Robotics Lab: Homework 1

Building your robot manipulator

Giulia Gelsomina Romano, P38000223

Group: Emanuele Cuzzocrea, Silvia Leo, Vito Daniele Perfetta, Giulia Gelsomina Romano.

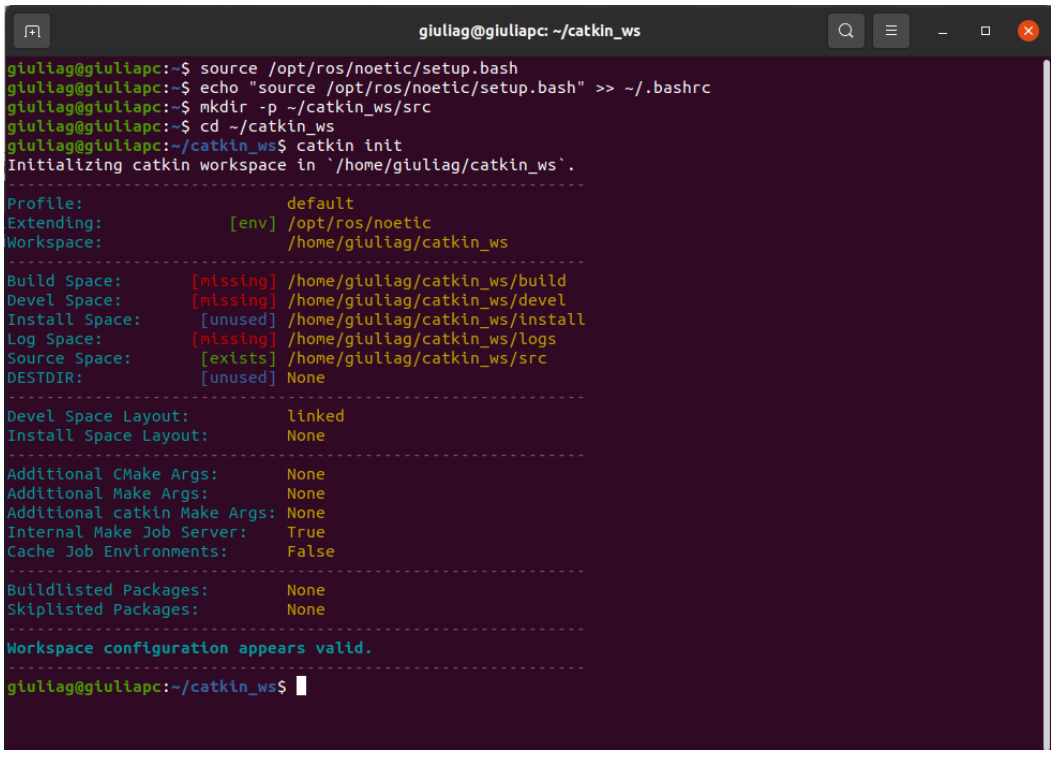
Here is the link to my public repo on github: https://github.com/giuliagromano/rl_hw1.git

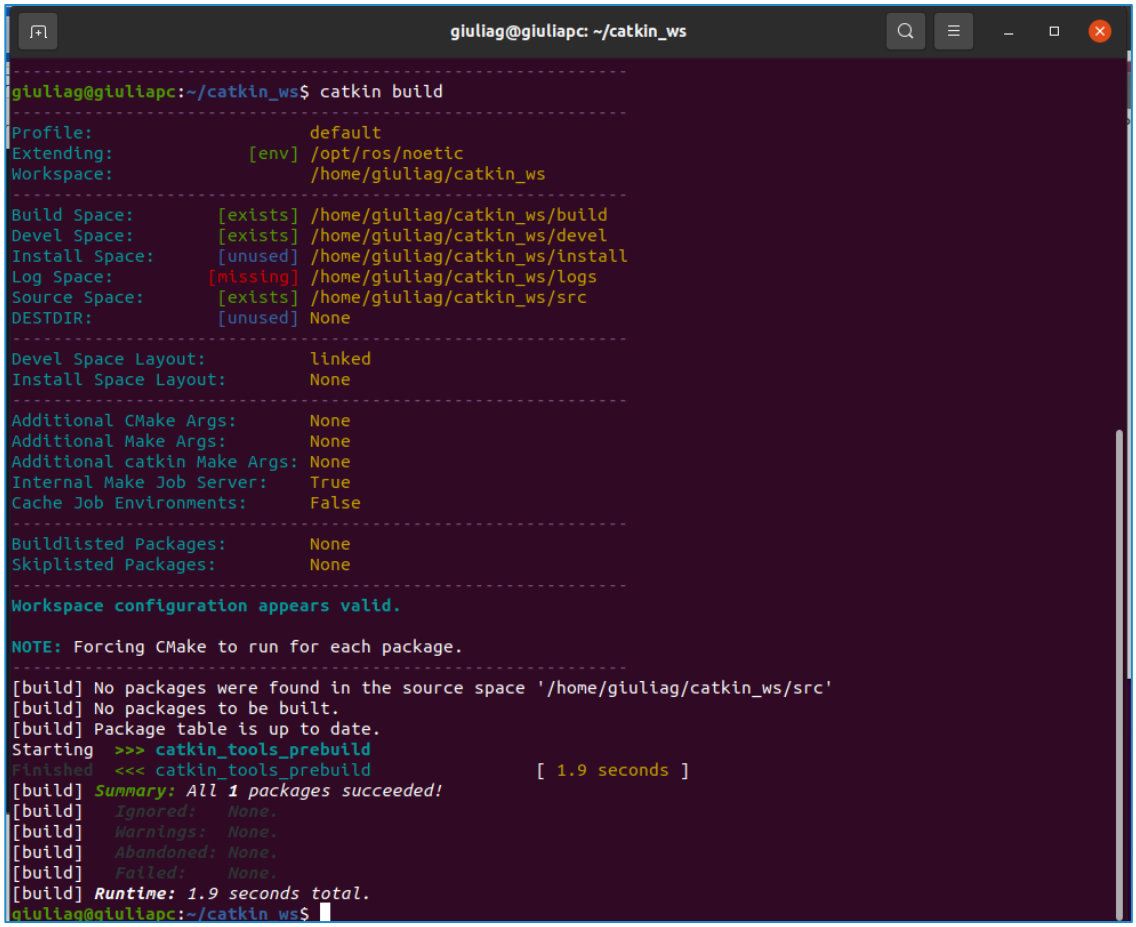
1. Create the description of your robot and visualize it in Rviz

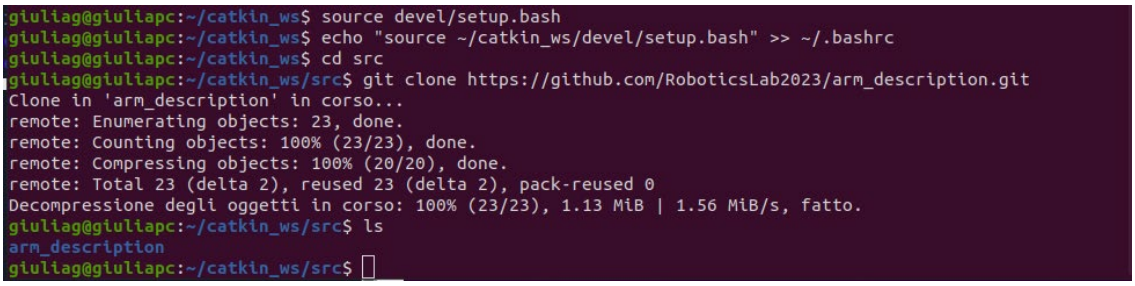
a. Download the arm_description package from the repo https://github.com/RoboticsLab2023/arm_description.git into your catkin_ws using git commands

Used commands

$ source /opt/ros/noetic/setup.bash
$ echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws
$ catkin init
$ catkin build
$ source devel/setup.bash
$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
$ cd src
$ git clone https://github.com/RoboticsLab2023/arm_description.git
$ ls



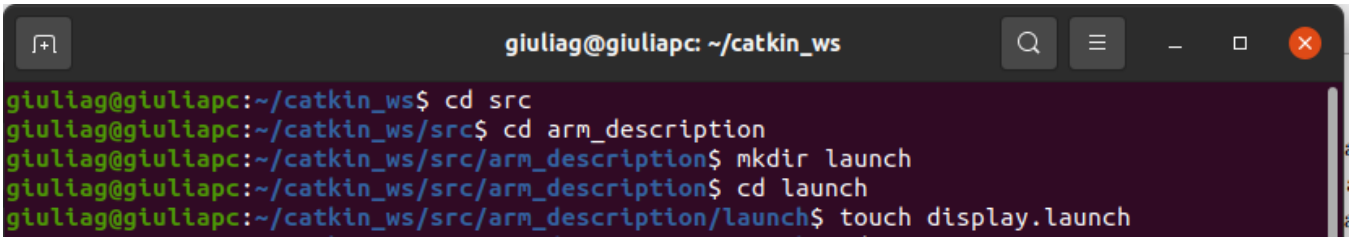


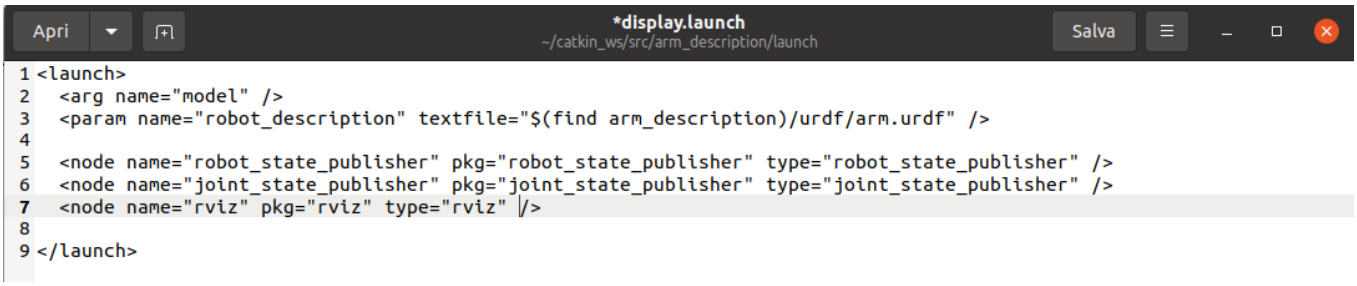


b. Within the package create a launch folder containing a launch file named display.launch that loads the URDF as a robot_description ROS param and starts the robot_state_publisher node, the joint_state_publisher node, and the rviz node. Launch the file using roslaunch

Used commands

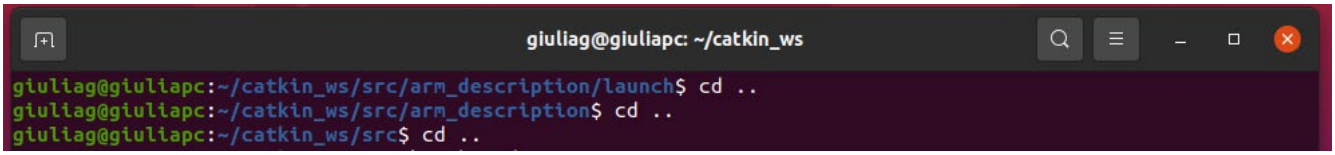
$ cd src
$ cd arm_description
$ mkdir launch
$ cd launch
$ touch display.launch

