Hands-on Session No.02

Title: Introduction to a Robot Arm Forward Kinematics task with ROS

1. Introduction

In this hands-on session you will learn how Robot Operating System (ROS) and its libraries can be used for the study of *robotic kinematics* concepts. ROS has a number of software packages that deal with kinematics. We use RVIZ as the visualization tool that can display a URDF (Unified Robotic Description Format) robot model in 3D along with data from other sensors such as cameras. URDF is an XML modeling format capable of modeling most robots. The *tf* software package provides a library of kinematic routines that provides all the mathematical functions we need to transform kinematic data from one frame of reference to another. MoveIt is a useful software package for controlling robot arms and manipulators (MoveIt will not be used in this hands-on session).

2. Implementation on ROS-DS

You will be provided with the files for ROS-DS inside '**src.zip**'. The github link for the files is https://github.com/sakunaharinda/ROS-Handson-Session-2.

Create a new ROSject according to the information given below.

- ROS CONFIGURATION: Select **Ubuntu v 16.04** and **ROS Kinetic** as the platform
- PRIVATE OR PUBLIC: Choose **Public**
- DESCRIPTION - Give a brief description on the project (OPTIONAL)

Go to the ROSject and open it. Once the project has loaded, go to the bottom **toolbar** and select the **Code editor** (VS Code) IDE.

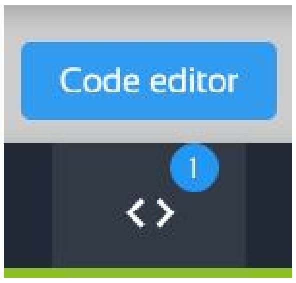


Figure 1 – Code editor icon

Upload the contents inside **src.zip** into **catkin_ws/src** folder shown in the VS Code IDE. (You can simply **drag-n-drop** to upload folders) It will look like below after you have completed uploading.

```
catkin_ws
  |___ build
  |___ devel
  |___ src
       |_ forward_kinematics
       |        |_ src
       |             |_ solution.py
       |_ lwr_robot
       |        |_ lwr_defs
       |             |_ robots
       |                  |_ kuka_lwr_arm.urdf
       |                  |_ kuka_lwr_arm.urdf.xml
       |_ robot_mover
       |_ robot_sim
```

Follow the terminal commands given as follows:

```
$ sudo apt-get install ros-kinetic-urdfdom-py
$ cd catkin_ws/
$ catkin_make
```

Once you have successfully compiled your ROS project (remember to have a separate roscore terminal running), complete the following:

