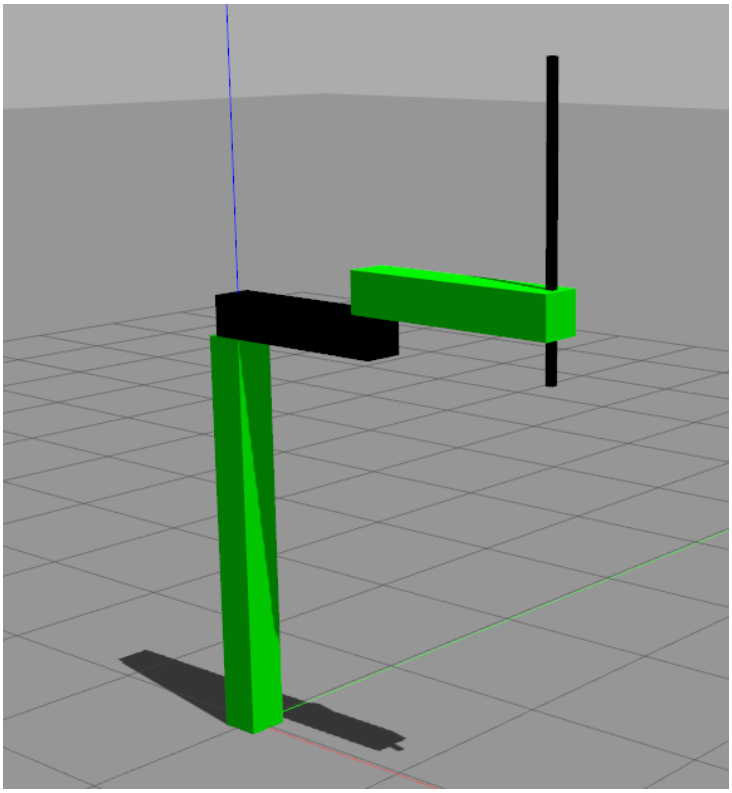
RBE 500 Group Assignment #1

Joshua Gross, Arjan Gupta, Melissa Kelly

Problem 1

Create SCARA Robot in Gazebo

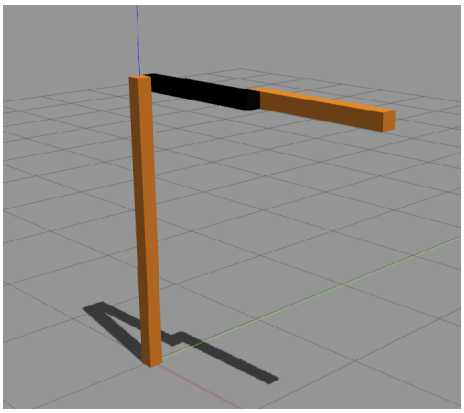
The 3 DOF SCARA robot we have built is shown below.



We undertook the following steps to create our SCARA robot.

1 — Modify joint locations

In the downloaded package, the RRBot robot has its revolute joints on the 'sides' of its links, as shown in the following figure.



However, for a standard SCARA robot, we want the revolute joints to sweep angles in the XY plane of the world frame, not in the XZ plane.

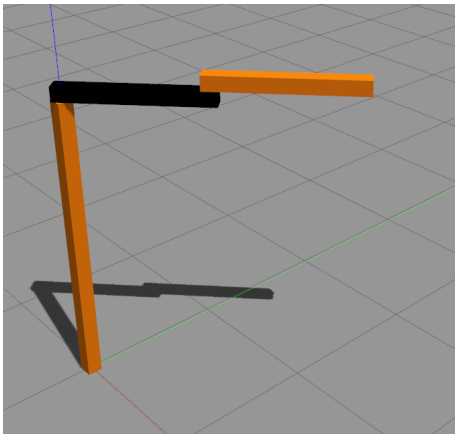
Hence, we edited the <joint> element blocks in the URDF file rrbot_description.urdf.xacro. For the first joint, we made the following change.

```
<joint name="${prefix}joint1" type="revolute">
  <parent link="${prefix}base_link"/>
  <child link="${prefix}link1"/>
  <!-- Set limits of revolute joint to -90deg to +90deg, 1000 N effort limit, velocity ...
       of 180 rad/s  -->
  <limit lower="-1.5708" upper="1.5708" effort="1000" velocity="3.14159"/>
  <origin xyz="0 0 ${height1 + axel_offset*2}" rpy="0 1.5708 0"/>
  <axis xyz="-1 0 0"/>
  <dynamics damping="0.7"/>
</joint>
```

In the above code snippet, we changed the type attribute of the joint element from continuous to revolute. We also added the limit sub-element, and modified the origin and axis sub-elements. We made similar changes for the second joint, for which the code snippet is shown below.

```
<joint name="${prefix}joint2" type="revolute">
  <parent link="${prefix}link1"/>
  <child link="${prefix}link2"/>
  <!-- Set limits of revolute joint to -90deg to +90deg, 1000 N effort limit, ...
       velocity of 180 rad/s  -->
  <limit lower="-1.5708" upper="1.5708" effort="1000" velocity="3.14159"/>
  <origin xyz="${width * -1} 0 ${height2 - axel_offset*2}" rpy="0 0 0"/>
  <axis xyz="-1 0 0"/>
  <dynamics damping="0.7"/>
</joint>
```

As a result, our robot now looked like the following image.



