ROS Lab 5

Vipul Dinesh, 220929024, MTE-A-09

1. Get the following required packages

```
sudo apt-get install ros-humble-teleop-twist-keyboard ros-humble-joint-state-publisher* ros-humble-joint-trajectory-controller ros-humble-controller-manager ros-humble-gazebo-* ros-humble-gazebo-msgs ros-humble-gazebo-ros ros-humble-gazebo-ros2-control-demos ros-humble-ros2-control ros-humble-ros2-controllers ros-humble-ros2controlcli ros-humble-xacro ros-humble-gazebo-dev ros-humble-gazebo-plugins
```

2. Create urdf_tutorial package

- `cd ~/ros2_ws/src`
- `ros2 pkg create --build-type ament_python urdf_tutorial --dependencies rclpy`

3. Make urdf, launch and config sub-folders in `urdf_tutorial` package folder

- `mkdir ~/ros2_ws/src/urdf_tutorial/urdf`
- `mkdir ~/ros2_ws/src/urdf_tutorial/launch`
- `mkdir ~/ros2_ws/src/urdf_tutorial/config`

4. Create a config file called `control.yaml` in the config directory just created and fill content as follows

`control.yaml`

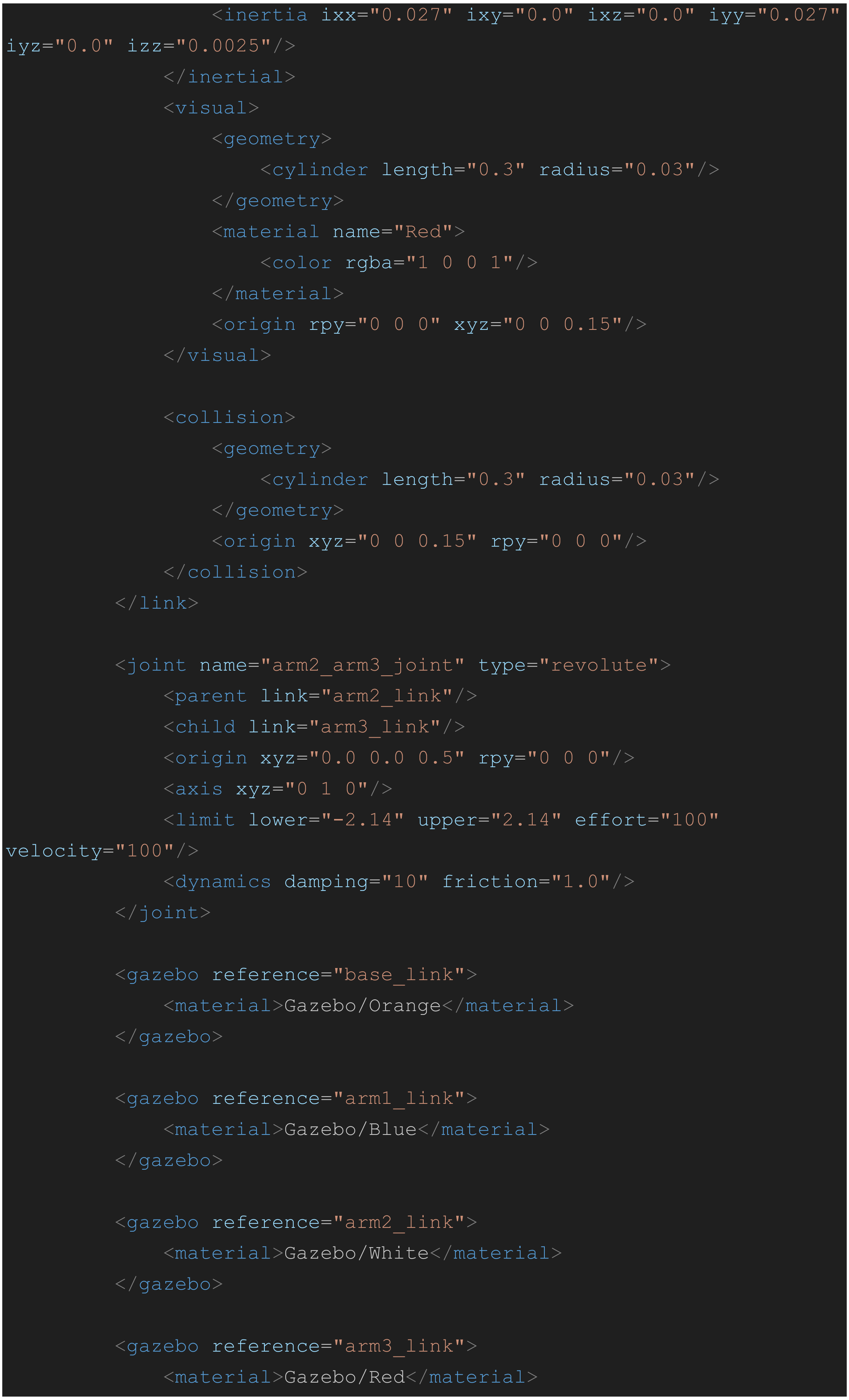
```
controller_manager:
 ros__parameters:
   update_rate: 100

   joint_state_broadcaster:
     type: joint_state_broadcaster/JointStateBroadcaster

   joint_trajectory_controller:
     type: joint_trajectory_controller/JointTrajectoryController

joint_trajectory_controller:
 ros__parameters:
   joints:
     - base_arm1_joint
     - arm1_arm2_joint
     - arm2_arm3_joint

   command_interfaces:
     - position

   state_interfaces:
     - position

   state_publish_rate: 50.0
   action_monitor_rate: 20.0

   allow_partial_joints_goal: false
   open_loop_control: true
   constraints:
     stopped_velocity_tolerance: 0.01
     goal_time: 0.0
     joint1:
       trajectory: 0.05
       goal: 0.03
```

