**ROS Workspace**

$ mkdir -p ~/catkin\_ws/src

$ cd ~/catkin\_ws/

$ catkin\_make

**Clone repository**

$ cd ~/catkin\_ws/src

$ git clone https://github.com/udacity/RoboND-Kinematics-Project.git

**Update all required ros dependencies and permissions**

$ cd ~/catkin\_ws

$ rosdep install --from-paths src --ignore-src --rosdistro=kinetic -y

$ cd ~/catkin\_ws/src/RoboND-Kinematics-Project/kuka\_arm/scripts

$ sudo chmod +x target\_spawn.py

$ sudo chmod +x IK\_server.py

$ sudo chmod +x safe\_spawner.sh

**Build project**

$ cd ~/catkin\_ws

$ catkin\_make

**Add to .bashrc file**

export GAZEBO\_MODEL\_PATH=~/catkin\_ws/src/RoboND-Kinematics-Project/kuka\_arm/models

source ~/catkin\_ws/devel/setup.bash

**inverse\_kinematics.launch file**

demo flag is set to "false” to run new IK\_server.py

**Run project**

$ cd ~/catkin\_ws/src/RoboND-Kinematics-Project/kuka\_arm/scripts

$ ./safe\_spawner.sh

$ cd ~/catkin\_ws/src/RoboND-Kinematics-Project/kuka\_arm/scripts

$ rosrun kuka\_arm IK\_server.py

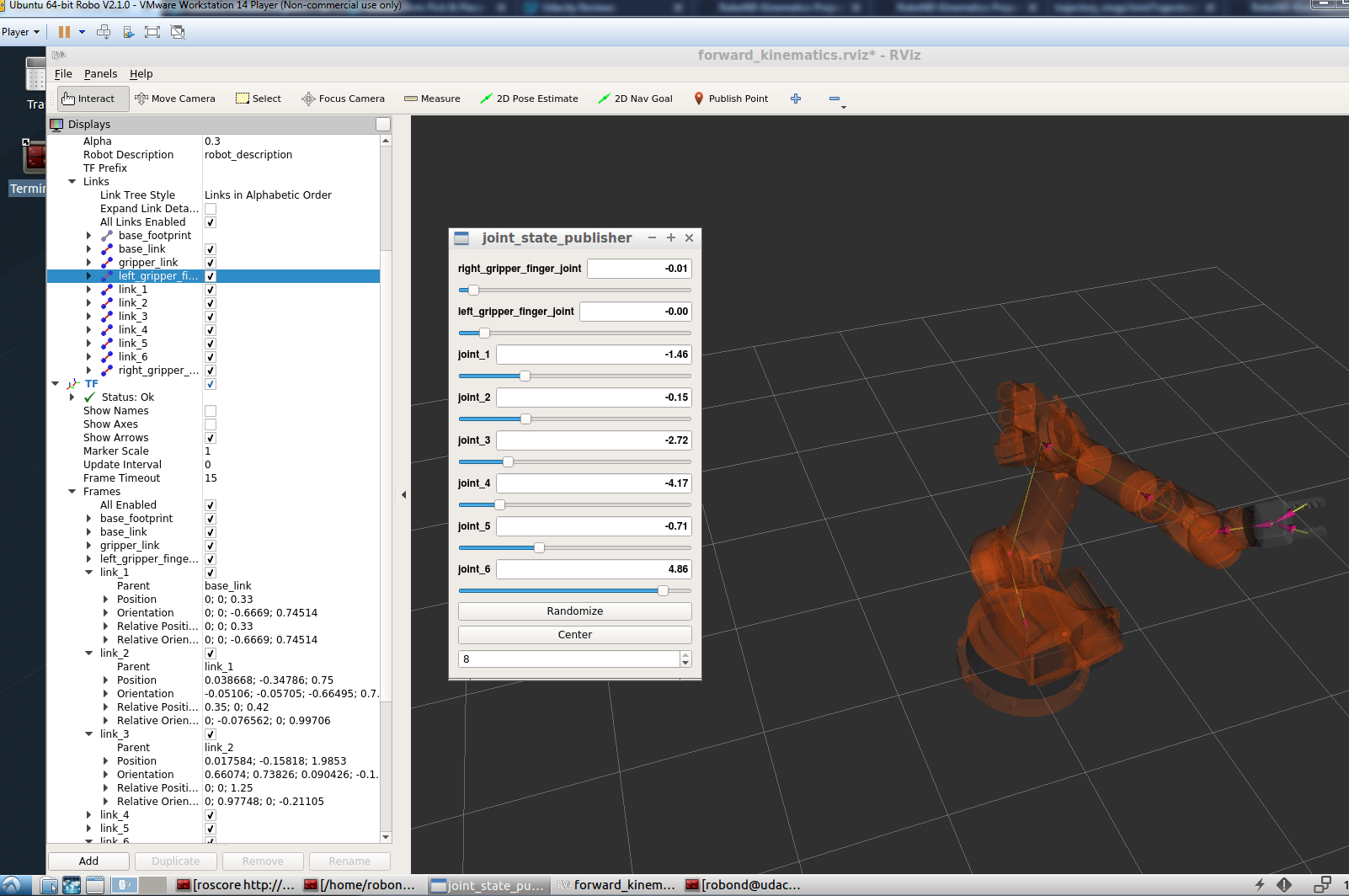
**FORWARD KINEMATIC**

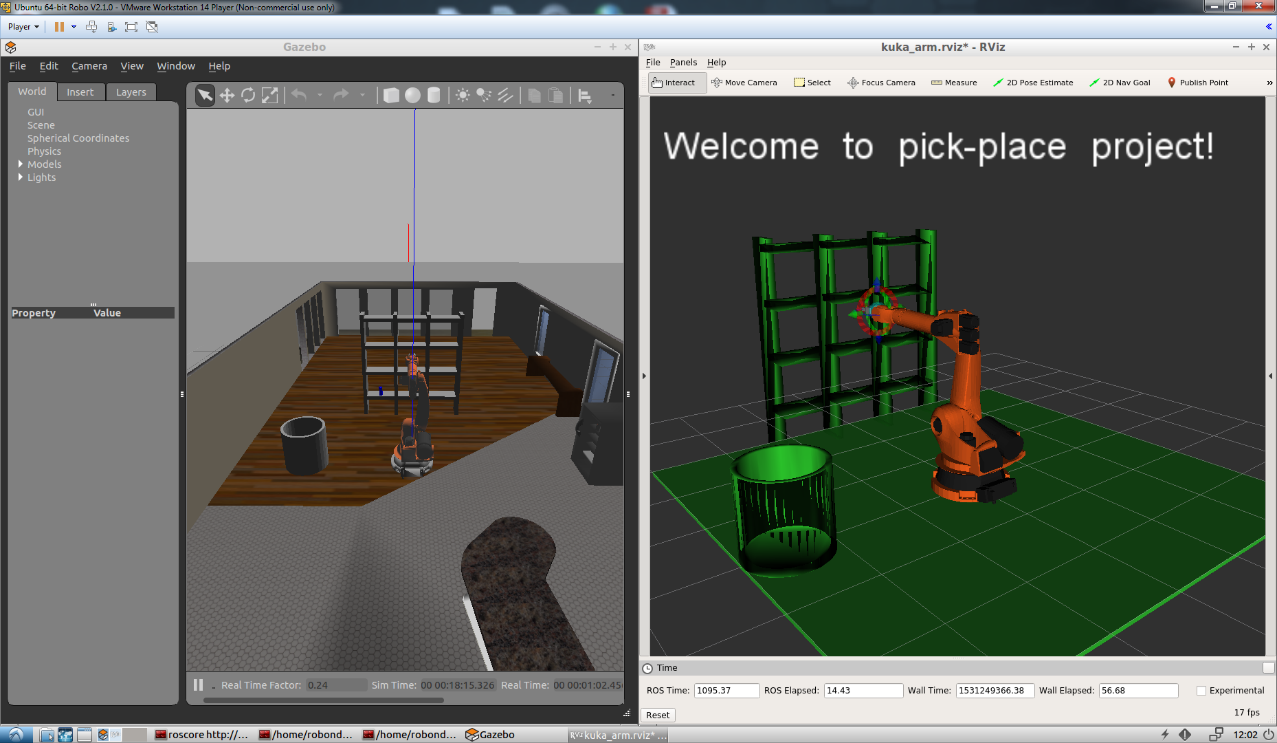
Experiment with the forward kinematics environment and get familiar with the robot.

$ roslaunch kuka\_arm forward\_kinematics.launch

ROS makes it very easy to get the transform between any two given frames with the tf\_echo command.

