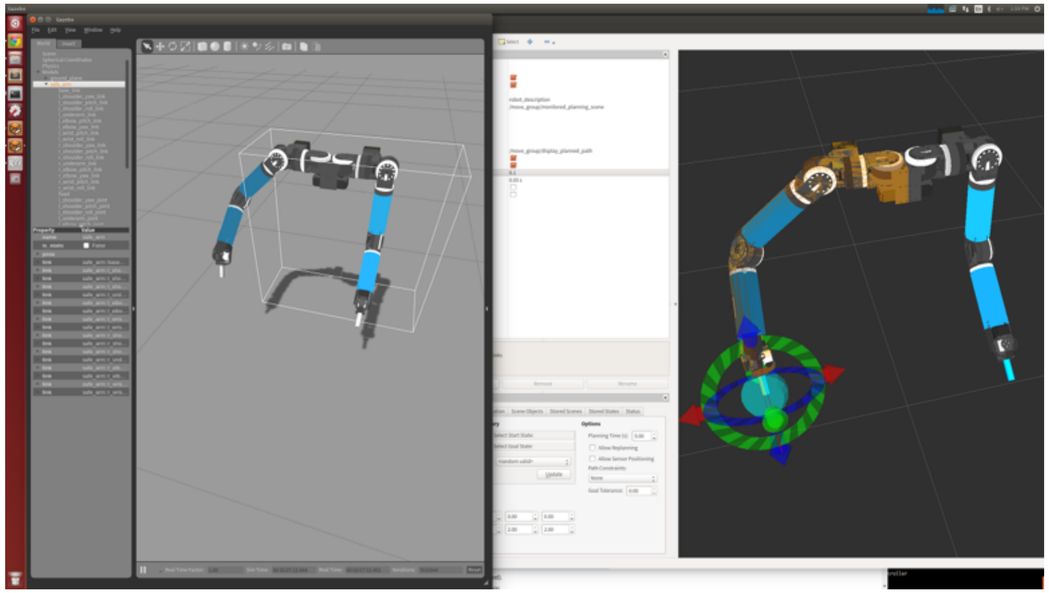


Gazebo, MoveIt!, ros_control을 활용한 로봇팔 모델링 및 제어

2nd Open Robotics Seminar

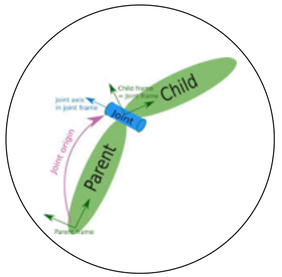
December 22, 2014

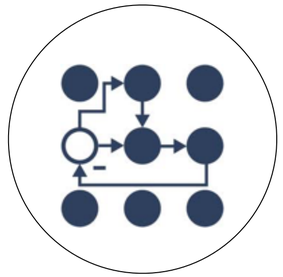
Byeong-Kyu Ahn (byeongkyu@gmail.com)

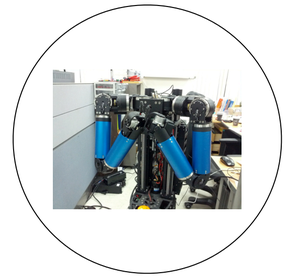
오로카

An Overview

- ✓ Prerequisite
- ✓ Robot (Target)
- ✓ UDRF
- ✓ Gazebo
- ✓ Controller
- ✓ MoveIt
- ✓ Demo
- ✓ Real Robot







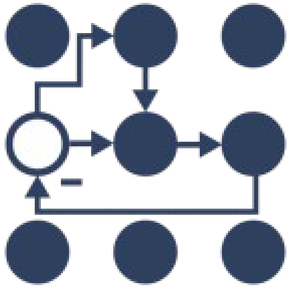
준비해야 할것들

Prerequisite

ROS Indigo Igloo [1)]
Ubuntu 14.04 LTS (Trusty)

Gazebo 4.1.0 [2)]

MoveIt [3)]

