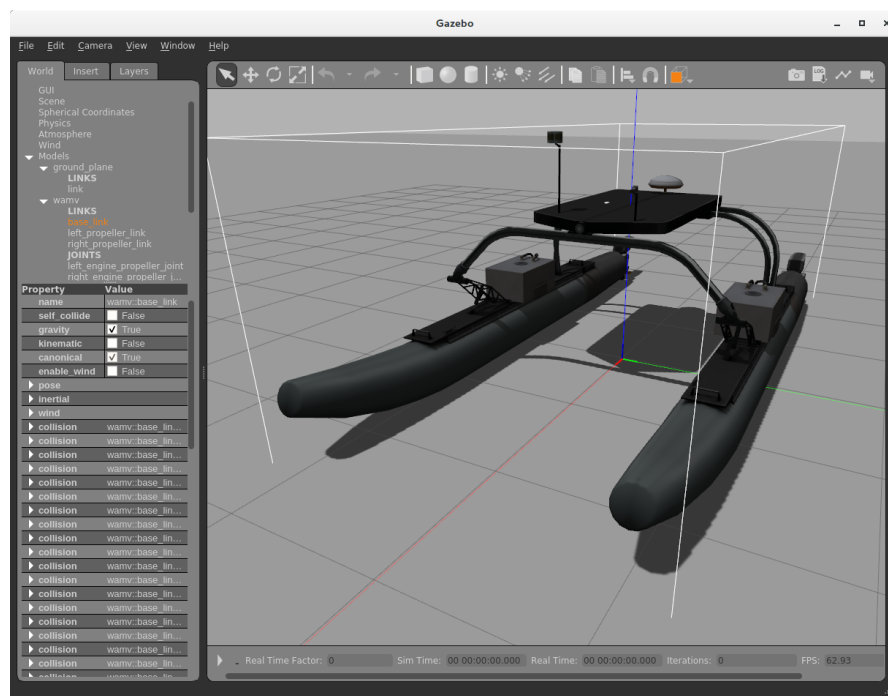
In addition to the descriptions below, the VRX project Wiki contains [VRX 2022 Task Tutorials](#) with working demonstrations for each of the tasks. Teams are encouraged to run these demonstrations to understand how the tasks will be implemented and scored.

General task information (e.g., task name, task status, etc.) are available in real-time through a ROS API. Each run of a specific task evolves through a set of sequential states Initial->Ready->Running->Finished. Participants should review this general task interface and structure as described in the VRX Technical Guide.

## 4.1. General Definitions

### 4.1.1. Simulated Pose Reference

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 wamv 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.



**Figure 1: 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.

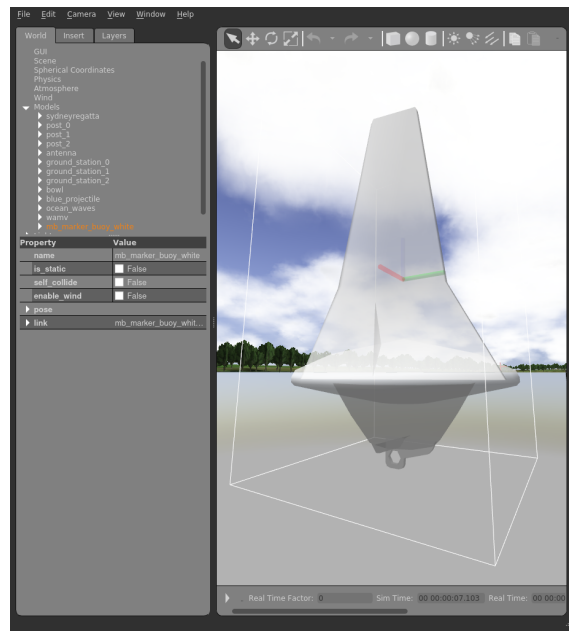


Figure 2: Illustration of origin link location for mb\_marker\_buoy\_white object.

#### 4.1.2. Task Names

During every task the status of the task is published as a custom ROS [task message](#) over a ROS topic as detailed in the VRX Technical Guide. The name field of this task message uniquely specifies the current task, as shown in the table below.

Table 3: Task Naming

Task	Task Message Name
Station-Keeping	station_keeping
Wayfinding	wayfinding
Landmark Localization and Characterization	perception
Wildlife Encounter and Avoid	wildlife
Channel Navigation, Acoustic Beacon Localization and Obstacle Avoidance	gymkhana
Scan and Dock and Deliver	scan_dock_deliver

## 4.2. Level 1: Control and Perception Fundamentals

### 4.2.1. Task 1: Station-Keeping

#### Capability:

System should be capable of performing localization by fusing sensor data (e.g., GPS, IMU, etc.) and maintaining USV position and heading in the presence of environmental forcing (e.g., wind and waves).

