MANIPULATOR DESIGN AND CONTROL

http://wiki.ros.org/urdf/Tutorials/Adding%20Physical%20and%20Collision%20Properties%20to%20a%20URDF%20Model

Execute in Terminal #1

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

```
cd ros2_ws/src/urdf_tutorial/urdf
touch manipulator.urdf
```

```
<?xml version="1.0"?>
<robot name="arm">

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

        <collision>
            <geometry>
                <cylinder length="0.05" radius="0.2"/>
            </geometry>
            <origin rpy="0 0 0" xyz="0 0 0.025"/>
        </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="fixed" type="fixed">
    <parent link="world"/>
    <child link="base_link"/>
    <dynamics damping="10" friction="1.0"/>
</joint>

<link name="link_1">
    <visual>
        <geometry>
            <cylinder length="0.5" radius="0.08"/>
        </geometry>
        <material name="blue">
            <color rgba="0 0 0.8 1"/>
        </material>
        <origin rpy="0 0 0" xyz="0 0 0.25"/>
    </visual>

    <collision>
        <geometry>
            <cylinder length="0.5" radius="0.08"/>
        </geometry>
        <origin rpy="0 0 0" xyz="0 0 0.25"/>
    </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="joint_1" type="continuous">
    <axis xyz="0 0 1"/>
    <parent link="base_link"/>
    <child link="link_1"/>
    <origin rpy="0 0 0" xyz="0.0 0.0 0.05"/>
    <dynamics damping="10" friction="1.0"/>
</joint>

<link name="link_2">
    <inertial>
        <origin rpy="0 0 0" xyz="0 0 0.2"/>
        <mass value="2.0"/>
        <inertia ixx="0.027" ixy="0.0" ixz="0.0" iyy="0.027" iyz="0.0" izz="0.0025"/>
    </inertial>
```

```
    <visual>
        <geometry>
        <cylinder length="0.1" radius="0.08"/>
        </geometry>
        <material name="Red">
            <color rgba="1 0 0 1"/>
        </material>
    </visual>

    <collision>
        <geometry>
            <cylinder length="0.1" radius="0.08"/>
        </geometry>
    </collision>
</link>

<joint name="joint_2" type="continuous">
    <axis xyz="0 0 1"/>
    <parent link="link_1"/>
    <child link="link_2"/>
    <origin rpy="0 1.5708 0" xyz="0.0 -0.005 0.58"/>
    <limit lower="-0.25" upper="3.34" effort="10" velocity="0.5"/>
    <dynamics damping="10" friction="1.0"/>
</joint>

<link name="link_3">
    <inertial>
        <origin rpy="0 0 0" xyz="0 0 0.2"/>
        <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="0.4" radius="0.05"/>
        </geometry>
        <material name="blue">
            <color rgba="0.5 0.5 0.5 1"/>
        </material>
    </visual>

    <collision>
        <geometry>
            <cylinder length="0.4" radius="0.05"/>
        </geometry>
    </collision>
</link>

<joint name="joint_3" type="fixed">
    <parent link="link_2"/>
    <child link="link_3"/>
    <origin rpy="1.57 0 0" xyz="0.0 0.2 0 "/>
```

```
    <dynamics damping="10" friction="1.0"/>
</joint>

<link name="link_4">
    <inertial>
        <origin rpy="0 0 0" xyz="0 0 0.2"/>
        <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="0.1" radius="0.06"/>
        </geometry>
        <material name="Red">
            <color rgba="1 0 0 1"/>
        </material>
    </visual>

    <collision>
        <geometry>
            <cylinder length="0.1" radius="0.06"/>
        </geometry>
    </collision>

</link>

<joint name="joint_4" type="continuous">
    <parent link="link_3"/>
    <child link="link_4"/>
    <origin rpy="1.57 0 0" xyz=" 0 0 -0.25"/>
    <axis xyz=" 0 0 1"/>
    <limit lower="-1.92" upper="1.92" effort="10" velocity="0.5"/>
    <dynamics damping="10" friction="1.0"/>
</joint>

<link name="link_5">
    <inertial>
        <origin rpy="0 0 0" xyz="0 0 0.2"/>
        <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="yello">
            <color rgba="0 1 0.5 1"/>
        </material>
        </visual>
```

```
    <collision>
        <geometry>
            <cylinder length="0.3" radius="0.03"/>
        </geometry>
    <dynamics damping="0.0" friction="0.0"/>

    </collision>
</link>

<joint name="joint_5" type="fixed">
    <parent link="link_4"/>
    <child link="link_5"/>
    <origin rpy="1.57 0 0" xyz="0.0 -0.2 0 "/>
    <dynamics damping="10" friction="1.0"/>
</joint>

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

<gazebo reference="link_1">
    <material>Gazebo/White</material>
</gazebo>

<gazebo reference="link_3">
    <material>Gazebo/White</material>
</gazebo>

<gazebo reference="link_2">
    <material>Gazebo/Blue</material>
</gazebo>

<gazebo reference="link_4">
    <material>Gazebo/Blue</material>
</gazebo>

<gazebo reference="link_5">
    <material>Gazebo/White</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/asha/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="joint_1">
        <command_interface name="position">
        <param name="min">-3.14</param>
        <param name="max">3.14</param>
        </command_interface>
        <state_interface name="position"/>
        <param name="initial_position">0.0</param>
    </joint>
    <joint name="joint_2">
        <command_interface name="position">
        <param name="min">-3.14</param>
        <param name="max">3.14</param>
        </command_interface>
        <state_interface name="position"/>
        <param name="initial_position">-1.57</param>
    </joint>
    <joint name="joint_4">
        <command_interface name="position">
        <param name="min">-3.14</param>
        <param name="max">3.14</param>
        </command_interface>
        <state_interface name="position"/>
        <param name="initial_position">0.0</param>
    </joint>

    </ros2_control>

</robot>
```

```
cd ..
cd launch
touch arm_rviz.launch.py
```