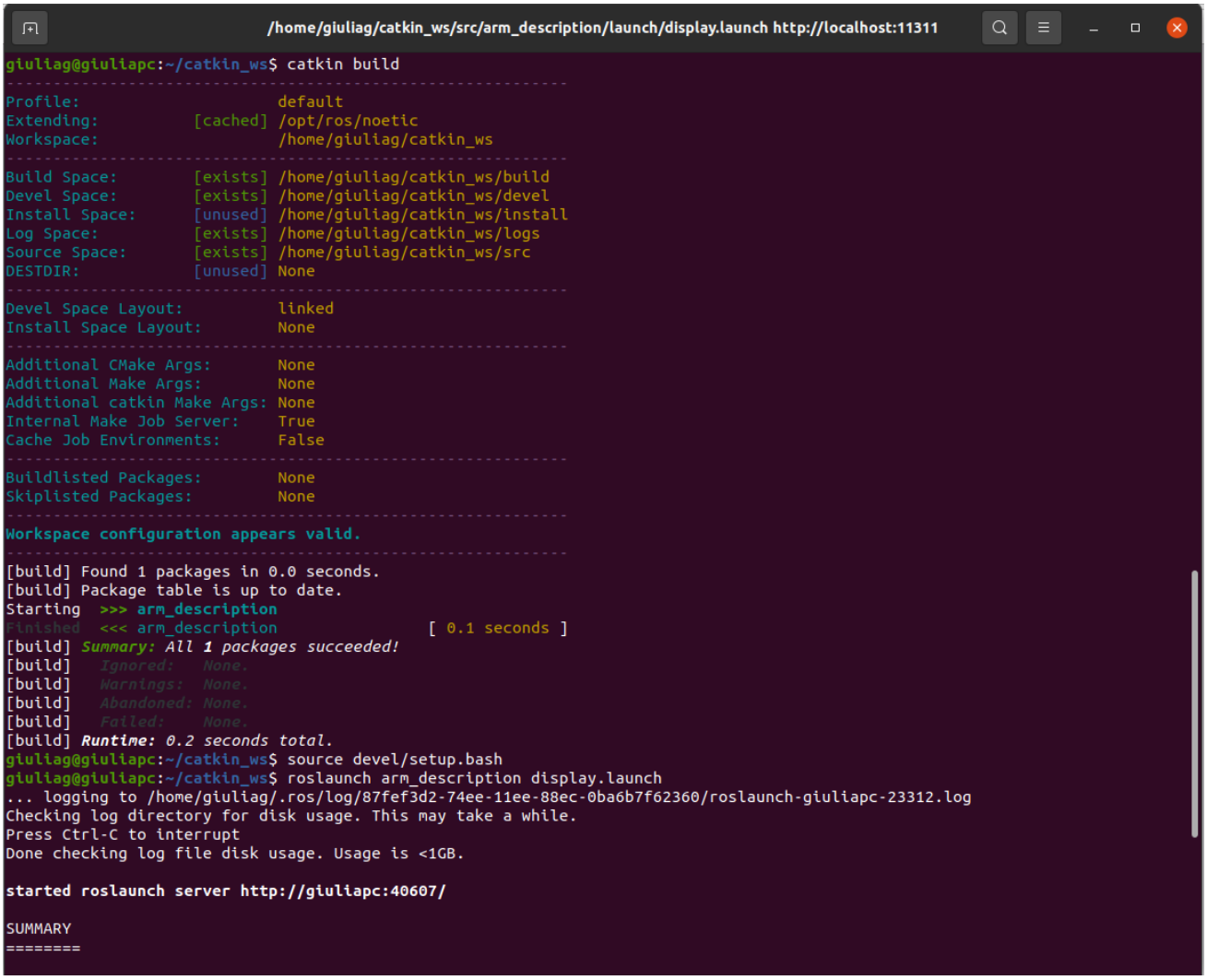




Used commands

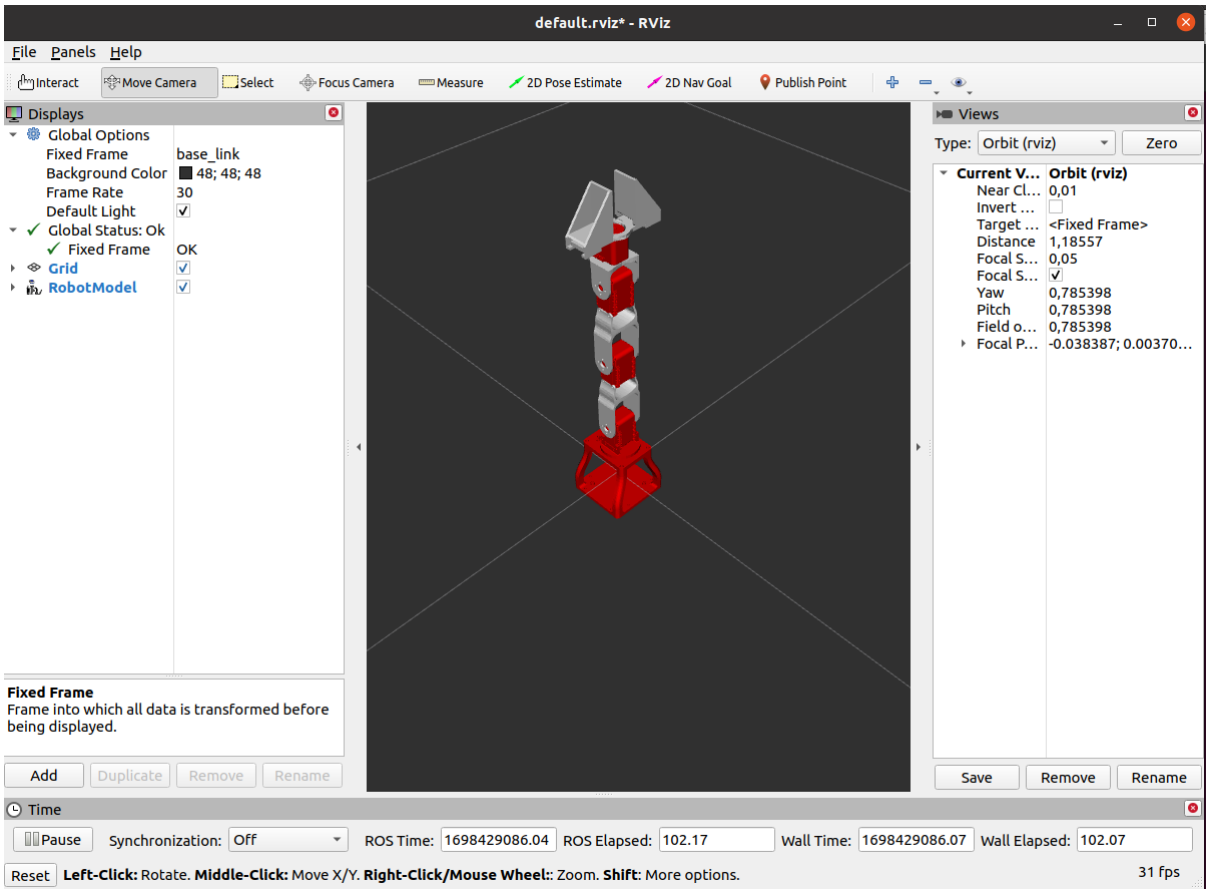
$ cd ..
$ cd ..
$ cd ..
$ catkin build
$ source devel/setup.bash
$ roslaunch arm_description display.launch




/home/giuliag/catkin_ws/src/arm_description/launch/display.launch http://localhost:11311
giuliag@giuliapc:~/catkin_ws$ catkin build
Profile: default
Extending: [cached] /opt/ros/noetic
Workspace: /home/giuliag/catkin_ws
Build Space: [exists] /home/giuliag/catkin_ws/build
Devel Space: [exists] /home/giuliag/catkin_ws/devel
Install Space: [unused] /home/giuliag/catkin_ws/install
Log Space: [exists] /home/giuliag/catkin_ws/logs
Source Space: [exists] /home/giuliag/catkin_ws/src
DESTDIR: [unused] None
Devel Space Layout: linked
Install Space Layout: None
Additional CMake Args: None
Additional Make Args: None
Additional catkin Make Args: None
Internal Make Job Server: True
Cache Job Environments: False
Buildlisted Packages: None
Skiplisted Packages: None
Workspace configuration appears valid.
[build] Found 1 packages in 0.0 seconds.
[build] Package table is up to date.
Starting >>> arm_description
Finished <<< arm_description [0.1 seconds]
[build] Summary: All 1 packages succeeded!
[build] Ignored: None.
[build] Warnings: None.
[build] Abandoned: None.
[build] Failed: None.
[build] Runtime: 0.2 seconds total.
giuliag@giuliapc:~/catkin_ws$ source devel/setup.bash
giuliag@giuliapc:~/catkin_ws$ roslaunch arm_description display.launch
... logging to /home/giuliag/.ros/log/87fef3d2-74ee-11ee-88ec-0ba6b7f62360/roslaunch-giuliapc-23312.log
Checking log directory for disk usage. This may take a while.
Press Ctrl-C to interrupt
Done checking log file disk usage. Usage is <1GB.
started roslaunch server http://giuliapc:40607/
SUMMARY
========

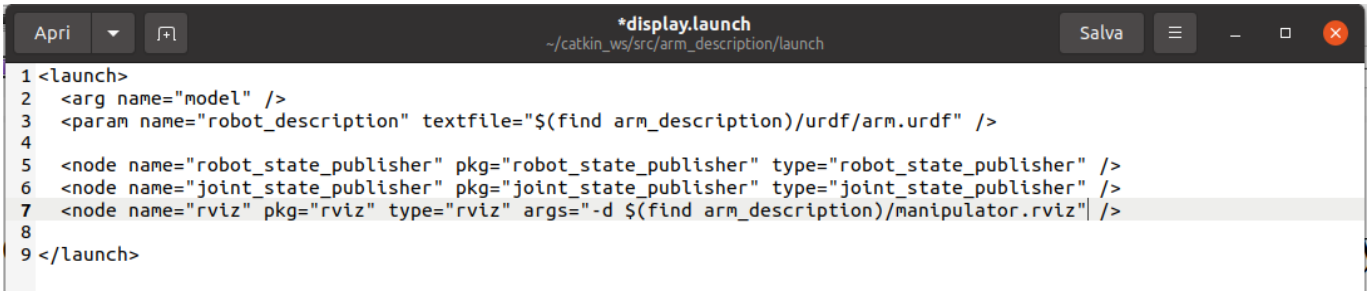

PARAMETERS
* /robot_description: <?xml version="1....
* /rosdistro: noetic
* /rosversion: 1.16.0
NODES
/
joint_state_publisher (joint_state_publisher/joint_state_publisher)
robot_state_publisher (robot_state_publisher/robot_state_publisher)
rviz (rviz/rviz)
auto-starting new master
process[master]: started with pid [23320]
ROS_MASTER_URI=http://localhost:11311
setting /run_id to 87fef3d2-74ee-11ee-88ec-0ba6b7f62360
process[rosout-1]: started with pid [23330]
started core service [/rosout]
process[robot_state_publisher-2]: started with pid [23333]
process[joint_state_publisher-3]: started with pid [23334]
process[rviz-4]: started with pid [23339]
[WARN] [1698427833.410640252]: The root link base_link has an inertia specified in the URDF, but KDL does not support a root link with an inertia. As a workaround, you can add an extra dummy link to your URDF.