### Implementation:

The performance in this task is measured using a combined pose error distance, given as:

$$E_{pose} = d + k^d h \quad (1)$$

where  $d$  is the Euclidean distance in meters between the true, 2D position of the USV and the goal,  $h$  is the positive difference in radians between the USV heading and the goal heading, and  $k^d$  is a weighting term. The value of  $k$  is set to 0.75.

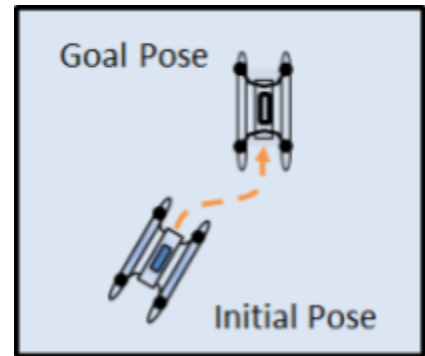
An instantaneous pose error is calculated at each time step. The total run score is the mean pose error over the total number of time steps in the simulation.

The task is implemented in the following sequential states:

- *Initial*: The simulation is initialized with the USV in a random location within the operating area.
- *Ready*: The goal pose is published to the `/vrx/station_keeping/goal` ROS topic.
- *Running*: Scoring begins. For the purposes of development and debugging, the following items are provided as ROS topics:
  - The instantaneous pose error is published to the `/vrx/station_keeping/pose_error` ROS topic.
  - The current mean pose error is published to the `/vrx/station_keeping/mean_pose_error` ROS topic.

**Note:** The above scoring publications will not be available to team software during the final, scored runs of the competition.

- *Finished*: Scoring ends.



### Scoring:

- Run score is the mean pose error for each run of the task.
- Task score is the mean of the run scores for all runs.
- Task rank is the ordering of task scores from lowest to highest.

The mean pose error is determined for each run of the task. The overall task score which determines the task ranking for each team is the mean pose error over all scored runs.

Table 4: Station Keeping API

Topic Name	Message Type	Description
/vrx/station_keeping/goal	Geographic_msgs::GeoPoseStamped	The goal pose, consisting of a position in spherical (WGS84) coordinates and a heading, given as a quaternion.
/vrx/station_keeping/pose_error	std_msgs::Float64	A 1 Hz sample of the current 2D pose error metric, which summarizes the current difference between USV and goal in terms of position and heading.
/vrx/station_keeping/mean_pose_error	std_msgs::Float64	The mean pose error accumulated over the duration of the run so far.

#### 4.2.2. Task 2: Wayfinding

**Capability:** System should be capable of command and control of USV to achieve a series of given goal states, specified as a series of locations/heading values.

**Implementation:**

Teams receive a list of goal states which they are free to visit in any order. The simulation continues until the maximum time is exceeded. At each time step, the combined pose error is calculated between the USV and every waypoint using the same formula for pose error distance described in the station keeping task. The performance for each waypoint is quantified by the minimum pose error achieved for that waypoint over all time steps.

The overall performance for a given run is calculated as the mean of the minimum pose errors over all waypoints.

The task is implemented in the following sequential states:

- *Initial:* The simulation is initialized with the USV in a random location within the operating area.
  - Note that in some run's obstacles may also be initialized in the operating area.
- *Ready:* The full list of goal states (waypoints) is published to the /vrx/wayfinding/waypoints ROS topic.
- *Running:* Scoring begins. For the purposes of development and debugging, the following items are provided as ROS topics:
  - The minimum error achieved so far for each waypoint is published to the /vrx/wayfinding/min\_errors ROS topic.
  - The mean of the current minimum errors for each waypoint is published to the /vrx/wayfinding/mean\_error ROS topic.

**Note:** The above scoring publications will not be available to team software during the final, scored runs of the competition.
- *Finished:* Scoring ends.

**Scoring:**

- Run score is the mean of the “minimum pose errors over all waypoints” (1) for a single run of the task.
- Task score is the mean of the run scores for all runs.



- Task rank is the ordering of task scores from lowest to highest.

Table 5: Wayfinding API

Topic Name	Message Type	Description
/vrx/wayfinding/waypoints	Geographic_msgs::GeoPath	An array of waypoints, each consisting of a position given in spherical (WGS84) coordinates and a heading given as a quaternion.
/vrx/wayfinding/min_errors	Std_msgs::Float64MultiArray	An array containing the minimum 2D pose error so far achieved between the USV and each waypoint.
/vrx/wayfinding/mean_error	std_msgs::Float64	The mean of the minimum errors so far achieved for all waypoints.