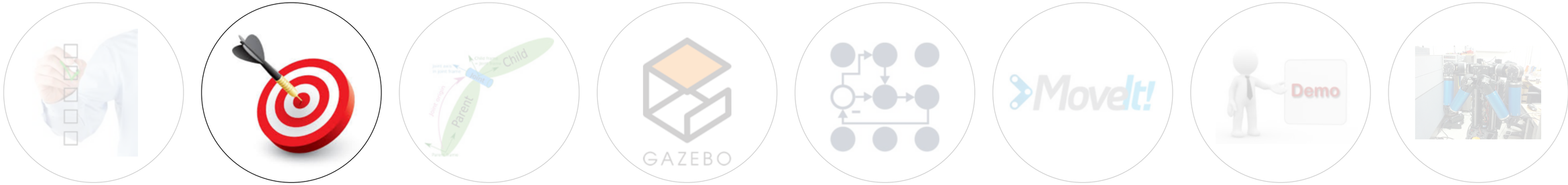


ros_control [4)]

1) ROS Installation: http://wiki.ros.org/indigo/Installation/Ubuntu
2) Gazebo Installation: http://gazebosim.org/tutorials?tut=install&ver=4.0&cat=get_started
3) MoveIt Installation: http://moveit.ros.org/install
4) ros_control: http://wiki.ros.org/ros_control

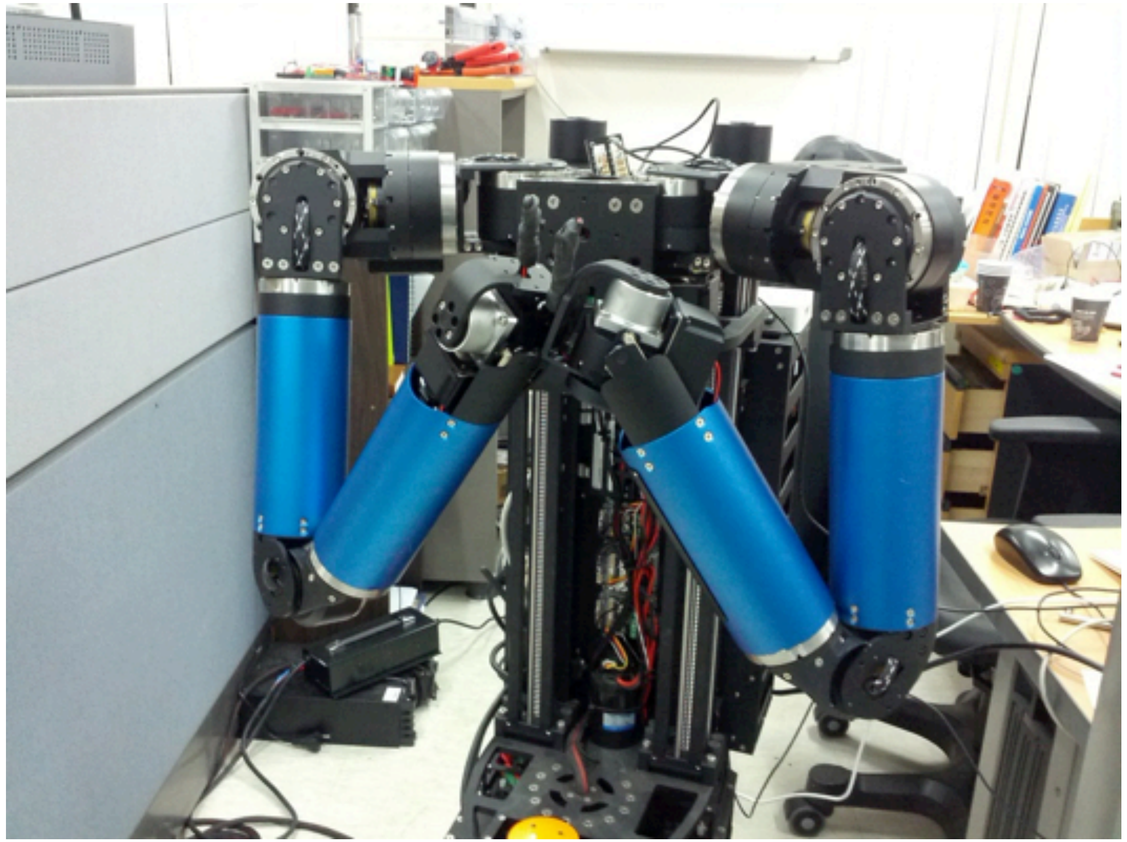
Install packages for MoveIt, ros_control

```
$ sudo apt-get install ros-indigo-moveit-*
$ sudo apt-get install ros-indigo-ros-control
$ sudo apt-get install ros-indigo-ros-controller
$ sudo apt-get install ros-indigo-joint-state-controller
$ sudo apt-get install ros-indigo-effort-controllers
$ sudo apt-get install ros-indigo-position-controllers
$ sudo apt-get install ros-indigo-joint-trajectory-controller
```