5. Create a urdf file called `arm.urdf` in the urdf directory just created and fill content as follows

`arm.urdf`

```
      <?xml version="1.0"?>
      <robot name="arm">
         <!-- https://www.rapidtables.com/web/color/RGB_Color.html --
>


         <link name="world"/>

         <link name="base_link">
             <visual>
                 <geometry>
                     <cylinder length="0.05" radius="0.2"/>
                 </geometry>
                 <material name="Orange">
                     <color rgba="1 0.5 0 1"/>
                 </material>
                 <origin xyz="0 0 0.025" rpy="0 0 0"/>
             </visual>

             <collision>
                 <geometry>
                     <cylinder length="0.05" radius="0.2"/>
                 </geometry>
                 <origin xyz="0 0 0.025" rpy="0 0 0"/>
             </collision>
```

```
            <inertial>
                <origin rpy="0 0 0" xyz="0 0 0.025"/>
                <mass value="5.0"/>
                <inertia ixx="0.0135" ixy="0.0" ixz="0.0" iyy="0.0135" iyz="0.0" izz="0.05"/>
            </inertial>
        </link>

        <joint name="world_base_joint" type="fixed">
            <parent link="world"/>
            <child link="base_link"/>
            <dynamics damping="10" friction="1.0"/>
        </joint>

        <link name="arm1_link">
            <visual>
                <geometry>
                    <cylinder length="0.5" radius="0.08"/>
                </geometry>
                <material name="Blue">
                    <color rgba="0 0 1 1"/>
                </material>
                <origin xyz="0 0 0.25" rpy="0 0 0"/>
            </visual>

            <collision>
                <geometry>
                    <cylinder length="0.5" radius="0.08"/>
                </geometry>
                <origin xyz="0 0 0.25" rpy="0 0 0"/>
            </collision>

            <inertial>
                <origin rpy="0 0 0" xyz="0 0 0.25"/>
                <mass value="5.0"/>
                <inertia ixx="0.107" ixy="0.0" ixz="0.0" iyy="0.107" iyz="0.0" izz="0.0125"/>
            </inertial>
        </link>

        <joint name="base_arm1_joint" type="revolute">
            <axis xyz="0 1 0"/>
            <parent link="base_link"/>
            <child link="arm1_link"/>
            <origin xyz="0.0 0.0 0.05" rpy="0 0 0"/>
            <limit lower="-2.14" upper="2.14" effort="100"
velocity="100"/>
            <dynamics damping="10" friction="1.0"/>
        </joint>

        <link name="arm2_link">
            <inertial>
                <origin xyz="0 0 0.25" rpy="0 0 0"/>
                <mass value="0.01"/>
                <inertia ixx="0.027" ixy="0.0" ixz="0.0" iyy="0.027"
iyz="0.0" izz="0.0025"/>
            </inertial>

            <visual>
                <geometry>
                    <cylinder length="0.5" radius="0.05"/>
                </geometry>
                <material name="White">
                    <color rgba="1 1 1 1"/>
                </material>
                <origin rpy="0 0 0" xyz="0 0 0.25"/>
            </visual>

            <collision>
                <geometry>
                    <cylinder length="0.4" radius="0.05"/>
                </geometry>
                <origin xyz="0 0 0.25" rpy="0 0 0"/>
            </collision>
        </link>

        <joint name="arm1_arm2_joint" type="revolute">
            <parent link="arm1_link"/>
            <child link="arm2_link"/>
            <origin xyz="0.0 0.0 0.5" rpy="0 0 0"/>
            <axis xyz="0 1 0"/>
            <limit lower="-2.14" upper="2.14" effort="100"