#### 4.2.3. Task 3: Object Localization and Characterization

##### Capability:

Using perceptive sensors (cameras, LiDAR, etc.), system should be capable of identifying, characterizing and localizing RobotX objects including buoys, totems, placards and docks. Perception should be robust with respect to vehicle motion (heave, pitch and roll) and environmental conditions (lighting, camera noise, etc.).

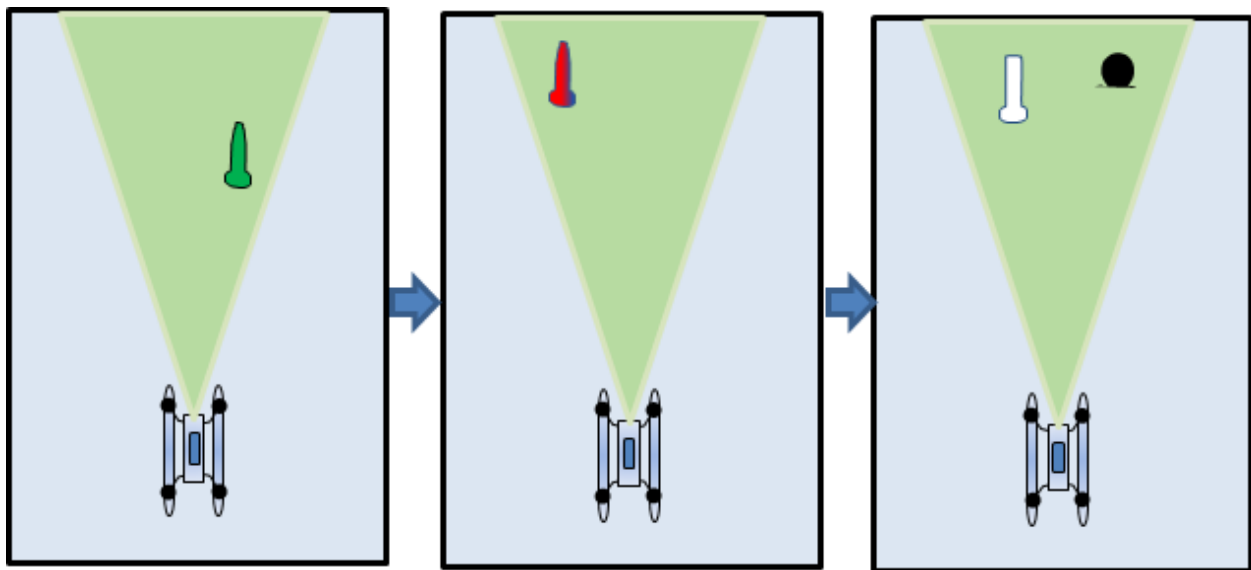


Figure 4: Three individual cases during the task as markers and objects are sequentially added to the field of view.

##### Implementation:


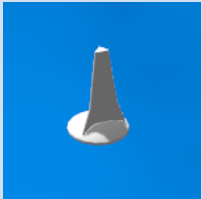



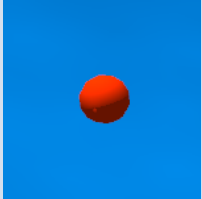
This task is made up of multiple *runs*, each under different environmental conditions. A single run is made up of multiple *trials*, where each trial consists of spawning one or more objects within the field of view of

the USV and then removing the objects for that trial. Objects for identification and localization may include models such as navigation markers, totems and obstacles. The *trial time* is the elapsed time between the appearance and disappearance of an object. For an identification/localization answer to be scored, the message must be received during the trial time for that object. In competition, the trial time for all trials will be fixed at five seconds.

The task is implemented in the following sequential states:

- *Initial*: The simulation is initiated with the USV in a fixed location. The USV remains fixed for the duration of the tasks in the X (surge), Y (sway) and yaw degrees of freedom, but free to move in Z (heave), pitch and roll.
- *Ready*: No change—the USV remains constrained in 2D position and heading.
- *Running*: Scoring begins. A series of simulated RobotX objects (Gazebo models) appear within the field of view of the USV. Teams will locate and identify the objects. Localization and identification is reported via the ROS API described below. For each trial, only one publication per object will be accepted; subsequent publications for that trial will be ignored.
- *Finished*: Scoring ends.

