

Robotics Lab: Homework 1

Building your robot manipulator

Giulia Gelsomina Romano, P38000223

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

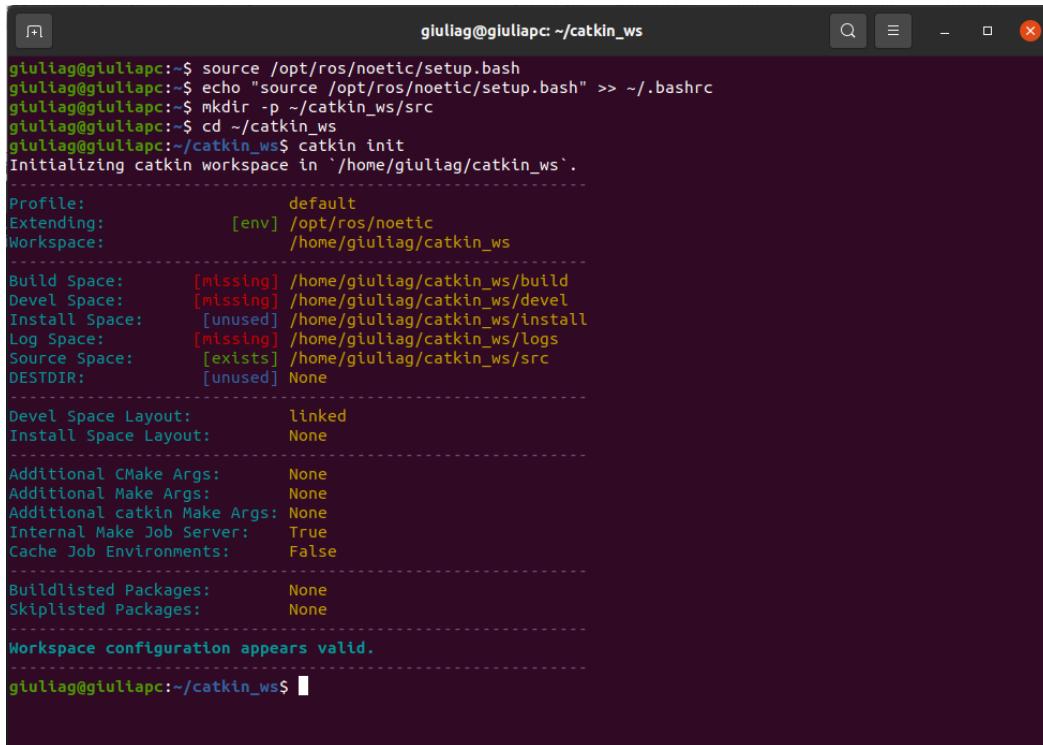
Here is the link to my public repo on github: https://github.com/giuliagromano/rl_hw1.git

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
```



```
giuliag@giuliapc:~$ source /opt/ros/noetic/setup.bash  
giuliag@giuliapc:~$ echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc  
giuliag@giuliapc:~$ mkdir -p ~/catkin_ws/src  
giuliag@giuliapc:~$ cd ~/catkin_ws  
giuliag@giuliapc:~/catkin_ws$ catkin init  
Initializing catkin workspace in '/home/giuliag/catkin_ws'.  
-----  
Profile: default  
Extending: [env] /opt/ros/noetic  
Workspace: /home/giuliag/catkin_ws  
-----  
Build Space: [missing] /home/giuliag/catkin_ws/build  
Devel Space: [missing] /home/giuliag/catkin_ws/devel  
Install Space: [unused] /home/giuliag/catkin_ws/install  
Log Space: [missing] /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.  
-----  
giuliag@giuliapc:~/catkin_ws$
```

```

giuliag@giuliapc:~/catkin_ws$ catkin build
-----
Profile:           default
Extending:        [env] /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:        [missing] /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.

NOTE: Forcing CMake to run for each package.

[build] No packages were found in the source space '/home/giuliag/catkin_ws/src'
[build] No packages to be built.
[build] Package table is up to date.
Starting >>> catkin_tools_prebuild
Finished <<< catkin_tools_prebuild [ 1.9 seconds ]
[build] Summary: All 1 packages succeeded!
[build] Ignored: None.
[build] Warnings: None.
[build] Abandoned: None.
[build] Failed: None.
[build] Runtime: 1.9 seconds total.
giuliag@giuliapc:~/catkin_ws$ 

```

```

giuliag@giuliapc:~/catkin_ws$ source devel/setup.bash
giuliag@giuliapc:~/catkin_ws$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc
giuliag@giuliapc:~/catkin_ws$ cd src
giuliag@giuliapc:~/catkin_ws/src$ git clone https://github.com/RoboticsLab2023/arm_description.git
Clone in 'arm_description' in corso...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 23 (delta 2), reused 23 (delta 2), pack-reused 0
Decompression degli oggetti in corso: 100% (23/23), 1.13 MiB | 1.56 MiB/s, fatto.
giuliag@giuliapc:~/catkin_ws/src$ ls
arm_description
giuliag@giuliapc:~/catkin_ws/src$ 

```

- 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

```

giuliag@giuliapc:~/catkin_ws\$ cd src
giuliag@giuliapc:~/catkin_ws/src\$ cd arm_description
giuliag@giuliapc:~/catkin_ws/src/arm_description\$ mkdir launch
giuliag@giuliapc:~/catkin_ws/src/arm_description\$ cd launch
giuliag@giuliapc:~/catkin_ws/src/arm_description/launch\$ touch display.launch

Aprí ▾ 🔍 *display.launch
~/catkin_ws/src/arm_description/launch Salva ▾ 🔍 ⌛ ✎

```
1 <launch>  
2   <arg name="model" />  
3   <param name="robot_description" textfile="$(find arm_description)/urdf/arm.urdf" />  
4  
5   <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />  
6   <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" />  
7   <node name="rviz" pkg="rviz" type="rviz" />  
8  
9 </launch>
```

Used commands

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

giuliag@giuliapc:~/catkin_ws\$ cd ..
giuliag@giuliapc:~/catkin_ws/src/arm_description\$ cd ..
giuliag@giuliapc:~/catkin_ws/src\$ cd ..

```

/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

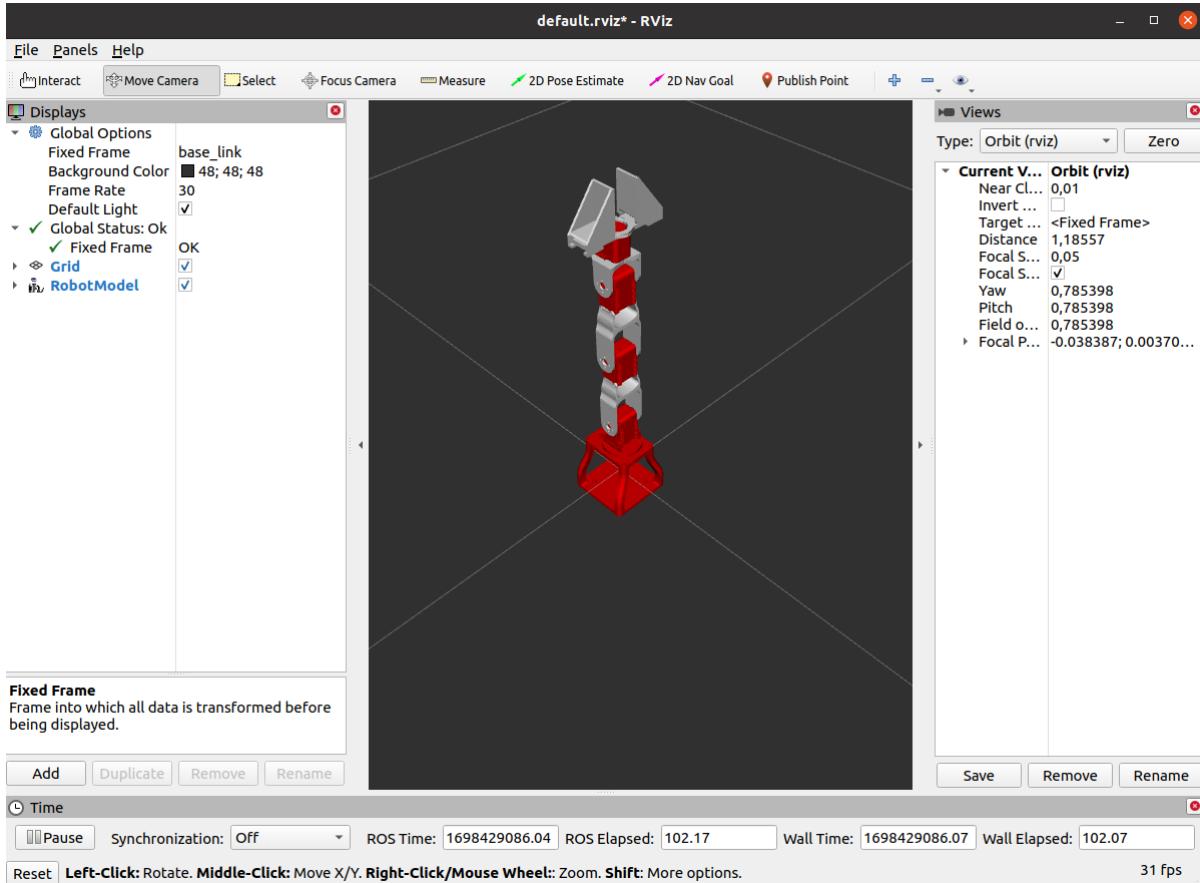
NODES
/
  joint_state_publisher (joint_state_publisher/joint_state_publisher)
  robot_state_publisher (robot_state_publisher/robot_state_publisher)
  rviz (rviz/rviz)

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, that automatically loads the RobotModel plugin by default, and give it as an argument to your node in the display.launch file.

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