$ rosrun tf tf\_echo base\_link link\_6

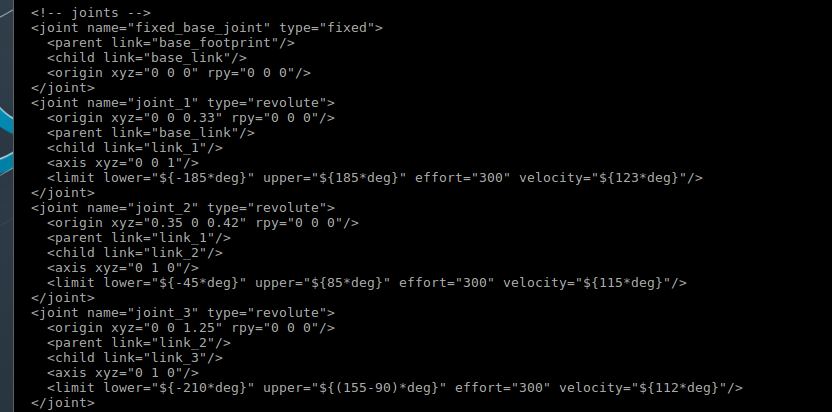


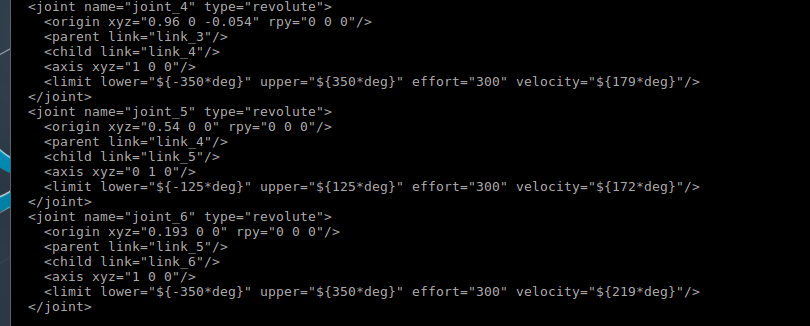


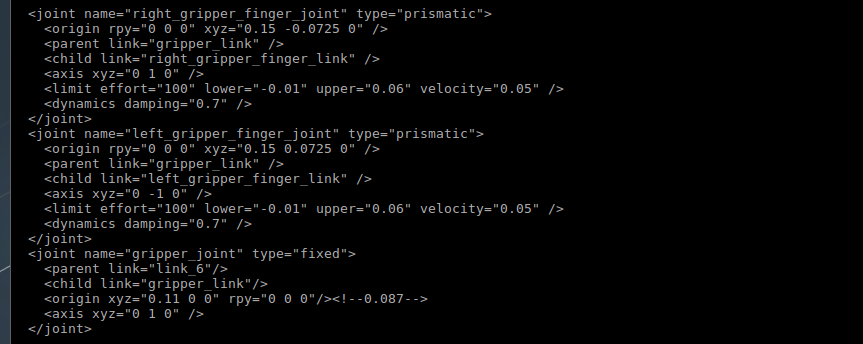
**KR210.urdf.xacro**

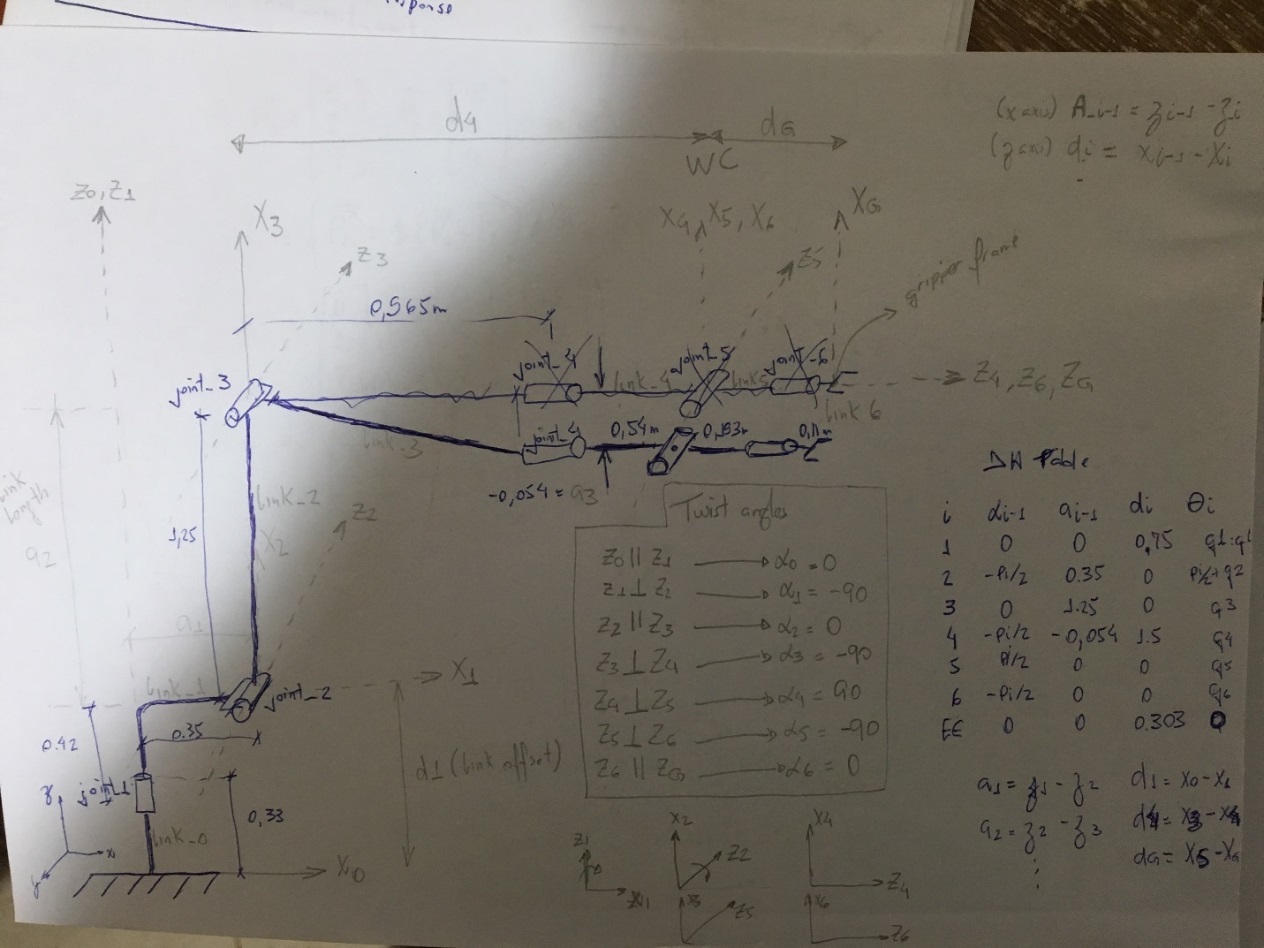
All the values for the robot geometry are contained inside urdf file.