**Table 6: List of 3D objects to be considered in Task 3**

3D object	Identification String	3D object	Identification String
	mb_marker_buoy_black		mb_marker_buoy_white
	mb_marker_buoy_green		mb_round_buoy_black
	mb_marker_buoy_red		mb_round_buoy_orange

#### Scoring:

Scoring is designed to incentivize both correctly identifying and precisely localizing objects.

*Localization error* is defined as the horizontal distance between the location reported by the team and the true location in meters.

During each trial, one or more objects are spawned in the field of view of the USV. For each object in the field of view, an error value is added to the total error for the run:

- If the object is not identified, or incorrectly identified, an error value of 10 m is added to the total error.
- If the object is correctly identified and the localization error is greater than or equal to 2 m, an error of 2 m is added to the total error.
- If the object is correctly identified and the localization error is less than 2 m, the localization error value is added to the total error.

Scoring for the task includes the following:

- Run score is the mean error per object, calculated as total error divided by the total number of objects in the run.
- Task score is the mean of run scores for all runs.
- Task rank is the ordering of task scores from lowest to highest

**Table 7: Landmark localization and characterization API**

Topic Name	Message Type	Description
/vrx/perception/landmark	Geographic_msgs::GeoPoseStamped	Teams report their estimated location (latitude and longitude) of a detected object. The identification of the object is reported using the message header <code>frame_id</code> string (see Table 3 for enumeration of object identifications).

## 4.3. Level 2: Integrating Control and Perception

### 4.3.1. Task 4: Wildlife Encounter and Avoid

**Capability:**

The system should be capable of identifying specific objects of interest in the arena and planning an appropriate action for each object while taking into account its properties and behavior. To test this ability, a bounded portion of the arena will be populated with mobile “animal” markers representing three different types of wildlife: platypus, turtle and crocodile. The system should plan and traverse a path that circles clockwise around platypus markers, counterclockwise around turtle markers, and avoids (i.e. remains at a distance from) crocodile markers.

**Implementation:**

For each run, three animals will be placed in the arena area. Animals may remain at rest or may move during the course of the run. The types of animals and their current spherical coordinates will be published to the /vrx/wildlife/animals ROS topic. At least one animal will be either a turtle or a platypus.

Each run of the task will progress through the following stages:

- *Initial*: The simulation is initiated with three animals in random locations, and the USV in the vicinity of the animals.
- *Ready*: The USV is free to move in all degrees of freedom.
- *Running*: The USV may begin to approach and circle the “encounter” animals (platypus and turtle) while avoiding the crocodile.