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


def generate_launch_description():

    urdf_file = urdf = '/home/asha/ros2_ws/src/urdf_tutorial/urdf/manipulator.urdf'



    joint_state_publisher_node = Node(
        package="joint_state_publisher_gui",
        executable="joint_state_publisher_gui",
    )
```

```
    robot_state_publisher_node = Node(
        package="robot_state_publisher",
        executable="robot_state_publisher",
        output="both",
        arguments=[urdf_file]
    )
    rviz_node = Node(
        package="rviz2",
        executable="rviz2",
        name="rviz2",
        output="log"
    )

    nodes_to_run = [
        joint_state_publisher_node,
        robot_state_publisher_node,
        rviz_node
    ]

    return LaunchDescription(nodes_to_run)
```

touch 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/asha/ros2_ws/src/urdf_tutorial/urdf/manipulator.urdf'

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


        ]
```

```
    )

cd ~/ros2_ws/src/urdf_tutorial
mkdir config
cd config
touch 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:
      - joint_1
      - joint_2
      - joint_4

    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


touch arm_control.launch.py
Edit arm_control.launch.py

import os
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

import xacro

def generate_launch_description():

    urdf_file = '/home/asha/ros2_ws/src/lab5/urdf/manipulator.urdf'
    controller_file = '/home/asha/ros2_ws/src/lab5/config/control.yaml'
    robot_description = {"robot_description": urdf_file}

    gazebo = IncludeLaunchDescription(
        PythonLaunchDescriptionSource([os.path.join(
            get_package_share_directory('gazebo_ros'), 'launch'), '/gazebo.launch.py']),
    )

    doc = xacro.parse(open(urdf_file))
    xacro.process_doc(doc)
    params = {'robot_description': doc.toxml()}

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

    spawn_entity = Node(package='gazebo_ros', executable='spawn_entity.py',
        arguments=["-entity","lab5","-b","-file", urdf_file],
        output='screen'
    )

    load_joint_state_controller = ExecuteProcess(
        cmd=['ros2', 'control', 'load_controller', '--set-state', 'start',
             'joint_state_broadcaster'],
        output='screen'
    )

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

    return LaunchDescription(
        [
            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"
        )
    ]
)
```