Once the display.launch is executed, adding the RobotModel plugin interface and changing the fixed frame into "base_link", the robot is visualized in Rviz.



Optional: save a .rviz configuration file, thad automatically loads the RobotModel plugin by default, and give it as an argument to your node in the display.launch file.

Modify the dispay.launch in order to save the .rviz configuration file:

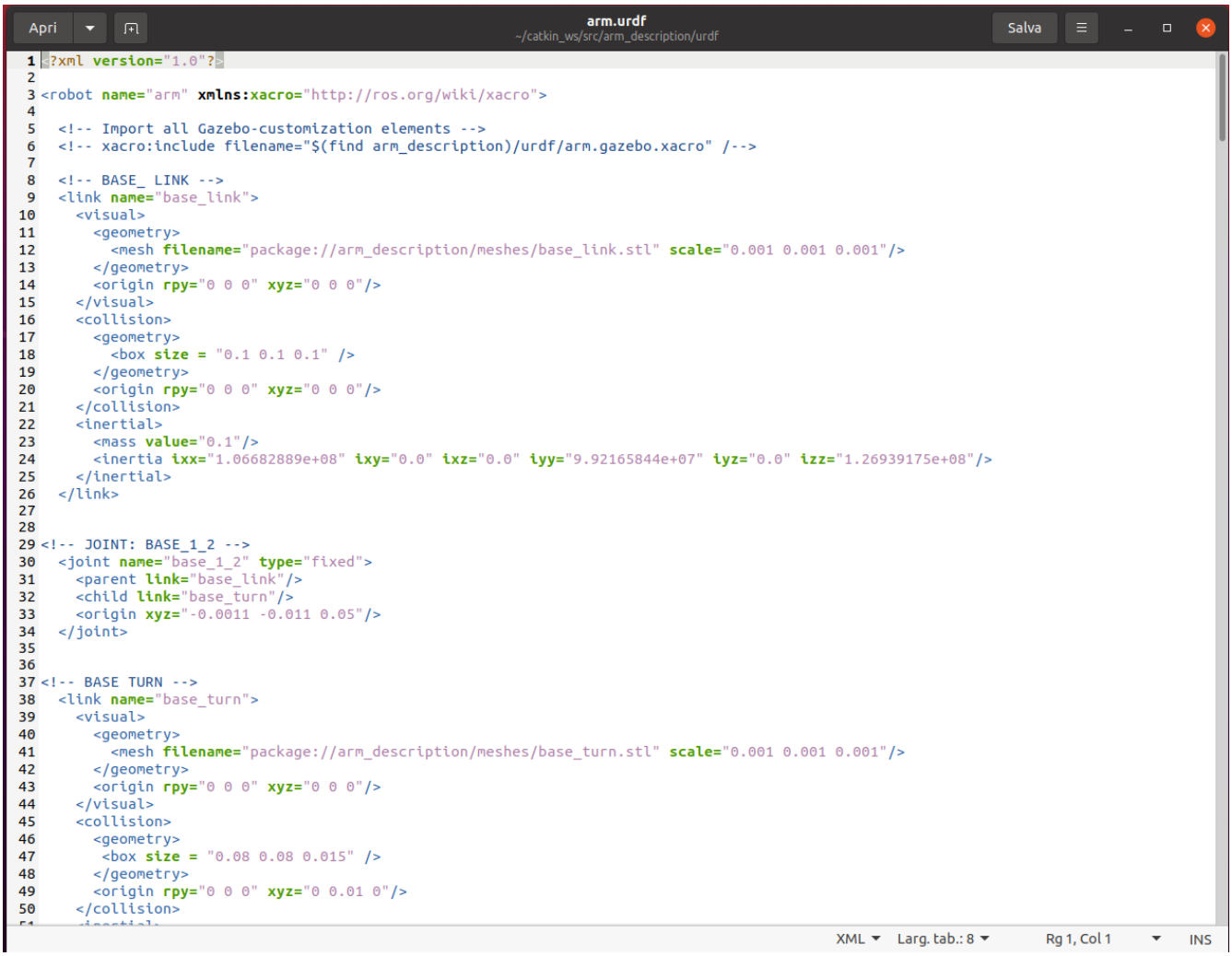


Used commands

```
$ catkin build
$ source devel/setup.bash
$ roslaunch arm_description display.launch
```

c. Substitute the collision meshes of your URDF with primitive shapes. Use geometries of reasonabe size approximating the links.

The arm.urdf file has been modified replaicing the collision meshes by box of reasonable size as in the following picture:




```
<?xml version="1.0"?>

<robot name="arm" xmlns:xacro="http://ros.org/wiki/xacro">

  <!-- Import all Gazebo-customization elements -->
  <!-- xacro:include filename="$(find arm_description)/urdf/arm.gazebo.xacro" /-->

  <!-- BASE_ LINK -->
  <link name="base_link">
    <visual>
      <geometry>
        <mesh filename="package://arm_description/meshes/base_link.stl" scale="0.001 0.001 0.001"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0"/>
    </visual>
    <collision>
      <geometry>
        <box size = "0.1 0.1 0.1" />
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0"/>
    </collision>
    <inertial>
      <mass value="0.1"/>
      <inertia ixx="1.06682889e+08" ixy="0.0" ixz="0.0" iyy="9.92165844e+07" iyz="0.0" izz="1.26939175e+08"/>
    </inertial>
  </link>


<!-- JOINT: BASE_1_2 -->
  <joint name="base_1_2" type="fixed">
    <parent link="base_link"/>
    <child link="base_turn"/>
    <origin xyz="-0.0011 -0.011 0.05"/>
  </joint>


<!-- BASE TURN -->
  <link name="base_turn">
    <visual>
      <geometry>
        <mesh filename="package://arm_description/meshes/base_turn.stl" scale="0.001 0.001 0.001"/>
      </geometry>
      <origin rpy="0 0 0" xyz="0 0 0"/>
    </visual>
    <collision>
      <geometry>
       <box size = "0.08 0.08 0.015" />
      </geometry>
      <origin rpy="0 0 0" xyz="0 0.01 0"/>
    </collision>
```

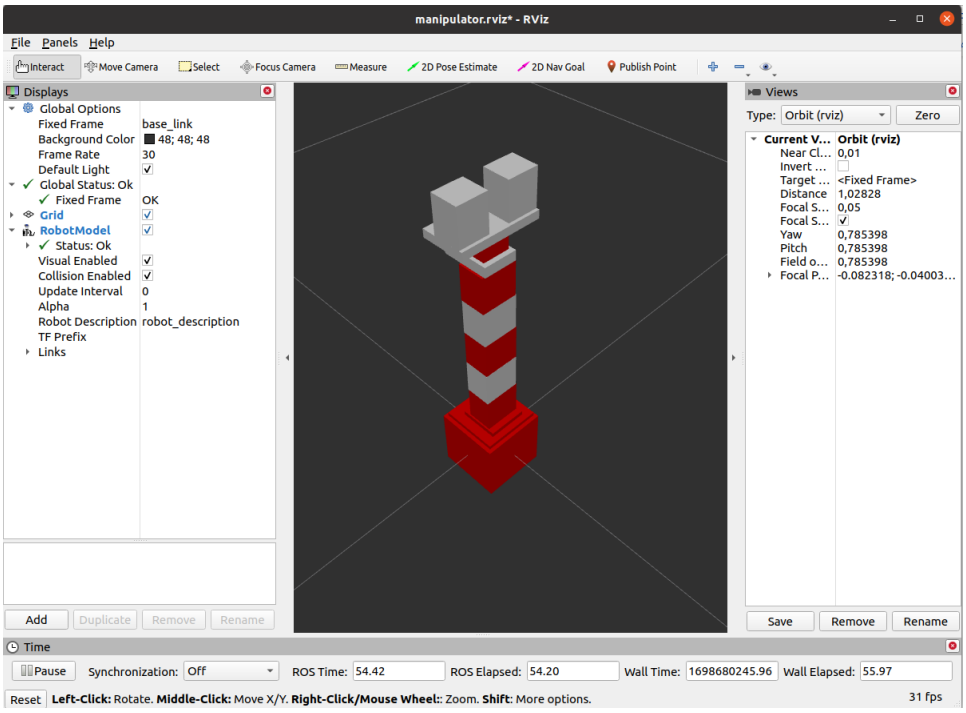
Used commands

$ catkin build

$ source devel/setup.bash

$ roslaunch arm_description display.launch

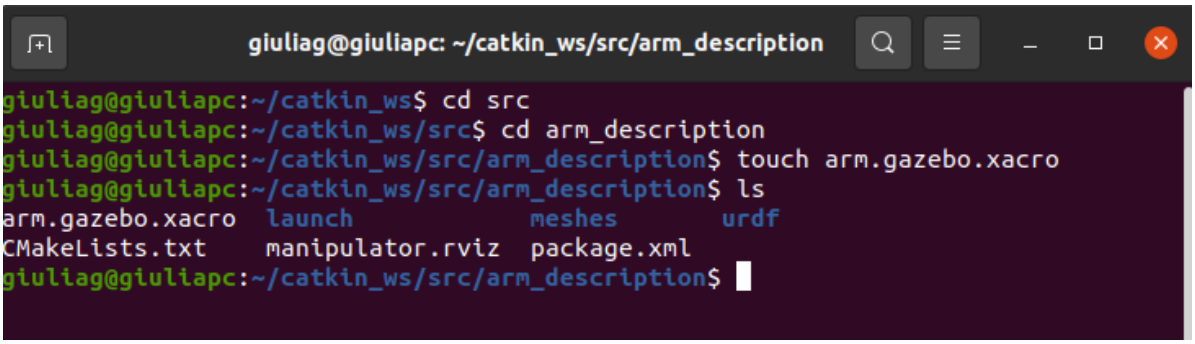
With the collision enabled, the robot is visualized in Rviz as:



d. Create a file named arm.gazebo.xacro within your package, define a xacro:macro inside your file containing all the tags you find within your arm.urdf and import it in your URDF using xacro:include.

