

# **RBE 500 Group Assignment #1**

**Joshua Gross, Arjan Gupta, Melissa Kelly**

## Problem 1

### Create ROS Package for PD Controller

#### Preliminary Work with Gazebo

Before creating the ROS package for the PD controller we performed some reconnaissance.

When we executed `ros2 topic list`, we saw that the `/forward_effort_controller/commands` topic was part of it. We then executed the following command in an attempt to move the joints.

```
1      ros2 topic pub --once /forward_effort_controller/commands ...  
      std_msgs/msg/Float64MultiArray "{data: [1, 1, 1]}"
```

However, this took no effect. We remember from the last group assignment (Part 1) that we executed a very similar command to make the joints move to a specific position, except in that case the topic we published to was `/forward_position_controller/commands`. This gave us a hint to the fact that Gazebo was preferring the position controller over the effort controller. Upon discovering the `controller_switch.cpp` file in the `rrbot` simulation files, we realized that we must ‘activate’ the effort controller in order to use it. Next, we did some more discovery using terminal commands. We executed

```
1      ros2 service list -t
```

This helped us see that `/controller_manager/switch_controller` was an available service we could use. Since we supplied the `-t` flag to this command, we could also see that `controller_manager_msgs/srv/SwitchController` was the type of message we had to use. To further take note of the message type, we executed

```
1      ros2 interface show controller_manager_msgs/srv/SwitchController
```

Here we could see that `activate_controllers` and `deactivate_controllers` were parameters of the message. Now we put together our findings and executed the following command that we constructed.

```
1      ros2 service call /controller_manager/switch_controller ...  
      controller_manager_msgs/srv/SwitchController "{activate_controllers: ...  
      [\"forward_effort_controller\"], deactivate_controllers: ...  
      [forward_position_controller, forward_velocity_controller]}"
```

Upon doing this, we saw the the prismatic joint drop. This means something took effect. We now tried our initial command,

```
1      ros2 topic pub --once /forward_effort_controller/commands ...  
      std_msgs/msg/Float64MultiArray "{data: [1, 1, 1]}"
```

Now we saw the robot in Gazebo move! The two revolute joints ‘spun’ in an anti-clockwise direction in the XY plane, whereas the prismatic joint did not do anything. With some more experimentation we realized:

- Acceleration due to gravity, approximated in magnitude to  $9.8m/s^2$  was acting on the prismatic joint. Therefore, in order to make it move upward we needed to publish an effort with an acceleration of anything less than  $-9.8m/s^2$ , and in order to hold the prismatic joint in that state, we needed to apply

exactly  $-9.8m/s^2$  acceleration. Since our joint mass is simply 1, we were able to publish the joint effort as simply  $-9.8$ , for example.