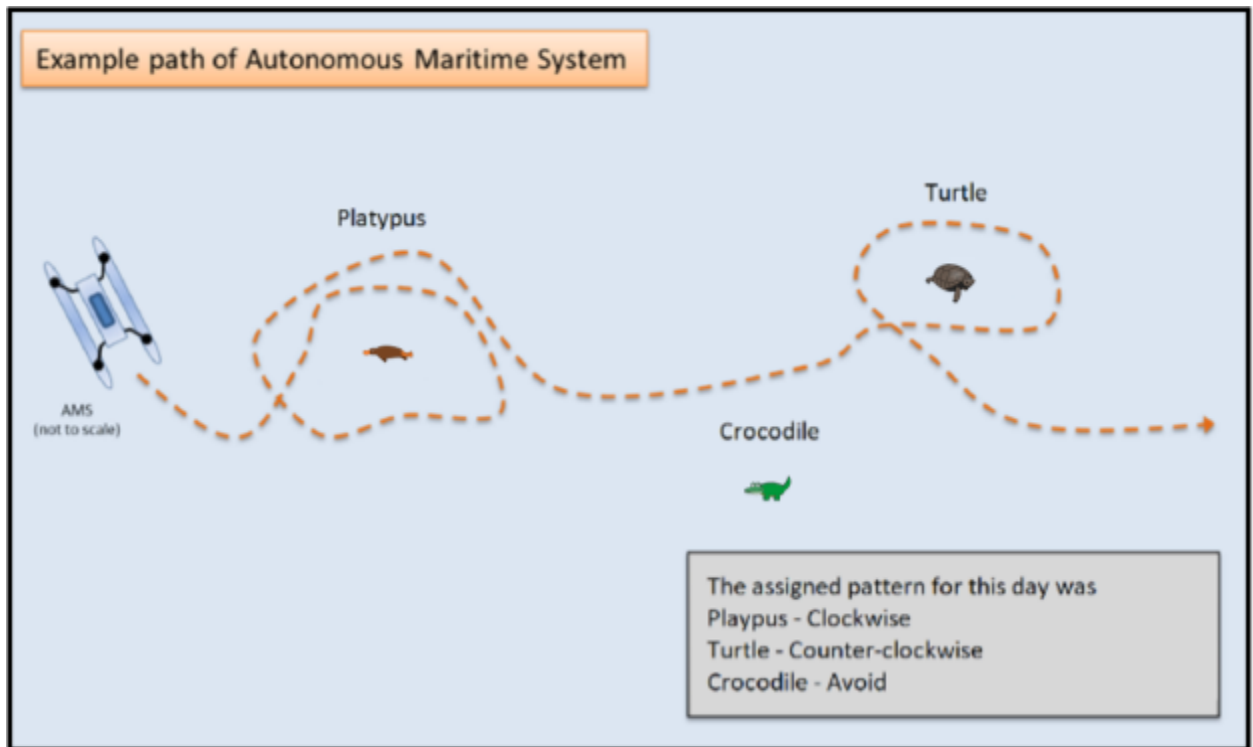


Figure 5: Example of a successful USV path

- The USV is considered to have “circled” an animal if the reference point of the USV traces a path that wraps a full 360 degrees around the animal while remaining within a 10 meter radius of the animal (the path itself does not need to be circular).
- If the USV changes direction (e.g. from clockwise to counter-clockwise) while circling or collides with the animal, any circle in progress is lost and must be restarted.
- To successfully avoid the crocodile, the USV must not pass within a 10 meter radius of it at any time during the run.
- A time bonus of 30s is awarded for each animal successfully circled or avoided.
- The simulation ends when the USV has successfully circled all appropriate animals or the maximum simulation time is exceeded.
- *Finished*: Scoring ends.

#### Scoring:

- Run score is equal to the total runtime, adjusted for any time bonuses earned.

- Task score is the sum of all run scores.
- Task rank is the ordering of task scores from lowest to highest.

API:

Table 8: Wildlife Encounter and Avoid API

Service Name	Message Type	Description
/vrx/wildlife/animals	Geographic_msgs:: GeoPath	An array of animal locations, each containing a position given in spherical (WGS84) coordinates. The identification of the animal is provided in the message header frame_id string. The possible identifiers are: crocodile, platypus, turtle.

#### 4.3.2. Task 5: Channel Navigation, Acoustic Beacon Localization and Obstacle Avoidance

##### Capability:

System should be capable of combining all level 1 tasks to complete a two-part, “gymkhana” challenge. The arena is divided into two areas: the channel and the obstacle field. To complete the first part of the task, teams should create and execute a motion plan to navigate the channel specified by red and green markers. After crossing the channel, the USV enters into the obstacle field made up of black and orange buoys, and begins the second part of the task. Here, the goal is to locate and minimize distance to an underwater black box containing an acoustic pinger without hitting any obstacles. Once the black box is localized, the USV will have to maintain its position as close to the black box as possible (while remaining on the surface of the water) until the end of the task. The solution should be robust with respect to environmental forcing and obstacles within the channel.

##### Channel implementation:

The navigation channel is defined as a series of gates, where each gate is a pair of colored buoys. The entrance gate is a white-red pair, the exit gate is a black-red pair and the other gates are green-red pairs. While the layout of a particular navigation channel will vary, all competition navigation channel runs will satisfy the following constraints:

- The width of a gate, measured as the distance between the two buoys making up the gate, will be between 15-25 m.
- The distance between gates, measured as the distance between the centroid of the two gate markers, will be greater than the maximum of the widths of the two closest gates.

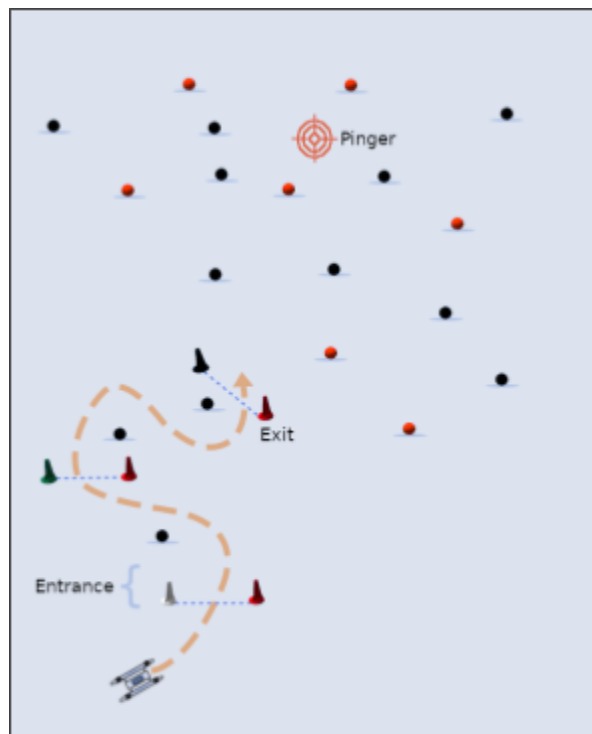


Figure 5: Black Box Gymkhana

- The number of gates in an individual run can be between 2 and 5.