Used commands

$ cd src
$ cd arm_description
$ touch arm.gazebo.xacro
$ ls



Apri arm.gazebo.xacro ~/catkin_ws/src/arm_description/urdf Salva

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://ros.org/wiki/xacro">

<xacro:macro name="arm_gazebo">
    <gazebo reference="f4">
    <material>Gazebo/Red</material>
  </gazebo>

  <gazebo reference="f5">
    <material>Gazebo/Red</material>
  </gazebo>

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

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

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

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

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

  <gazebo reference="dyn2">
    <material>Gazebo/Black</material>
  </gazebo>

  <gazebo reference="dyn3">
    <material>Gazebo/Black</material>
  </gazebo>

  <gazebo reference="dyn4">
    <material>Gazebo/Black</material>
  </gazebo>

  <gazebo reference="dyn5">
    <material>Gazebo/Black</material>
  </gazebo>

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


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

</xacro:macro>
</robot>

```

XML Larg. tab.: 8 Rg 30, Col 21 INS

In the arm.urdf.xacro file the above <gazebo reference> are commented out and the arm.gazebo.xacro is included.

Apri *arm.urdf.xacro ~/catkin_ws/src/arm_description/urdf Salva

```
<?xml version="1.0"?>

<robot name="arm" xmlns:xacro="http://ros.org/wiki/xacro">

  <!-- Import all Gazebo-customization elements -->
  <xacro:include filename="$(find arm_description)/urdf/arm.gazebo.xacro" />
```

…

```
  <xacro:arm_gazebo/>

</robot>
```

XML Larg. tab.: 8 Rg 467, Col 14 INS

The display.launch is modified as:

Apri display.launch ~/catkin_ws/src/arm_description/launch Salva

```
<?xml version="1.0"?>
<launch>
  <arg name="model" />
  <param name="robot_description" command=" $(find xacro)/xacro --inorder '$(find arm_description)/urdf/arm.urdf.xacro'"/>

  <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" />
  <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find arm_description)/manipulator.rviz" />


</launch>
```

Used commands

$ catkin build

$ source devel/setup.bash

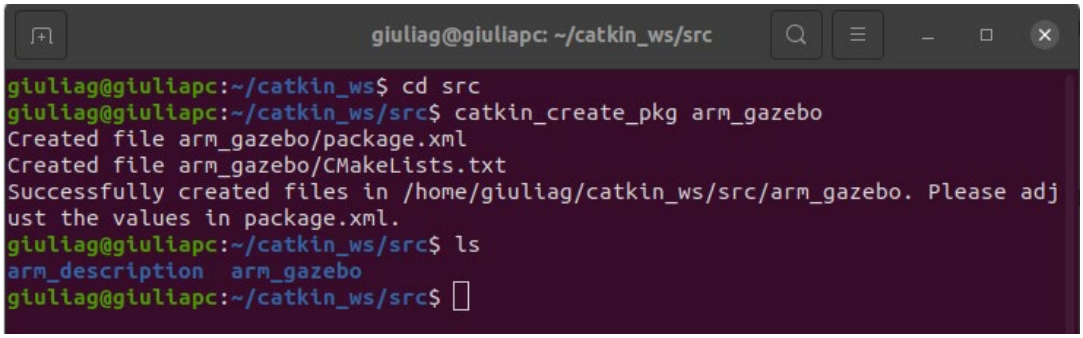
$ roslaunch arm_description display.launch

2. Add transmission and controllers to your robot and spawn it in Gazebo.

a. Create a package named arm_gazebo

Used commands

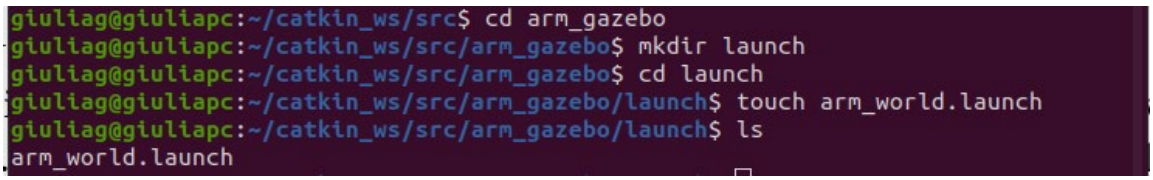
$ cd src
$ carkin_create_pkg arm_gazebo
$ ls



b. Within this package create a launch folder containing a arm_world.launch file

Used commands

$ cd arm_gazebo
$ mkdir launch
$ cd launch
$ touch arm.world.launch
$ ls



c. Fill this launch file with commands that load the URDF into the ROS Parameter Server and spawn your robot using the spawn_model node.

Used commands

$ cd src/arm_gazebo
$ mkdir world
$ cd world
$ touch arm.world
$ ls

```
giuliag@giuliapc: ~/catkin_ws/src/arm_gazebo
giuliag@giuliapc:~/catkin_ws$ cd src
giuliag@giuliapc:~/catkin_ws/src$ cd arm_gazebo
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo$ mkdir world
```

```
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo$ cd world
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo/world$ touch arm.world
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo/world$ ls
arm.world
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo/world$
```

A arm.world file is created to describe the world around the robot.

arm.world
~/catkin_ws/src/arm_gazebo/world

```
<?xml version="1.0" ?>
<sdf version="1.4">
  <!-- We use a custom world for the iiwa so that the camera angle is launched correctly. -->
  <!-- One can change this world to his needs or use another one. -->

  <world name="default">

    <include>
      <uri>model://ground_plane</uri>
    </include>

    <!-- Global light source -->
    <include>
      <uri>model://sun</uri>
    </include>

    <physics name="default_physics" default="0" type="ode">
      <max_step_size>0.01</max_step_size>
      <real_time_factor>1</real_time_factor>
      <real_time_update_rate>100</real_time_update_rate>
      <ode>
        <solver>
          <type>quick</type>
          <iters>50</iters>
          <sor>1.0</sor> <!-- Important, see issue #2209 -->
          <use_dynamic_moi_rescaling>false</use_dynamic_moi_rescaling>
        </solver>
      </ode>
    </physics>

    <!-- Focus camera -->
    <gui fullscreen='0'>
      <camera name='user_camera'>
        <pose>4.927360 -4.376610 3.740080 0.000000 0.275643 2.356190</pose>
        <view_controller>orbit</view_controller>
      </camera>
    </gui>

  </world>

</sdf>
```

The arm_world.launch is modified as:

arm_world.launch
~/catkin_ws/src/arm_gazebo/launch

```
<?xml version="1.0"?>
<launch>

    <!-- Loads the arm.world environment in Gazebo. -->

    <arg name="paused" default="false"/>
    <arg name="use_sim_time" default="true"/>
    <arg name="gui" default="true"/>
    <arg name="headless" default="false"/>
    <arg name="debug" default="false"/>
    <arg name="hardware_interface" default="PositionJointInterface"/>
    <arg name="robot_name" default="arm" />
    <arg name="model" default="arm5"/>

    <!-- We resume the logic in empty_world.launch, changing only the name of the world to be launched -->
    <include file="$(find gazebo_ros)/launch/empty_world.launch">
        <arg name="world_name" value="$(find arm_gazebo)/worlds/arm.world"/>
        <arg name="debug" value="$(arg debug)" />
        <arg name="gui" value="$(arg gui)" />
        <arg name="paused" value="$(arg paused)"/>
        <arg name="use_sim_time" value="$(arg use_sim_time)"/>
        <arg name="headless" value="$(arg headless)"/>
    </include>

    <!-- Load the URDF into the ROS Parameter Server  -->
    <include file="$(find arm_description)/launch/display.launch">


    <!-- Run a python script to send a service call to gazebo_ros to spawn a URDF robot -->
    <node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
    args="-urdf -model arm -param robot_description"/>


</launch>

```

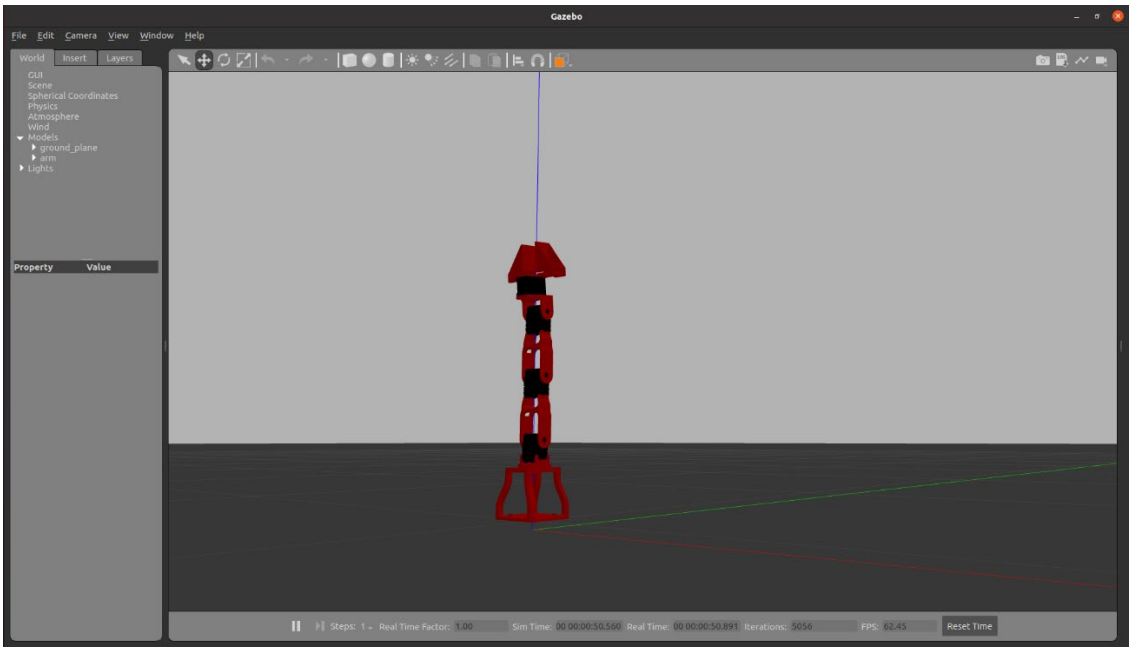
To display the robot in Gazebo:

Used commands

$ catkin build

$ source devel/setup.bash

$ roslaunch arm_gazebo arm_world.launch



d. Now add a PositionJointInterface as hardware interface to your robot: create a arm.transmission.xacro file into your arm_description/urdf folder containing a xacro:macro with the hardware interface and load it into your arm.urdf.xacro file using xacro:include. Launch the file.

Used commands

$ cd src/arm_description/udrf/

$ touch arm.trasmission.xacro

$ ls

```
giuliag@giuliapc:~/catkin_ws$ cd src
giuliag@giuliapc:~/catkin_ws/src$ cd arm_description
giuliag@giuliapc:~/catkin_ws/src/arm_description$ cd urdf/
giuliag@giuliapc:~/catkin_ws/src/arm_description/urdf$ touch arm.transmission.xacro
giuliag@giuliapc:~/catkin_ws/src/arm_description/urdf$ ls
arm.gazebo.xacro  arm.transmission.xacro  arm.urdf.xacro
giuliag@giuliapc:~/catkin_ws/src/arm_description/urdf$
```

To add the hardware interface the following files are modified as:

Apri — display.launch
~/catkin_ws/src/arm_description/launch

```
<?xml version="1.0"?>
<launch>

  <arg name="hardware_interface" default="PositionJointInterface"/>

  <param name="robot_description" command=" $(find xacro)/xacro --inorder '$(find arm_description)/urdf/arm.urdf.xacro' hardware_interface:=$(arg hardware_interface)"/>

  <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" />
  <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />
  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find arm_description)/manipulator.rviz" />


