1. **Create the robot:**

   - Spawn the robot to the ROS-Gazebo environment. Include and image of the robot.
   - Include the robot definition file.

   Include `joint_state_publisher`

   Reference this connecting-gazebo-and-rviz

   To begin the simulation:

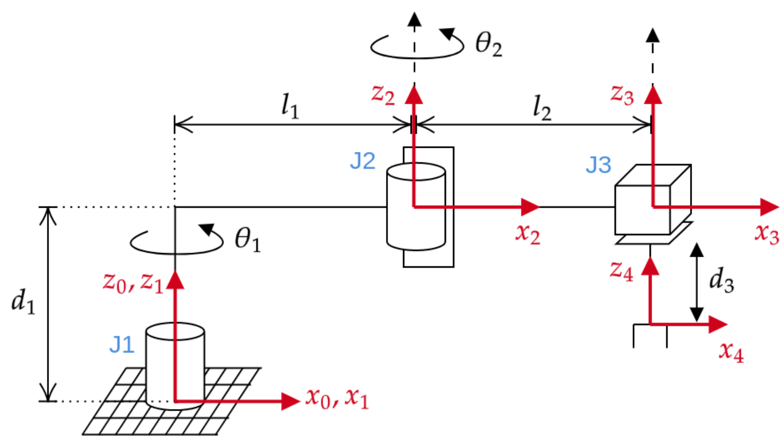
   - `roslaunch rrbot_gazebo rrbot_world.launch`

2. **Forward Kinematics:** Implement a FK node that

   - Subscribes to the joint values topic and reads them from the Gazebo simulator.
   - Calculates the pose of the end-effector.
   - Publishes the pose as a ROS topic (inside the callback function that reads the joint values).

   Print the resulting pose to the terminal using the `rostopic echo` command and include a screenshot of the results.

   We begin by assigning coordinate frames to the manipulator:



   We can then formulate the DH parameters coordinating to the links:

Link	θ_i	d_i	a_i	α_i
1	θ_1^*	d_1	l_1	0
2	θ_2^*	0	l_2	0
3	0	$-d_3^*$	0	0

   Next, we can calculate the transformations for each frame:

$$T_{i+1}^{i} = \text{Rot}_z(\theta_i)\text{Trans}_z(d_i)\text{Trans}_x(a_i)\text{Rot}_x(\alpha_i) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i\cos\alpha_i & \sin\theta_i\sin\alpha_i & a_i\cos\theta_i \\ \sin\theta_i & \cos\theta_i\cos\alpha_i & -\cos\theta_i\sin\alpha_i & a_i\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_2^1 = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1\cos 0 & \sin\theta_1\sin 0 & l_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1\cos 0 & -\cos\theta_1\sin 0 & l_1\sin\theta_1 \\ 0 & \sin 0 & \cos 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 & l_1\cos\theta_1 \\ \sin\theta_1 & \cos\theta_1 & 0 & l_1\sin\theta_1 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_3^2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2\cos 0 & \sin\theta_2\sin 0 & l_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2\cos 0 & -\cos\theta_2\sin 0 & l_2\sin\theta_2 \\ 0 & \sin 0 & \cos 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & l_2\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & l_2\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_4^3 = \begin{bmatrix} \cos 0 & -\sin 0\cos 0 & \sin 0\sin 0 & 0\cos 0 \\ \sin 0 & \cos 0\cos 0 & -\cos 0\sin 0 & 0\sin 0 \\ 0 & \sin 0 & \cos 0 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The combined transformation of the end effector is:

$$T_4^0 = T_1^0 T_2^1 T_3^2 T_4^3$$

This calculation is used in the forward kinematic function `calc_homogeneous_transform(q)`:

```
def calc_homogeneous_transform(q): # calculate the homogeneous transform from the base frame to EE
        # q = [th1, th2, d3]
        th1 = q[0]
        th2 = q[1]
        d3 = q[2]

        T1_0 = np.matrix([[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]) # frame 1 w.r.t 0
        T2_1 = np.matrix([[math.cos(th1),-math.sin(th1),0,l1*math.cos(th1)],[math.sin(th1),math.cos(th1),0,l1*math.sin(th1)],[0,0,1,d1],[0,0,0,1]])      # frame 2 w.r.t 1
        T3_2 = np.matrix([[math.cos(th2),-math.sin(th2),0,l2*math.cos(th2)],[math.sin(th2),math.cos(th2),0,l2*math.sin(th2)],[0,0,1,0],[0,0,0,1]])      # frame 3 w.r.t 2
        T4_3 = np.matrix([[1,0,0,0],[0,1,0,0],[0,0,1,-d3],[0,0,0,1]]) # frame 4 w.r.t 3

        T_EE = T1_0.dot(T2_1).dot(T3_2).dot(T4_3)

        return T_EE
```