**Obstacle area implementation:**

The obstacle area is an open water zone with an unknown number of black and orange buoys. The black box is located in this area at a maximum depth of 5 meters.

**Combined implementation:**

Each run of the task will progress through the following stages:

- *Initial*: The simulation is initiated with the USV in a random location, in the vicinity of the entrance gate (white-red buoys).
- *Ready*: The USV is free to move in all degrees of freedom.
- *Running*: The USV may begin traversing gates. The USV is considered to have passed through the gate if the reference point of the USV crosses the line connecting the reference points of the two buoys making up that gate. The direction of travel is specified as “Red, Right, Returning”. Gates must be traversed in the appropriate order and direction.
- Note that traversing a gate in the wrong direction or out of order will be considered a failure to traverse the navigation channel and will terminate the task.
- Once the USV crosses the exit gate, it will enter the obstacle area.
- In the obstacle area, the USV should use its acoustic sensor to localize the pinger within the black box.
- The run lasts until the predetermined maximum time.
- *Finished*: Scoring ends.

**Scoring:**

This task’s run score will be calculated according to the same criteria as task 1, except in this task the goal pose is the vertical projection from the black box to the surface of the water, and no goal heading is specified (i.e., the heading error term is dropped and only 2D Euclidean distance is taken into account). In addition, the following rules also apply:

- The USV must cross the navigation channel exit gate to get a score. Otherwise, the score will default to the maximum error.
- Despite the above condition, the calculation of the error distance to the goal still begins as soon as the task enters the *Running* state. This is intended to incentivize speedy navigation of the channel.
- For each collision with an object on the course, a 1 m penalty will be added to the final run score.
- Task score is the mean of the run scores for all runs.
- Task rank is the ordering of task scores from lowest to highest.

## API:

**Table 4: API for Channel Navigation, Acoustic Beacon Localization and Obstacle Avoidance**

Topic Name	Message Type	Description
/wamv/sensors/pingers/pinger/range_bearing	usv_msgs::RangeBearing	A distance (range) and two angles (bearing and elevation) indicating the relative position of the blackbox from the USV, with noise. These values should be used in the solution to locate the blackbox.
/vrx/gymkhana_blackbox/goal	Geographic_msgs::GeoPoseStamped	For debugging purposes only. The goal pose, consisting of a position in spherical (WGS84) coordinates. This topic will not be available to the system in the actual competition.
/vrx/gymkhana_blackbox/pose_error	std_msgs::Float64	A 1 Hz sample of the current 2D pose error metric, which summarizes the current Euclidean distance between the USV and goal.
/vrx/gymkhana_blackbox/mean_pose_error	std_msgs::Float64	The mean pose error accumulated over the duration of the run so far.

Note that there is no task-specific ROS API for the navigation channel portion of the task. However, development and debugging information, which teams may find helpful, is printed to standard output by the `navigation_scoring_plugin` when verbose mode is enabled (e.g., "[Msg] New gate crossed!" or "[Msg] Transited the gate in the wrong direction. Gate invalidated!").

