3 Creating the RazBot Control Configuration

Previous: 2 Expanding RazBot Description for Simulation
Next: 4 Bringing Up RazBot Simulator

Create razbot_control package, used for storing any environment-independent (live or simulation) components and configuration, such as ros_control. You can find the output for this stage of the tutorials in https://github.com/clearpathrobotics/razbot_tutorials/tree/add-control.

1. Create the skeleton for the package:

```
$ catkin_create_pkg razbot_control roslaunch
$ mkdir razbot_control/config razbot_control/launch
```

2. Modify razbot_control/package.xml. This file contains dependencies and other relevant information about our package:

```
<?xml version="1.0"?>
<package>
  <name>razbot_control</name>
  <version>0.0.1</version>
  <description>RazBot controller configuration, contains environment-independent
configuration</description>

  <maintainer email="pbovbel@clearpathrobotics.com">Paul Bovbel</maintainer>
  <license>BSD</license>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roslaunch</build_depend>
  <run_depend>interactive_marker_twist_server</run_depend>
  <run_depend>diff_drive_controller</run_depend>
  <run_depend>joint_state_controller</run_depend>
</package>
```

3. Modify razbot_control/CMakeLists.txt. This file tells catkin, the ROS build system, how to build our package:

```
cmake_minimum_required(VERSION 2.8.3)
project(razbot_control)

find_package(catkin REQUIRED COMPONENTS roslaunch)
catkin_package()

roslaunch_add_file_check(launch)

install(DIRECTORY config launch
  DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}
)
```

4. Create razbot_control/config/control.yaml.

```
# This file contains configuration for the RazBot controllers. These should be
# the same for simulated and live robots.

# The joint state controller handles publishing transforms for any moving joints
razbot_joint_state_controller:
  type: "joint_state_controller/JointStateController"
  publish_rate: 50

# The diff drive controller can control a Differential Drive or Skid Steer
vehicle
# with any arbitrary number of wheels.
razbot_diff_drive_controller:
  type: "diff_drive_controller/DiffDriveController"
  publish_rate: 50
  left_wheel: ['front_left_wheel', 'rear_left_wheel']
  right_wheel: ['front_right_wheel', 'rear_right_wheel']

  # Odometry covariances for the encoder output of the robot. These values should
  # be tuned to your robot's sample odometry data, but these values are a good
place
  # to start
  pose_covariance_diagonal: [0.001, 0.001, 0.001, 0.001, 0.001, 0.03]
  twist_covariance_diagonal: [0.001, 0.001, 0.001, 0.001, 0.001, 0.03]

  # Top level frame (link) of the robot description
  base_frame_id: base_footprint

  # Velocity and acceleration limits for the robot
  linear:
    x:
      has_velocity_limits    : true
      max_velocity           : 0.2   # m/s
      has_acceleration_limits: true
      max_acceleration       : 0.6   # m/s^2
  angular:
    z:
      has_velocity_limits    : true
      max_velocity           : 0.4   # rad/s
      has_acceleration_limits: true
      max_acceleration       : 1.2   # rad/s^2
```

5. Create razbot_control/launch/control.launch:

```
<?xml version="1.0"?>
<launch>

  <!-- This launch file is where any environment-agnostic (live or simulation)
  components should be loaded, including controllers -->
  <rosparam command="load" file="$(find husky_control)/config/control.yaml" />

  <!-- This node loads the two controllers into a controller manager (real or
simulated). The
  controllers are defined in config/control.yaml -->
  <node name="razbot_controller_spawner" pkg="controller_manager" type="spawner"
    args="razbot_joint_state_controller razbot_diff_drive_controller
--shutdown-timeout 3"/>

</launch>
```