91.548 Robot Design
Spring 2015
Prof. Yanco

Assignment 2: ROS Race

Out: Tuesday, 24 February 2015
Due: Tuesday, 10 March 2015, with files submitted before class

Preparation:

You should do the tutorials at ros.org before this race!

Dependencies (in addition to ros hydro or indigo):

```
sudo apt-get install python-wstool git-core
```

In your homedir, create a directory with an "src" folder inside it.

```
mkdir -p race_ws/src; cd race_ws/src
```

```
rosws init
rosws merge /opt/ros/$ROS_DISTRO
RI=https://raw.githubusercontent.com/uml-robotics/uml_race_ws/master/workspace.rosinstall
rosws merge $RI
rosws up
source setup.bash
rosmake uml_race
```

In the src folder of the workspace you've created, create a rosmake package (roscreate-pkg) or catkin package (roscreate-catkin-pkg) to contain the node you will be writing.
Name it something containing your name.

```
[appropriate create-pkg cmd] [your pkg name] roscpp rospy geometry_msgs sensor_msgs std_msgs
```

For this assignment, you can use either python or C++, but either requires dependencies:
geometry_msgs ***sensor_msgs*** ***std_msgs***

If you use C++, don't forget to rosmake or catkin_make between test+edit iterations

---

If you are using **rosmake**, you will need to source the *setup.bash* in your src folder in each new terminal to be able to run the racetrack and your driving node.

If you are using **catkin**, you need to do a few additional setup steps, then source a different setup.bash.

```
cd ~/race_ws/src
catkin_init_workspace
cd ..
catkin_make # catkin_make must be run from the directory containing the src folder
```

In each new terminal, you need to source *~/race_ws/devel/setup.bash* to use your **catkin workspace.**

What to do:

Write a control program to complete the race track simulation provided in the uml_race package. Your robot should travel no faster than 5m/s. This will be enforced by a referee node that is watching you when the robot starts. The starting location of the robot may be anywhere within a 4x10m bounding box around the start location so do not hard code the turns.

The robot has a 5 meter laser scanner that can see 180 degrees in front of it. The output of this topic is published to **/robot/base_scan** and messages are typed ***sensor_msgs/LaserScan***.

The robot listens to commands on **/robot/cmd_velocity**. Messages are of type ***geometry_msgs/Twist***. Twists are composed of an angular Vecto3 and a linear Vector3. Forward velocity is stored in ***linear.x*** while rotational "yaw" velocity is stored in ***angular.z*** .

To run the racetrack and referee: roslaunch uml_race racetrack.launch

In the stage window you may see the output of the laser by selecting View > Data. Alternatively, you could *rosrun rviz rviz*, and add a LaserScan display with its topic set appropriately.

It turns out that msg.ranges[0] is the right-hand laser reading. Use msg.ranges[179] to get the left-side reading. You can use any/all of the readings as you see fit to complete the task.

After launching the racetrack, start your node with rosrun in a separate terminal. The racetrack will kill itself after the robot reaches the finish line.

Completion of the race without collision is sufficient. However, if you are inspired by competition, Eric McCann's node runs the track in 49-50 seconds.

How to Get Help:

Email emcann@cs.uml.edu if you need assistance on this assignment.

Eric will also be in 302 on Tuesday 3/3 from 6-7:30pm to provide help to anyone who needs it.

What To Turn In:

A zip file containing ONLY your race controller package folder.
(from your src folder: *zip –r yourname_race.zip your_race_pkg*)

Copy the zip to your homedir on mercury.cs.uml.edu, then submit it with:

```
    submit emccann race yourname_race.zip
```