```
*display.launch
~/catkin_ws/src/arm_description/launch
1 <launch>
2   <arg name="model" />
3   <param name="robot_description" textfile="$(find arm_description)/urdf/arm.urdf" />
4
5   <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />
6   <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" />
7   <node name="rviz" pkg="rviz" type="rviz" args="-d $(find arm_description)/manipulator.rviz" />
8
9 </launch>
```

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 reasonable size approximating the links.

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

```

1 <?xml version="1.0"?>
2
3 <robot name="arm" xmlns:xacro="http://ros.org/wiki/xacro">
4
5   <!-- Import all Gazebo-customization elements -->
6   <!-- xacro:include filename="$(find arm_description)/urdf/arm.gazebo.xacro" /-->
7
8   <!-- BASE_LINK -->
9   <link name="base_link">
10    <visual>
11      <geometry>
12        <mesh filename="package://arm_description/meshes/base_link.stl" scale="0.001 0.001 0.001"/>
13      </geometry>
14      <origin rpy="0 0 0" xyz="0 0 0"/>
15    </visual>
16    <collision>
17      <geometry>
18        <box size = "0.1 0.1 0.1" />
19      </geometry>
20      <origin rpy="0 0 0" xyz="0 0 0"/>
21    </collision>
22    <inertial>
23      <mass value="0.1"/>
24      <inertia ixz="1.06682889e+08" ixy="0.0" ixz="0.0" iyy="9.92165844e+07" iyz="0.0" izz="1.26939175e+08"/>
25    </inertial>
26  </link>
27
28
29 <!-- JOINT: BASE_1_2 -->
30  <joint name="base_1_2" type="fixed">
31    <parent link="base_link"/>
32    <child link="base_turn"/>
33    <origin xyz="-0.0011 -0.011 0.05"/>
34  </joint>
35
36
37 <!-- BASE TURN -->
38  <link name="base_turn">
39    <visual>
40      <geometry>
41        <mesh filename="package://arm_description/meshes/base_turn.stl" scale="0.001 0.001 0.001"/>
42      </geometry>
43      <origin rpy="0 0 0" xyz="0 0 0"/>
44    </visual>
45    <collision>
46      <geometry>
47        <box size = "0.08 0.08 0.015" />
48      </geometry>
49      <origin rpy="0 0 0" xyz="0 0.01 0"/>
50    </collision>

```

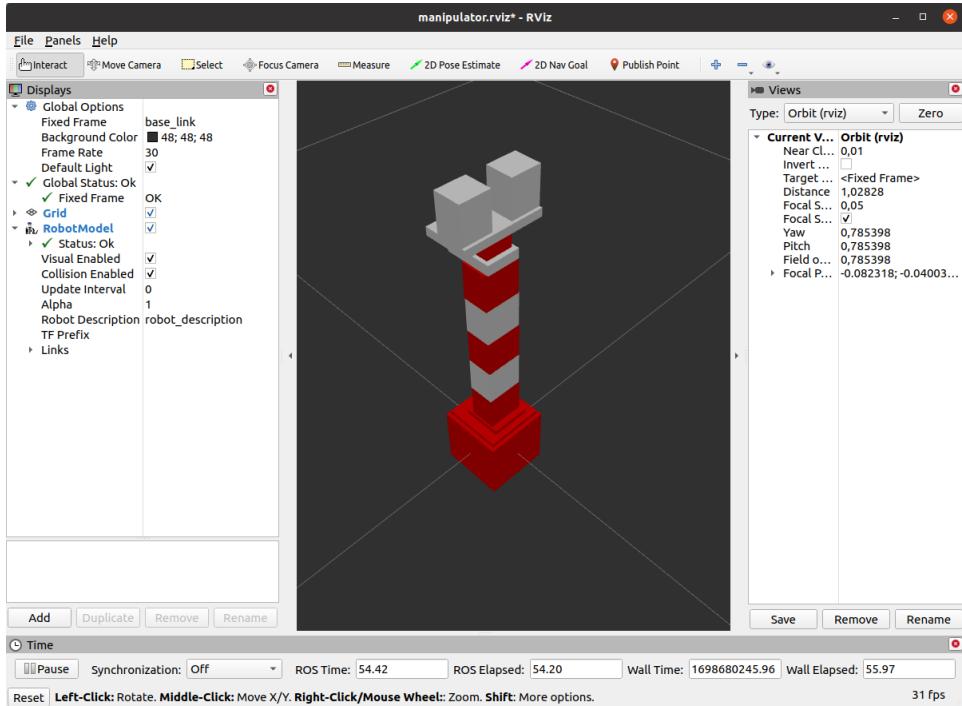
Used commands

```

$ 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
```

A screenshot of a terminal window titled "giuliag@giuliapc: ~/catkin_ws/src/arm_description". The terminal shows the following command history:

```
giuliag@giuliapc:~/catkin_ws$ cd src  
giuliag@giuliapc:~/catkin_ws/src$ cd arm_description  
giuliag@giuliapc:~/catkin_ws/src/arm_description$ touch arm.gazebo.xacro  
giuliag@giuliapc:~/catkin_ws/src/arm_description$ ls  
arm.gazebo.xacro  launch      meshes      urdf  
CMakeLists.txt    manipulator.rviz  package.xml  
giuliag@giuliapc:~/catkin_ws/src/arm_description$
```

```

1 <?xml version="1.0"?>
2 <robot xmlns:xacro="http://ros.org/wikl/xacro">
3
4 <xacro:macro name="arm_gazebo">
5   <gazebo reference="f4">
6     <material>Gazebo/Red</material>
7   </gazebo>
8
9   <gazebo reference="f5">
10  <material>Gazebo/Red</material>
11 </gazebo>
12
13 <gazebo reference="wrist">
14   <material>Gazebo/Red</material>
15 </gazebo>
16
17 <gazebo reference="crawler_base">
18   <material>Gazebo/Red</material>
19 </gazebo>
20
21 <gazebo reference="base_link">
22   <material>Gazebo/Red</material>
23 </gazebo>
24
25 <gazebo reference="base_turn">
26   <material>Gazebo/Red</material>
27 </gazebo>
28
29 <gazebo reference="base_turn_rot">
30   <material>Gazebo/Black</material>
31 </gazebo>
32
33 <gazebo reference="dyn2">
34   <material>Gazebo/Black</material>
35 </gazebo>
36
37 <gazebo reference="dyn3">
38   <material>Gazebo/Black</material>
39 </gazebo>
40
41 <gazebo reference="dyn4">
42   <material>Gazebo/Black</material>
43 </gazebo>
44
45 <gazebo reference="dyn5">
46   <material>Gazebo/Black</material>
47 </gazebo>
48
49 <gazebo reference="crawler_left">
50   <material>Gazebo/Red</material>
51 </gazebo>
52
53   <gazebo reference="crawler_right">
54     <material>Gazebo/Red</material>
55   </gazebo>
56
57 </xacro:macro>
58 </robot>
59

```

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

```

*arm.urdf.xacro
~/catkin_ws/src/arm_description/urdf
Salva - x
1 <?xml version="1.0"?>
2
3 <robot name="arm" xmlns:xacro="http://ros.org/wikl/xacro">
4
5   <!-- Import all Gazebo customization elements -->
6   <xacro:include filename="$(find arm_description)/urdf/arm.gazebo.xacro" />
...
669   <xacro:arm_gazebo/>
670 </robot>

```

The display.launch is modified as:

```

display.launch
~/catkin_ws/src/arm_description/launch
Salva - x
1 <?xml version="1.0"?>
2 <launch>
3   <arg name="model" />
4   <param name="robot_description" command=" $(find xacro)/xacro --inorder '$(find arm_description)/urdf/arm.urdf.xacro'" />
5
6   <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" />
7   <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />
8   <node name="rviz" pkg="rviz" type="rviz" args="-d $(find arm_description)/manipulator.rviz" />
9
10
11 </launch>

```

Used commands

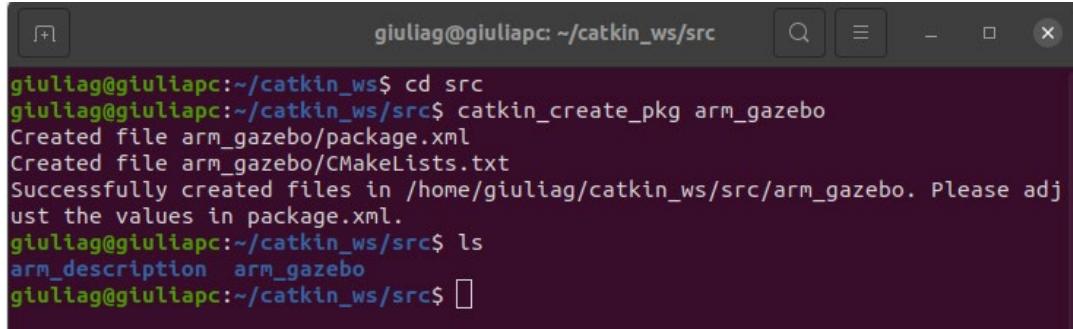
- \$ catkin build
- \$ source devel/setup.bash
- \$ roslaunch arm_description display.launch

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

- Create a package named arm_gazebo

Used commands

```
$ cd src  
$ carkin_create_pkg arm_gazebo  
$ ls
```



A terminal window titled "giuliag@giuliapc: ~/catkin_ws/src". The command \$ catkin_create_pkg arm_gazebo is run, creating files package.xml and CMakeLists.txt. The command \$ ls shows the directory contains arm_description and arm_gazebo.

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

- 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
```



A terminal window titled "giuliag@giuliapc: ~/catkin_ws/src". The user navigates to the arm_gazebo directory, creates a launch folder, and touches an arm_world.launch file. The command \$ ls shows the contents of the launch folder.

```
giuliag@giuliapc:~/catkin_ws/src$ cd arm_gazebo  
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo$ mkdir launch  
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo$ cd launch  
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo/launch$ touch arm_world.launch  
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo/launch$ ls  
arm_world.launch
```

- 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$ 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.

```

Apr 11 2023 10:45:23 arm.world ~/catkin_ws/src/arm_gazebo/world

1 <?xml version="1.0" ?>
2 <sdf version="1.4">
3   <!-- We use a custom world for the iiwa so that the camera angle is launched correctly. -->
4   <!-- One can change this world to his needs or use another one. -->
5
6   <world name="default">
7
8     <include>
9       <uri>model://ground_plane</uri>
10      </include>
11
12     <!-- Global light source -->
13     <include>
14       <uri>model://sun</uri>
15     </include>
16
17     <physics name="default_physics" default="0" type="ode">
18       <max_step_size>0.01</max_step_size>
19       <real_time_factor>1</real_time_factor>
20       <real_time_update_rate>100</real_time_update_rate>
21     </ode>
22       <solver>
23         <type>quick</type>
24         <iters>50</iters>
25         <sor>1.0</sor> <!-- Important, see issue #2209 -->
26         <use_dynamic_mol_rescaling>false</use_dynamic_mol_rescaling>
27       </solver>
28     </ode>
29   </physics>
30
31   <!-- Focus camera -->
32   <gui fullscreen='1'>
33     <camera name='user_camera'>
34       <pose>4.927360 -4.376610 3.740080 0.000000 0.275643 2.356190</pose>
35     <view_controller>orbit</view_controller>
36   </camera>
37 </gui>
38
39 </world>
40
41 
```

The arm_world.launch is modified as:

```

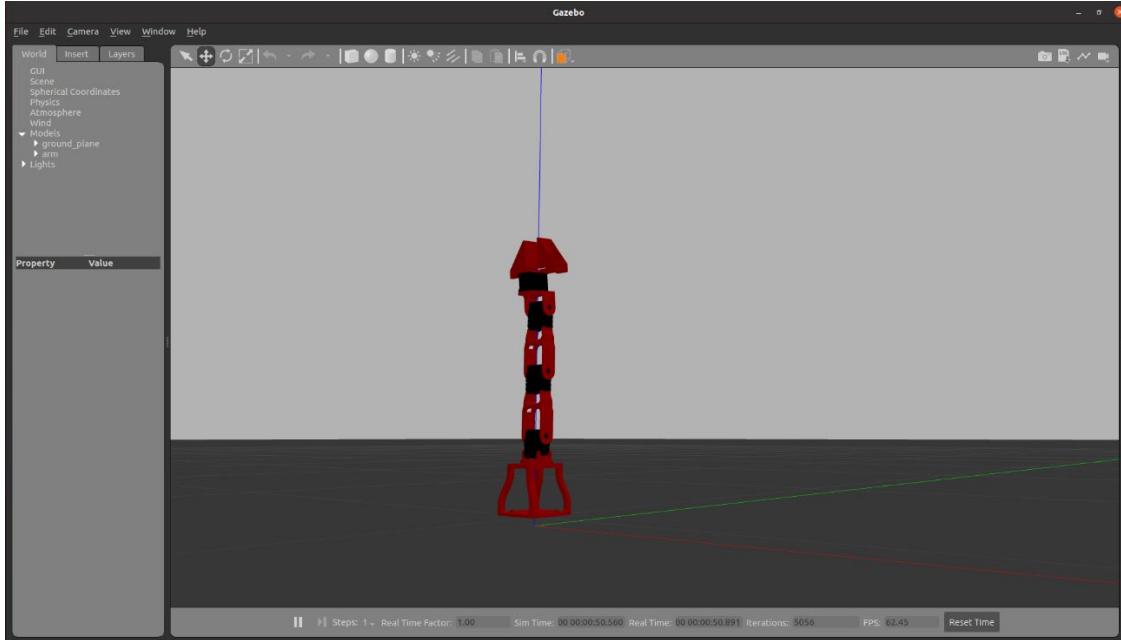
Apr 11 2023 10:45:23 arm_world.launch ~/catkin_ws/src/arm_gazebo/launch

1 <?xml version="1.0"?
2 <launch>
3
4   <!-- Loads the arm.world environment in Gazebo. -->
5
6   <arg name="paused" default="false"/>
7   <arg name="use_sim_time" default="true"/>
8   <arg name="gui" default="true"/>
9   <arg name="headless" default="false"/>
10  <arg name="debug" default="false"/>
11  <arg name="hardware_interface" default="PositionJointInterface"/>
12  <arg name="robot_name" default="arm" />
13  <arg name="model" default="arm5"/>
14
15  <!-- We reuse the logic in empty_world.launch, changing only the name of the world to be launched -->
16  <include file="$(find gazebo_ros)/launch/empty_world.launch">
17    <arg name="world_name" value="$(find arm_gazebo)/worlds/arm.world"/>
18    <arg name="debug" value="$(arg debug)" />
19    <arg name="gui" value="$(arg gui)" />
20    <arg name="paused" value="$(arg paused)" />
21    <arg name="use_sim_time" value="$(arg use_sim_time)" />
22    <arg name="headless" value="$(arg headless)" />
23  </include>
24
25  <!-- Load the URDF into the ROS Parameter Server -->
26  <include file="$(find arm_description)/launch/display.launch">
27
28
29  <!-- Run a python script to send a service call to gazebo_ros to spawn a URDF robot -->
30  <node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
31  args="-urdf -model arm -param robot_description"/>
32
33
34 
```

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/urdf/  
$ touch arm.transmission.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:

```

display.launch
~/catkin_ws/src/arm_description/launch
1 <?xml version="1.0"?>
2 <launch>
3
4   <arg name="hardware_interface" default="PositionJointInterface"/>
5
6   <param name="robot_description" command=" $(find xacro)/xacro --inorder '$(find arm_description)/urdf/arm.urdf.xacro' hardware_interface:=$(arg hardware_interface)"/>
7
8   <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" />
9   <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" />
10  <node name="rviz" pkg="rviz" type="rviz" args="-d $(find arm_description)/manipulator.rviz" />
11
12
13 </launch>