In order to test our changes, we moved our robot by publishing joint values in the format of the following ROS command in our terminal.

```
ros2 topic pub --once /forward_position_controller/commands ...
    std_msgs/msg/Float64MultiArray "{data: [0.75 0.82]}"
```

Problem 1 continued on next page...

2 — Add prismatic joint

Now our revolute joints resemble those of a SCARA robot, but we still need a prismatic joint. In order to do this, we first made the following changes to the rrbot_description.urdf.xacro file.

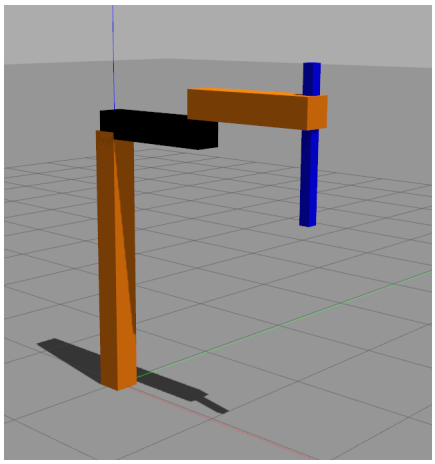
```
<!-- Tool Joint -->
<joint name="${prefix}tool_joint" type="prismatic">
  <parent link="${prefix}link2"/>
  <child link="${prefix}tool_link" />
  <!-- Set limits of prismatic joint -->
  <limit lower="${prismatic_offset * -1}" upper="1.1" effort="1000" velocity="1"/>
  <origin xyz="${prismatic_offset - height4} 0 ${height3 - axel_offset*2}" rpy="0 ...
      1.5708 0" />
  <axis xyz="0 0 1"/>
  <dynamics damping="0.7"/>
</joint>

<!-- Tool Link -->
<link name="${prefix}tool_link">
  <collision>
    <origin xyz="0 0 ${height4/2 - axel_offset}" rpy="0 0 0"/>
    <geometry>
      <box size="${prismatic_width} ${prismatic_width} ${height4}"/>
    </geometry>
  </collision>

  <visual>
    <origin xyz="0 0 ${height4/2 - axel_offset}" rpy="0 0 0"/>
    <geometry>
      <box size="${prismatic_width} ${prismatic_width} ${height4}"/>
    </geometry>
  </visual>

  <inertial>
    <origin xyz="0 0 ${height4/2 - axel_offset}" rpy="0 0 0"/>
    <mass value="${mass}"/>
    <inertia
      ixx="${mass / 12.0 * (prismatic_width*prismatic_width + height4*height4)}" ...
          ixy="0.0" ixz="0.0"
      iyy="${mass / 12.0 * (height4*height4 + prismatic_width*prismatic_width)}" ...
          iyz="0.0"
      izz="${mass / 12.0 * (prismatic_width*prismatic_width + ...
          prismatic_width*prismatic_width)}"/>
    </inertial>
</link>
```

In the above code snippet, we have changed the tool joint from fixed to prismatic, defined its translation axis, set its limits, and defined its origin. We defined a special `prismatic_width` so that we could make the tool link thinner than the regular links. We also defined a `prismatic_offset` so that the tool link is slightly 'lowered'. This makes it so that the zero position of the tool link is at the same height as our first link, which makes our calculations consistent with our forward-kinematics derivation.

Next, we defined the tool joint in the rrbot.ros2_control.xacro file, so that the command and state interfaces for ROS2 could be made available for that joint. Finally, we also edited the gazebo_controllers.yaml file to define the tool joint as part of the three controllers — `forward_position_controller`, `forward_velocity_controller`, and `forward_effort_controller`. All these modified files have been included as part of our submission inside the rrbot_description.zip compressed directory.

Problem 1 continued on next page...

At this point, our robot looked as follows.



After this, we tweaked some offsets, adjusted some link lengths, changed the colors of the robot, and decreased the thickness of the tool link in order to arrive at our finished implementation of the SCARA robot. We also used the same ROS command as earlier (except with 3 data-points this time) to ensure that we were able to move all the joints of our robot.