Add extensions in Visual Studio

ros
ros snippet
xml
xml tools
urdf
xml complete
icons

Execute in Terminal #1
colcon build --packages-select urdf_tutorial
Execute in Terminal #1
ros2 launch urdf_tutorial arm_rviz.launch.py
Execute in Terminal #2
ros2 launch urdf_tutorial arm_gazebo.launch.py
Execute in Terminal #3
ros2 launch urdf_tutorial arm_control.launch.py

cd ros2_ws/src/urdf_tutorial/urdf_tutorial/
touch controller.py
chmod +x controller.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.publisher_ = self.create_publisher(JointTrajectory, topic_, 10)
        self.timer_ = self.create_timer(1,self.timer_callback)
        self.joints = ['joint_1', 'joint_2', 'joint_4']
        self.goal_ =[1.5, 0.5, 1.2]


    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()
```

Edit setup.py as

```
from setuptools import setup
import os
from glob import glob
package_name = 'urdf_tutorial'

setup(
    name=package_name,
    version='0.0.0',
    packages=[package_name],
    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='asha',
    maintainer_email='asha@todo.todo',
    description='TODO: Package description',
    license='TODO: License declaration',
    tests_require=['pytest'],
    entry_points={
        'console_scripts': [
            'controller = urdf_tutorial.controller:main'
        ],
    },
)
```

Execute in Terminal #1