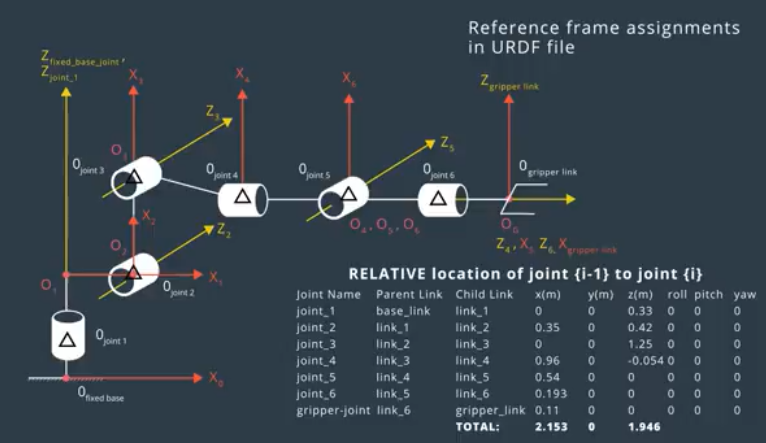
It contains static and dynamic friction coefficient for the links, each link origin w.r.t to its local C.S., mass, inertia and also each JOINT type, origin, parent/child LINK, axis, and physical limits.





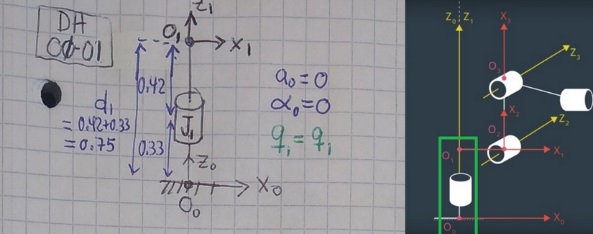
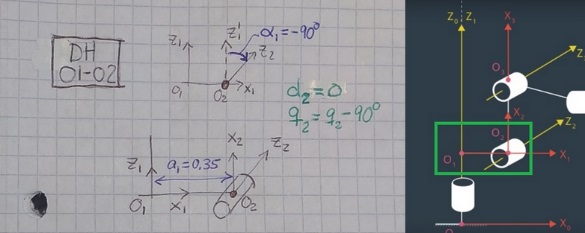


