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 `scara_pd_controller` package implements a proportional and derivative controller for joint 3 (prismatic joint). The controller functions by reading the current joint position using the `gazebo/get_joint_properties` service, calculating the necessary input into the joint, and applying the input force using the `gazebo/apply_joint_effort` service.
  - Old packages used (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.

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.

   The following changes were made to the XACRO file:

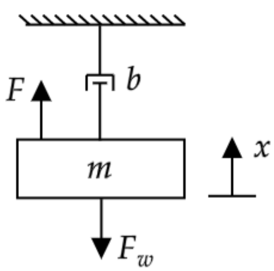
   ...

   ...

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.

   The E.O.M. of our third link for our controller is below:



$$\sum F = ma \Rightarrow F - b\dot{x} - F_w = m\ddot{x}$$

$$m\ddot{x} + b\dot{x} + F_w = F$$

$$H(s) = \frac{X(s)}{F(s)} = \frac{1}{ms^2 + bs + F_w}$$

   Our controller (PD) will have the following form:

$$F(s) = k_p + k_d s \Rightarrow f(t) = k_p E + k_d \frac{dx}{dt}$$

   where $E = d_{3,\text{desired}} - d_{3,\text{current}}$, $dx = d_{3,\text{current}} - d_{3,\text{old}}$, and dt is the time step of the controller loop. k_p and k_d are the gains for the proportional and derivative terms, respectively.

   The main file for the controller exists in the `scara_pd_controller` package under the `pd_control.py` file.

   The control values are calculated within the `pd_control` function:

```
def pd_control(joint, pos_cur, pos_old, pos_des, kp, kd):

        p_err = pos_des - pos_cur
        d_err = (pos_old - pos_cur)*rate
        f = -(kp*p_err + kd*d_err)

        if debug == True:
        print("\np_err = %f,  d_err = %f" % (p_err, d_err))
```

```
        print("\npos_cur = %f, pos_old = %f" % (pos_cur, pos_old))
        print("\nSending joint force f = [%f]" % (f)) # printing calculated values to terminal

        je_service = rospy.ServiceProxy('/gazebo/apply_joint_effort', ApplyJointEffort)
        zero_time = rospy.Time()
        tick = rospy.Duration(0, int((1/rate)*10**9))
        je_service(joint, f, zero_time, tick)

        return f
```

The joint positions are obtained in the following function:

```
def request_joint_status(joint):
        global d3
        global d3_old

        joint_stauts = rospy.ServiceProxy('/gazebo/get_joint_properties', GetJointProperties)
        resp = joint_stauts(joint)

        d3 = -resp.position[0]

        if debug == True:
        print("\n\nReceived joint position: [%f] (d3) (meters)" % (d3)) # printing received data to terminal

        pd_control('joint5', d3, d3_old, d3_des, kp, kd)

        d3_old = d3

        return resp
```

The reference position is set with the following function

```
def service_handle(data):
        global d3_des

        d3_des = data.d3_des

        if debug == True:
        print("\nReceived reference position d3 = %f]" % (d3_des)) # printing converted values to terminal

        if d3_des >= 0 or d3_des <=1:
        success = True
        else:
        success = False

        return success
```

The results from MATALB are shown below:

...

...

...

Steps to run:

(a) `catkin_make`

(b) `source devel/setup.bash`

(c) `roslaunch scara_gazebo scara_world.launch`

(d) In a new window, `rosrun scara_pd_controller pd_control.py`. The controller will begin controlling the joint to its home position ($d_3 = 0$).

(e) In a new window, `rosservice call /scara/JointControlReference "d3_des:  X.XX"` where X.XX is any number between 0 and 1 (joint limits).