velocity="100"/>
            <dynamics damping="10" friction="1.0"/>
        </joint>

        <link name="arm3_link">
            <inertial>
                <origin xyz="0 0 0.15" rpy="0 0 0"/>
                <mass value="0.01"/>
```



```
                <inertia ixx="0.027" ixy="0.0" ixz="0.0" iyy="0.027"
iyz="0.0" izz="0.0025"/>
            </inertial>
            <visual>
                <geometry>
                    <cylinder length="0.3" radius="0.03"/>
                </geometry>
                <material name="Red">
                    <color rgba="1 0 0 1"/>
                </material>
                <origin rpy="0 0 0" xyz="0 0 0.15"/>
            </visual>

            <collision>
                <geometry>
                    <cylinder length="0.3" radius="0.03"/>
                </geometry>
                <origin xyz="0 0 0.15" rpy="0 0 0"/>
            </collision>
        </link>

        <joint name="arm2_arm3_joint" type="revolute">
            <parent link="arm2_link"/>
            <child link="arm3_link"/>
            <origin xyz="0.0 0.0 0.5" rpy="0 0 0"/>
            <axis xyz="0 1 0"/>
            <limit lower="-2.14" upper="2.14" effort="100"
velocity="100"/>
            <dynamics damping="10" friction="1.0"/>
        </joint>

        <gazebo reference="base_link">
            <material>Gazebo/Orange</material>
        </gazebo>

        <gazebo reference="arm1_link">
            <material>Gazebo/Blue</material>
        </gazebo>

        <gazebo reference="arm2_link">
            <material>Gazebo/White</material>
        </gazebo>

        <gazebo reference="arm3_link">
            <material>Gazebo/Red</material>
        </gazebo>

        <gazebo>
            <plugin filename="libgazebo_ros2_control.so"
name="gazebo_ros2_control">

<robot_sim_type>gazebo_ros2_control/GazeboSystem</robot_sim_type>

<parameters>/home/vipul/ros2_ws/src/urdf_tutorial/config/control.yaml
</parameters>
            </plugin>
        </gazebo>

        <ros2_control name="GazeboSystem" type="system">
            <hardware>
                <plugin>gazebo_ros2_control/GazeboSystem</plugin>
            </hardware>

            <joint name="base_arm1_joint">
                <command_interface name="position">
                    <param name="min">-2.14</param>
                    <param name="max">2.14</param>
                </command_interface>
                <state_interface name="position"/>
                <param name="initial_position">0.0</param>
            </joint>

            <joint name="arm1_arm2_joint">
                <command_interface name="position">
                    <param name="min">-2.14</param>
                    <param name="max">2.14</param>
                </command_interface>
                <state_interface name="position"/>
                <param name="initial_position">0.1</param>
            </joint>

            <joint name="arm2_arm3_joint">
                <command_interface name="position">
                    <param name="min">-2.14</param>
                    <param name="max">2.14</param>
                </command_interface>
                <state_interface name="position"/>
                <param name="initial_position">0.2</param>
            </joint>
        </ros2_control>
```

```
      </robot>
```

6. In the launch folder, create launch files for gazebo, rviz and arm_control such that they use the `arm.urdf` file. Name them as `arm_gazebo.launch.py`, `rviz.launch.py` and `arm_control.launch.py` respectively and fill the content as follows