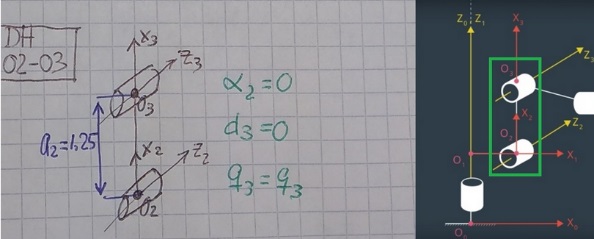
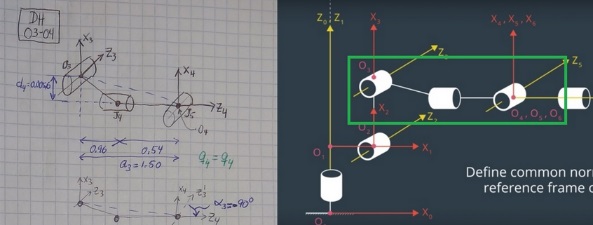
Taking parameter values from URDF file to write down the manipulator:

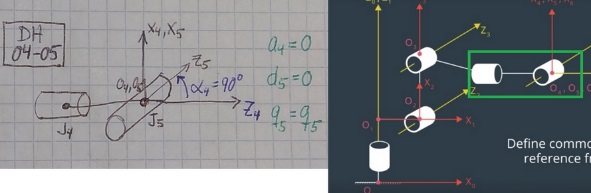
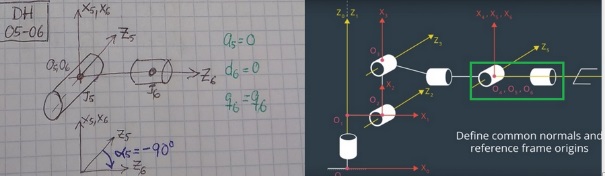


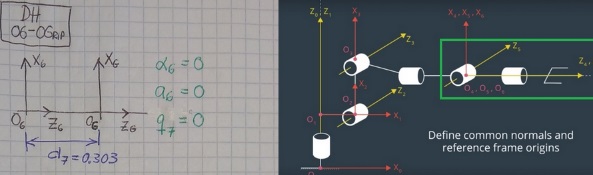
**DH Table**

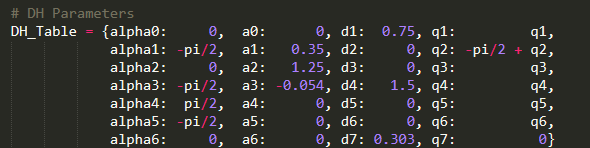
* alpha = twist angle
* a= link length
* d= link offset
* q= joint angle

