

# VRX 2023 Task Descriptions

## 1. Introduction

This document describes the individual tasks that make up the Virtual RobotX (VRX) 2023 Competition. Each task description includes a name and objective as well as implementation and scoring details. All tasks are under active development and details may change before the final release of this document. 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 the competition.

In addition to the descriptions below, the VRX project Wiki will provide VRX 2023 Task Tutorials<sup>1</sup> 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. Participants should also review the general task interface and structure as described in the VRX Technical Guide.

## 2. Task Names

During every task the status of the task is published as a ROS 2 [ParamVec message](#) over a ROS 2 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 1: Task Naming

Task	Task Message Name
Station-Keeping	stationkeeping
Wayfinding	wayfinding
Landmark Localization and Characterization	perception
Acoustic Perception	acoustic_perception
Wildlife Encounter and Avoid	wildlife
Follow the Path	gymkhana
Acoustic Tracking	acoustic_tracking
Scan and Dock and Deliver	scan_dock_deliver

## 3. Level 1: Control and Perception Fundamentals

### 3.1. 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:

---

<sup>1</sup> Coming soon.

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

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/stationkeeping/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/stationkeeping/pose_error` ROS topic.
  - The current mean pose error is published to the `/vrx/stationkeeping/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.

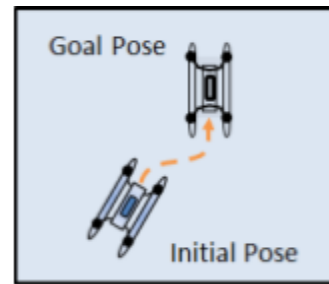


Figure 1: Stationkeeping Task

Table 2: Station Keeping API

Topic Name	Message Type	Description
/vrx/stationkeeping/goal	geometry_msgs::msg::PoseStamped	The goal pose, consisting of a position in spherical (WGS84) coordinates and a heading, given as a quaternion.
/vrx/stationkeeping/pose_error	std_msgs::msg::Float32	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/stationkeeping/mean_pose_error	std_msgs::msg::Float32	The mean pose error accumulated over the duration of the run so far.

### 3.2. 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 runs 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 2 topic.
- *Running*: Scoring begins. For the purposes of development and debugging, the following items are provided as ROS 2 topics:
  - The minimum error achieved so far for each waypoint is published to the /vrx/wayfinding/min\_errors ROS 2 topic.
  - The mean of the current minimum errors for each waypoint is published to the /vrx/wayfinding/mean\_error ROS 2 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 3: Wayfinding API**

Topic Name	Message Type	Description
/vrx/wayfinding/waypoints	geometry_msgs::msg::PoseArray	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	ros_gz_interfaces::msg::Float32Array	An array containing the minimum 2D pose error so far achieved between the USV and each waypoint.
/vrx/wayfinding/mean_error	std_msgs::msg::Float32	The mean of the minimum errors so far achieved for all waypoints.

