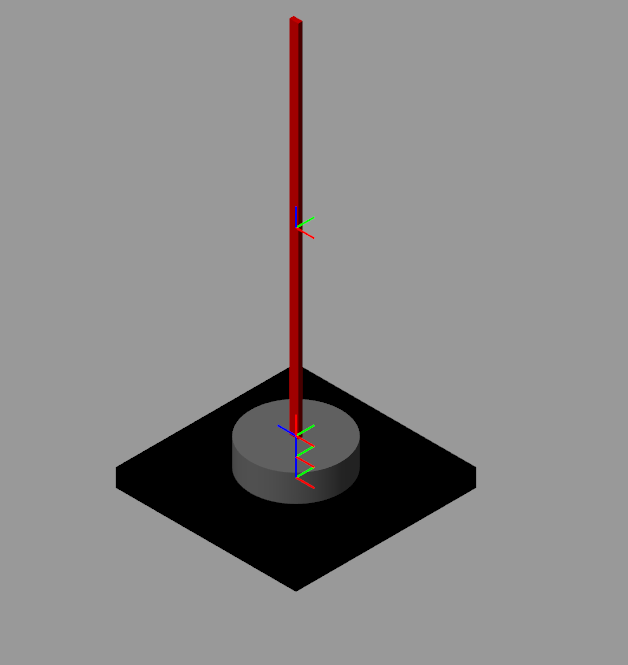
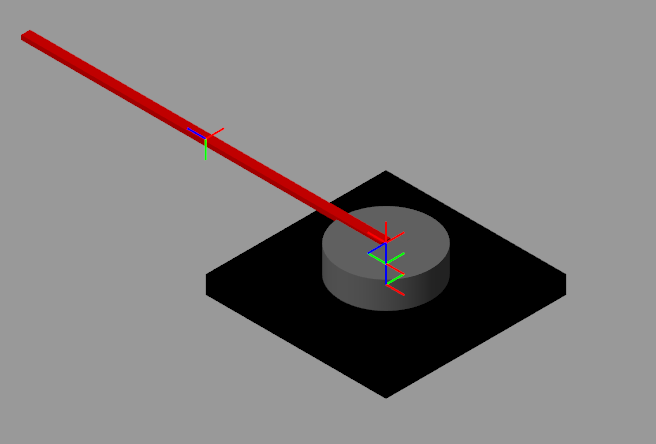
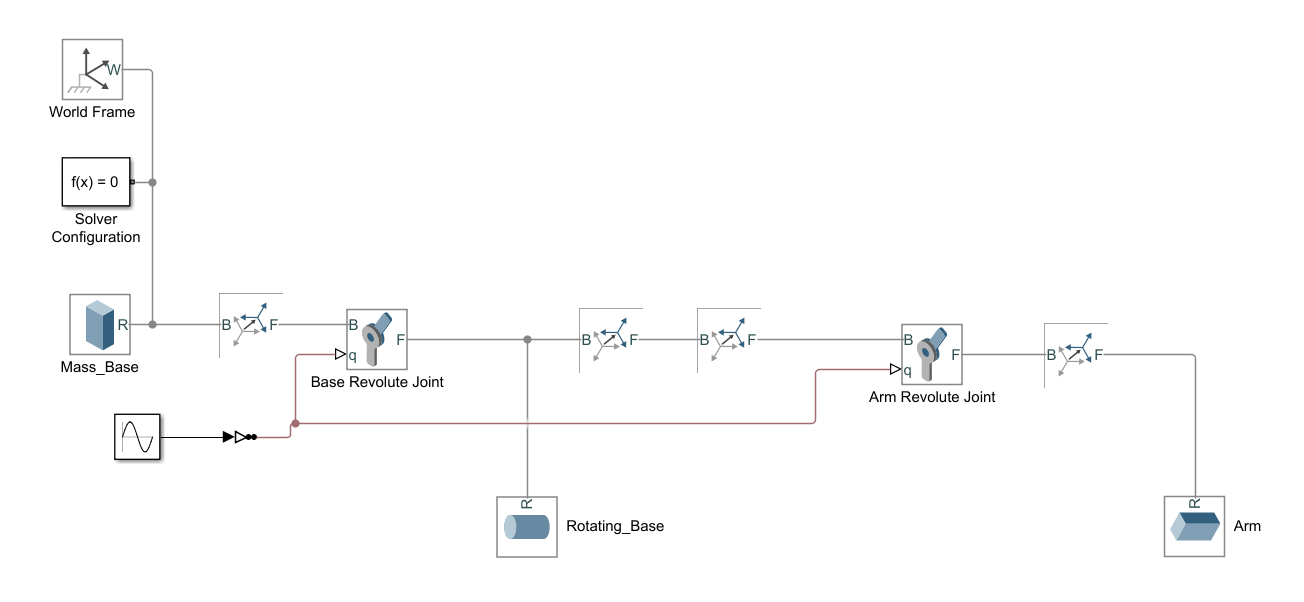
**Model File**