타겟 선정 (로봇)

Robot



고기능 작업용 팔

로봇산업원천기술개발사업 - 로봇 H/W플랫폼기술 개발 (2009. 06 ~ 2011.05)

Specification

항목	설명
재원	길이: 670.73mm 무게: 7.63kg
자유도	8 DoF (어깨 4, 팔꿈치 1, 손목 3)
가반하중	3.0kg
종단속도	0.6m/s
통신	CAN
전원	24V
제어방식	위치, 속도, 전류제어

Characteristic

- 어깨와 몸통 사이의 자유도 구현 (총 8자유도)
- 외력 반응성 (Backdrivablity) 향상을 위한 저기어비 설계 (100/1)
- 타이밍 벨트 및 풀리에서 생기는 마찰 및 손실 방지를 위해 모터/기어 직결모듈 방식
- 각 관절의 절대위치 인식 (엡솔루트 엔코더)
- 토크 제어 (전류제어)
- 외부와 접촉 시 접촉 힘 측정 및 대응 (F/T 센서 장착시)
- 중공형 기어, 모터, 엔코더 사용

알아내야 할 것들

Separate
Parts
(Link)

Part별
Mass(kg)
Inertia Tensor
Center of Mass

Part별
COLLADA 파일
or STL

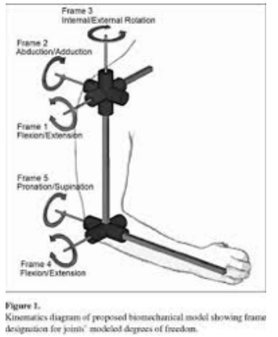


Figure 1.
Kinematics diagram of proposed biomechanical model showing frame designation for joints' modeled degrees of freedom.

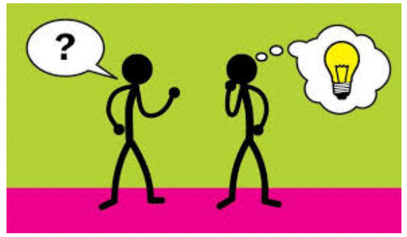
Joint 정보
회전축, 한계값
최대속도 등

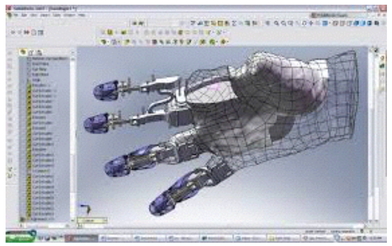
COLLADA.org - https://collada.org

http://ko.wikipedia.org/wiki/관성_모멘트

어떻게 구하나?