```



```

arm.urdf.xacro
~/catkin_ws/src/arm_description/urdf
1 <?xml version="1.0"?>
2
3 <robot name="arm" xmlns:xacro="http://ros.org/wiki/xacro">
4
5   <arg name="hardware_interface" default="PositionJointInterface"/>
6
7   <xacro:include filename="$(find arm_description)/urdf/arm.gazebo.xacro" />
8
9   <xacro:include filename="$(find arm_description)/urdf/arm.transmission.xacro" />
10

```



```

arm.world.launch
~/catkin_ws/src/arm_gazebo/launch
1 <?xml version="1.0"?>
2 <launch>
3
4   <!-- Loads the arm.world environment in Gazebo. -->
5
6   <arg name="paused" default="false"/>
7   <arg name="use_sim_time" default="true"/>
8   <arg name="gui" default="true"/>
9   <arg name="headless" default="false"/>
10  <arg name="debug" default="false"/>
11  <arg name="hardware_interface" default="PositionJointInterface"/>
12  <arg name="robot_name" default="arm" />
13  <arg name="model" default="arm5"/>
14
15  <!-- We resume the logic in empty_world.launch, changing only the name of the world to be launched -->
16  <include file="$(find gazebo_ros)/launch/empty_world.launch">
17    <arg name="world_name" value="$(find arm_gazebo)/worlds/arm.world"/>
18    <arg name="debug" value="$(arg debug)" />
19    <arg name="gui" value="$(arg gui)" />
20    <arg name="paused" value="$(arg paused)"/>
21    <arg name="use_sim_time" value="$(arg use_sim_time)"/>
22    <arg name="headless" value="$(arg headless)"/>
23  </include>
24
25  <!-- Load the URDF into the ROS Parameter Server -->
26  <include file="$(find arm_description)/launch/display.launch">
27    <arg name="hardware_interface" value="$(arg hardware_interface)"/>
28  </include>
29
30  <!-- Run a python script to send a service call to gazebo_ros to spawn a URDF robot -->
31  <node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
32  args="-urdf -model arm -param robot_description"/>
33
34
35 </launch>
36

```

arm.transmission.xacro

```

1 <?xml version="1.0"?>
2 <robot xmlns:xacro="http://ros.org/wiki/xacro">
3   <xacro:macro name="arm_transmission" params="hardware_interface">
4     <transmission name="arm_tran_0">
5       <robotNamespace>/arm</robotNamespace>
6       <type>transmission_interface/SimpleTransmission</type>
7       <joint name="j0">
8         <hardwareInterface>hardware_interface/S{hardware_interface}</hardwareInterface>
9       </joint>
10      <actuator name="arm_motor_0">
11        <hardwareInterface>hardware_interface/S{hardware_interface}</hardwareInterface>
12        <mechanicalReduction>1</mechanicalReduction>
13      </actuator>
14    </transmission>
15
16    <transmission name="arm_tran_1">
17      <robotNamespace>/arm</robotNamespace>
18      <type>transmission_interface/SimpleTransmission</type>
19      <joint name="j1">
20        <hardwareInterface>hardware_interface/S{hardware_interface}</hardwareInterface>
21      </joint>
22      <actuator name="arm_motor_1">
23        <hardwareInterface>hardware_interface/S{hardware_interface}</hardwareInterface>
24        <mechanicalReduction>1</mechanicalReduction>
25      </actuator>
26    </transmission>
27
28    <transmission name="arm_tran_2">
29      <robotNamespace>/arm</robotNamespace>
30      <type>transmission_interface/SimpleTransmission</type>
31      <joint name="j2">
32        <hardwareInterface>hardware_interface/S{hardware_interface}</hardwareInterface>
33      </joint>
34      <actuator name="arm_motor_2">
35        <hardwareInterface>hardware_interface/S{hardware_interface}</hardwareInterface>
36        <mechanicalReduction>1</mechanicalReduction>
37      </actuator>
38    </transmission>
39
40    <transmission name="arm_tran_3">
41      <robotNamespace>/arm</robotNamespace>
42      <type>transmission_interface/SimpleTransmission</type>
43      <joint name="j3">
44        <hardwareInterface>hardware_interface/S{hardware_interface}</hardwareInterface>
45      </joint>
46      <actuator name="arm_motor_3">
47        <hardwareInterface>hardware_interface/S{hardware_interface}</hardwareInterface>
48        <mechanicalReduction>1</mechanicalReduction>
49      </actuator>
50    </transmission>
51
52  </xacro:macro>
53 </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.



```

Aprile - arm_control.launch
~/catkin_ws/src/arm_control/launch
Salva

1<?xml version="1.0"?>
2<Launch>
3
4    <!-- Launches the controllers according to PositionJointInterface-->
5    <arg name="hardware_interface" default="PositionJointInterface"/>
6    <arg name="controllers" default="joint_state_controller joint1_position_controller joint2_position_controller joint3_position_controller joint4_position_controller"/>
7    <arg name="joint_state_frequency" default="100" />
8    <arg name="robot_state_frequency" default="100" />
9
10   <!-- Loads joint controller configurations from YAML file to parameter server -->
11   <rosparam file="$(find arm_control)/config/arm_control.yaml" command="load" />
12   <param name="/arm/joint_state_controller/publish_rate" value="$(arg joint_state_frequency)" />
13
14   <!-- Loads the controllers -->
15   <node name="controller_spawner" pkg="controller_manager" type="spawner" respawn="false" output="screen" args="$(arg controllers)" />
16
17   <!-- Converts joint states to TF transforms for rviz, etc -->
18   <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher" respawn="false" output="screen">
19     <remap from="joint_states" to="/arm/joint_states" />
20     <param name="publish_frequency" value="$(arg robot_state_frequency)" />
21   </node>
22
23</launch>

```

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



```

Aprile - arm_control.yaml
~/catkin_ws/src/arm_control/config
Salva

1#arm:
2
3# Publish all joint states -----
4 joint_state_controller:
5  type: joint_state_controller/JointStateController
6  publish_rate: 50
7
8# Forward Position Controllers -----
10 joint0_position_controller:
11  type: position_controllers/JointPositionController
12  joint: j0
13 joint1_position_controller:
14  type: position_controllers/JointPositionController
15  joint: j1
17 joint2_position_controller:
18  type: position_controllers/JointPositionController
19  joint: j2
21 joint3_position_controller:
22  type: position_controllers/JointPositionController
24  joint: j3
25

```

- 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
```

```
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo/launch$ cd src/
giuliag@giuliapc:~/catkin_ws/src$ cd arm_gazebo/
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo$ cd launch/
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo/launch$ touch arm_gazebo.launch
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo/launch$ ls
arm_gazebo.launch  arm_world.launch
giuliag@giuliapc:~/catkin_ws/src/arm_gazebo/launch$
```

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

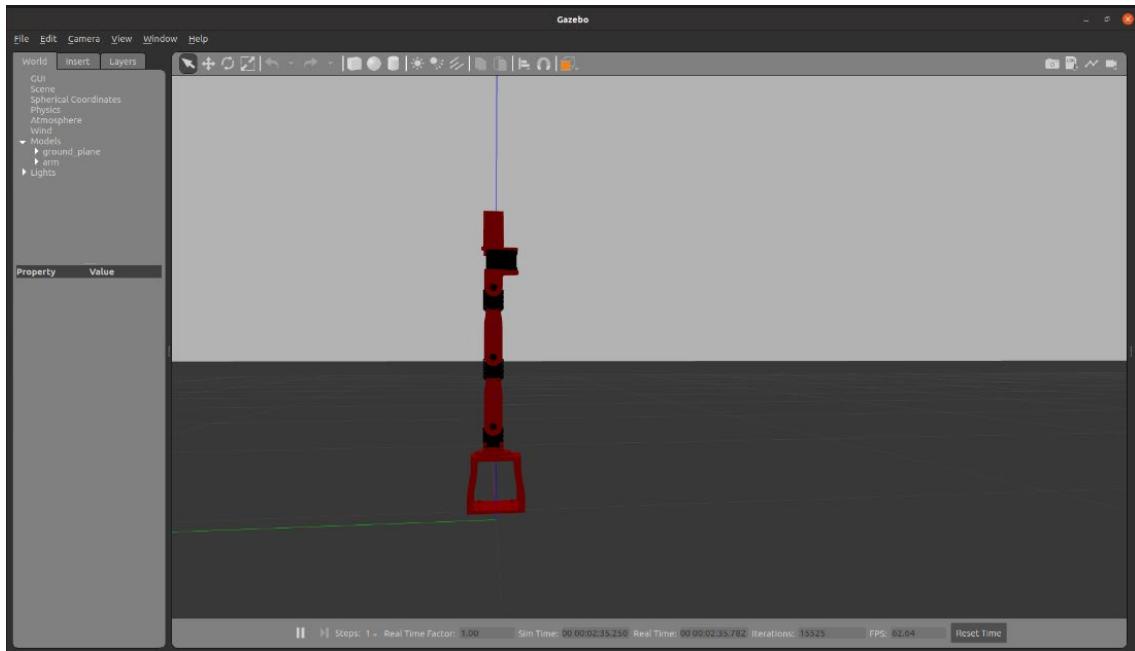
```
1 <?xml version="1.0"?>
2 <robot xmlns:xacro="http://ros.org/wiki/xacro">
3
4 <xacro:macro name="arm_gazebo">
5
6 <!-- Controller -->
7   <gazebo>
8     <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
9       <robotNamespace>/arm</robotNamespace>
10    </plugin>
11  </gazebo>
12
```