   `arm_gazebo.launch.py`

```
      import os
      from launch import LaunchDescription
      from launch.actions import ExecuteProcess
      from launch_ros.actions import Node

      def generate_launch_description():
         urdf_file =
'/home/vipul/ros2_ws/src/urdf_tutorial/urdf/arm.urdf'

         return LaunchDescription(
             [
                 ExecuteProcess(
                     cmd=["gazebo", "-s",
"libgazebo_ros_factory.so"],
                     output="screen",
                 ),
                 Node(
                     package="gazebo_ros",
                     executable="spawn_entity.py",
                     arguments=["-entity", "manipulator", "-b", "-
file", urdf_file],
                     output="screen",
                 ),
                 Node(
                     package="robot_state_publisher",
                     executable="robot_state_publisher",
                     arguments=[urdf_file],
                     output="screen",
                 ),
             ]
         )
```

   `arm_rviz.launch.py`

```
      from launch import LaunchDescription
      from launch_ros.actions import Node

      def generate_launch_description():
         urdf_file =
'/home/vipul/ros2_ws/src/urdf_tutorial/urdf/arm.urdf'

         joint_state_publisher_node = Node(
             package="joint_state_publisher_gui",
             executable="joint_state_publisher_gui",
             name="joint_state_publisher_gui",
             output="screen",
             arguments=[urdf_file]
         )

         robot_state_publisher_node = Node(
             package="robot_state_publisher",
             executable="robot_state_publisher",
             name="robot_state_publisher",
             output="screen",
             arguments=[urdf_file]
         )

         rviz_node = Node(
             package="rviz2",
             executable="rviz2",
             name="rviz2",
             output="screen"
         )

         nodes_to_run = [
             joint_state_publisher_node,
             robot_state_publisher_node,
             rviz_node
         ]

         return LaunchDescription(nodes_to_run)
```

arm_control.launch.py

```
      import os
      import xacro

      from launch import LaunchDescription
      from launch.actions import ExecuteProcess,
IncludeLaunchDescription, RegisterEventHandler
      from launch_ros.actions import Node
      from launch.event_handlers import OnProcessExit
      from launch.launch_description_sources import
PythonLaunchDescriptionSource
```

```
      from ament_index_python.packages import
get_package_share_directory

      def generate_launch_description():
         urdf_file =
'/home/vipul/ros2_ws/src/urdf_tutorial/urdf/arm.urdf'
         controller_file =
'/home/vipul/ros2_ws/src/urdf_tutorial/config/control.yaml'
         robot_description = {"robot_description": urdf_file}

         # Include Gazebo launch file
         gazebo = IncludeLaunchDescription(

PythonLaunchDescriptionSource("/home/vipul/ros2_ws/src/urdf_tutorial/
launch/arm_gazebo.launch.py"),
         )

         # Process the URDF file with Xacro
         doc = xacro.parse(open(urdf_file))
         xacro.process_doc(doc)
         params = {'robot_description': doc.toxml()}

         # Create Node for robot_state_publisher
         node_robot_state_publisher = Node(
             package='robot_state_publisher',
             executable='robot_state_publisher',
             output='screen',
             parameters=[params]
         )

         # Create Node for spawning the entity in Gazebo
         spawn_entity = Node(
             package='gazebo_ros',
             executable='spawn_entity.py',
             arguments=["-entity", "manipulator", "-b", "-file",
urdf_file],
             output='screen'
         )

         # Create ExecuteProcess actions to load controllers
         load_joint_state_controller = ExecuteProcess(
             cmd=['ros2', 'control', 'load_controller', '--set-
state', 'active', 'joint_state_broadcaster'],
             output='screen'
         )
```

```
        load_joint_trajectory_controller = ExecuteProcess(
            cmd=['ros2', 'control', 'load_controller', '--set-
state', 'active', 'joint_trajectory_controller'],
            output='screen'
        )

        return LaunchDescription(
            [
                # Register event handlers to load controllers
sequentially
                RegisterEventHandler(
                    event_handler=OnProcessExit(
                        target_action=spawn_entity,
                        on_exit=[load_joint_state_controller],
                    )
                ),
                RegisterEventHandler(
                    event_handler=OnProcessExit(
                        target_action=load_joint_state_controller,
                        on_exit=[load_joint_trajectory_controller],
                    )
                ),
                gazebo,
                node_robot_state_publisher,
                spawn_entity,
                Node(
                    package="controller_manager",
                    executable="ros2_control_node",
                    parameters=[robot_description, controller_file],
                    output="screen"
                )
            ]
        )
```