</launch>
```

Apri — arm.urdf.xacro
~/catkin_ws/src/arm_description/urdf

```
<?xml version="1.0"?>

<robot name="arm" xmlns:xacro="http://ros.org/wiki/xacro">

  <arg name="hardware_interface" default="PositionJointInterface"/>

  <xacro:include filename="$(find arm_description)/urdf/arm.gazebo.xacro" />

  <xacro:include filename="$(find arm_description)/urdf/arm.transmission.xacro" />

```

Apri — arm_world.launch
~/catkin_ws/src/arm_gazebo/launch

```
<?xml version="1.0"?>
<launch>

    <!-- Loads the arm.world environment in Gazebo. -->

    <arg name="paused" default="false"/>
    <arg name="use_sim_time" default="true"/>
    <arg name="gui" default="true"/>
    <arg name="headless" default="false"/>
    <arg name="debug" default="false"/>
    <arg name="hardware_interface" default="PositionJointInterface"/>
    <arg name="robot_name" default="arm" />
    <arg name="model" default="arm5"/>

    <!-- We resume the logic in empty_world.launch, changing only the name of the world to be launched -->
    <include file="$(find gazebo_ros)/launch/empty_world.launch">
        <arg name="world_name" value="$(find arm_gazebo)/worlds/arm.world"/>
        <arg name="debug" value="$(arg debug)" />
        <arg name="gui" value="$(arg gui)" />
        <arg name="paused" value="$(arg paused)"/>
        <arg name="use_sim_time" value="$(arg use_sim_time)"/>
        <arg name="headless" value="$(arg headless)"/>
    </include>

    <!-- Load the URDF into the ROS Parameter Server  -->
    <include file="$(find arm_description)/launch/display.launch">
      <arg name="hardware_interface" value="$(arg hardware_interface)"/>
    </include>

    <!-- Run a python script to send a service call to gazebo_ros to spawn a URDF robot -->
    <node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
    args="-urdf -model arm -param robot_description"/>


    </launch>
```

Apri

arm.transmission.xacro
~/catkin_ws/src/arm_description/urdf

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://ros.org/wiki/xacro">
<xacro:macro name="arm_transmission" params="hardware_interface">

  <transmission name="arm_tran_0">
      <robotNamespace>/arm</robotNamespace>
      <type>transmission_interface/SimpleTransmission</type>
      <joint name="j0">
        <hardwareInterface>hardware_interface/${hardware_interface}</hardwareInterface>
      </joint>
      <actuator name="arm_motor_0">
        <hardwareInterface>hardware_interface/${hardware_interface}</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
      </actuator>
    </transmission>

    <transmission name="arm_tran_1">
      <robotNamespace>/arm</robotNamespace>
      <type>transmission_interface/SimpleTransmission</type>
      <joint name="j1">
        <hardwareInterface>hardware_interface/${hardware_interface}</hardwareInterface>
      </joint>
      <actuator name="arm_motor_1">
        <hardwareInterface>hardware_interface/${hardware_interface}</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
      </actuator>
    </transmission>

    <transmission name="arm_tran_2">
      <robotNamespace>/arm</robotNamespace>
      <type>transmission_interface/SimpleTransmission</type>
      <joint name="j2">
        <hardwareInterface>hardware_interface/${hardware_interface}</hardwareInterface>
      </joint>
      <actuator name="arm_motor_2">
        <hardwareInterface>hardware_interface/${hardware_interface}</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
      </actuator>
    </transmission>

    <transmission name="arm_tran_3">
      <robotNamespace>/arm</robotNamespace>
      <type>transmission_interface/SimpleTransmission</type>
      <joint name="j3">
        <hardwareInterface>hardware_interface/${hardware_interface}</hardwareInterface>
      </joint>
      <actuator name="arm_motor_3">
        <hardwareInterface>hardware_interface/${hardware_interface}</hardwareInterface>
        <mechanicalReduction>1</mechanicalReduction>
      </actuator>
    </transmission>

</xacro:macro>
</robot>
```

e. Add joint position controllers to your robot: create a arm_control package with a arm_control.launch file inside its launch folder and a arm_control.yaml file within its config folder

Used commands

$ cd src
$ catkin_create_pkg arm_control
$ ls
$ cd arm_control
$ mkdir launch
$ cd launch
$ touch arm_control.launch
$ ls
$ cd ..
$ mkdir config
$ cd config
$ touch arm_control.yaml
$ ls

```
giuliag@giuliapc:~/catkin_ws$ cd src/
giuliag@giuliapc:~/catkin_ws/src$ catkin_create_pkg arm_control
Created file arm_control/package.xml
Created file arm_control/CMakeLists.txt
Successfully created files in /home/giuliag/catkin_ws/src/arm_control. Please adjust the values in
 package.xml.
giuliag@giuliapc:~/catkin_ws/src$ ls
arm_control  arm_description  arm_gazebo
giuliag@giuliapc:~/catkin_ws/src$ cd arm_control
giuliag@giuliapc:~/catkin_ws/src/arm_control$ mkdir launch
giuliag@giuliapc:~/catkin_ws/src/arm_control$ cd launch
giuliag@giuliapc:~/catkin_ws/src/arm_control/launch$ touch arm_control.launch
giuliag@giuliapc:~/catkin_ws/src/arm_control/launch$ ls
arm_control.launch
giuliag@giuliapc:~/catkin_ws/src/arm_control/launch$ cd ..
giuliag@giuliapc:~/catkin_ws/src/arm_control$ mkdir config
giuliag@giuliapc:~/catkin_ws/src/arm_control$ cd config/
giuliag@giuliapc:~/catkin_ws/src/arm_control/config$ touch arm_control.yaml
giuliag@giuliapc:~/catkin_ws/src/arm_control/config$ ls
arm_control.yaml
```

f. Fill the arm_control.launch file with commands that load the joint controller configurations from the .yaml file to the parameter server and spawn the controllers using the controller_manager package.

Apri — arm_control.launch — ~/catkin_ws/src/arm_control/launch — Salva

```
<?xml version="1.0"?>
<launch>

    <!-- Launches the controllers according to PosITIonJointInterface-->
    <arg name="hardware_interface" default="PositionJointInterface"/>
    <arg name="controllers" default="joint_state_controller joint1_position_controller joint2_position_controller joint3_position_controller joint4_position_controller"/>
    <arg name="joint_state_frequency" default="100" />
    <arg name="robot_state_frequency" default="100" />

    <!-- Loads joint controller configurations from YAML file to parameter server -->
    <rosparam file="$(find arm_control)/config/arm_control.yaml" command="load" />
    <param name="/arm/joint_state_controller/publish_rate" value="$(arg joint_state_frequency)" />

    <!-- Loads the controllers -->
    <node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false" output="screen" args="$(arg controllers)" />

    <!-- Converts joint states to TF transforms for rviz, etc -->
    <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" respawn="false" output="screen">
      <remap from="joint_states" to="/arm/joint_states" />
      <param name="publish_frequency" value="$(arg robot_state_frequency)" />
    </node>

</launch>
```

g. Fill the arm arm_control.yaml adding a joint_state_controller and a JointPositionController to all the joints

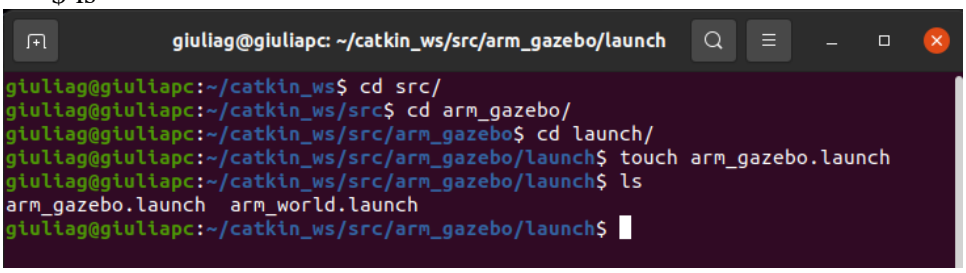
Apri — arm_control.yaml — ~/catkin_ws/src/arm_control/config — Salva

```
#arm:

  # Publish all joint states -----------------------------------
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 50


  # Forward Position Controllers ---------------------------------------
  joint0_position_controller:
    type: position_controllers/JointPositionController
    joint: j0

  joint1_position_controller:
    type: position_controllers/JointPositionController
    joint: j1

  joint2_position_controller:
    type: position_controllers/JointPositionController
    joint: j2

  joint3_position_controller:
    type: position_controllers/JointPositionController
    joint: j3

```

h. Create an arm_gazebo.launch file into the launch folder of the arm_gazebo package loading the Gazebo world with arm_world.launch and spawning the controllers within arm_control.launch. Go to the arm_description package and add the gazebo_ros_control plugin to your main URDF into the arm.gazebo.xacro file. Launch the simulation and check if your controllers are correctly loaded

Used commands

$ cd src/arm_gazebo/launch/
$ touch arm_gazebo.launch
$ ls



Apri — arm.gazebo.xacro — ~/catkin_ws/src/arm_description/urdf

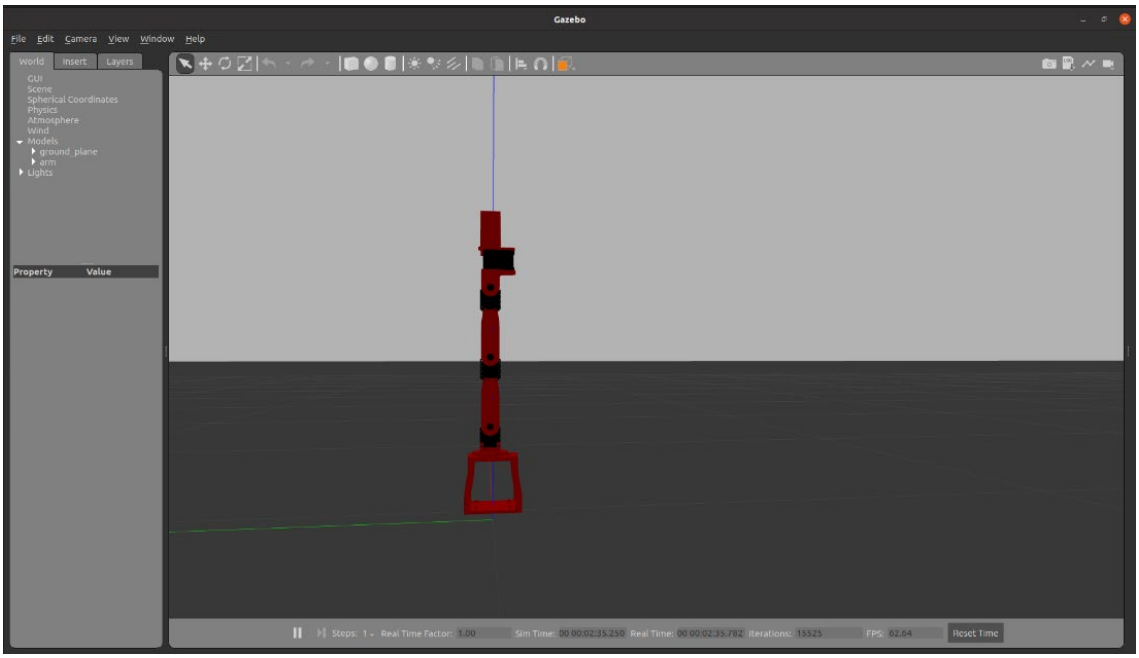
```
<?xml version="1.0"?>
<robot xmlns:xacro="http://ros.org/wiki/xacro">

<xacro:macro name="arm_gazebo">

<!-- Controller -->
    <gazebo>
      <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
        <robotNamespace>/arm</robotNamespace>
     </plugin>
    </gazebo>