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

```
1 <?xml version="1.0"?>
2 <launch>
3
4   <arg name="hardware_interface" default="PositionJointInterface" />
5
6   <!-- Loads the Gazebo world. -->
7   <include file="$(find arm_gazebo)/launch/arm_world.launch">
8     <arg name="hardware_interface" value="$(arg hardware_interface)" />
9   </include>
10
11   <group ns="arm">
12     <include file="$(find arm_control)/launch/arm_control.launch">
13       <arg name="hardware_interface" value="$(arg hardware_interface)" />
14       <arg name="controllers" value="joint_state_controller joint0_position_controller joint1_position_controller
joint2_position_controller joint3_position_controller" />
15     </include>
16   </group>
17
18 </launch>
```

Used commands

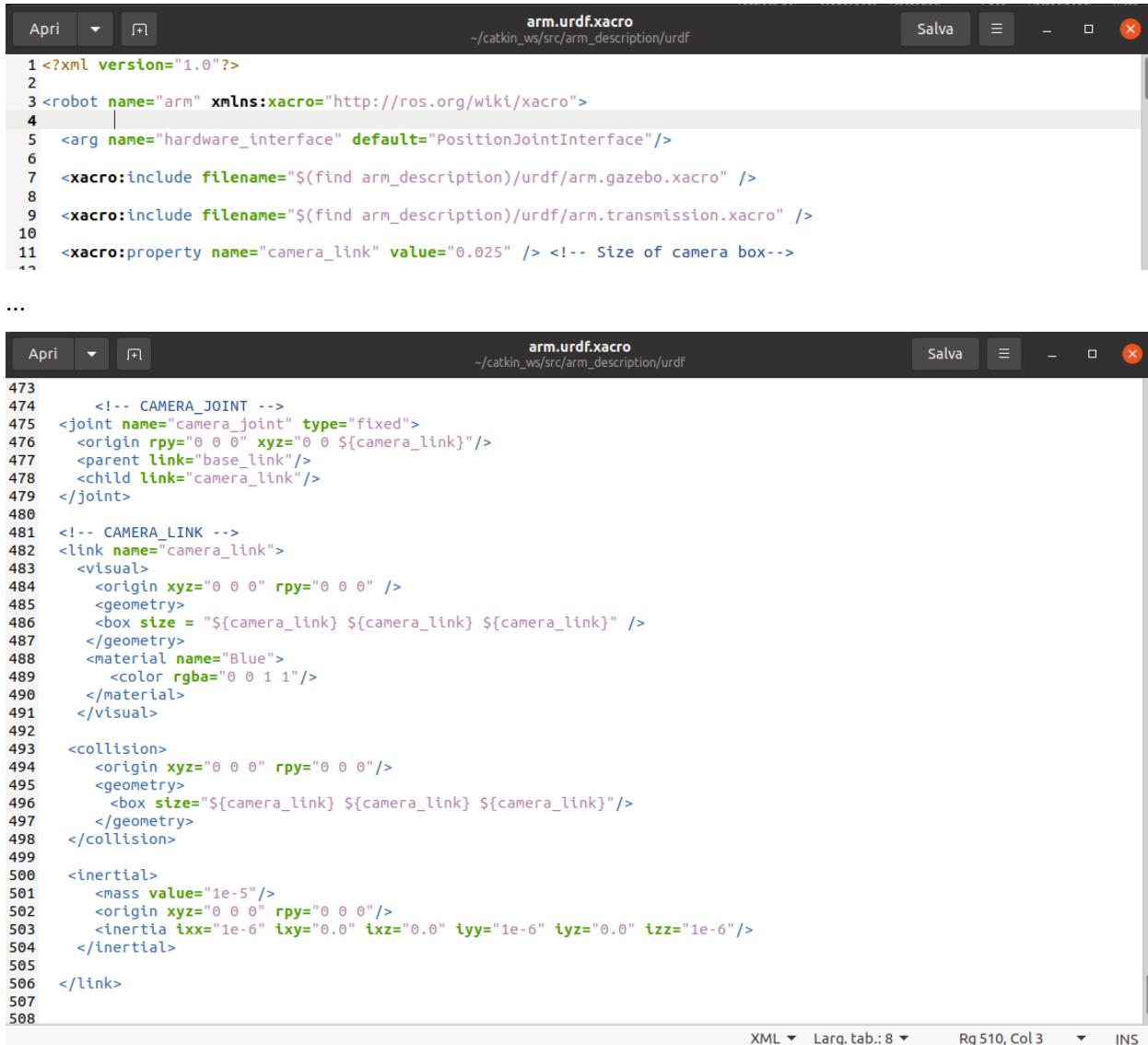
```
$ catkin build
$ source devel/setup.bash
$ roslaunch arm_gazebo arm_gazebo.launch
```



3. Add a camera sensor to your robot

- Go into your arm.urdf.xacro file 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:

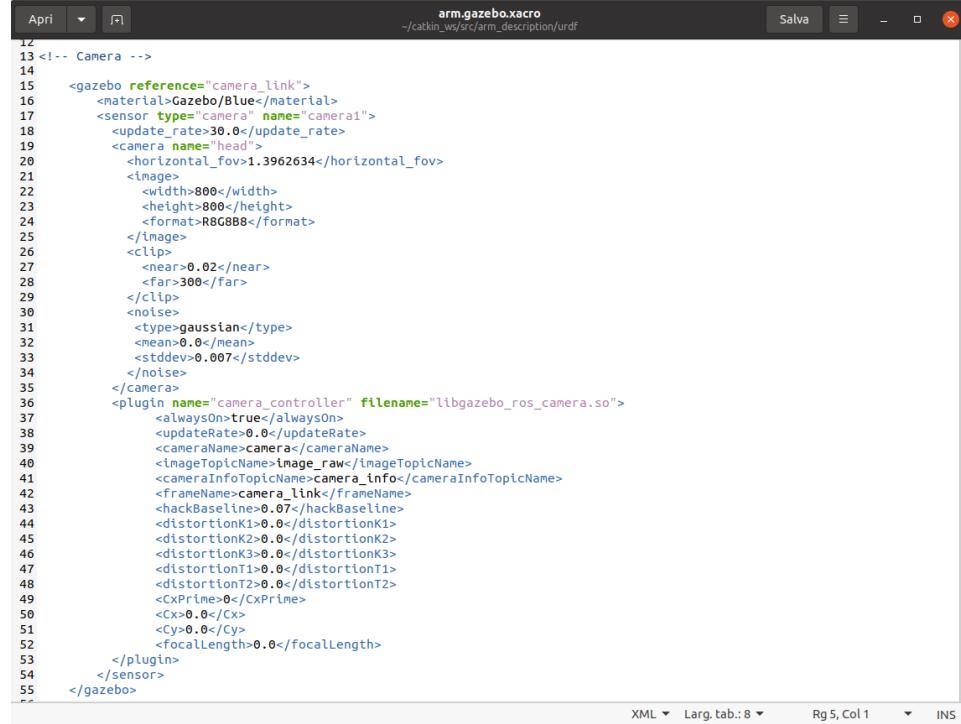


The image shows a code editor with two tabs open, both titled "arm.urdf.xacro". The top tab contains lines 1 through 12 of the XML code. The bottom tab contains lines 473 through 508. The code defines a robot named "arm" with hardware_interface and various xacro includes. It also defines a camera_link with a fixed camera_joint as its parent, positioned at (0, 0, 0) and oriented along the z-axis. The camera_link has a blue material, a size of 0.025, and a collision volume. The inertial properties are set to zero.

```
1 <?xml version="1.0"?>
2
3 <robot name="arm" xmlns:xacro="http://ros.org/wiki/xacro">
4   ...
5   <arg name="hardware_interface" default="PositionJointInterface"/>
6
7   <xacro:include filename="$(find arm_description)/urdf/arm.gazebo.xacro" />
8
9   <xacro:include filename="$(find arm_description)/urdf/arm.transmission.xacro" />
10
11  <xacro:property name="camera_link" value="0.025" /> <!-- Size of camera box-->
12
...
473      <!-- CAMERA_JOINT -->
474    <joint name="camera_joint" type="fixed">
475      <origin rpy="0 0 0" xyz="0 0 ${camera_link}" />
476      <parent link="base_link"/>
477      <child link="camera_link"/>
478    </joint>
479
480    <!-- CAMERA_LINK -->
481    <link name="camera_link">
482      <visual>
483        <origin xyz="0 0 0" rpy="0 0 0" />
484        <geometry>
485          <box size = "${camera_link} ${camera_link} ${camera_link}" />
486        </geometry>
487        <material name="Blue">
488          <color rgba="0 0 1 1"/>
489        </material>
490      </visual>
491
492      <collision>
493        <origin xyz="0 0 0" rpy="0 0 0"/>
494        <geometry>
495          <box size="${camera_link} ${camera_link} ${camera_link}" />
496        </geometry>
497      </collision>
498
499      <inertial>
500        <mass value="1e-5"/>
501        <origin xyz="0 0 0" rpy="0 0 0"/>
502        <inertia ixx="1e-6" ixy="0.0" ixz="0.0" iyy="1e-6" iyz="0.0" izz="1e-6"/>
503      </inertial>
504
505    </link>
506
507
508
```

- 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.



```

12 <!-- Camera -->
13
14     <gazebo reference="camera_link">
15         <material>Gazebo/Blue</material>
16         <sensor type="camera" name="camera1">
17             <update_rate>30.0</update_rate>
18             <camera name="head">
19                 <horizontal_fov>1.3962634</horizontal_fov>
20                 <image>
21                     <width>800</width>
22                     <height>800</height>
23                     <format>RGB8B8</format>
24                 </image>
25                 <clip>
26                     <near>0.02</near>
27                     <far>300</far>
28                 </clip>
29                 <noise>
30                     <type>gaussian</type>
31                     <mean>0.0</mean>
32                     <stddev>0.007</stddev>
33                 </noise>
34             </camera>
35             <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
36                 <alwaysOn>true</alwaysOn>
37                 <updateRate>0.0</updateRate>
38                 <cameraName>camera</cameraName>
39                 <imageTopicName>image_raw</imageTopicName>
40                 <cameraInfoTopicName>camera_info</cameraInfoTopicName>
41                 <frameName>camera_link</frameName>
42                 <hackBaseline>0.07</hackBaseline>
43                 <distortionK1>0.0</distortionK1>
44                 <distortionK2>0.0</distortionK2>
45                 <distortionK3>0.0</distortionK3>
46                 <distortionT1>0.0</distortionT1>
47                 <distortionT2>0.0</distortionT2>
48                 <cX>0.0</cX>
49                 <cY>0.0</cY>
50                 <focalLength>0.0</focalLength>
51             </plugin>
52         </sensor>
53     </gazebo>

```

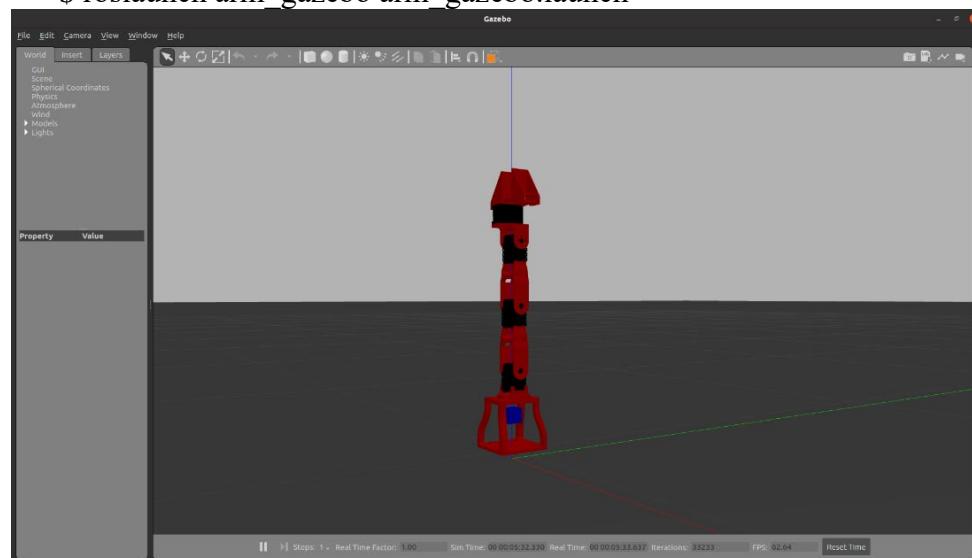
- c. Launch the Gazebo simulation with using arm_gazebo.launch and check if the image topic is correctly published using rqt_image_view

Used commands

```

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

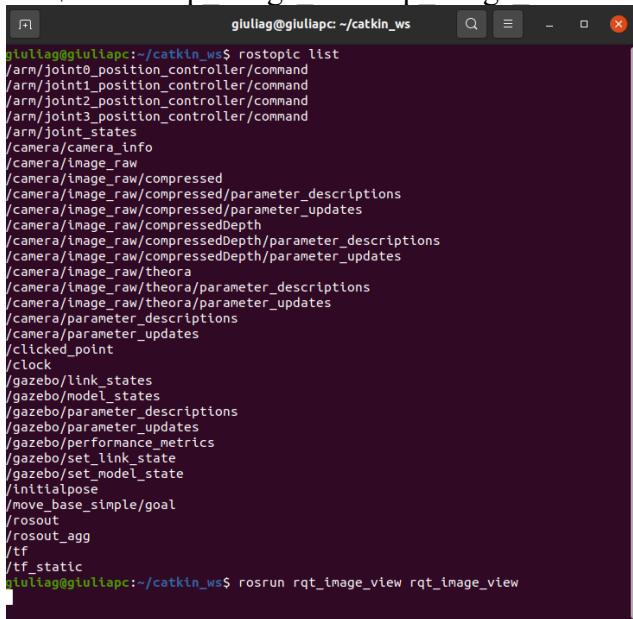
```



Used commands

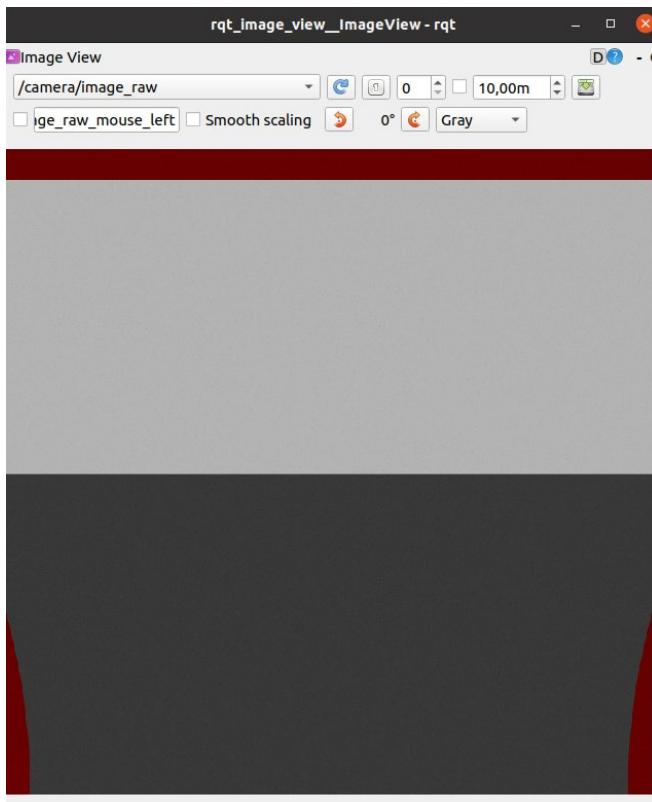
```
$ rostopic list
```

```
$ rosrun rqt image view rqt image view
```



```
gluлаг@gluliapc:~/catkin_ws$ rostopic list
/arm/joint0_position_controller/command
/arm/joint1_position_controller/command
/arm/joint2_position_controller/command
/arm/joint3_position_controller/command
/arm/joint_states
/camera/camera_info
/camera/image_raw
/camera/image_raw/compressed
/camera/image_raw/compressed/parameter_descriptions
/camera/image_raw/compressed/parameter_updates
/camera/image_raw/compressedDepth
/camera/image_raw/compressedDepth/parameter_descriptions
/camera/image_raw/compressedDepth/parameter_updates
/camera/image_raw/theora
/camera/image_raw/theora/parameter_descriptions
/camera/image_raw/theora/parameter_updates
/camera/parameter_descriptions
/camera/parameter_updates
/clicked_point
/clock
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/performance_metrics
/gazebo/set_link_state
/gazebo/set_model_state
/initialpose
/move_base_simple/goal
/rosout
/rosout_agg
/tf
/tf_static
gluлаг@gluliapc:~/catkin_ws$ rosrun rqt_image_view rqt_image_view
```