7. Create nodes called `control.py` and `controller_fk.py` in the directory `~/ros2_ws/src/urdf_tutorial/urdf_tutorial`

control.py

```
    #!/usr/bin/env python3
    import rclpy
    from rclpy.node import Node
    from builtin_interfaces.msg import Duration
    from trajectory_msgs.msg import JointTrajectory,
JointTrajectoryPoint
```

```
class TrajectoryPublisher(Node):

   def __init__(self):
       super().__init__('trajectory_node')
       topic_ = "/joint_trajectory_controller/joint_trajectory"
       self.joints = ['base_arm1_joint', 'arm1_arm2_joint', 'arm2_arm3_joint']
       #self.goal_ =[1.5, 0.5, 1.2]
       self.declare_parameter("joint_angles", [1.5, 0.5, 1.2])
       self.goal_=self.get_parameter("joint_angles").value
       self.publisher_ = self.create_publisher(JointTrajectory, topic_, 10)
       self.timer_ = self.create_timer(1,self.timer_callback)



   def timer_callback(self):
       msg = JointTrajectory()
       msg.joint_names = self.joints
       point = JointTrajectoryPoint()
       point.positions = self.goal_
       point.time_from_start = Duration(sec=2)
       msg.points.append(point)
       self.publisher_.publish(msg)

def main(args=None):
   rclpy.init(args=args)
   node = TrajectoryPublisher()
   rclpy.spin(node)
   node.destroy_node()
   rclpy.shutdown()

if __name__ == '__main__':
   main()
```

controller_fk.py

```
import rclpy
from rclpy.node import Node
from sensor_msgs.msg import JointState
from rclpy.clock import Clock
import sys

class TrajectoryPublisher(Node):
   def __init__(self):
```

```
             super().__init__('trajectory_node')

             topic_ = "/joint_states"
             self.joints = ['base_arm1_joint', 'arm1_arm2_joint',
'arm2_arm3_joint']

             # Handle command-line arguments
             if len(sys.argv) < 4:
                 self.get_logger().error("Not enough arguments
provided. Using default values.")
                 self.goal_ = [0.5, 0.5, 0.5]  # Default values
             else:
                 try:
                     self.goal_ = [float(sys.argv[1]),
float(sys.argv[2]), float(sys.argv[3])]
                 except ValueError:
                     self.get_logger().error("Invalid argument(s)
provided. Using default values.")
                     self.goal_ = [0.0, 0.0, 0.0]  # Default values

             self.publisher_ = self.create_publisher(JointState,
topic_, 10)
             self.timer_ = self.create_timer(0.1,
self.timer_callback)

         def timer_callback(self):
             msg = JointState()
             current_time = Clock().now().to_msg()

             msg.header.stamp.sec = current_time.sec
             msg.header.stamp.nanosec = current_time.nanosec
             msg.name = self.joints
             msg.position = self.goal_

             self.publisher_.publish(msg)
             self.get_logger().info("Publishing position:
{}".format(self.goal_))

      def main(args=None):
         rclpy.init(args=args)
         node = TrajectoryPublisher()
         rclpy.spin(node)
         node.destroy_node()
         rclpy.shutdown()
```

```
if __name__ == '__main__':
    main()
```

8. Make changes in `setup.py` to set `control.py` and `controller_fk.py` as executables named `control` and `control_fk` respectively, initialize the gazebo, rviz and arm_control launch files, recognize the urdf file and also recognize the config file

`setup.py`

```
from setuptools import find_packages, setup
from glob import glob
import os

package_name = 'urdf_tutorial'

setup(
    name=package_name,
    version='0.0.0',
    packages=find_packages(exclude=['test']),
    data_files=[
        ('share/ament_index/resource_index/packages',
            ['resource/' + package_name]),
        ('share/' + package_name, ['package.xml']),
        (os.path.join('share', package_name), glob('urdf/*')),
        (os.path.join('share', package_name), glob('launch/*')),
        (os.path.join('share', package_name), glob('config/*')),
    ],
    install_requires=['setuptools'],
    zip_safe=True,
    maintainer='vipul',
    maintainer_email='vipul@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
        'control=urdf_tutorial.control:main',
        'control_fk=urdf_tutorial.controller_fk:main'
        ],
    },
)
```