```
colcon build --packages-select urdf_tutorial
```

Execute in Terminal #1

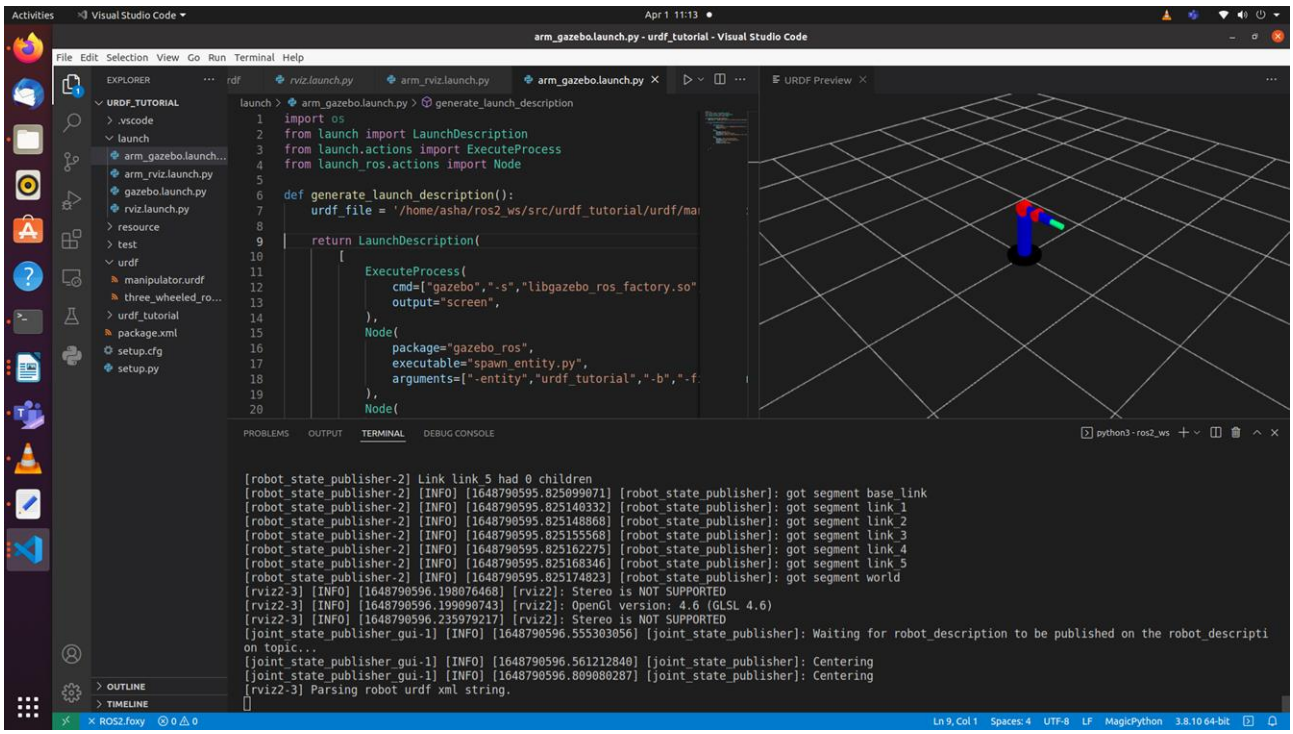
```
ros2 launch urdf_tutorial arm_rviz.launch.py
```

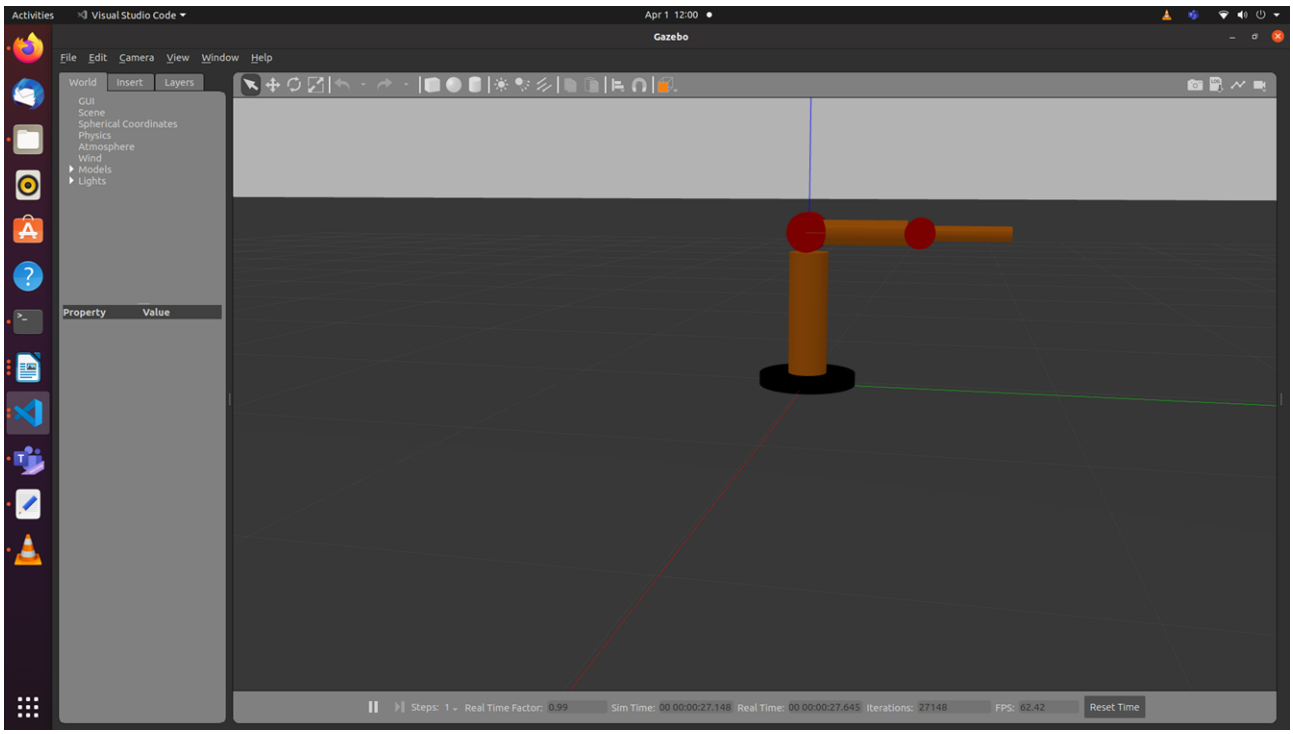