The image topic is shown:



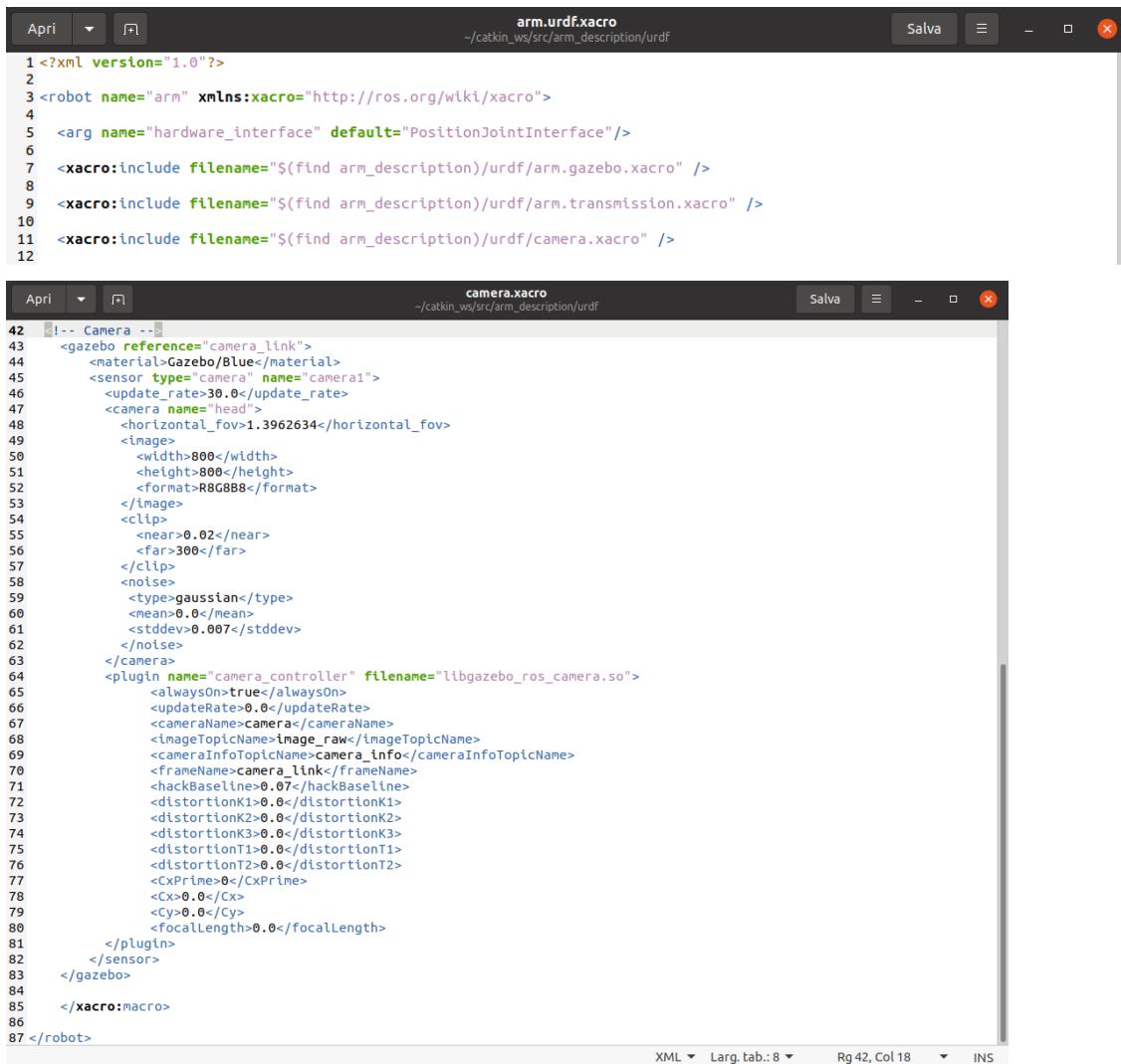
- d. Optionally: You can create a camera.xacro file and add it to your robot URDF using <xacro:include>

Used commands

```
$ cd src/arm_description/urdf
$ touch camera.xacro
$ ls
```



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



The screenshot shows a code editor with two tabs: "arm.urdf.xacro" and "camera.xacro". The "arm.urdf.xacro" tab contains the following XML code:

```
1 <?xml version="1.0"?>
2
3 <robot name="arm" xmlns:xacro="http://ros.org/wiki/xacro">
4
5   <arg name="hardware_interface" default="PositionJointInterface"/>
6
7   <xacro:include filename="$(find arm_description)/urdf/arm.gazebo.xacro" />
8
9   <xacro:include filename="$(find arm_description)/urdf/arm.transmission.xacro" />
10
11  <xacro:include filename="$(find arm_description)/urdf/camera.xacro" />
12
```

The "camera.xacro" tab contains the following XML code, which defines a camera named "camera1" with specific parameters like resolution and noise characteristics:

```
42 <!-- Camera -->
43   <gazebo reference="camera_link">
44     <material>Gazebo/Blue</material>
45     <sensor type="camera" name="camera1">
46       <update_rate>30.0</update_rate>
47       <camera name="head">
48         <horizontal_fov>1.3962634</horizontal_fov>
49         <image>
50           <width>800</width>
51           <height>800</height>
52           <format>R8G8B8</format>
53         </image>
54         <clip>
55           <near>0.02</near>
56           <far>300</far>
57         </clip>
58         <noise>
59           <type>gaussian</type>
60           <mean>0.0</mean>
61           <stddev>0.007</stddev>
62         </noise>
63       </camera>
64       <plugin name="camera_controller" filename="libgazebo_ros_camera.so">
65         <alwaysOn>true</alwaysOn>
66         <updateRate>0.0</updateRate>
67         <cameraName>camera</cameraName>
68         <imageTopicName>image_raw</imageTopicName>
69         <cameraInfoTopicName>Camera_info</cameraInfoTopicName>
70         <frameName>camera_link</frameName>
71         <hackBaseline>0.07</hackBaseline>
72         <distortionK1>0.0</distortionK1>
73         <distortionK2>0.0</distortionK2>
74         <distortionK3>0.0</distortionK3>
75         <distortionT1>0.0</distortionT1>
76         <distortionT2>0.0</distortionT2>
77         <CxPrime>0.0</CxPrime>
78         <CyPrime>0.0</CyPrime>
79         <focalLength>0.0</focalLength>
80       </plugin>
81     </sensor>
82   </gazebo>
83 </xacro:macro>
84
85 </robot>
```

```

1 <?xml version="1.0"?>
2 <robot xmlns:xacro="http://www.ros.org/wiki/xacro" name="camera">
3
4   <xacro:macro name="camera">
5     <xacro:property name="camera_link" value="0.025" /> <!-- Size of camera box-->
6
7     <!-- CAMERA_JOINT -->
8     <joint name="camera_joint" type="fixed">
9
10      <origin rpy="0 0 0" xyz="0 0 0 ${camera_link}" />
11      <parent link="base_link"/>
12      <child link="camera_link"/>
13    </joint>
14
15    <!-- CAMERA_LINK -->
16    <link name="camera_link">
17      <visual>
18        <origin xyz="0 0 0" rpy="0 0 0" />
19        <geometry>
20          <box size = "${camera_link} ${camera_link} ${camera_link}" />
21        </geometry>
22        <material name="Blue">
23          <color rgba="0 0 1 1"/>
24        </material>
25      </visual>
26
27      <collision>
28        <origin xyz="0 0 0" rpy="0 0 0" />
29        <geometry>
30          <box size=" ${camera_link} ${camera_link} ${camera_link}" />
31        </geometry>
32      </collision>
33
34      <inertial>
35        <mass value="1e-5"/>
36        <origin xyz="0 0 0" rpy="0 0 0" />
37        <inertia ixz="1e-6" ixy="0.0" txz="0.0" iyy="1e-6" iyz="0.0" tzz="1e-6" />
38      </inertial>
39
40    </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

- 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

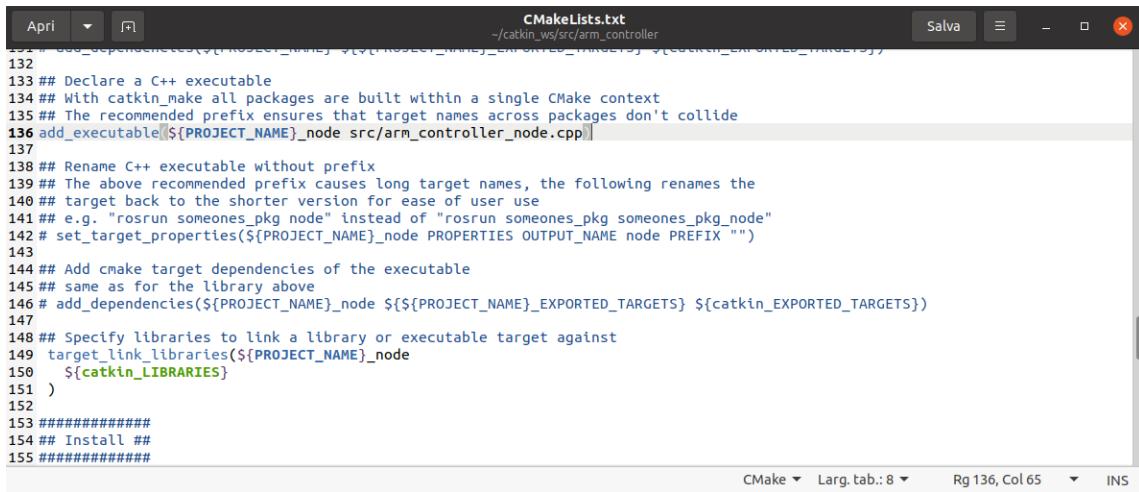
```

```

giuliag@giuliapc:~/catkin_ws/src$ 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 values 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.



A screenshot of a code editor window titled "CMakeLists.txt". The file path is "-/catkin_ws/src/arm_controller". The code is a CMake script for building an executable named "arm_controller_node". It includes sections for dependencies, executable creation, target renaming, cmake target properties, library linking, and installation. Lines 136 and 140 show the addition of the executable target. Line 141 shows a note about target prefixing. Line 142 sets target properties. Line 144 adds dependencies. Line 148 specifies libraries to link. Line 150 lists catkin libraries. Line 154 starts an install section. Line 155 ends it. The code editor interface includes tabs for "CMake", "Larg. tab.: 8", and "Rg 136, Col 65", and a status bar with "INS".