9. Rebuild the package and source it again

- `cd ~/ros2_ws/`
- `colcon build --packages-select urdf_tutorial`

- `source ~/ros2_ws/install/setup.zsh`

10. Launch rviz using the launch file and make the following changes

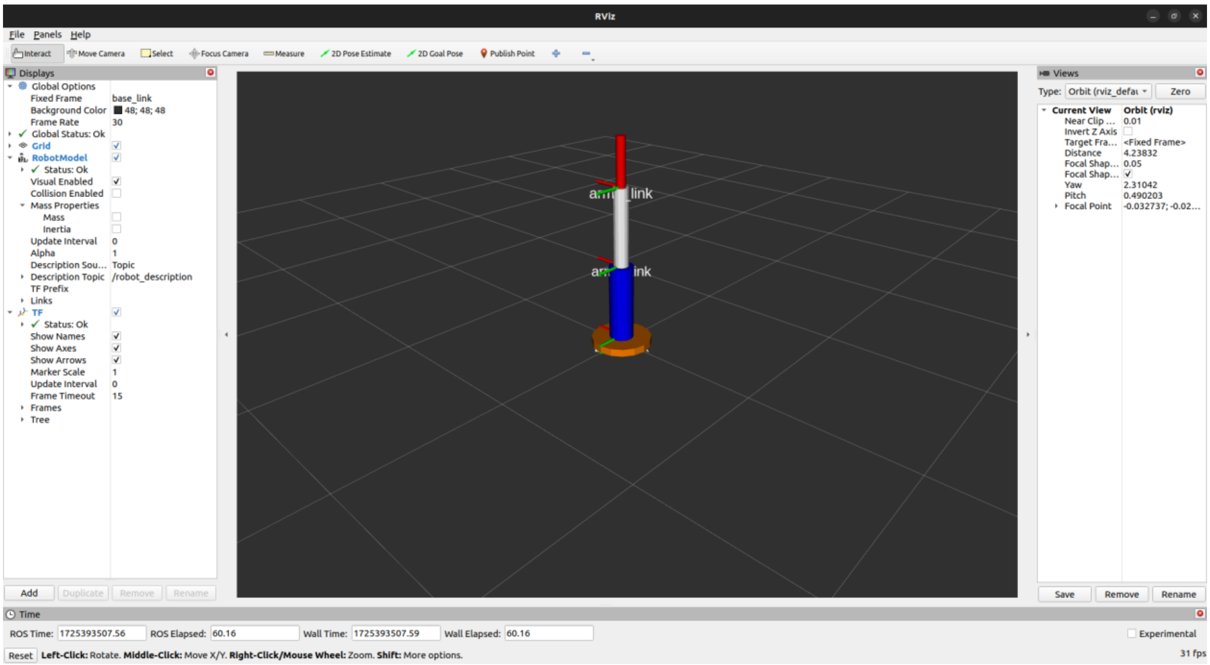
    `terminal_1`

    - `ros2 launch urdf_tutorial arm_rviz.launch.py`

– Under `Global Options`, set `Fixed Frame` as `base_link`

– Click on `Add` at the bottom of the window and include `RobotModel` and `TF` (one at a time)

– Under `RobotModel`, set `Description Topic` as `/robot_description`



```
Global Options
    Fixed Frame          base_link
    Background Color     48; 48; 48
    Frame Rate           30
✓ Global Status: Ok
Grid                     ☑
RobotModel               ☑
  ✓ Status: Ok
    Visual Enabled       ☑
    Collision Enabled    □
    Mass Properties
        Mass             □
        Inertia          □
    Update Interval      0
    Alpha                1
    Description Sou...   Topic
    Description Topic    /robot_description
    TF Prefix
    Links
TF                       ☑
  ✓ Status: Ok
    Show Names           ☑
    Show Axes            ☑
    Show Arrows          ☑
    Marker Scale         1
    Update Interval      0
    Frame Timeout        15
    Frames
    Tree
```

```
Type: Orbit (rviz_defa   Zero
Current View       Orbit (rviz)
    Near Clip ...      0.01
    Invert Z Axis      □
    Target Fra...      <Fixed Frame>
    Distance           4.23832
    Focal Shap...      0.05
    Focal Shap...      ☑
    Yaw                2.31042
    Pitch              0.490203
    Focal Point        -0.032737; -0.02...
```