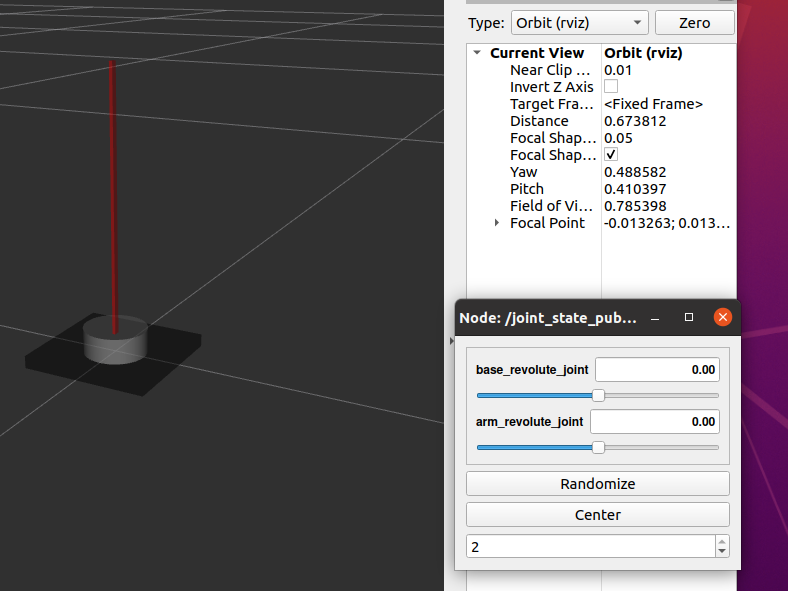
I begin with a simple 2 DOF actuator, a revolute joint at the base around the z-axis, where the base connects to a single arm through a revolute joint around the y-axis. The arm can only rotate +/- 90 degrees from vertical; coinciding with the z-axis is the origin. To help visualize the system I modeled it first in MATLAB Simscape as that is the tool that I am most familiar with.