### 3.3. 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.).

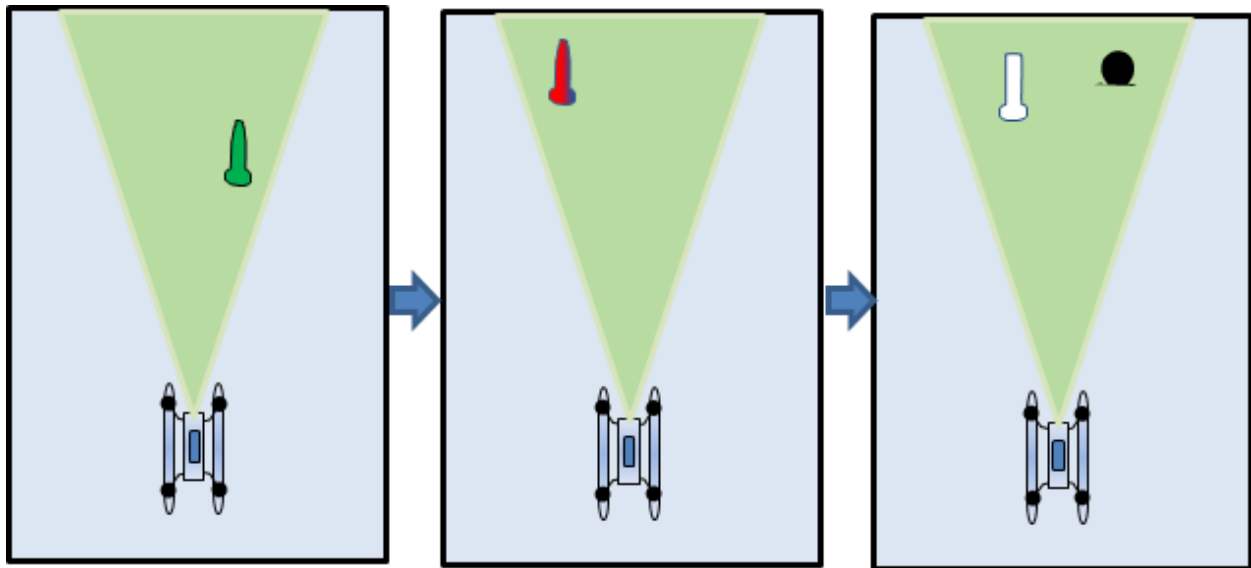


Figure 2: Three individual cases during the task as markers and objects are sequentially added to the field of view.

#### Implementation:


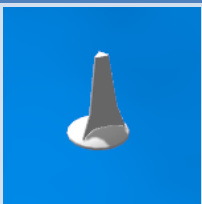
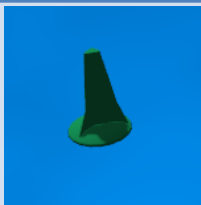
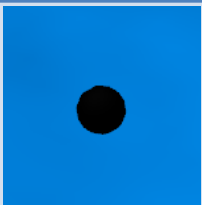

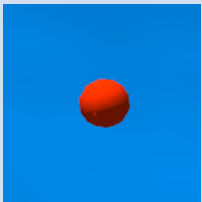
This task is made up of multiple runs, each under different environmental conditions. A single run is made up

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 4: 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 5: Landmark localization and characterization API**

Topic Name	Message Type	Description
/vrx/perception/landmark	geometry_msgs::msg::PoseStamped	Teams report their estimated location (latitude and longitude) of a detected object. The identification of the object is reported using the message header frame_id string (see Table 4 for enumeration of object identifications).

### 3.4. Task 4: Acoustic Perception

**Capability:**

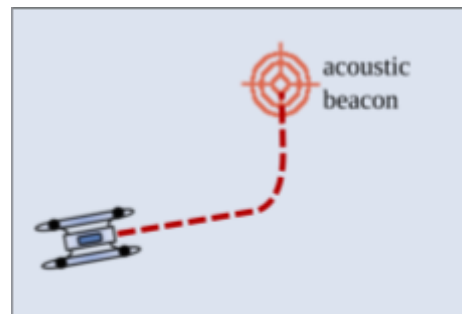
System should be capable of command and control of USV to achieve a goal state marked by an underwater acoustic beacon.

**Implementation:**

An underwater acoustic beacon broadcasts range, bearing and elevation indicating its position relative to the USV, with noise. Teams should navigate to the beacon as quickly as possible. The task ends when the distance between the USV and the beacon falls below 1m. The overall performance for a given run is determined by the total time required to complete the task.

The task is implemented in the following sequential states:

- *Initial:* The simulation is initialized with the USV and acoustic beacon in random locations within the operating area.



**Figure 3: Acoustic Perception Task**

- *Ready*: The USV is free to move in all degrees of freedom. The acoustic beacon begins to advertise its location using the `/wamv/pingers/pinger/range_bearing` ROS 2 topic.
- *Running*: Scoring begins.
- *Finished*: Scoring ends.

**Scoring:**

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

**Table 6: Acoustic Perception API**

Topic Name	Message Type	Description
<code>/wamv/pingers/pinger/range_bearing</code>	<code>ros_gz_interfaces::msg::ParamVec</code>	A distance (range) and two angles (bearing and elevation) indicating the relative position of the beacon from the USV, with noise.

## 4. Level 2: Integrating Control and Perception

### 4.1. Task 5: 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 2 topic. At least one animal will be either a turtle or a platypus.

The rate at which coordinates are published will remain fixed during a single run, but may vary from one run to another. For any given run, the rate will be set at a value between 1 Hz and 0.1 Hz. The purpose of introducing lower publication rates is to encourage teams to use perception capabilities in conjunction with the published coordinates to localize the animals.

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.