To run and test the subscriber/publisher function:

- Run `roslaunch scara_gazebo scara_world.launch`
- In a new window `roslaunch gazebo_publish gazebo_publish.launch` which allows the joint states to be published.
- In a new window `rostopic list` should show /scara/joint_states
- `rostopic echo /scara/joint_states` should show the joint states printing
- `rosrun scara_forward_kinematics configuration_to_operational_sub.py` will run the program that subscribes to the joint states, calculates the forward kinematics, and publishes the end-effector pose back to the `Pose` topic. The print out can be seen below:

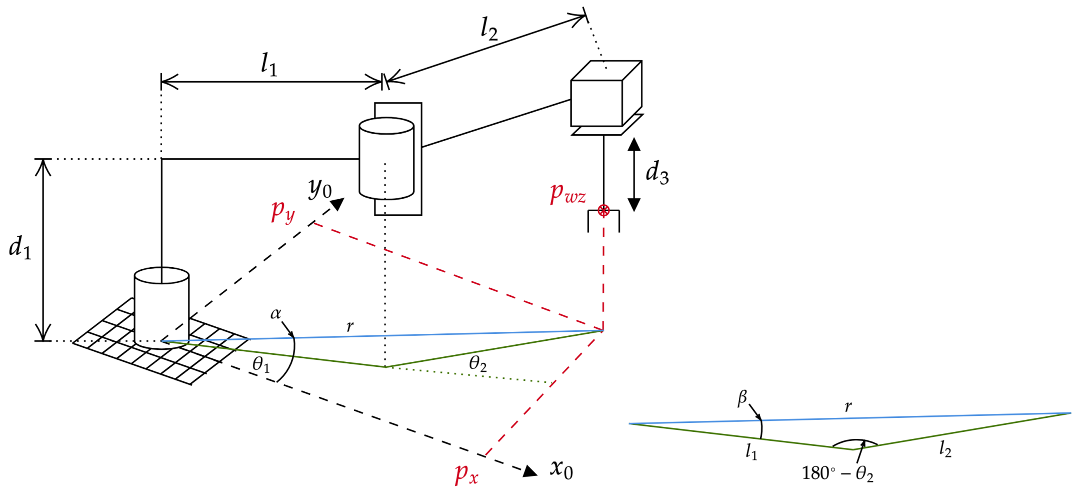
- We can also see the `Pose` topic getting published by running `rostopic echo /scara_robot/pose`

3. **Inverse Kinematics:**

- Implement an IK node (separate node) that has a service client that take a desired pose of the end effector and returns joint positions as a response.

- Test your node with `rosservice call`. Include a screenshot of the results.

The following definitions can be used to calculate the inverse kinematics:



The large right triangle can be used to calculate the following:

$$r = \sqrt{p_x^2 + p_y^2}$$
$$A = \frac{p_x}{r} = \cos\alpha \Rightarrow \sin\alpha = \pm\sqrt{1 - A^2} \Rightarrow \alpha = \text{atan2}\left(\pm\sqrt{1 - A}, A\right)$$

The law of cosines can be used on the other triangle to calculate β and θ_1:

$$l_2^2 = r^2 + l_1^2 - 2rl_1\cos\beta$$
$$\cos\beta = \frac{r^2 + l_1^2 - l_2^2}{2rl_1} = C \Rightarrow \sin\beta = \pm\sqrt{1 - C^2}$$
$$\beta = \text{atan2}\left(\pm\sqrt{1 - C^2}, C\right)$$
$$\theta_1 = \alpha - \beta$$

The law of cosines can be used again to calculate θ_1:

$$r^2 = l_1^2 + l_2^2 - 2l_1l_2\cos(180 - \theta_2)$$
$$r^2 = l_1^2 + l_2^2 + 2l_1l_2\cos(\theta_2)$$
$$\cos\theta_2 = \frac{r^2 - l_1^2 - l_2^2}{2l_1l_2} = D \Rightarrow \sin\theta_2 = \pm\sqrt{1 - D^2}$$
$$\theta_2 = \text{atan2}\left(\pm\sqrt{1 - D^2}, D\right)$$

Lastly, $d3$ is calculated simply as follows:

$$d_3 = d_1 - p_z$$

The above equations are included in the server `inverse_server.cpp`:

```
double D = - (((l1*l1)+(l2*l2)-(x*x+y*y))/(2*l1*l2));
double C = (((l1*l1)+x*x+y*y-(l2*l2))/(2*l1*sqrt(x*x+y*y)));

double B = sqrt(1-D*D);
double E = sqrt(1-C*C);

double alpha = atan2(y, x);

```

```
res.theta1 = alpha-atan2(E, C);
res.theta2 = atan2(B, D);
res.d3 = d1 - z;
```