설계한 사람한테 물어보기





Mass(kg)
Inertia Tensor
Center of Mass

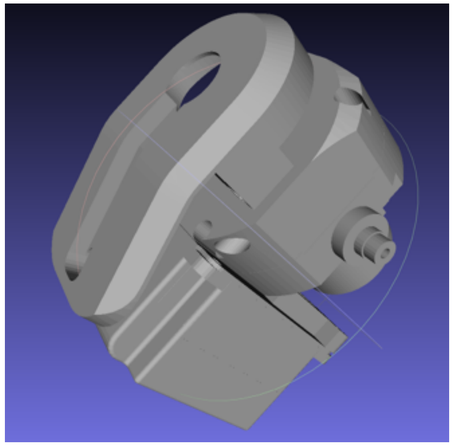
직접 구하기(근사치)

MeshLab

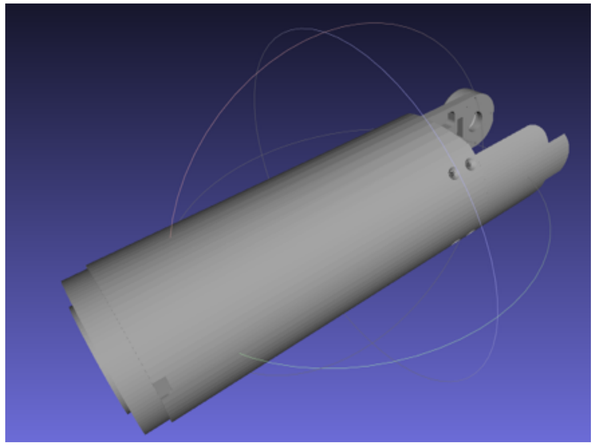
- Find out inertial parameters: http://gazebosim.org/tutorials?tut=inertia&cat=

Menu > Filters > Quality Measure and Computations Compute > Geometric Measures

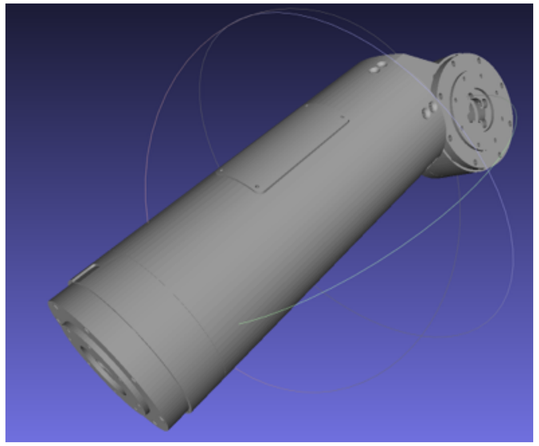
```
Mesh Bounding Box Size 115.100006 83.000015 85.500000
Mesh Bounding Box Diag 165.672150
Mesh Volume is 223733.125000
Mesh Surface is 126935.031250
Thin shell barycenter 14.123314 -0.036923 8.072087
Center of Mass is 8.950365 0.008595 7.867443
Inertia Tensor is :
| 252065328.000000 150681.546875 -32695376.000000 |
| 150681.546875 417160000.000000 -490.519348 |
| -32695376.000000 -490.519348 423229216.000000 |
Principal axes are :
| -0.983400 0.000865 -0.181451 |
| -0.000434 -0.999997 -0.002418 |
| -0.181453 -0.002299 0.983397 |
axis momenta are :
| 246032432.000000 417160032.000000 429262048.000000 |
```