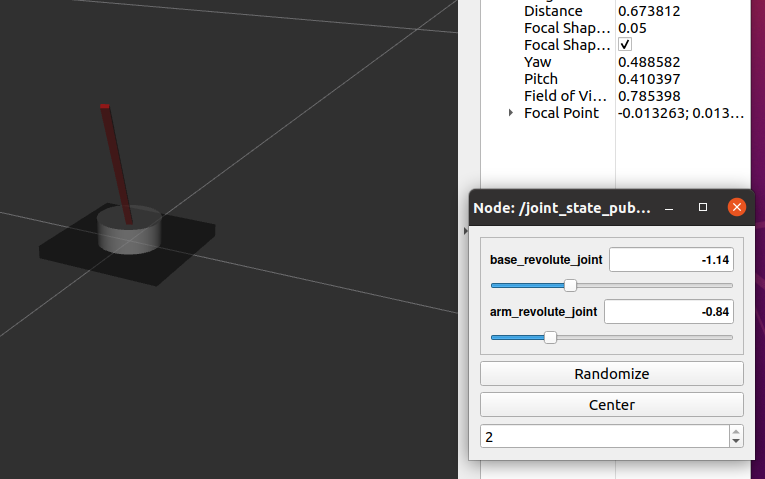
 



There is currently no method for exporting a Simscape model to a URDF or SDF, though you can import a URDF into Simscape. I selected a simple actuator model so that creating a URDF from scratch would be straightforward.

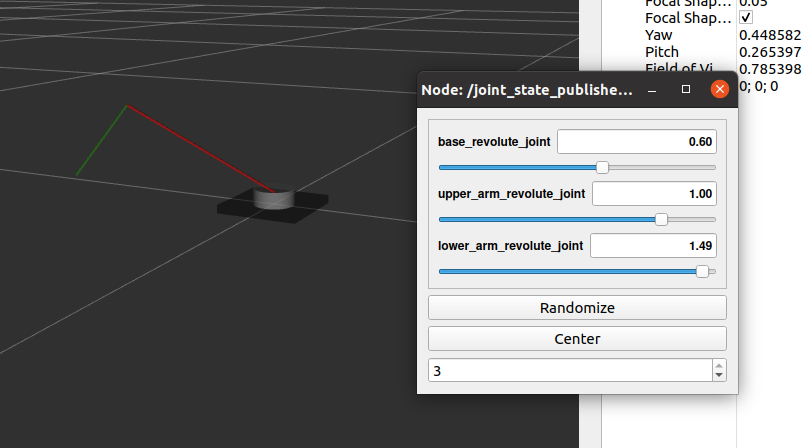
There are three links and two joints in the URDF, the same as in Simulink. The dimensions are the same, and I created material colors to match closely.





The joint state publisher is used to move the arm as shown in the second figure.

Before the model can be used in Gazebo it needs two things. First, I will add a second revolute joint, per the assignment, and make it’s length half the length of the first arm.

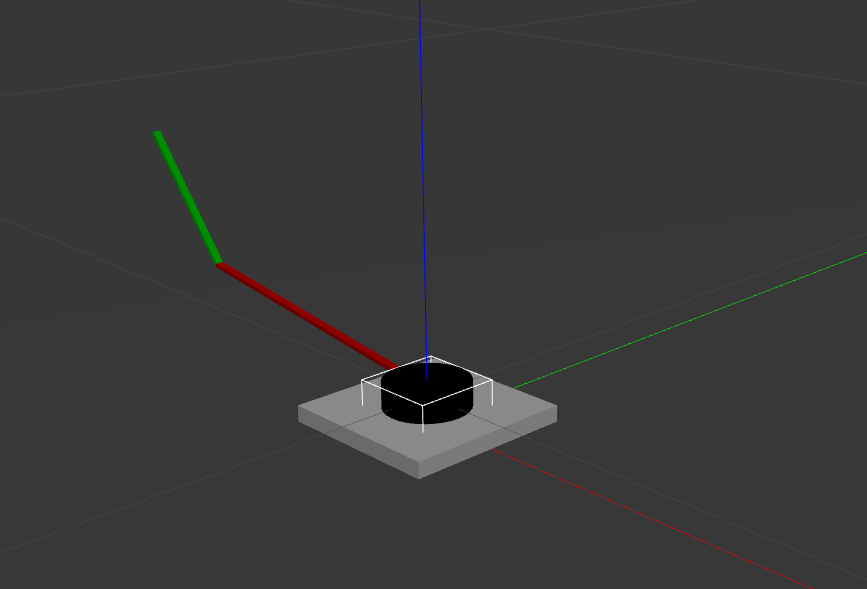
The next step was to integrate my URDF with the rrbot demo packages so that I could articulate the model using the tutorials. I have one extra joint than the rrbot, so I’ll have to add some code to make that transmission controllable.