Execute in Terminal #2

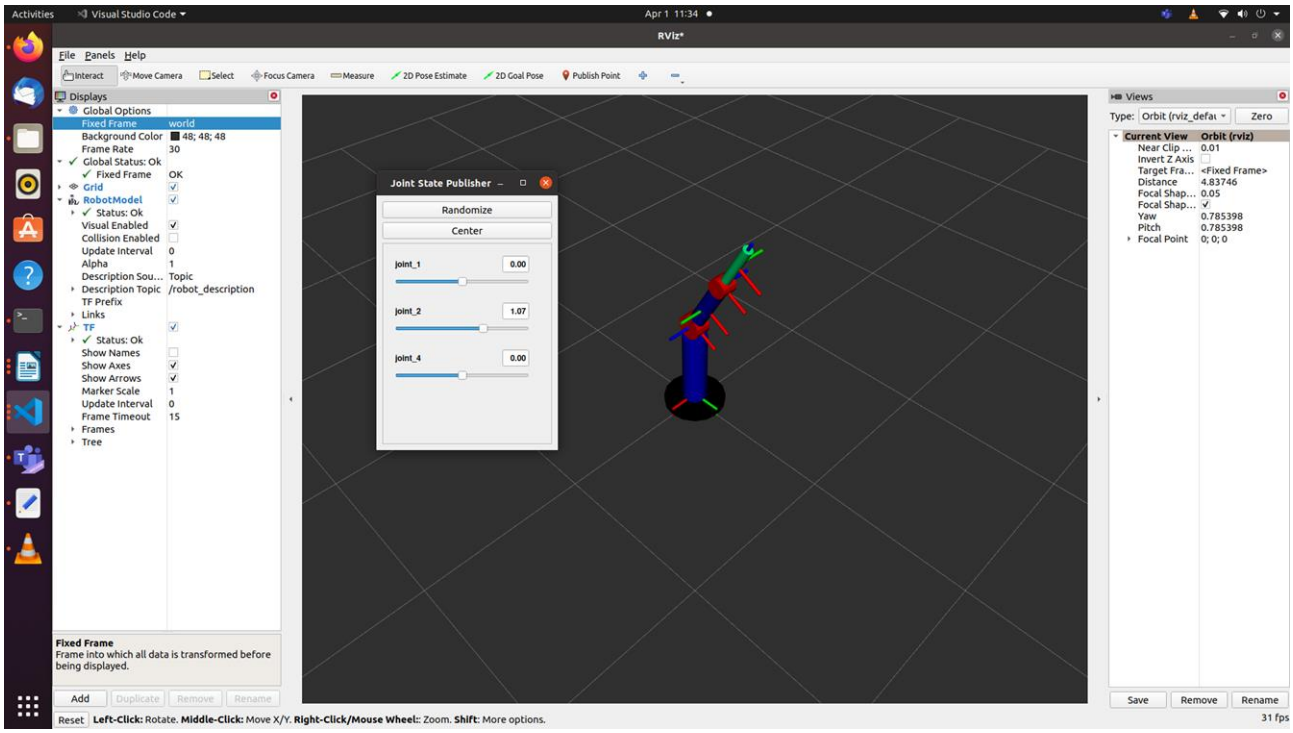
```
ros2 launch urdf_tutorial arm_control.launch.py
```

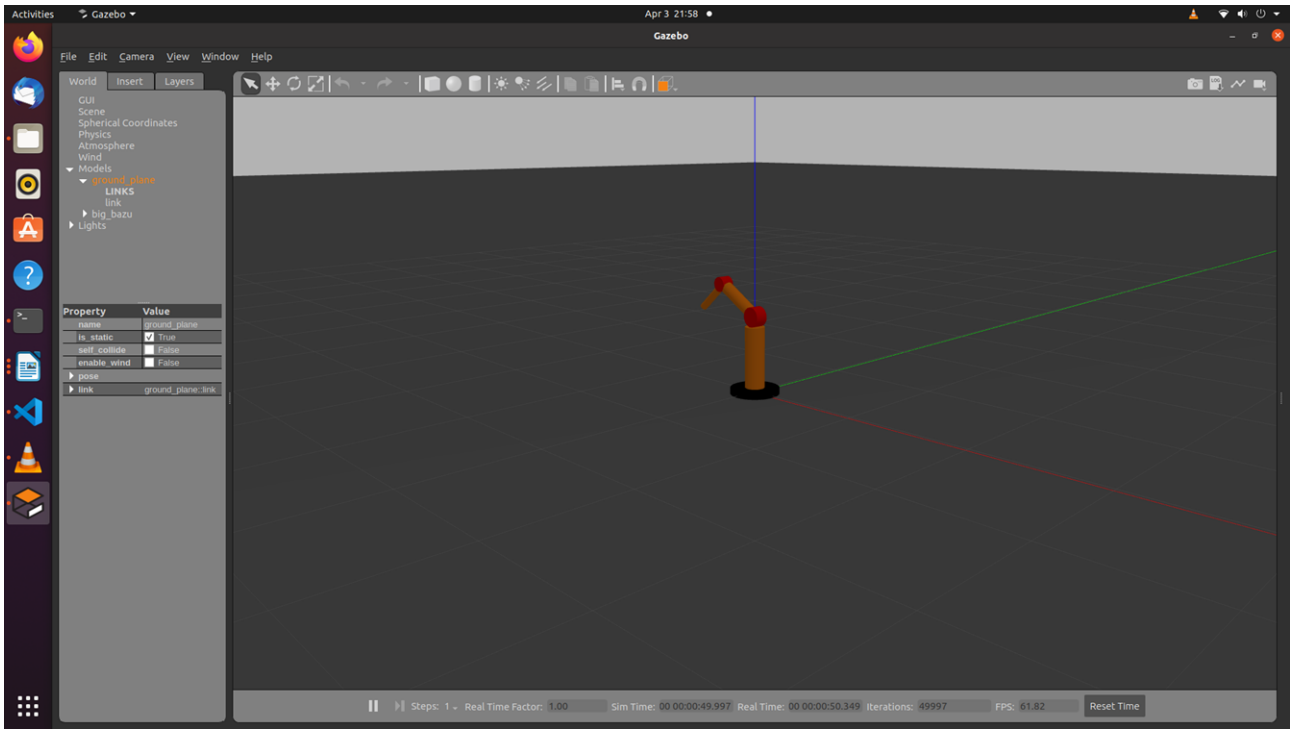
Execute in Terminal #3