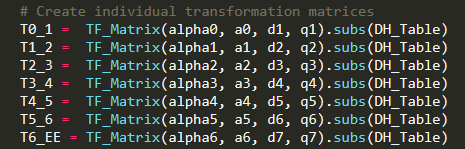
 



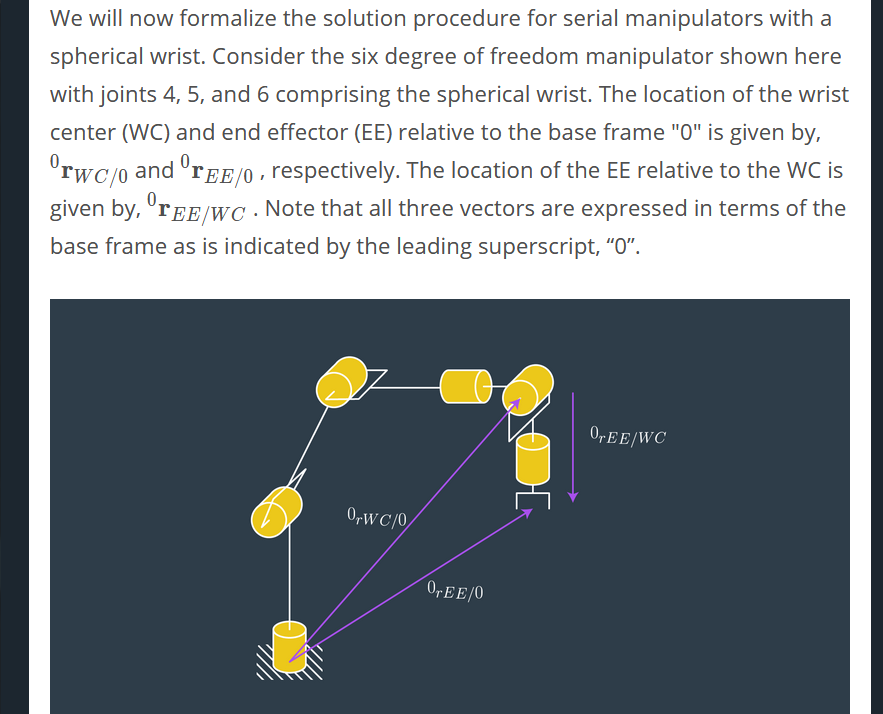


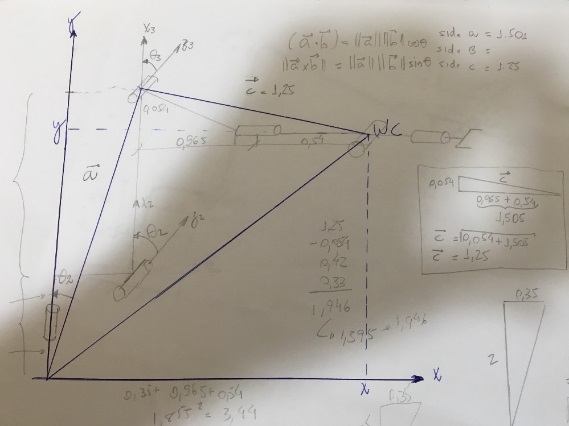
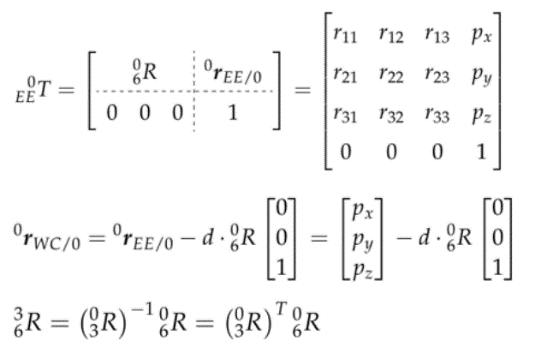
**Matrix for translation and orientation**

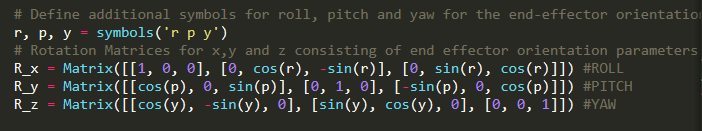


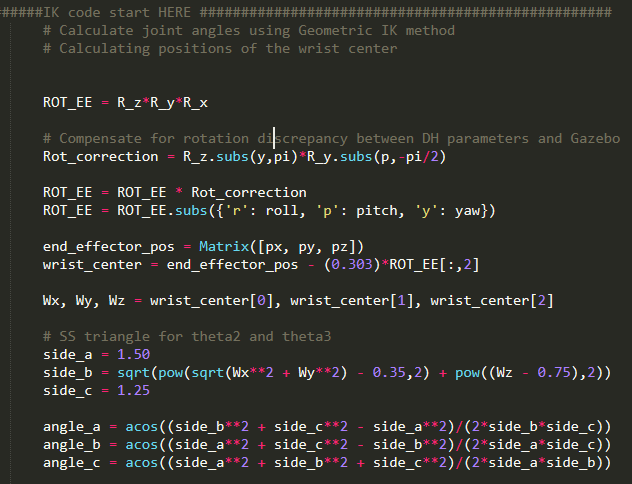


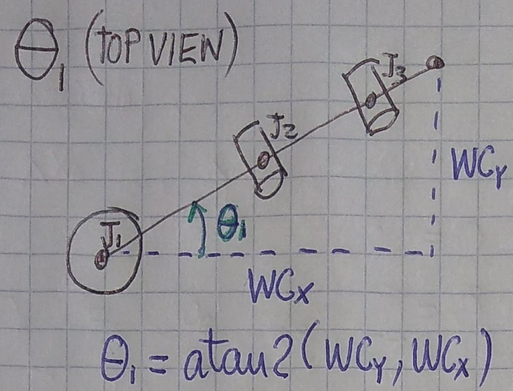
**Inverse position and orientation kinematics**