```

Apri — arm_gazebo.launch — ~/catkin_ws/src/arm_gazebo/launch — Salva

```
<?xml version="1.0"?>
<launch>

    <arg name="hardware_interface" default="PositionJointInterface" />

    <!-- Loads the Gazebo world. -->
    <include file="$(find arm_gazebo)/launch/arm_world.launch">
        <arg name="hardware_interface" value="$(arg hardware_interface)" />
    </include>

      <group ns="arm">
        <include file="$(find arm_control)/launch/arm_control.launch">
            <arg name="hardware_interface" value="$(arg hardware_interface)" />
            <arg name="controllers" value="joint_state_controller joint0_position_controller joint1_position_controller
joint2_position_controller joint3_position_controller" />

        </include>
    </group>

</launch>
```

Used commands

$ catkin build
$ source devel/setup.bash
$ roslaunch arm_gazebo arm_gazebo.launch


Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
ground_plane
arm
Lights
Property Value
Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:02:35.250 Real Time: 00 00:02:35.702 Iterations: 15525 FPS: 62.64
Reset Time

3. Add a camera sensor to your robot

a. Go into your arm.urdf.xacro _le and add a camera_link and a fixed camera_joint with base_link as a parent link. Size and position the camera link opportunely

To add the camera, the file arm.urdf.xacro is modified as:

Apri — arm.urdf.xacro — ~/catkin_ws/src/arm_description/urdf — Salva

```
<?xml version="1.0"?>

<robot name="arm" xmlns:xacro="http://ros.org/wiki/xacro">

  <arg name="hardware_interface" default="PositionJointInterface"/>

  <xacro:include filename="$(find arm_description)/urdf/arm.gazebo.xacro" />

  <xacro:include filename="$(find arm_description)/urdf/arm.transmission.xacro" />

  <xacro:property name="camera_link" value="0.025" /> <!-- Size of camera box-->
```

...

Apri — arm.urdf.xacro — ~/catkin_ws/src/arm_description/urdf — Salva

```
    <!-- CAMERA_JOINT -->
  <joint name="camera_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0 0 ${camera_link}"/>
    <parent link="base_link"/>
    <child link="camera_link"/>
  </joint>

  <!-- CAMERA_LINK -->
  <link name="camera_link">
    <visual>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
      <box size = "${camera_link} ${camera_link} ${camera_link}" />
     </geometry>
     <material name="Blue">
        <color rgba="0 0 1 1"/>
     </material>
    </visual>

   <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <box size="${camera_link} ${camera_link} ${camera_link}"/>
      </geometry>
   </collision>

   <inertial>
      <mass value="1e-5"/>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <inertia ixx="1e-6" ixy="0.0" ixz="0.0" iyy="1e-6" iyz="0.0" izz="1e-6"/>
    </inertial>

  </link>
```

XML — Larg. tab.: 8 — Rg 510, Col 3 — INS

b. In the arm.gazebo.xacro add the gazebo sensor reference tags and the libgazebo_ros_camera plugin to your xacro.

In the arm.gazebo.xacro the <gazebo reference> for the camera is added.

Apri — arm.gazebo.xacro — ~/catkin_ws/src/arm_description/urdf — Salva

```
<!-- Camera -->

  <gazebo reference="camera_link">
    <material>Gazebo/Blue</material>
    <sensor type="camera" name="camera1">
      <update_rate>30.0</update_rate>
      <camera name="head">
        <horizontal_fov>1.3962634</horizontal_fov>
        <image>
          <width>800</width>
          <height>800</height>
          <format>R8G8B8</format>
        </image>
        <clip>
          <near>0.02</near>
          <far>300</far>
        </clip>
        <noise>
         <type>gaussian</type>
         <mean>0.0</mean>
         <stddev>0.007</stddev>
        </noise>
      </camera>
      <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
            <alwaysOn>true</alwaysOn>
            <updateRate>0.0</updateRate>
            <cameraName>camera</cameraName>
            <imageTopicName>image_raw</imageTopicName>
            <cameraInfoTopicName>camera_info</cameraInfoTopicName>
            <frameName>camera_link</frameName>
            <hackBaseline>0.07</hackBaseline>
            <distortionK1>0.0</distortionK1>
            <distortionK2>0.0</distortionK2>
            <distortionK3>0.0</distortionK3>
            <distortionT1>0.0</distortionT1>
            <distortionT2>0.0</distortionT2>
            <CxPrime>0</CxPrime>
            <Cx>0.0</Cx>
            <Cy>0.0</Cy>
            <focalLength>0.0</focalLength>
      </plugin>
    </sensor>
  </gazebo>
```

XML — Larg. tab.: 8 — Rg 5, Col 1 — INS

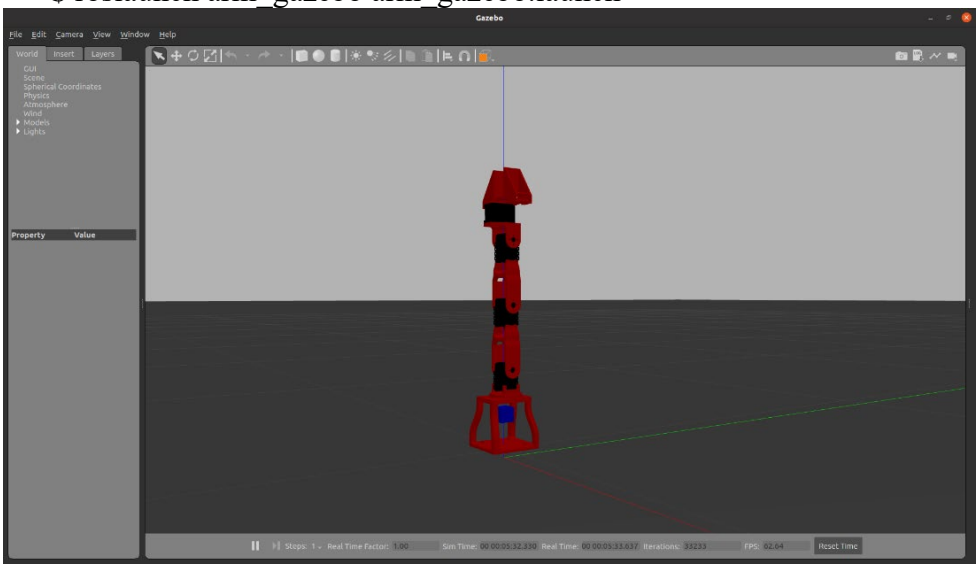
c. Launch the Gazebo simulation with using arm_gazebo.launch and check if the image topic is correctly published using rqt_image_view

Used commands

$ catkin build

$ source devel/setup.bash

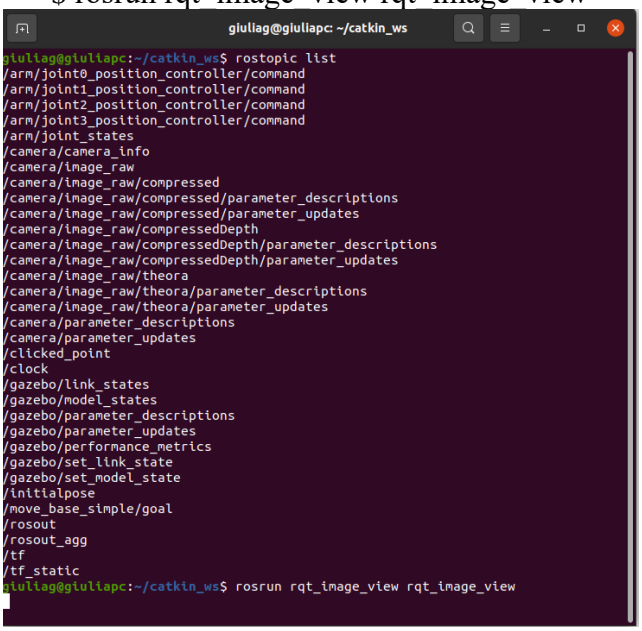
$ roslaunch arm_gazebo arm_gazebo.launch



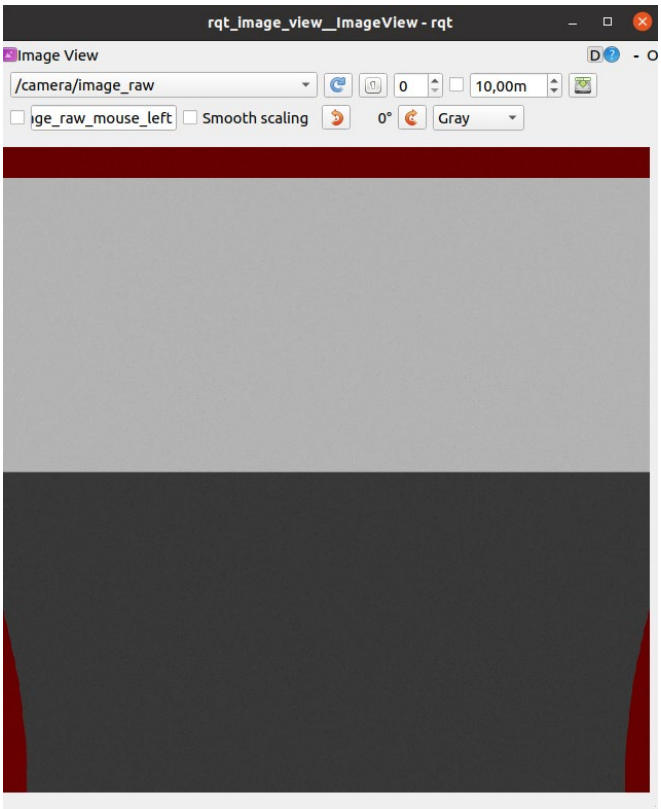
Used commands

$ rostopic list

$ rosrun rqt_image_view rqt_image_view



The image topic is shown:



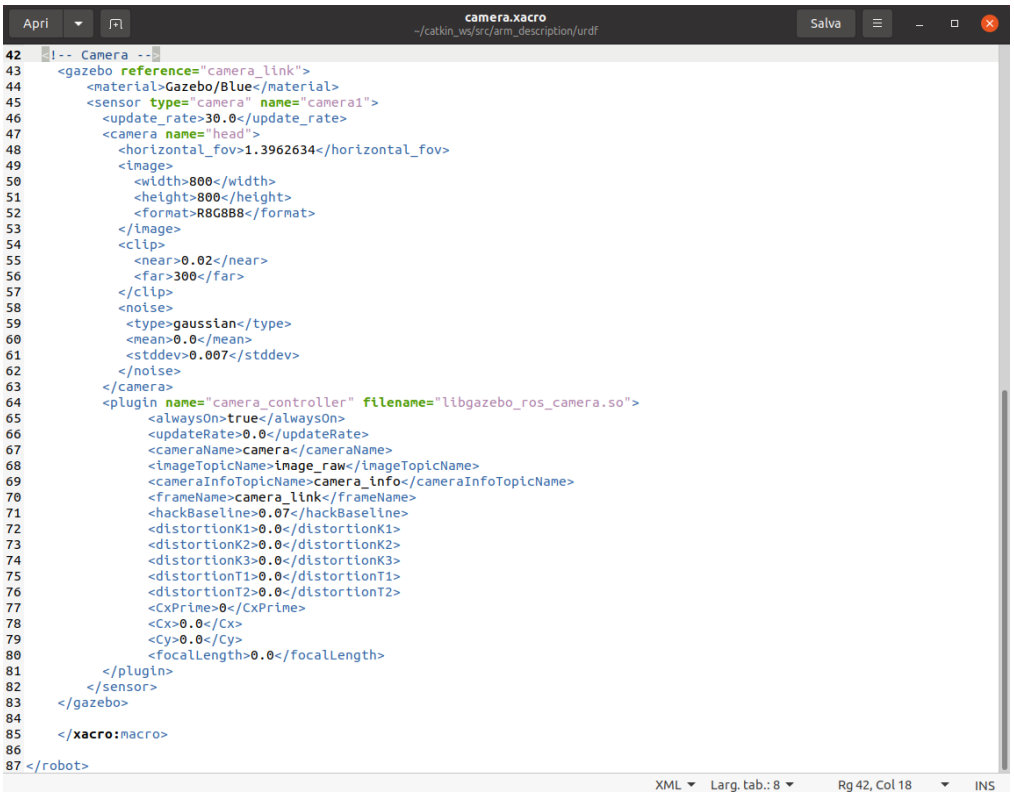
d. Optionally: You can create a camera.xacro file and add it to your robot URDF using <xacro:include>

Used commands

$ cd src/arm_description/urdf

$ touch camera.xacro

$ ls



camera.xacro
~/catkin_ws/src/arm_description/urdf

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="camera">

    <xacro:macro name="camera">
    <xacro:property name="camera_link" value="0.025" /> <!-- Size of camera box-->

     <!-- CAMERA_JOINT -->
        <joint name="camera_joint" type="fixed">

              <origin rpy="0 0 0" xyz="0 0 ${camera_link}"/>
               <parent link="base_link"/>
              <child link="camera_link"/>
       </joint>

     <!-- CAMERA_LINK -->
  <link name="camera_link">
    <visual>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
      <box size = "${camera_link} ${camera_link} ${camera_link}" />
     </geometry>
     <material name="Blue">
        <color rgba="0 0 1 1"/>
     </material>
    </visual>

   <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
        <box size="${camera_link} ${camera_link} ${camera_link}"/>
      </geometry>
   </collision>

   <inertial>
      <mass value="1e-5"/>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <inertia ixx="1e-6" ixy="0.0" ixz="0.0" iyy="1e-6" iyz="0.0" izz="1e-6"/>
    </inertial>

  </link>
```

The camera joint and camera link in arm.urdf.xacro and the <gazebo reference> in arm.gazebo.xacro are commented out if using the camera.xacro.

4. Create a ROS publisher node that reads the joint state and sends joint position commands to your robot

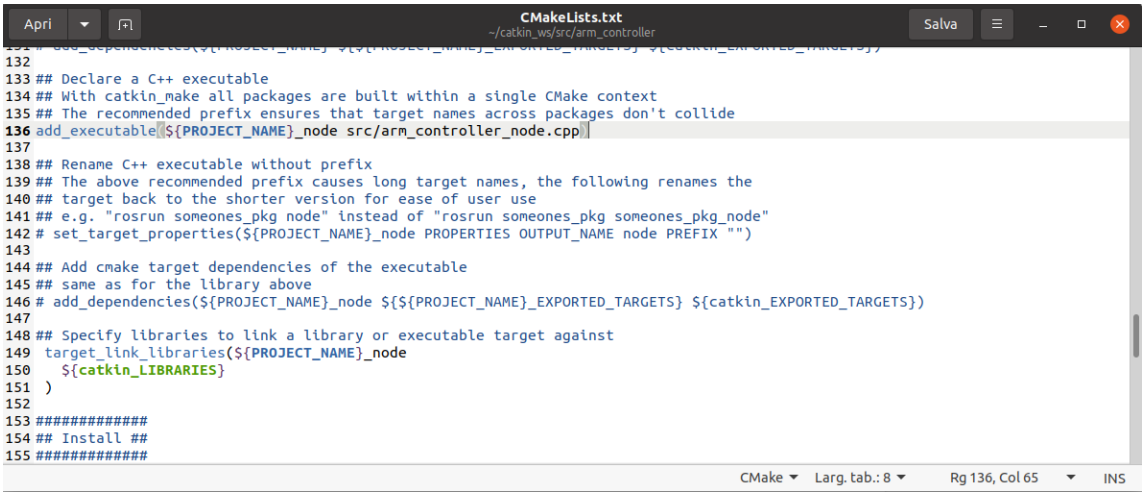
a. Create an arm_controller package with a ROS C++ node named arm_controller_node. The dependencies are roscpp, sensor_msgs and std_msgs. Modify opportunely the CMakeLists.txt file to compile your node. Hint: uncomment add_executable and target_link_libraries lines

Used commands

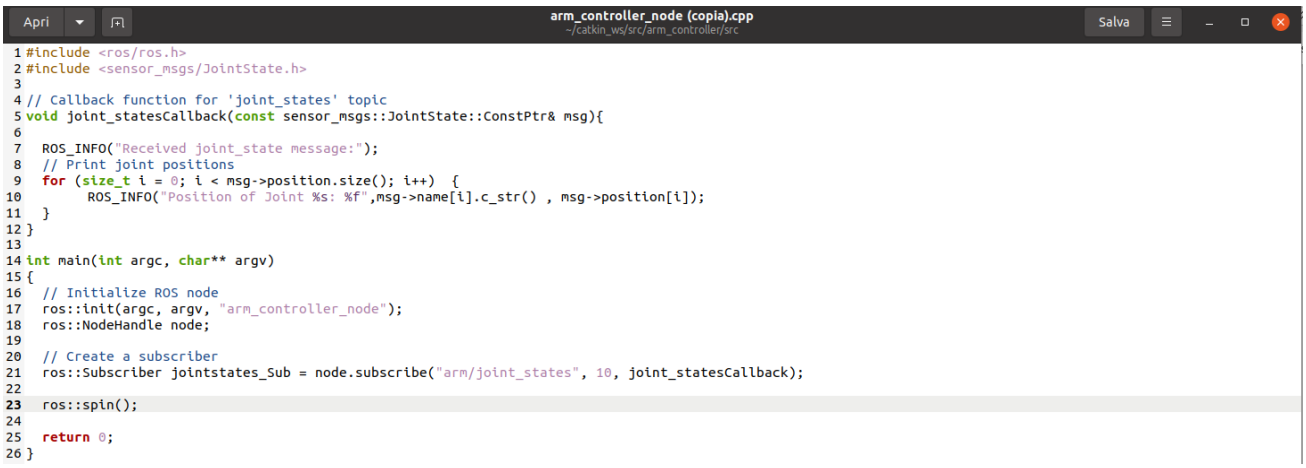
$ cd src
$ catkin_create_pkg arm_controller roscpp sensor_msgs std_msgs
$ cd arm_controller/src
$ touch arm_controller_node.cpp
$ ls

```
giuliag@giuliapc: ~/catkin_ws/src/arm_controller/src
giuliag@giuliapc:~/catkin_ws/src$ catkin_create_pkg arm_controller roscpp sensor_msgs std_msgs
Created file arm_controller/package.xml
Created file arm_controller/CMakeLists.txt
Created folder arm_controller/include/arm_controller
Created folder arm_controller/src
Successfully created files in /home/giuliag/catkin_ws/src/arm_controller. Please adjust the value
s in package.xml.
giuliag@giuliapc:~/catkin_ws/src$ cd arm_controller/src
giuliag@giuliapc:~/catkin_ws/src/arm_controller/src$ touch arm_controller_node.cpp
giuliag@giuliapc:~/catkin_ws/src/arm_controller/src$ ls
arm_controller_node.cpp
giuliag@giuliapc:~/catkin_ws/src/arm_controller/src$
```

The CMakeLists.txt has been modified as follows to compile the arm_controller_node.



b. Create a subscriber to the topic joint_states and a callback function that prints the current joint positions



c. Create publishers that write commands onto the controllers' /command topics

Apri — arm_controller_node.cpp — ~/catkin_ws/src/arm_controller/src — Salva

```cpp
#include <ros/ros.h>
#include <sensor_msgs/JointState.h>
#include <std_msgs/Float64.h>

// Callback function for 'joint_states' topic
void joint_statesCallback(const sensor_msgs::JointState::ConstPtr& msg){

  ROS_INFO("Received joint_state message:");
  // Print joint positions
  for (size_t i = 0; i < msg->position.size(); i++)  {
        ROS_INFO("Position of Joint %s: %f",msg->name[i].c_str() , msg->position[i]);
  }
}

int main(int argc, char** argv)
{
  // Initialize ROS node
  ros::init(argc, argv, "arm_controller_node");
  ros::NodeHandle node;

  // Create a subscriber
  ros::Subscriber jointstates_Sub = node.subscribe("arm/joint_states", 10, joint_statesCallback);

  // Create publishers
  ros::Publisher joint0_Pub = node.advertise<std_msgs::Float64>("arm/joint0_position_controller/command", 1);
  ros::Publisher joint1_Pub = node.advertise<std_msgs::Float64>("arm/joint1_position_controller/command", 1);
  ros::Publisher joint2_Pub = node.advertise<std_msgs::Float64>("arm/joint2_position_controller/command", 1);
  ros::Publisher joint3_Pub = node.advertise<std_msgs::Float64>("arm/joint3_position_controller/command", 1);
  ros::Rate loopRate(10);

  if(argc != 5){
        ROS_INFO("Repeat the command with the right number of the desired joint positions...");
        return 0;
  }

  double position_j0 = std::stod(argv[1]);
  double position_j1 = std::stod(argv[2]);
  double position_j2 = std::stod(argv[3]);
  double position_j3 = std::stod(argv[4]);

  std_msgs::Float64 joint0_command, joint1_command, joint2_command, joint3_command;

  joint0_command.data = position_j0;
  joint1_command.data = position_j1;
  joint2_command.data = position_j2;
  joint3_command.data = position_j3;

  while (ros::ok()) {

        ROS_INFO("Position command to joint0: %.2f", joint0_command.data);
        joint0_Pub.publish(joint0_command);

        ROS_INFO("Position command to joint1: %.2f", joint1_command.data);
        joint1_Pub.publish(joint1_command);

        ROS_INFO("Position command to joint2: %.2f", joint2_command.data);
        joint2_Pub.publish(joint2_command);