```
ros2 run urdf_tutorial controller
```


Activities
Visual Studio Code
Apr 1 11:13
arm_gazebo.launch.py - urdf_tutorial - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
URDF_TUTORIAL
.vscode
launch
arm_gazebo.launch...
arm_rviz.launch.py
gazebo.launch.py
rviz.launch.py
resource
test
urdf
manipulator.urdf
three_wheeled_ro...
urdf_tutorial
package.xml
setup.cfg
setup.py
rdf
rviz.launch.py
arm_rviz.launch.py
arm_gazebo.launch.py
launch > arm_gazebo.launch.py > generate_launch_description
1 import os
2 from launch import LaunchDescription
3 from launch.actions import ExecuteProcess
4 from launch_ros.actions import Node
5
6 def generate_launch_description():
7 urdf_file = '/home/asha/ros2_ws/src/urdf_tutorial/urdf/ma
8
9 return LaunchDescription(
10 [
11 ExecuteProcess(
12 cmd=["gazebo","-s","libgazebo_ros_factory.so"
13 output="screen",
14),
15 Node(
16 package="gazebo_ros",
17 executable="spawn_entity.py",
18 arguments=["-entity","urdf_tutorial","-b","-f
19),
20 Node(
URDF Preview
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
python3 - ros2_ws
[robot_state_publisher-2] Link link_5 had 0 children
[robot_state_publisher-2] [INFO] [1648790595.825099071] [robot_state_publisher]: got segment base_link
[robot_state_publisher-2] [INFO] [1648790595.825140332] [robot_state_publisher]: got segment link_1
[robot_state_publisher-2] [INFO] [1648790595.825148868] [robot_state_publisher]: got segment link_2
[robot_state_publisher-2] [INFO] [1648790595.825155568] [robot_state_publisher]: got segment link_3
[robot_state_publisher-2] [INFO] [1648790595.825162275] [robot_state_publisher]: got segment link_4
[robot_state_publisher-2] [INFO] [1648790595.825168346] [robot_state_publisher]: got segment link_5
[robot_state_publisher-2] [INFO] [1648790595.825174823] [robot_state_publisher]: got segment world
[rviz2-3] [INFO] [1648790596.198076468] [rviz2]: Stereo is NOT SUPPORTED
[rviz2-3] [INFO] [1648790596.199090743] [rviz2]: OpenGl version: 4.6 (GLSL 4.6)
[rviz2-3] [INFO] [1648790596.235979217] [rviz2]: Stereo is NOT SUPPORTED
[joint_state_publisher_gui-1] [INFO] [1648790596.555303056] [joint_state_publisher]: Waiting for robot_description to be published on the robot_descripti
on topic...
[joint_state_publisher_gui-1] [INFO] [1648790596.561212840] [joint_state_publisher]: Centering
[joint_state_publisher_gui-1] [INFO] [1648790596.809080287] [joint_state_publisher]: Centering
[rviz2-3] Parsing robot urdf xml string.
OUTLINE
TIMELINE
ROS2.foxy
Ln 9, Col 1
Spaces: 4
UTF-8
LF
MagicPython
3.8.10 64-bit


Activities
Visual Studio Code
Apr 1 12:00
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Steps: 1 Real Time Factor: 0.99 Sim Time: 00 00:00:27.148 Real Time: 00 00:00:27.645 Iterations: 27148 FPS: 62.42
Reset Time


Activities
Visual Studio Code
Apr 1 11:34
RViz*
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Goal Pose Publish Point
Displays
Global Options
Fixed Frame world
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid
RobotModel
Status: Ok
Visual Enabled
Collision Enabled
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
Fixed Frame
Frame into which all data is transformed before being displayed.
Add Duplicate Remove Rename
Joint State Publisher
Randomize
Center
joint_1 0.00
joint_2 1.07
joint_4 0.00
Views
Type: Orbit (rviz_defa
Zero
Current View Orbit (rviz)
Near Clip ... 0.01
Invert Z Axis
Target Fra... <Fixed Frame>
Distance 4.83746
Focal Shap... 0.05
Focal Shap...
Yaw 0.785398
Pitch 0.785398
Focal Point 0; 0; 0
Save Remove Rename
Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
31 fps



ROS2 parameters:

Parameters help to provide the values while running the code.

ros2 param list

Edit the controller.py