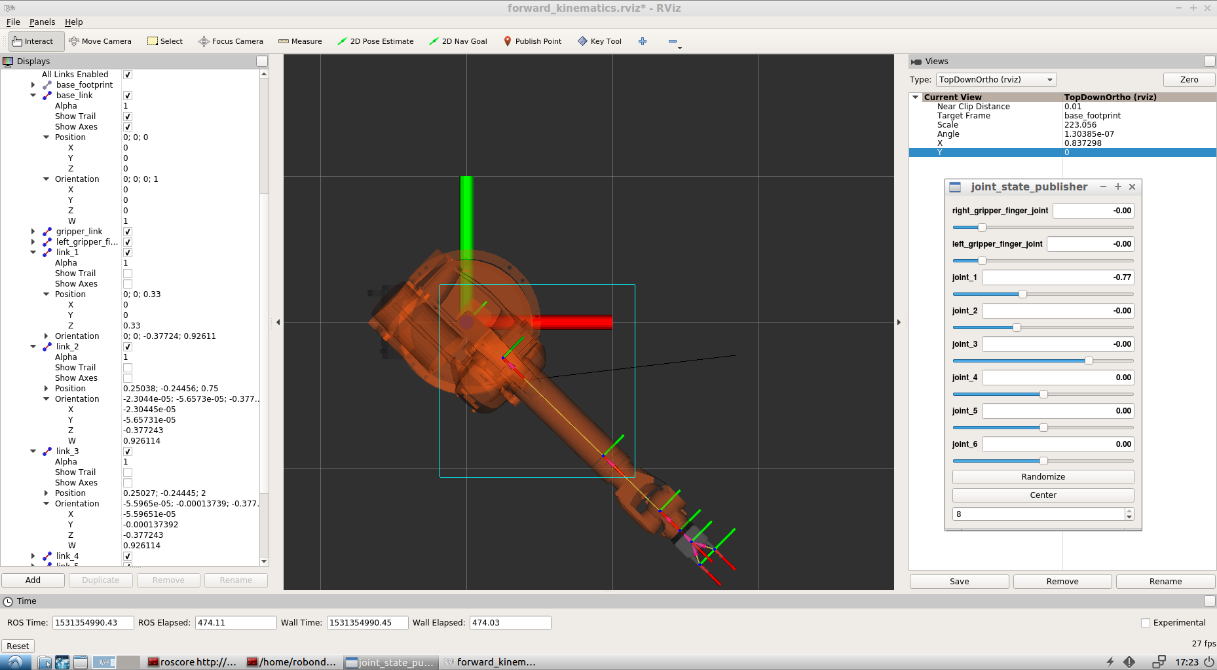


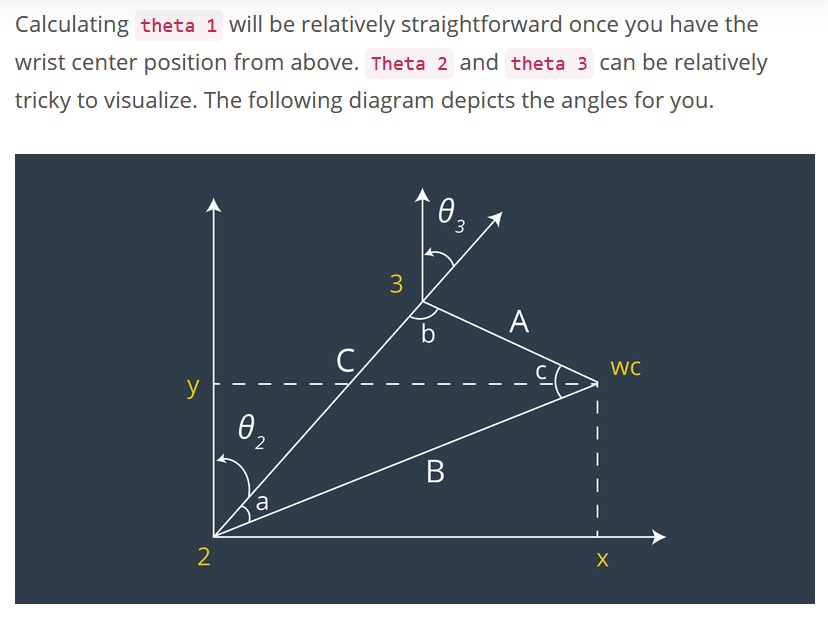


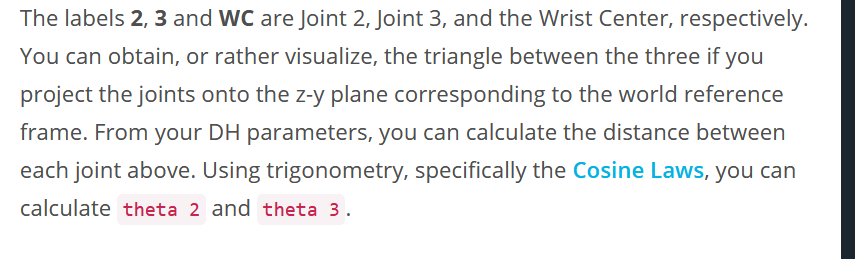


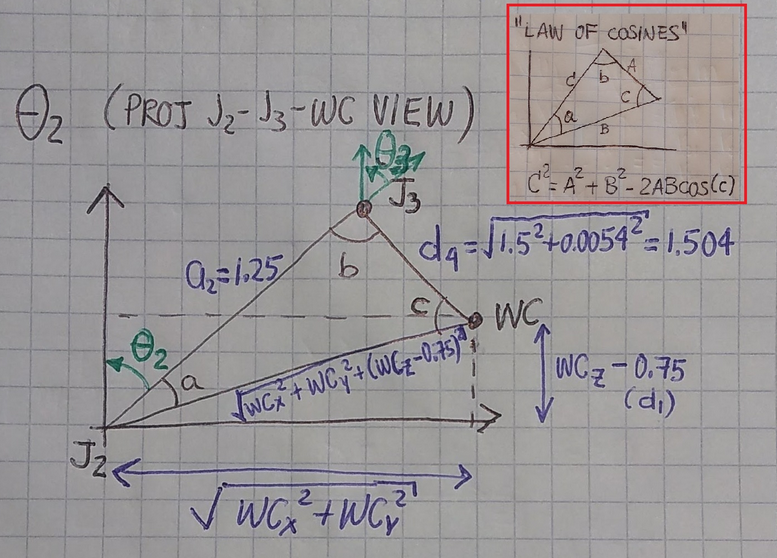


theta1 = atan2(Wy, Wx)

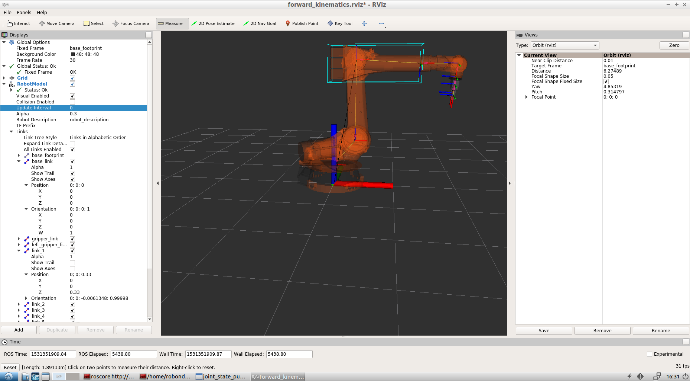