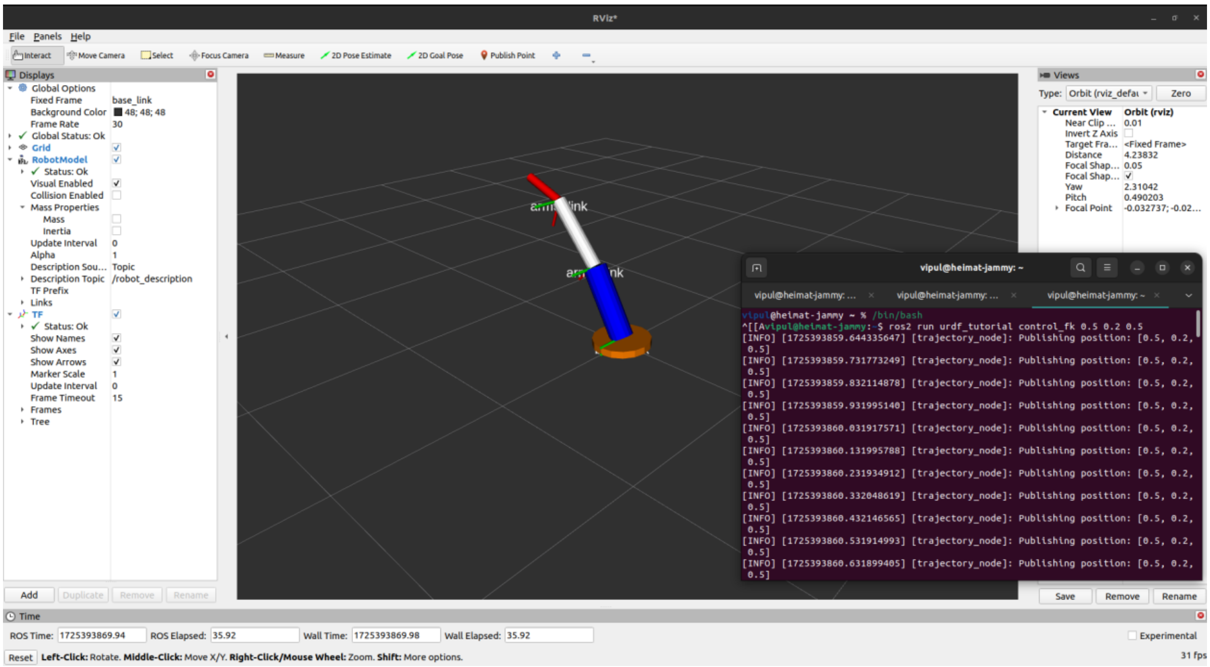
```
ROS Time: 1725393507.56   ROS Elapsed: 60.16   Wall Time: 1725393507.59   Wall Elapsed: 60.16   □ Experimental
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.   31 fps
```

11. Launch arm_control using the launch file

- `ros2 launch urdf_tutorial arm_control.launch.py`

12. Run the control_fk node

- `ros2 run urdf_tutorial control_fk 0.5 0.2 0.5`



Global Options
Fixed Frame: base_link
Background Color: 48; 48; 48
Frame Rate: 30
✓ Global Status: Ok
Grid ☑
RobotModel ☑
✓ Status: Ok
Visual Enabled ☑
Collision Enabled □
Mass Properties
Mass □
Inertia □
Update Interval: 0
Alpha: 1
Description Sou...: Topic
Description Topic: /robot_description
TF Prefix
Links
TF ☑
✓ Status: Ok
Show Names ☑
Show Axes ☑
Show Arrows ☑
Marker Scale: 1
Update Interval: 0
Frame Timeout: 15
Frames
Tree