```
#!/usr/bin/env python3


#colcon build --packages-select urdf_tutorial
#ros2 run urdf_tutorial controller --ros-args -p end_location:=[3.5,1.5,-1.2]

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 = ['joint_1', 'joint_2', 'joint_4']
        #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()
```

Execute in Terminal #1

```
#colcon build --packages-select urdf_tutorial
```

Execute in Terminal #2

```
ros2 launch urdf_tutorial arm_control.launch.py
ros2 control load_controller --set-state start joint_state_broadcaster
ros2 control load_controller --set-state start joint_trajectory_controller
```

Execute in Terminal #3

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

Exercise 1: Replicate the process for UR5e robot given its urdf file.

Exercise 2: Write a python code to move the manipulator to end location using inverse kinematics.

Viva Questions: Compute the inertia parameters for the each block used in the three_wheeled_robot and manipulator.

References

https://docs.ros.org/en/foxy/Tutorials/URDF/Using-URDF-with-Robot-State-Publisher.html

https://github.com/benbongalon/ros2-urdf-tutorial/tree/master/urdf_tutorial

https://github.com/cra-ros-pkg/robot_localization/tree/foxy-devel

https://github.com/ros/robot_state_publisher/tree/foxy

https://github.com/ros/joint_state_publisher/tree/foxy

http://gazebosim.org/tutorials?tut=ros_urdf