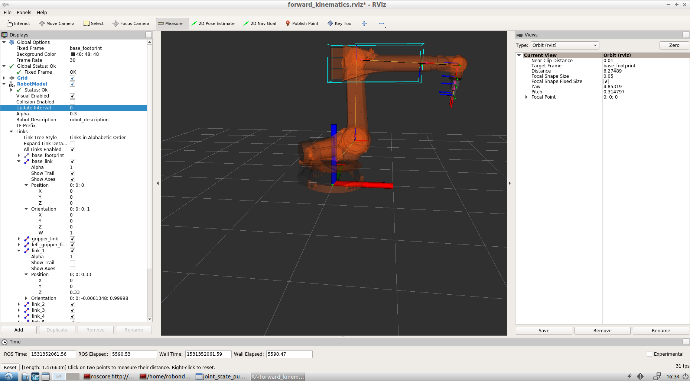


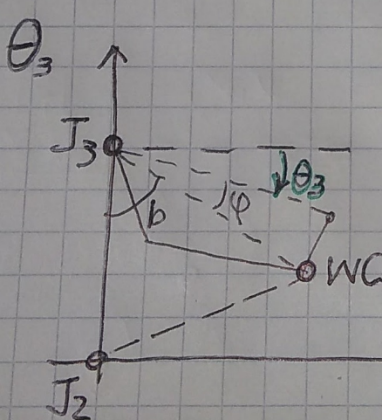






theta2 = pi/2 - angle\_a - atan2((Wz - 0.75), sqrt(Wx\*\*2 + Wy\*\*2) - 0.35)



theta3 = pi/2 - angle\_b + 0.036