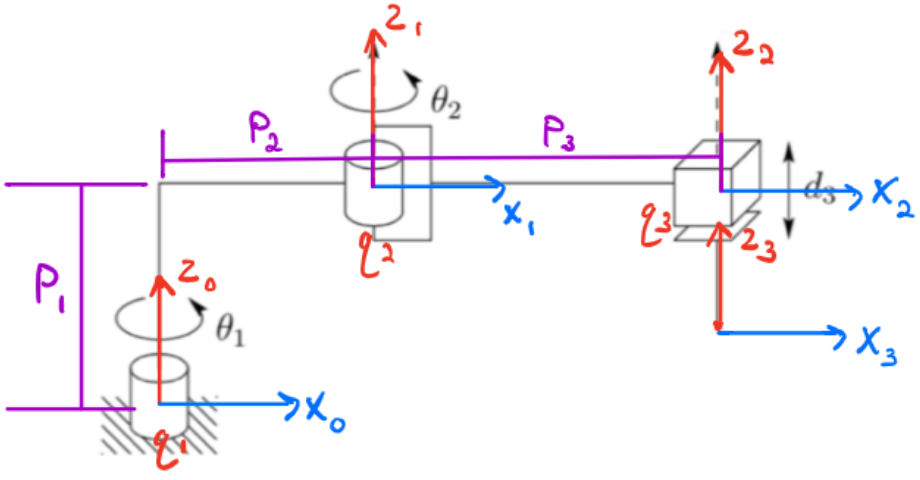
Question 2

Forward Kinematics for SCARA

Before implementing forward kinematics in ROS, we worked out a derivation for the forward kinematics of the SCARA robot model.

Derviation

The image below shows the frame assignments for the robot.



Based on this, we can form the DH table as shown on the next page.

Link	α_i	a_i	θ_i	d_i
1	0	p_2	q_1	p_1
2	0	p_3	q_2	0
3	0	0	0	q_3

Next, we create a MATLAB script as shown below. We use the matrix obtained from equation 3.10 of the textbook to write A_1, A_2, A_3. By multiplying these matrices, we obtain the T_3^0.

```
%% Forward Kinematics
clc
clear

syms P1 P2 P3 q1 q2 q3
a = [P2 P3 0];
theta = [ q1 q2 0];
d = [P1 0 q3];
alpha = [ 0 0 0];

% NOTE: Please enter theta in degrees!

i = 1;
A1 = [cosd(theta(i)) (-sind(theta(i))*cosd(alpha(i))) (sind(theta(i))*sind(alpha(i))) ...
    (a(i)*cosd(theta(i)));
    sind(theta(i)) (cosd(theta(i))*cosd(alpha(i))) (-cosd(theta(i))*sind(alpha(i))) ...
        (a(i)*sind(theta(i)));
    0 sind(alpha(i)) cosd(alpha(i)) d(i);
    0 0 0 1];

i = 2;
A2 = [cosd(theta(i)) (-sind(theta(i))*cosd(alpha(i))) (sind(theta(i))*sind(alpha(i))) ...
    (a(i)*cosd(theta(i)));
    sind(theta(i)) (cosd(theta(i))*cosd(alpha(i))) (-cosd(theta(i))*sind(alpha(i))) ...
        (a(i)*sind(theta(i)));
    0 sind(alpha(i)) cosd(alpha(i)) d(i);
    0 0 0 1];

i = 3;
A3 = [cosd(theta(i)) (-sind(theta(i))*cosd(alpha(i))) (sind(theta(i))*sind(alpha(i))) ...
    (a(i)*cosd(theta(i)));
    sind(theta(i)) (cosd(theta(i))*cosd(alpha(i))) (-cosd(theta(i))*sind(alpha(i))) ...
        (a(i)*sind(theta(i)));
    0 sind(alpha(i)) cosd(alpha(i)) d(i);
    0 0 0 1];

T30 = A1*A2*A3;

% Export to latex
latex(T30)
```

This MATLAB script gives us the following T matrix, which is the homogeneous transformation for the end-effector with respect to the base frame.

$$T_3^0 = \begin{bmatrix} \cos q_1 \cos q_2 - \sin q_1 \sin q_2 & -\cos q_1 \sin q_2 - \cos q_2 \sin q_1 & 0 & P_2 \cos q_1 + P_3 \cos q_1 \cos q_2 - P_3 \sin q_1 \sin q_2 \\ \cos q_1 \sin q_2 + \cos q_2 \sin q_1 & \cos q_1 \cos q_2 - \sin q_1 \sin q_2 & 0 & P_2 \sin q_1 + P_3 \cos q_1 \sin q_2 + P_3 \cos q_2 \sin q_1 \\ 0 & 0 & 1 & P_1 + q_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$