### 4.3.3. Task 6: Scan and Dock and Deliver

#### Capability:

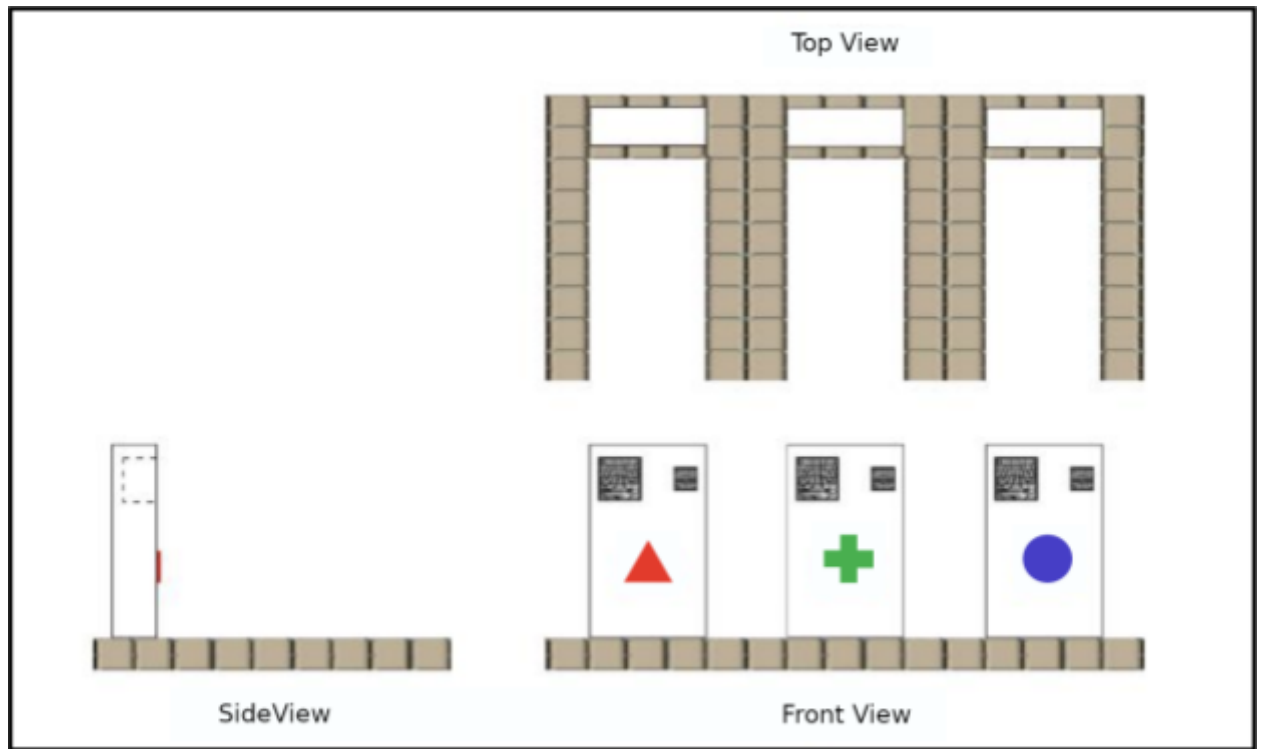
Given multiple docking bays (similar to the arrangement in the RobotX Challenge) the USV should be capable of deciding on the appropriate bay for docking, executing a safe and controlled docking maneuver and then exiting the dock. Additional points will be awarded for vehicles that can successfully propel a projectile through one of the two holes in the placard at the head of the correct docking bay.

#### Implementation:

This tasks make use of virtual models of the “symbols” used in the 2016 and 2018 RobotX competitions<sup>1</sup>. The USV must demonstrate the ability to successfully dock in the bay identified by the placard symbol indicated by the scan-the-code buoy. The symbols may be red, green, or blue in color and may be in the shape of a circle, triangle, or cruciform (cross) on a white background. Additionally, the placards will feature

<sup>1</sup> Task descriptions for both 2016 and 2018 RobotX are available at <https://www.robotx.org>

a pair of square target holes, one small and one large located above the symbol and outlined in black on a white background. The larger hole will be a square 0.5m on a side, and the smaller hole will be a square 0.25m on a side. A maximum simulation time is specified.



**Figure 6: Docking and Delivery Bays**

To learn the target symbol, teams must read the color sequence from the scan-the-code buoy and report this color sequence via the ROS API using the `/vrX/scan_dock/color_sequence` service. The color sequence of the scan-the-code buoy uniquely determines the color and shape of the placard symbol that indicates the correct docking bay using a constant mapping, as follows:

- The first color in the sequence determines the placard symbol color: Red, Blue, Green or Yellow.
- The last color in the sequence determines the placard symbol shape:
  - Red = Circle
  - Green = Triangle
  - Blue = Cruciform/Cross
  - Yellow = Rectangle
- The middle color in the sequence can be any color that does not cause the same color to appear twice in a row in the sequence.

Once the system has correctly perceived the color sequence from the scan-the-code buoy, the USV should attempt to dock in the docking bay with the placard symbol that corresponds to the color sequence of the scan-the-code buoy, according to the mapping given above.



Once docked, the USV will deliver a payload into one of the two holes above the placard symbol using the provided ball shooter. Delivery into the smaller hole will be worth more points (see scoring, below). The ball shooter has a fixed pitch and projectile velocity that can be configured statically before launching the simulation. It will be equipped with four projectiles.

