AVL Device Registry Documentation

What is the Device Registry?

The Device Registry Class serves as the primary interface for interacting with devices (sensors and actuators.) The Device Registry utilizes a variety of sensor memory cells, each of which is frequently evaluated through a FixedUpdate() method (inherited from the MonoBehaviour base class) to collect their own unique environmental data, allowing further automation within the robot. These devices can be controlled by writing collected data into simulated memory cells that take the form of arrays of floating-point numbers. It is NOT possible to command or control the object in any way without going through the Device Registry.

How is the Device Registry passed to each Task?

As the robot moves within its environment, the various sensors are continuously collecting data about the robot's relationship to and findings within the environment. This is accomplished with the FixedUpdate() method within the RTOS Class. For this particular project, FixedUpdate() is called every 0.125 seconds. The RTOS Class starts pulling tasks from a list of tasks in the Task List Class. The Task List Class inherits from the Task Interface Class, which utilizes the Execute() function with one argument of Device Registry type. These tasks are then executed and passed to the Device Registry, where it will read the sensors and actuators in order to interact with the environment and store the collected readings in simulated sensor memory cells.

What are the Sensor Memory Cells?

The sensor memory cells are contained within the Device Registry Class. These memory cells correspond to the sensors collecting data within the FixedUpdate() method, and store the collected readings in an array of floating-point numbers. These serve as the simulated memory cells that one can interact with to collect data and command the robot.

The Sensor Memory Cells are as follows:

**All are float type*

GPS: Provides the position of the robot in the environment.

Shape is: [2]; [0]: Latitude, [1]: Longitude

Lidar: Provides the readings from 16 lidar sensors on the robot. The sensors store the distance to any object that intersect their individual paths in an associated memory cell 0 – 15. If there is no object within the sensor's range, the maximum range value will be stored instead.

Shape is: [16]; [0]: Measured Distance for Lidar Sensor 0 … [15]: Measure Distance for Lidar Sensor 15

Pixels: Collects the RGB color readings from the pixels of the color camera on the robot. The first-dimension lists height, second lists width, and the third dimension lists the color channel (red is 0, green is 1, blue is 2.)

Shape is: [7, 15, 3]; [0, 0, 0]: Contains the red value at pixel (0,0) - top left of the camera,

[7, 15, 2]: Contains the blue value at pixel (7,15) - bottom right of the camera

Compass: Provides the difference angle between the direction the robot is facing and the direction of North. Possible values are between (-180, 180), where 0 is aligned with North, -90 is pointing West, and 90 is pointing East.

Shape is [1]; [0]: Difference Angle

Target Alignment: Gives the difference angle between the direction the robot is facing and the direction to a target position in the environment. Possible values are between (-180, 180), where 0 means the robot is facing directly at the object, -90 means the robot is perpendicular and pointing left, and 90 means perpendicular and pointing right.

Shape is [1]; [0]: Difference Angle

Speedometer: Gives the current speed of the robot in m/s (meters per second).

Shape is [1]; [0]: Speed