        ROS_INFO("Position command to joint3: %.2f", joint3_command.data);
        joint3_Pub.publish(joint3_command);

        ros::spinOnce();
        loopRate.sleep();
  }

  return 0;
}
```

C++ — Larg. tab.: 8 — Rg 34, Col 4 — INS

Used commands
Terminal 1:

$ catkin build
$ source devel/setup.bash
$ roslaunch arm_gazebo arm_gazebo.launch

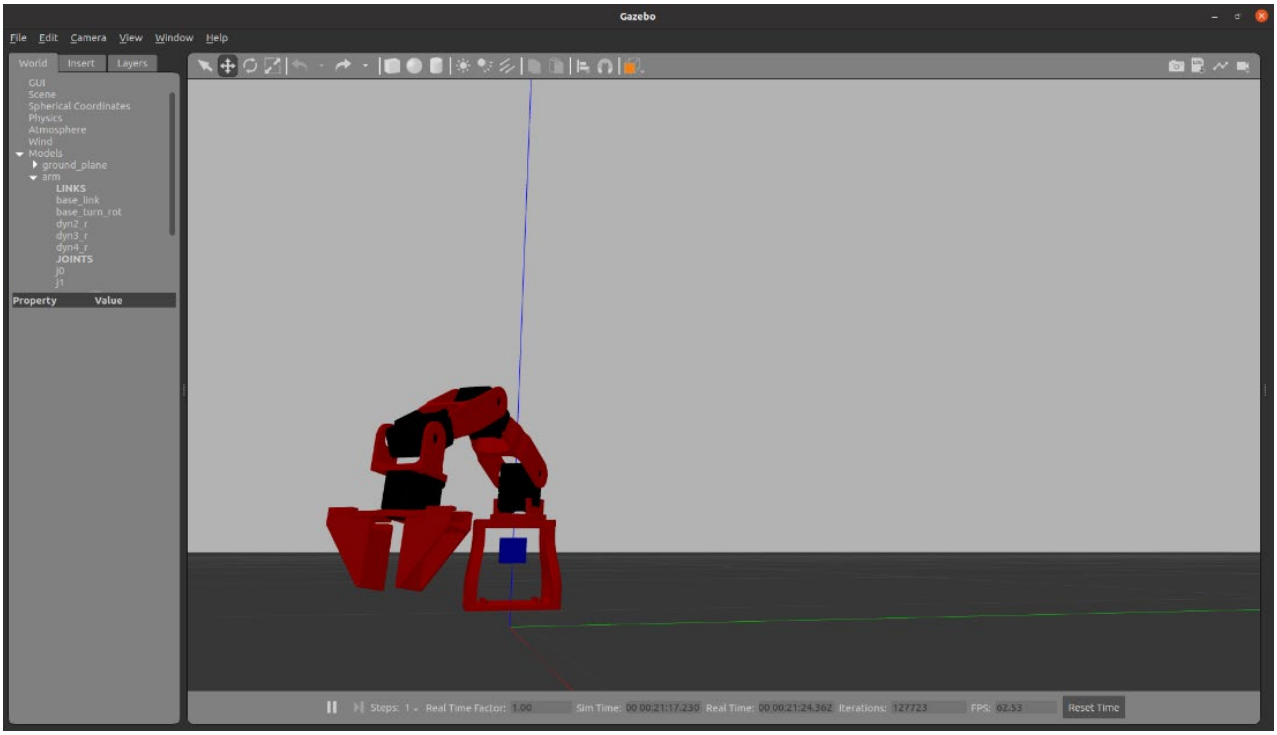
Terminal 2:

$ rosrun arm_controller arm_controller node 1 1 1 1

```
giuliag@giuliapc:~/catkin_ws/src/arm_controller/src$ rosrun arm_controller arm_controller_node 1 1 1 1
[ INFO] [1698744063.092354388]: Position command to joint0: 1.00
[ INFO] [1698744063.093662155]: Position command to joint1: 1.00
[ INFO] [1698744063.093688096]: Position command to joint2: 1.00
[ INFO] [1698744063.093700211]: Position command to joint3: 1.00
[ INFO] [1698744063.438863060, 1008.130000000]: Position command to joint0: 1.00
[ INFO] [1698744063.439043753, 1008.130000000]: Position command to joint1: 1.00
[ INFO] [1698744063.439127652, 1008.130000000]: Position command to joint2: 1.00
[ INFO] [1698744063.439235240, 1008.130000000]: Position command to joint3: 1.00
[ INFO] [1698744063.439460523, 1008.130000000]: Received joint_state message:
[ INFO] [1698744063.439564022, 1008.130000000]: Position of Joint j0: 0.000000
[ INFO] [1698744063.439700705, 1008.130000000]: Position of Joint j1: 0.000000
[ INFO] [1698744063.439775756, 1008.130000000]: Position of Joint j2: 0.000000
[ INFO] [1698744063.439848528, 1008.130000000]: Position of Joint j3: -0.000000
[ INFO] [1698744063.439946067, 1008.130000000]: Received joint_state message:
[ INFO] [1698744063.440018638, 1008.130000000]: Position of Joint j0: -0.000000
[ INFO] [1698744063.440130929, 1008.130000000]: Position of Joint j1: -0.000000
[ INFO] [1698744063.440311837, 1008.130000000]: Position of Joint j2: -0.000000
[ INFO] [1698744063.440402416, 1008.130000000]: Position of Joint j3: 0.000000
[ INFO] [1698744063.440602614, 1008.130000000]: Received joint_state message:
[ INFO] [1698744063.440682173, 1008.130000000]: Position of Joint j0: 0.000000
[ INFO] [1698744063.440762163, 1008.130000000]: Position of Joint j1: 0.000000
[ INFO] [1698744063.440841849, 1008.130000000]: Position of Joint j2: 0.000000
[ INFO] [1698744063.441002762, 1008.130000000]: Position of Joint j3: -0.000000
[ INFO] [1698744063.441133653, 1008.130000000]: Received joint_state message:
[ INFO] [1698744063.441198015, 1008.130000000]: Position of Joint j0: -0.000000
[ INFO] [1698744063.441253453, 1008.130000000]: Position of Joint j1: -0.000000
[ INFO] [1698744063.441392486, 1008.130000000]: Position of Joint j2: -0.000000
[ INFO] [1698744063.441477271, 1008.130000000]: Position of Joint j3: 0.000000
[ INFO] [1698744063.441567795, 1008.130000000]: Received joint_state message:
[ INFO] [1698744063.441631790, 1008.130000000]: Position of Joint j0: 0.000000
[ INFO] [1698744063.441698884, 1008.130000000]: Position of Joint j1: 0.000000
[ INFO] [1698744063.441760687, 1008.130000000]: Position of Joint j2: 0.000000
[ INFO] [1698744063.441832047, 1008.130000000]: Position of Joint j3: -0.000000
[ INFO] [1698744063.441997675, 1008.130000000]: Received joint_state message:
[ INFO] [1698744063.442307036, 1008.130000000]: Position of Joint j0: -0.000000
[ INFO] [1698744063.442374965, 1008.130000000]: Position of Joint j1: -0.000000
[ INFO] [1698744063.442425476, 1008.130000000]: Position of Joint j2: -0.000000
[ INFO] [1698744063.442507828, 1008.130000000]: Position of Joint j3: 0.000000
[ INFO] [1698744063.442617824, 1008.130000000]: Received joint_state message:
[ INFO] [1698744063.442699251, 1008.130000000]: Position of Joint j0: 0.000000
```

...

```
giuliag@giuliapc: ~/catkin_ws/src/arm_controller/src
[ INFO] [1698744428.276250622, 16.890000000]: Received joint_state message:
[ INFO] [1698744428.276457712, 16.890000000]: Position of Joint j0: 0.993000
[ INFO] [1698744428.276535735, 16.890000000]: Position of Joint j1: 0.993000
[ INFO] [1698744428.276606270, 16.890000000]: Position of Joint j2: 0.993000
[ INFO] [1698744428.276761355, 16.890000000]: Position of Joint j3: 0.993000
[ INFO] [1698744428.276860254, 16.890000000]: Received joint_state message:
[ INFO] [1698744428.276960335, 16.890000000]: Position of Joint j0: 0.994000
[ INFO] [1698744428.277088606, 16.890000000]: Position of Joint j1: 0.994000
[ INFO] [1698744428.277182108, 16.890000000]: Position of Joint j2: 0.994000
[ INFO] [1698744428.277292670, 16.890000000]: Position of Joint j3: 0.994000
[ INFO] [1698744428.277448798, 16.890000000]: Received joint_state message:
[ INFO] [1698744428.277558103, 16.890000000]: Position of Joint j0: 0.995000
[ INFO] [1698744428.277703433, 16.890000000]: Position of Joint j1: 0.995000
[ INFO] [1698744428.277803778, 16.890000000]: Position of Joint j2: 0.995000
[ INFO] [1698744428.277945318, 16.890000000]: Position of Joint j3: 0.995000
[ INFO] [1698744428.278074882, 16.890000000]: Received joint_state message:
[ INFO] [1698744428.278163897, 16.890000000]: Position of Joint j0: 0.996000
[ INFO] [1698744428.278249659, 16.890000000]: Position of Joint j1: 0.996000
[ INFO] [1698744428.278332569, 16.890000000]: Position of Joint j2: 0.996000
[ INFO] [1698744428.278409278, 16.890000000]: Position of Joint j3: 0.996000
[ INFO] [1698744428.278603583, 16.890000000]: Received joint_state message:
[ INFO] [1698744428.278753786, 16.890000000]: Position of Joint j0: 0.997000
[ INFO] [1698744428.278972900, 16.890000000]: Position of Joint j1: 0.997000
[ INFO] [1698744428.279044780, 16.890000000]: Position of Joint j2: 0.997000
[ INFO] [1698744428.279098422, 16.890000000]: Position of Joint j3: 0.997000
[ INFO] [1698744428.279278431, 16.890000000]: Received joint_state message:
[ INFO] [1698744428.279377148, 16.890000000]: Position of Joint j0: 0.998000
[ INFO] [1698744428.279494535, 16.890000000]: Position of Joint j1: 0.998000
[ INFO] [1698744428.279655445, 16.890000000]: Position of Joint j2: 0.998000
[ INFO] [1698744428.279779852, 16.890000000]: Position of Joint j3: 0.998000
[ INFO] [1698744428.279908245, 16.890000000]: Received joint_state message:
[ INFO] [1698744428.280002310, 16.890000000]: Position of Joint j0: 0.999000
[ INFO] [1698744428.280137302, 16.890000000]: Position of Joint j1: 0.999000
[ INFO] [1698744428.280217906, 16.890000000]: Position of Joint j2: 0.999000
[ INFO] [1698744428.280331459, 16.890000000]: Position of Joint j3: 0.999000
[ INFO] [1698744428.374121608, 16.990000000]: Position command to joint0: 1.00
[ INFO] [1698744428.374317836, 16.990000000]: Position command to joint1: 1.00
[ INFO] [1698744428.374551224, 16.990000000]: Position command to joint2: 1.00
[ INFO] [1698744428.374707830, 16.990000000]: Position command to joint3: 1.00
[ INFO] [1698744428.375481803, 16.990000000]: Received joint_state message:
[ INFO] [1698744428.375649216, 16.990000000]: Position of Joint j0: 1.000000
```


Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
ground_plane
arm
LINKS
base_link
base_turn_rot
dyn2_r
dyn3_r
dyn4_r
JOINTS
j0
j1
Property Value
Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:21:17.230 Real Time: 00 00:21:24.362 Iterations: 127723 FPS: 62.53
Reset Time