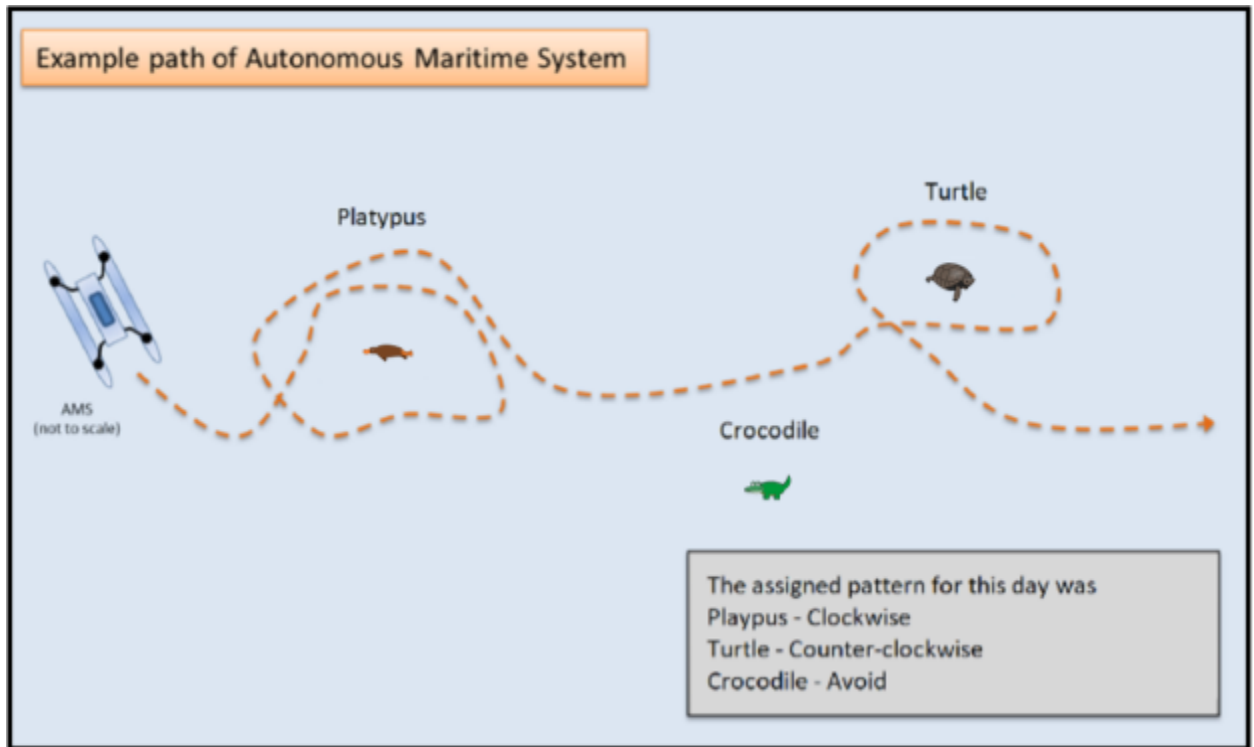


Figure 4: Example of a successful USV path

- 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 7: Wildlife Encounter and Avoid API**

Service Name	Message Type	Description
/vrx/wildlife/animal<#>/poses <sup>2</sup>	geometry_msgs::msg::PoseStamped	An animal name and 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.

**4.2. Task 6: Follow the Path**

**Capability:**

System should be capable of creating and executing a motion plan to traverse a navigation channel specified by red and green markers. The solution should be robust with respect to environmental forcing and obstacles within the 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 blue-red pair and the other gates are green-red pairs. Obstacles may be included within and around the navigation channel.

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.
- The number of gates in an individual run can be between 2 and 10.

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



**Figure 5: Follow the Path**

<sup>2</sup> A separate topic will be published for each of the three animals, i.e. /vrx/wildlife/animal0/poses, /vrx/wildlife/animal1/poses, /vrx/wildlife/animal2/poses.

- *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 runtime clock starts and the USV may begin traversing gates. Performance will be assessed as follows:
  - The USV is considered to have passed through a 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 direction.
  - Scoring begins when the USV crosses the start gate in the correct direction; the entrance gate must be traversed first before any other gates will be counted.
  - If a gate is crossed in the wrong direction no points are awarded and the incorrectly crossed gate is out of play for the rest of the run (i.e. it is no longer possible to score points by crossing it in the correct direction).
  - A team may miss one or more gates and still receive points for traversing gates that come later in the course; however, once a gate is crossed (correctly or incorrectly), all preceding gates are out of play for the rest of the run.
  - The run lasts until the predetermined maximum time or until the USV passes through both the entrance and the exit gates.
- The maximum run time for each run is 300s.
- *Finished*: Scoring ends and elapsed time for the run is recorded.

#### Scoring:

- Run score is calculated as follows:
  - 10 points are awarded for each gate crossed correctly.
  - 1 bonus point is awarded for crossing a gate correctly if the immediately preceding gate was also crossed correctly.
  - 1 bonus point is awarded for correctly crossing the finish gate.
  - 3 points are deducted for each collision (however, the score cannot fall below 0).
- Task score is the sum of run scores over all runs.
- Task rank is the ordering of task scores from lowest to highest.
- Ties will be broken in favor of the team with the lowest total simulation time over all runs.