Inertias are the most important part of the URDF for gazebo. I have a cylinder and two blocks, and the the equations for **I**x, **I**y, and **I**z are taken from the wikipedia page <https://en.wikipedia.org/wiki/List_of_moments_of_inertia>.

|  |  |  |
| --- | --- | --- |
| **Joint** | **Parameters** | **Ixx, Iyy, Izz** |
| rotating\_base\_link | m=5, r=0.025, h=0.02 | 0.000947917, 0.000947917, 0.0015625 |
| rotating\_arm\_link | m=0.1, l=0.2 | 0.00133333, 0, 0.0013333 |
| second\_arm\_link | m=0.05, l=0.1 | 0.00016667, 0, 0.00016667 |

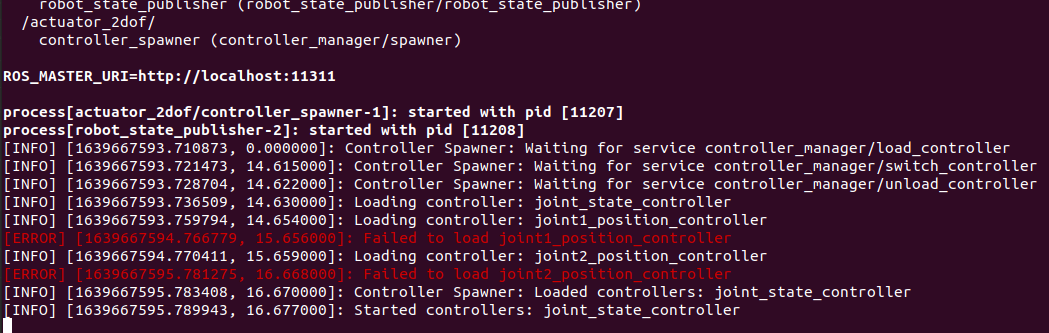
I used the rrbot demo as a blueprint and created my own actuator\_2dof robot. When I ran it in Gazebo it doesn’t fall down however. I ran the rrbot code and it does fall down once the code starts to run.

I did not include any collisions, but I didn’t expect that to stop the arms from falling. According to the tutorial if I don’t specify collision elements then Gazebo will not default to the visual elements. I added them in and now I see the arms moving. Actually, they are vibrating which is usually due to issues with the inertia values. I see that the base link is rotating slowly which is what’s probably causing the arms to vibrate. I don’t yet know enough about the transmission and closed loop control that I mimicked from the rrbot to know if this is correct behavior at this point. The arms eventually fall down.

**ROS Control**

The next step is to use the actuator\_2dof\_control package that I created, again based off of the rrbot example, and see how the .yaml file is used to control the joint positions. I’m very familiar with PID control loops, so this should be easy assuming there is adequate friction in the joints. I know that at the moment my joints don’t have any defined.

I’m getting an error message when it tries to load the controllers. I googled it and apparently I need to install ros-noetic-ros-control and ros-noetic-ros-controllers.



That worked.

Next I can manually send joint angle commands using rostopic pub to the individual position controllers. This worked! I can see that some tuning is needed as there is a little bit of overshoot reaching the position commands, but honestly not bad for using the default values from the other controller. See the captured video.

There is a lot more to this, and my next steps would be:

* Add an end-effector. I was thinking a yellow ball.
* Add a target location in x,y,z space. I was thinking of placing a ball on a table that the yellow ball had to bump into to push it off the table.
* Add simple path planning. This would be without sensors, but the algorithm would know that it had to approach the ball from the top because of the table. There are many solutions to not only the path but also the final pose of the three joints in order to make contact with the target. The first goal would be to find any solution. The second goal would be to find a solution that provided the most force with the least stress on the arm joints.