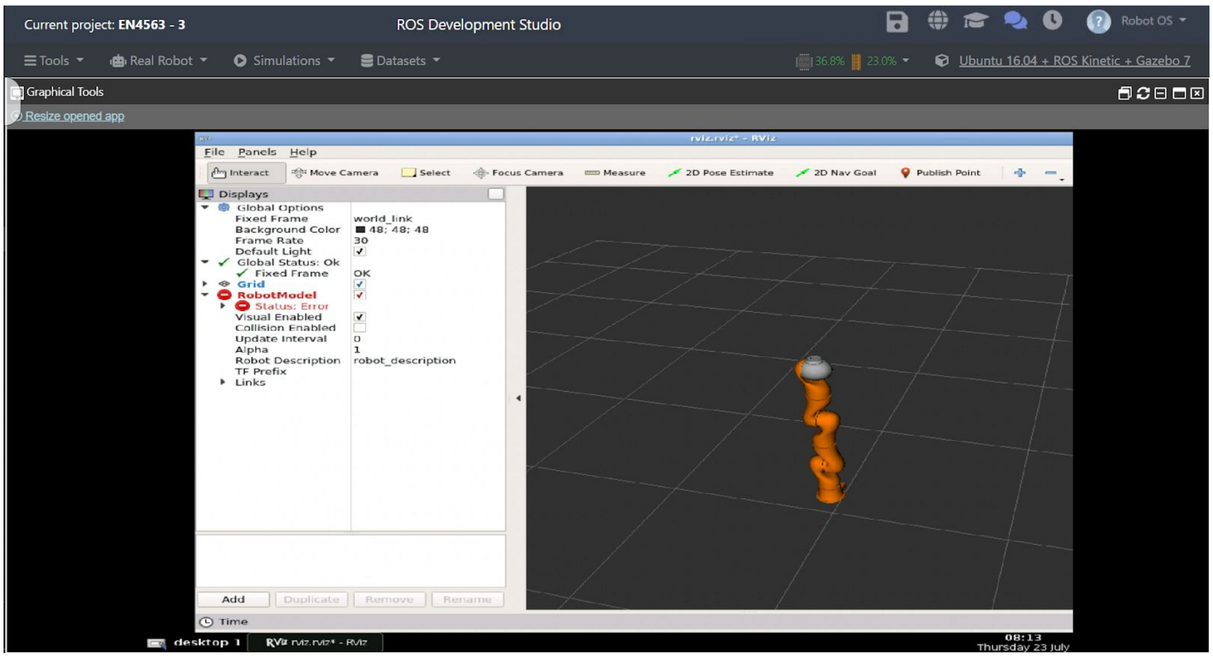
2.1 Project Task

In this project you will implement the forward kinematics for a **robot arm defined in a URDF file**, running in a ROS environment.

The setup contains a “simulated” robot that continuously publishes its own joint values. You can check whether the robot is publishing its joint values by using the ‘**rostopic echo/joint_states**’ command. However, that is not enough for the robot to be correctly displayed: a forward kinematics module must use the joint values to compute the transforms from the world coordinate frame to each link of the robot. Let's check the **forward kinematics code and compute transforms function and the launch file architecture.**

```
$ source devel/setup.bash
$ chmod +x src/forward_kinematics/src/solution.py
$ roslaunch robot_sim kuka_lwr.launch
```

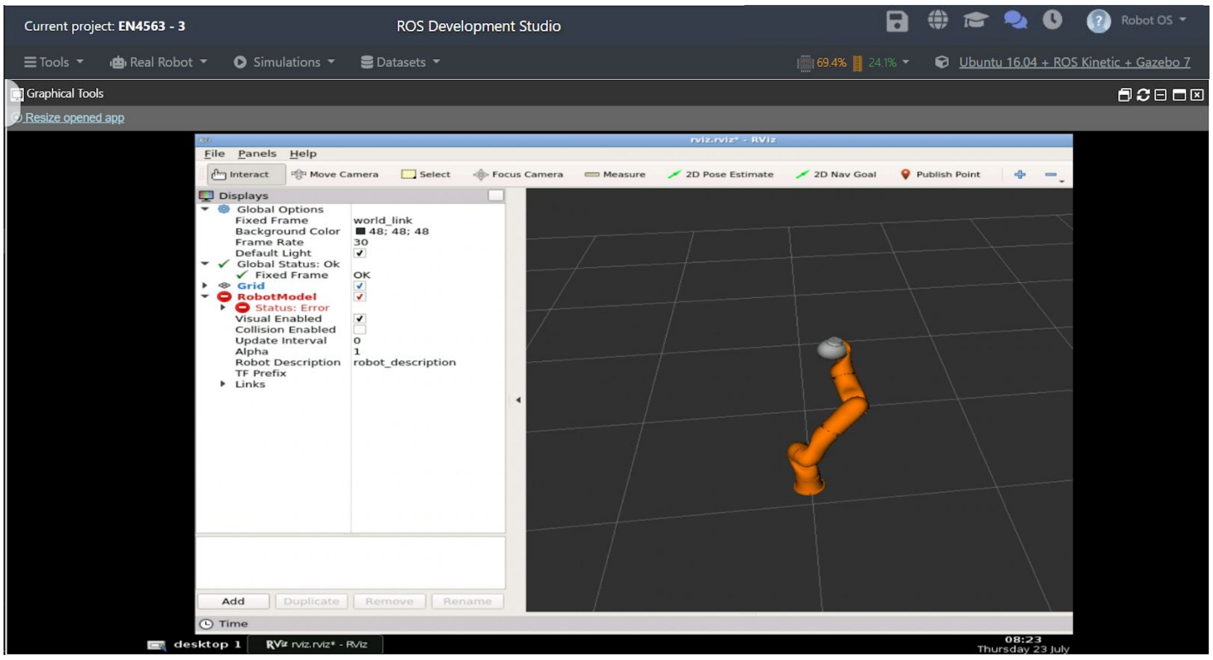
Note: You will need to open **Tools□Graphical Tools** window to visualize **rviz viewer.** Initially, the robot should be in the following position shown in Fig. 2, without moving.



```
Global Options
  Fixed Frame          world_link
  Background Color     48; 48; 48
  Frame Rate           30
  Default Light        ✓
Global Status: Ok
  Fixed Frame          OK
Grid                   ✓
RobotModel             ✓
  Status: Error
  Visual Enabled       ✓
  Collision Enabled
  Update Interval      0
  Alpha                1
  Robot Description    robot_description
  TF Prefix
  Links
```

Figure 2 – The Kuka Robot Arm ecosystem has been generated in RViz tool and is at the initial position

If you have done everything correctly, you should see the robot arm move back and forth in a physically correct fashion with its final position relating to the one shown in Fig. 3. You can also check by publishing its joint values by using the 'rostopic echo/joint_states' command. Understand why the robot is moving in this situation.