#### API:

There is no task-specific ROS 2 API for this 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. Task 7: Acoustic Tracking

**Capability:**

System should be capable of tracking a moving underwater acoustic beacon while avoiding obstacles.

**Implementation:**

The obstacle area is an open water zone with an unknown number of black and orange buoys. The acoustic beacon moves through this area at a maximum depth of 5 meters.

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

- *Initial:* The simulation is initiated with the USV and acoustic beacon in random locations within the operating area.
- *Ready:* The USV is free to move in all degrees of freedom. The acoustic beacon begins to advertise its location using the `/wamv/pingers/pinger/range_bearing` ROS 2 topic.
- *Running:* The USV should use its acoustic sensor to localize the acoustic beacon and minimize the distance between the beacon and USV.
- The run lasts until the predetermined maximum time.
- *Finished:* Scoring ends.

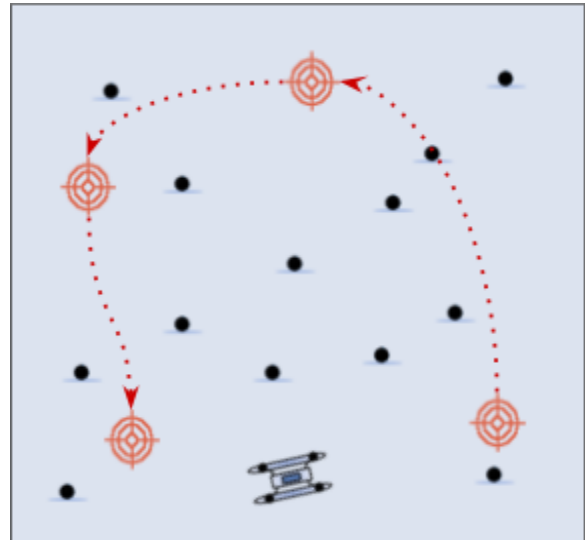


Figure 6: Acoustic Beacon Moving Through Obstacle Field

**Scoring:**

- Run score will be calculated according to the same criteria as task 1, with the following modifications:
  - the goal pose is the vertical projection from the acoustic beacon to the surface of the water (of course, since the beacon moves, so will the goal pose),
  - no goal heading is specified (i.e., the heading error term is dropped and only 2D Euclidean distance is taken into account),
  - and 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 8: API for Channel Navigation, Acoustic Beacon Localization and Obstacle Avoidance**

Topic Name	Message Type	Description
/wamv/pingers/pinger/ range_bearing	ros_gz_interfaces::msg:: ParamVec	A distance (range) and two angles (bearing and elevation) indicating the relative position of the beacon from the USV, with noise. These values should be used in the solution to locate the beacon.
/vrx/acoustic_tracking/ pose_error	std_msgs::msg::Float32	A 1 Hz sample of the current 2D pose error metric, which summarizes the current Euclidean distance between the USV and goal.
/vrx/acoustic_tracking/ mean_pose_error	std_msgs::msg::Float32	The mean pose error accumulated over the duration of the run so far.

**4.4. Task 8: 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 task makes use of virtual models of the “symbols” used in the 2016 and 2018 RobotX competitions<sup>3</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 (+) on a white background. Additionally, the placards will feature 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. 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 2 API using the /vrx/scan\_dock\_deliver/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

<sup>3</sup> Task descriptions for both 2016 and 2018 RobotX are available at <https://www.robotx.org>

- 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.

