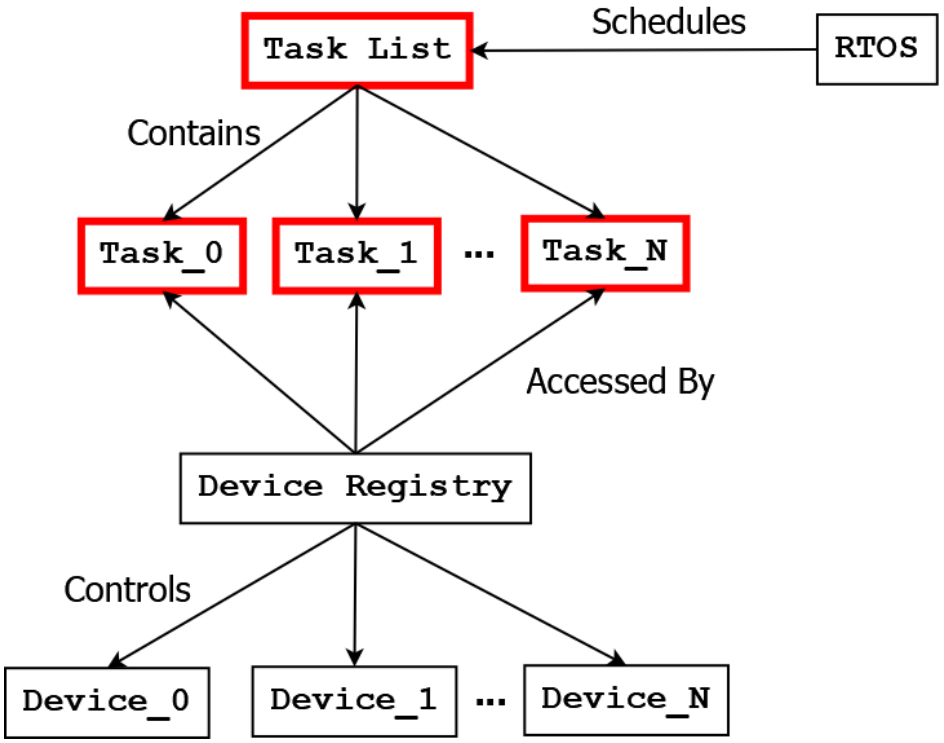
Creating Tasks



Abstract

Tasks serve as the primary means of decision making and information processing for the Ego vehicle. While the RTOS is in charge of determining which task is run each frame based on a scheduling algorithm, the RTOS does not do any “thinking” about the environment, nor does the RTOS issue commands to the devices on the Ego vehicle. The responsibility for higher level planning and the execution of desired actions falls entirely with the tasks.

All classes related to Tasks and the Task List are located in the folder:

- Unity-AVL/Assets/Scripts/Tasks/

When you create code for your own task classes, you should place them in this folder as well.

Task List

The Task List is a data structure which contains all the tasks that will run on the RTOS. In order for a task to be registered for execution, an instance of the task's class must be instantiated within the "TaskInterface[] tasks" class variable. Tasks are executed according to Round-Robin scheduling, and the order they are instantiated into the class variable determines the order which they are executed.

Let us assume there are two student-created task classes called Task1 and Task2. These tasks can be added into the Task List as shown below:

```
public class TaskList : MonoBehaviour
{
    protected TaskInterface[] tasks = new TaskInterface[] {
        new Task1(),
        new Task2()
    };

    ...
}
```

Thus during the Round-Robin scheduling algorithm, the RTOS will execute Task1 in the first frame, and then it will execute Task2 in the second frame, then Task1 in the third frame, Task 2 in the fourth frame, and so on. Once a task has been added to the task list, there is no way at this time to remove it.

Task Interface

The Task Interface is an interface data structure that all student-created task classes must implement. By implementing this interface, the different classes that students define for their tasks can be instantiated within the single "TaskInterface[] tasks" variable in the Task List class. Because the Task Interface is a C# interface and not a class, it will not interfere with any class inheritance that students may wish to implement.

The definition of the TaskInterface is:

```
public interface TaskInterface
{
    void Execute(DeviceRegistry devices);
}
```

Thus the Task Interface requires only a single method to be defined by classes that implement it. This method must be public, it must have returned type void, it must be titled Execute, and it must receive as input a single argument of type DeviceRegistry.

Writing a New Task

Defining a new student-created task is very straightforward, and an example class definition is given below:

```
public class Task1 : TaskInterface
{
    public void Execute(DeviceRegistry devices) {
        // Print a message to the command line
        Debug.Log($"Executing example task.");
    }
}
```

Task class files have a ".cs" file ending. Task class definition files should be placed in the same folder as the Task List and Task Interface, the location of which was given in the Abstract section. You should not add any namespace definitions to your class files.

Once a task has been defined, you will also need to instantiate an instance of the task's class in the Task List, as was shown previously.