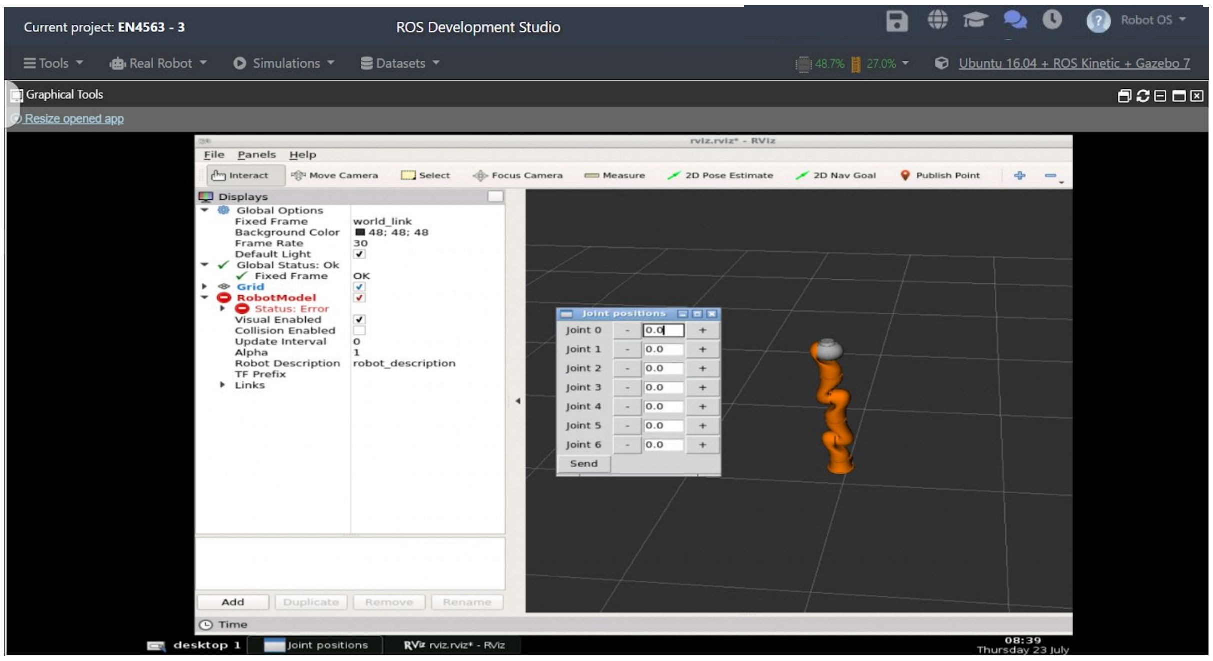
```
Global Options
  Fixed Frame          world_link
  Background Color     48; 48; 48
  Frame Rate           30
  Default Light        ✓
Global Status: Ok
  Fixed Frame          OK
Grid                   ✓
RobotModel             ✓
  Status: Error
  Visual Enabled       ✓
  Collision Enabled
  Update Interval      0
  Alpha                1
  Robot Description    robot_description
  TF Prefix
  Links
```

Figure 3 - The Kuka Robot Arm ecosystem has been generated in RViz tool and is at the final position

We are required to move the robot using user defined joint positions, as well. In order to visualize the forward kinematics of the robot in real-time, you may use the following commands:

```
$ source devel/setup.bash
$ chmod +x src/robot_sim/scripts/position_command.py
$ rosrun robot_sim position_command.py
$ roslaunch robot_sim kuka_lwr_test.launch
```

Your final output should look like Fig. 4.



```
Global Options
  Fixed Frame          world_link
  Background Color     48; 48; 48
  Frame Rate           30
  Default Light        ✓
Global Status: Ok
  Fixed Frame          OK
Grid                   ✓
RobotModel             ✓
  Status: Error
  Visual Enabled       ✓
  Collision Enabled
  Update Interval      0
  Alpha                1
  Robot Description    robot_description
  TF Prefix
  Links
```

Joint 0	-	0.0	+
Joint 1	-	0.0	+
Joint 2	-	0.0	+
Joint 3	-	0.0	+
Joint 4	-	0.0	+
Joint 5	-	0.0	+
Joint 6	-	0.0	+

Figure 4: The Kuka Robot Arm ecosystem has been generated in RViz tool with manual joint positions

Thank you!