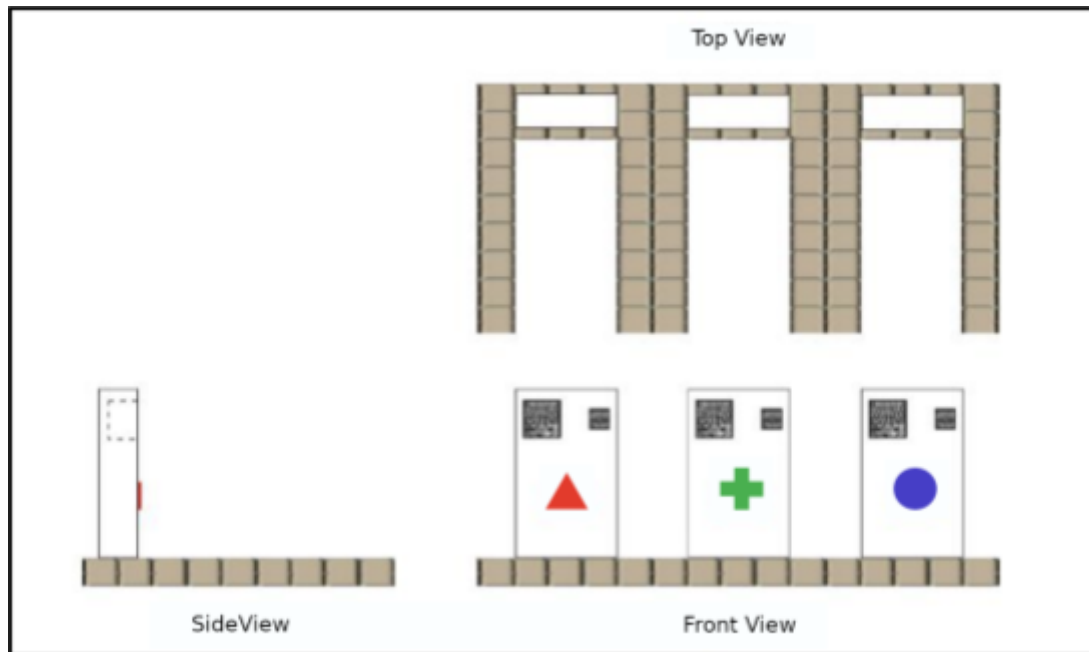


Figure 7: Docking and Delivery Bays

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.

- *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.

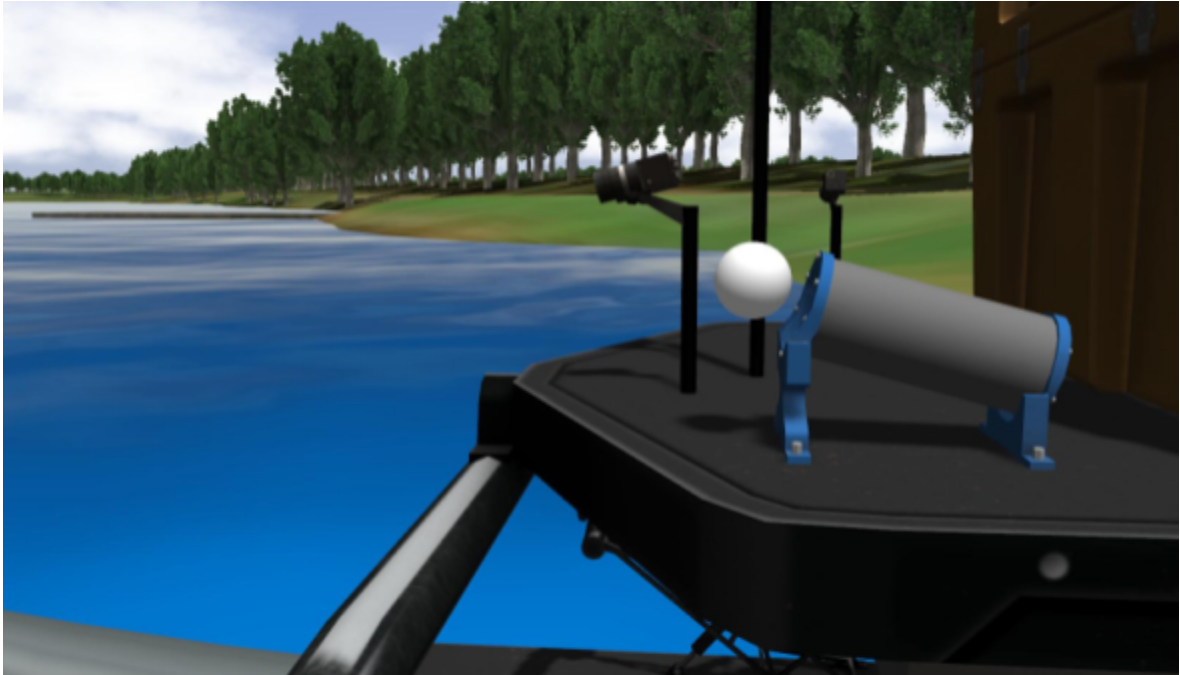


Figure 8: Ball Shooter for “Deliver” Portion of Task

**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	ros_gz_interfaces::msg:: StringVec	The sequence of three colors perceived. Allowed values are "red", "green", "blue" and "yellow".
/wamv/shooters/ball_shooter/ fire	std_msgs::msg:: Bool	An boolean message that causes the ballshooter to fire its projectile when it receives a value of "false."