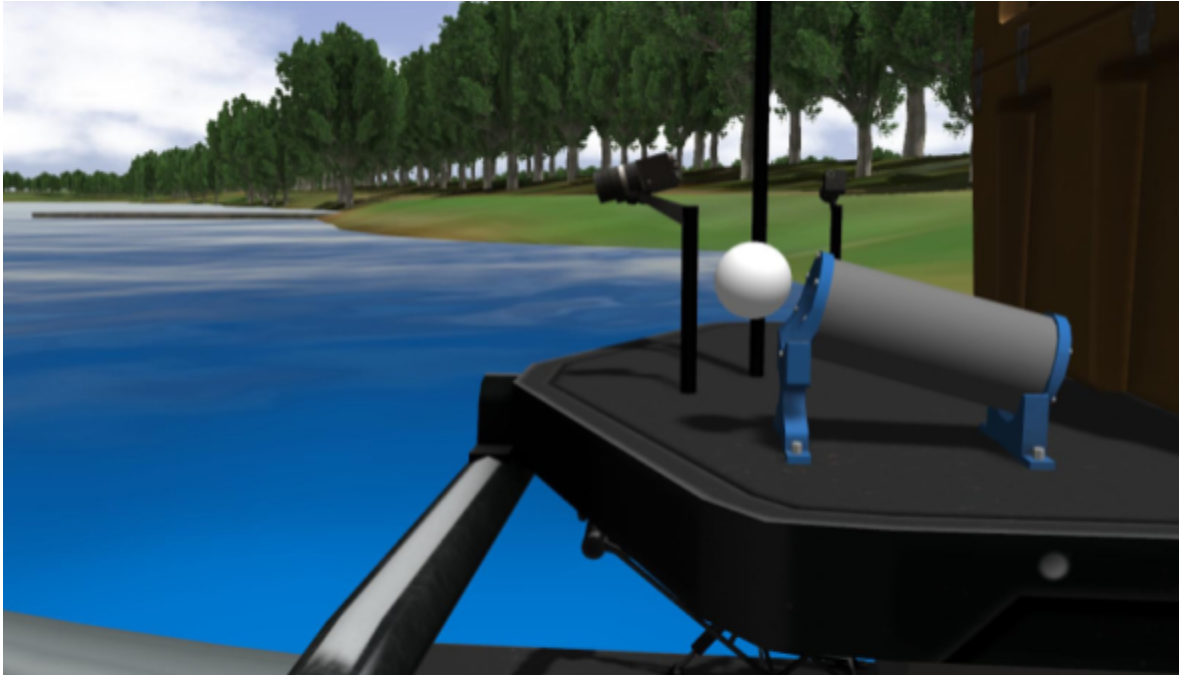


Figure 7: Ball Shooter for “Deliver” Portion of Task

- *Initial:* The simulation is initiated with the USV in a random location in the vicinity of the dock. Each docking bay has an associated placard symbol with a unique color and shape.
- *Ready:* The USV is free to move in all degrees of freedom.
- *Running:* The USV must read the color sequence from the scan-the-code buoy, report the sequence to the `/vrx/scan_dock_deliver/color_sequence` service, and dock in the correct docking bay.
- Before exiting the dock, the USV will use the ball shooter to launch a projectile through the hole in the placard.
- Successful docking consists of having the USV fully enter the dock, stay within the dock area for 10 s and then exit the dock. The USV may only dock once per simulation run.
- The simulation ends when the USV has successfully docked and exited or the maximum simulation time is exceeded.
- *Finished:* Scoring ends.

#### Scoring:

A *docking event* consists of entering an activation zone that delineates the docking bay, staying within the activation zone for 10 s, and exiting the docking bay. The details of this process and how it is evaluated are described in the [Docking Details wiki](#). No more than one docking event is allowed for each simulation run.

- Run score is the sum of the following points:
  - 15 points for a docking event
  - 20 points if that docking event occurs in the correct docking bay, as specified by the placard color and shape
  - 10 points for correctly reporting the color sequence from the scan-the-code buoy.
  - 5 points for each projectile successfully launched through the large hole in the placard.
  - 10 points for each projectile successfully launched through the small hole in the placard.
- Task score is the sum of run scores over all runs.
- Task rank is the ordering of task scores from highest to lowest.
- Ties will be broken in favor of the team with the lowest total simulation time over all runs.

**API:****Table 9: Dock and Scan-and-Dock Task API**

Service Name	Message Type	Description
/vrx/scan_dock_deliver/color_sequence	vrx_gazebo::ColorSequence	The sequence of three colors perceived. Allowed values are “red”, “green”, “blue” and “yellow”.
/wamv/shooters/ball_shooter/fire	std_msgs::Empty	An empty message that causes the ballshooter to fire its projectile.