Type: Orbit (rviz_defau Zero
Current View: Orbit (rviz)
Near Clip ...: 0.01
Invert Z Axis □
Target Fra...: <Fixed Frame>
Distance: 4.23832
Focal Shap...: 0.05
Focal Shap... ☑
Yaw: 2.31042
Pitch: 0.490203
Focal Point: -0.032737; -0.02...

```
vipul@heimat-jammy ~ % /bin/bash
^[[Avipul@heimat-jammy:~$ ros2 run urdf_tutorial control_fk 0.5 0.2 0.5
[INFO] [1725393859.644335647] [trajectory_node]: Publishing position: [0.5, 0.2,
 0.5]
[INFO] [1725393859.731773249] [trajectory_node]: Publishing position: [0.5, 0.2,
 0.5]
[INFO] [1725393859.832114878] [trajectory_node]: Publishing position: [0.5, 0.2,
 0.5]
[INFO] [1725393859.931995140] [trajectory_node]: Publishing position: [0.5, 0.2,
 0.5]
[INFO] [1725393860.031917571] [trajectory_node]: Publishing position: [0.5, 0.2,
 0.5]
[INFO] [1725393860.131995788] [trajectory_node]: Publishing position: [0.5, 0.2,
 0.5]
[INFO] [1725393860.231934912] [trajectory_node]: Publishing position: [0.5, 0.2,
 0.5]
[INFO] [1725393860.332048619] [trajectory_node]: Publishing position: [0.5, 0.2,
 0.5]
[INFO] [1725393860.432146565] [trajectory_node]: Publishing position: [0.5, 0.2,
 0.5]
[INFO] [1725393860.531914993] [trajectory_node]: Publishing position: [0.5, 0.2,
 0.5]
[INFO] [1725393860.631899405] [trajectory_node]: Publishing position: [0.5, 0.2,
 0.5]
```

ROS Time: 1725393869.94 ROS Elapsed: 35.92 Wall Time: 1725393869.98 Wall Elapsed: 35.92 □ Experimental

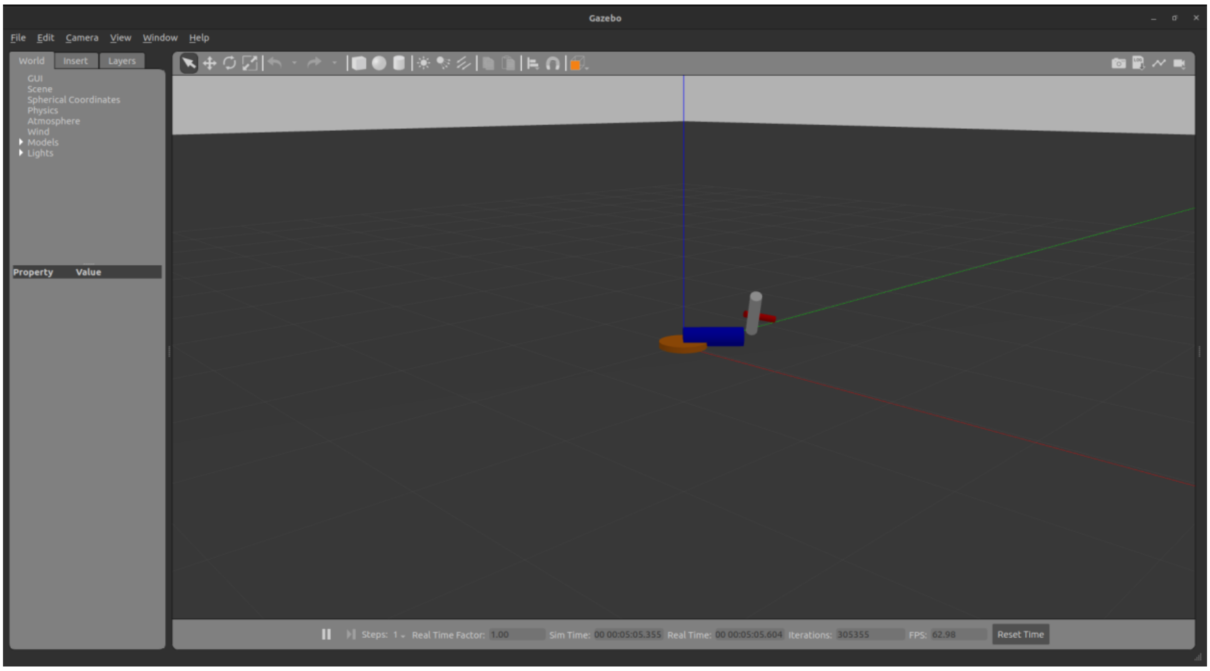
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options. 31 fps

13. Run the following commands in a new terminal to load controllers (with arm_control launch file still running)

- `ros2 control load_controller --set-state active joint_state_broadcaster`
- `ros2 control load_controller --set-state active joint_trajectory_controller`

14. Run the `control` node and pass the parameters as follows

- `ros2 run urdf_tutorial controller --ros-args -p joint_angles:=[3.5,1.5,-1.2]`



Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:05:05.355 Real Time: 00 00:05:05.604 Iterations: 305355 FPS: 62.98