Package overview:

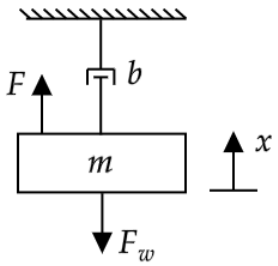
- Inside `catkin_ws/src`, the main package is `scara_robot`. It does not directly contain any nodes or launch files, but is a way to organize all of the other nodes.
  - New package:
    - The `pd_controller` package implements a proportional and derivative controller for joint 3 (prismatic joint). The controller functions by reading the current joint position, calculating the necessary input into the joint, and applying the input force using the `gazebo/apply_joint_effort` topic.
  - Old packages (from PA #1):
    - The `scara_gazebo` package includes the launch files for the gazebo world.
    - The `scara_description` package includes the URDF files for the robot as well as the rviz launch files.
    - The `gazebo_publish` package includes the launch file to allow for the joint states to be published from gazebo.
    - The `scara_forward_kinematics` folder is the pub/sub package that subscribes to the joint states, calculates the forward kinematics, and publishes the pose.
    - The `scara_inverse_kinematics` folder is the service/client package that ingests a desired end effector pose and returns the joint position.

1. Fix all of the joints except the last joint by changing the joint type field of the corresponding joints to "fixed" in the robot description file.

2. Write a position controller node.

   - Get positions from Gazebo and be able to send joint efforts.
   - Design PD controller (tune gains, don't calculate)
   - Implement service that takes in a reference (desired) position for the last joint.
   - Record both the reference position and current position in a text file. Plot the comparison in MATLAB.

   We need to start by determining the E.O.M. of our third link for our controller:



$$\sum F = ma \Rightarrow F - b\dot{x} - F_w = m\ddot{x}$$
$$m\ddot{x} + b\dot{x} + F_w = F$$
$$H(s) = \frac{X(s)}{F(s)} = \frac{1}{ms^2 + bs + F_w}$$

   Steps to run:

   (a) `catkin_make`
   (b) `source devel/setup.bash`
   (c) `roslaunch scara_gazebo scara_world.launch`