```
131 ## add_dependencies(${PROJECT_NAME} ${${PROJECT_NAME}_EXPORTED_TARGETS} ${${PROJECT_NAME}_IMPORTED_TARGETS})
132
133 ## Declare a C++ executable
134 ## With catkin_make all packages are built within a single CMake context
135 ## The recommended prefix ensures that target names across packages don't collide
136 add_executable(${PROJECT_NAME}_node src/arm_controller_node.cpp)
137
138 ## Rename C++ executable without prefix
139 ## The above recommended prefix causes long target names, the following renames the
140 ## target back to the shorter version for ease of user use
141 ## e.g. "rosrun someones_pkg node" instead of "rosrun someones_pkg someones_pkg_node"
142 # set_target_properties(${PROJECT_NAME}_node PROPERTIES OUTPUT_NAME node PREFIX "")
143
144 ## Add cmake target dependencies of the executable
145 ## same as for the library above
146 # add_dependencies(${PROJECT_NAME}_node ${${PROJECT_NAME}_EXPORTED_TARGETS} ${${PROJECT_NAME}_IMPORTED_TARGETS})
147
148 ## Specify libraries to link a library or executable target against
149 target_link_libraries(${PROJECT_NAME}_node
150   ${catkin_LIBRARIES}
151 )
152
153 #####
154 ## Install ##
155 #####
```

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



A screenshot of a code editor window titled "arm_controller_node (copia).cpp". The file path is "-/catkin_ws/src/arm_controller/src". The code is a ROS node named "arm_controller_node" that subscribes to the "joint_states" topic. It defines a callback function "joint_statesCallback" that prints the received joint state message. The main function initializes the ROS node and starts the spin loop. The code editor interface includes tabs for "CMake", "Larg. tab.: 8", and "Rg 136, Col 65", and a status bar with "INS".

```
1 #include <ros/ros.h>
2 #include <sensor_msgs/JointState.h>
3
4 // Callback function for 'joint_states' topic
5 void joint_statesCallback(const sensor_msgs::JointState::ConstPtr& msg){
6
7   ROS_INFO("Received joint_state message:");
8   // Print joint positions
9   for (size_t i = 0; i < msg->position.size(); i++) {
10     ROS_INFO("Position of Joint %s: %f",msg->name[i].c_str() , msg->position[i]);
11   }
12 }
13
14 int main(int argc, char** argv)
15 {
16   // Initialize ROS node
17   ros::init(argc, argv, "arm_controller_node");
18   ros::NodeHandle node;
19
20   // Create a subscriber
21   ros::Subscriber jointstates_Sub = node.subscribe("arm/joint_states", 10, joint_statesCallback);
22
23   ros::spin();
24
25   return 0;
26 }
```

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

```

1 #include <ros/ros.h>
2 #include <sensor_msgs/JointState.h>
3 #include <std_msgs/Float64.h>
4
5 // Callback function for 'joint_states' topic
6 void joint_statesCallback(const sensor_msgs::JointState::ConstPtr& msg){
7
8     ROS_INFO("Received joint_state message:");
9     // Print joint positions
10    for (size_t i = 0; i < msg->position.size(); i++) {
11        ROS_INFO("Position of Joint %s: %f", msg->name[i].c_str(), msg->position[i]);
12    }
13 }
14
15 int main(int argc, char** argv)
16 {
17     // Initialize ROS node
18     ros::init(argc, argv, "arm_controller_node");
19     ros::NodeHandle node;
20
21     // Create a subscriber
22     ros::Subscriber jointstates_Sub = node.subscribe("arm/joint_states", 10, joint_statesCallback);
23
24     // Create publishers
25     ros::Publisher joint0_Pub = node.advertise<std_msgs::Float64>("arm/joint0_position_controller/command", 1);
26     ros::Publisher joint1_Pub = node.advertise<std_msgs::Float64>("arm/joint1_position_controller/command", 1);
27     ros::Publisher joint2_Pub = node.advertise<std_msgs::Float64>("arm/joint2_position_controller/command", 1);
28     ros::Publisher joint3_Pub = node.advertise<std_msgs::Float64>("arm/joint3_position_controller/command", 1);
29     ros::Rate loopRate(10);
30
31     if(argc != 5){
32         ROS_INFO("Repeat the command with the right number of the desired joint positions...");
33         return 0;
34     }
35
36     double position_j0 = std::stod(argv[1]);
37     double position_j1 = std::stod(argv[2]);
38     double position_j2 = std::stod(argv[3]);
39     double position_j3 = std::stod(argv[4]);
40
41     std_msgs::Float64 joint0_command, joint1_command, joint2_command, joint3_command;
42
43     joint0_command.data = position_j0;
44     joint1_command.data = position_j1;
45     joint2_command.data = position_j2;
46     joint3_command.data = position_j3;
47
48     while (ros::ok()) {
49
50         ROS_INFO("Position command to joint0: %.2f", joint0_command.data);
51         joint0_Pub.publish(joint0_command);
52
53         ROS_INFO("Position command to joint1: %.2f", joint1_command.data);
54         joint1_Pub.publish(joint1_command);
55
56         ROS_INFO("Position command to joint2: %.2f", joint2_command.data);
57         joint2_Pub.publish(joint2_command);
58
59         ROS_INFO("Position command to joint3: %.2f", joint3_command.data);
60         joint3_Pub.publish(joint3_command);
61
62         ros::spinOnce();
63         loopRate.sleep();
64     }
65
66     return 0;
67 }

```

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
```

```

gluliag@gluliapc:~/catkin_ws/src/arm_controller/src$ rosrun arm_controller arm_controller_node 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] [1698744063.093706211]: 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] [1698744063.439706705], 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] [1698744063.440136929], 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] [1698744063.441417271], 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] [1698744063.441766687], 1008.130000000]: Position of Joint j2: 0.000000
[ INFO] [1698744063.441832047], 1008.130000000]: Position of Joint j3: -0.000000
[ INFO] [1698744063.441997765], 1008.130000000]: Received joint_state message:
[ INFO] [1698744063.442370736], 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

```

...

```

gluliag@gluliapc:~/catkin_ws/src/arm_controller/src
[ INFO] [1098744428.270025430], 10.890000000]: Position of Joint j3: 0.992000
[ 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] [1698744428.276866254], 16.890000000]: Received joint_state message:
[ INFO] [1698744428.276960335], 16.890000000]: Position of Joint j0: 0.994000
[ INFO] [1698744428.277088866], 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] [1698744428.278972906], 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

```