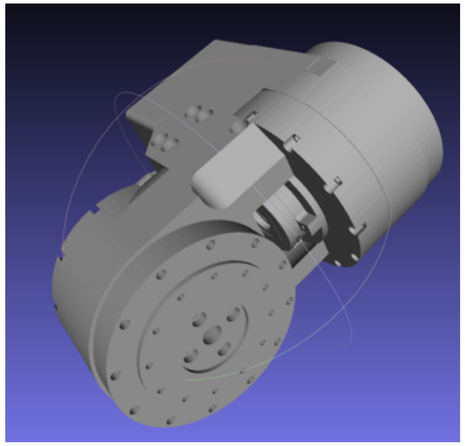
wrist_pitch



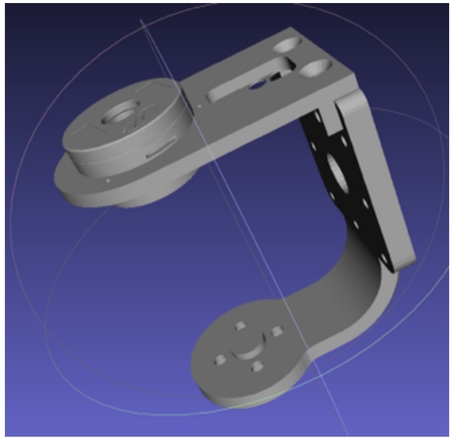
elbow_yaw



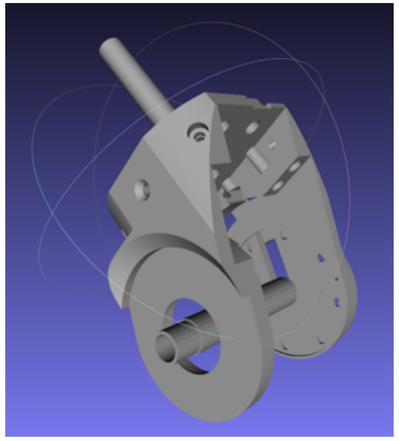
underarm



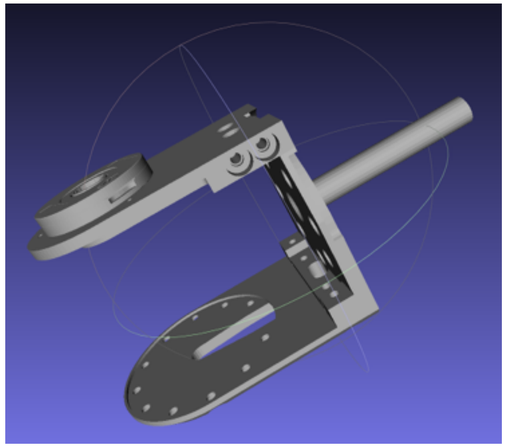
shoulder_pitch



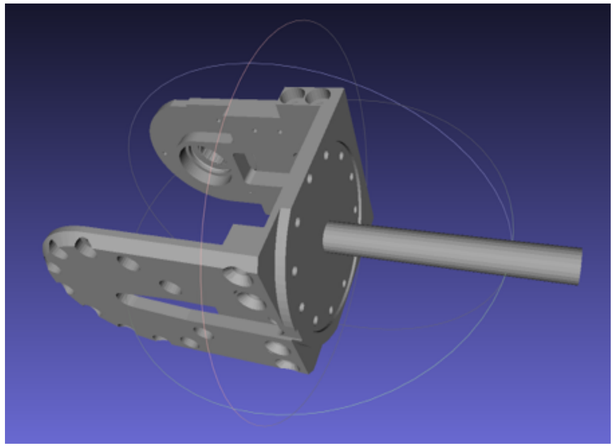
wrist_roll



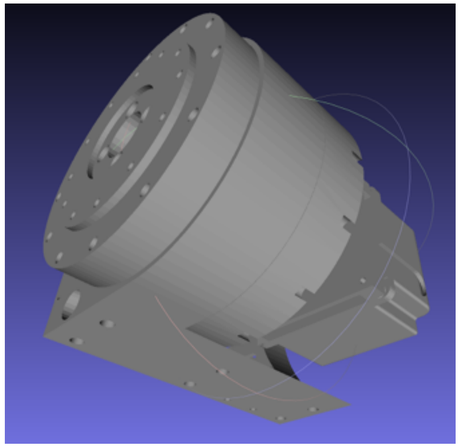
elbow_pitch



shoulder_roll



shoulder_yaw

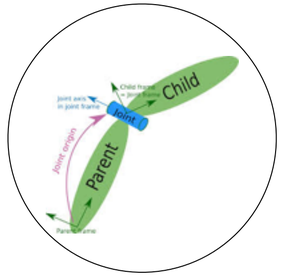


shoulder_base

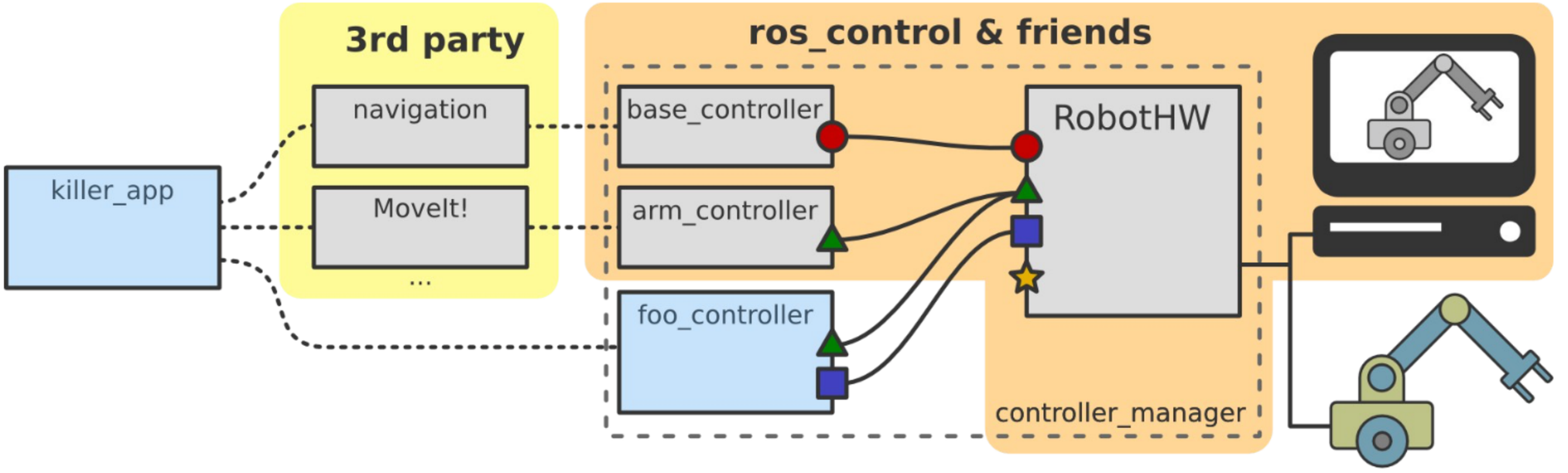
					Mass	Inertial Tensor						X	Y	Z
Link1	423229248.0000000	-499.4013370	-32695388.0000000	223733.1250000	1.1884610	0.0022482	0.0000000	-0.0001737	36.1625560	-0.0086030	-8.9503600	0.0361626	-0.0000086	-0.0089504
	-499.4013370	417160000.0000000	-490.5193480			0.0000000	0.0022159	0.0000000						
	-32695388.0000000	-490.5193480	252065344.0000000			-0.0001737	0.0000000	0.0013390						
Link2	60088316.0000000	24240.5195310	-781203.1250000	57622.3906250	0.1586760	0.0001655	0.0000001	-0.0000022	31.9138600	0.0109980	-3.1692050	0.0319139	0.0000110	-0.0031692
	24240.5195310	86699512.0000000	17580.4687500			0.0000001	0.0002387	0.0000000						
	-781203.1250000	17580.4687500	57394384.0000000			-0.0000022	0.0000000	0.0001580						
Link3	1141275648.0000000	-20703460.0000000	-159318880.0000000	455601.8125000	2.4071690	0.0060299	-0.0001094	-0.0008418	-14.7117800	-1.0823950	64.0507430	-0.0147118	-0.0010824	0.0640507
	-20703460.0000000	1230401920.0000000	-13909991.0000000			-0.0001094	0.0065008	-0.0000735						
	-159318880.0000000	-13909991.0000000	683165824.0000000			-0.0008418	-0.0000735	0.0036095						
Link4	87160008.0000000	-38024.3398440	-40232.7890620	57862.9882810	0.1549730	0.0002334	-0.0000001	-0.0000001	0.0194220	2.0935930	-30.6698440	0.0000194	0.0020936	-0.0306698
	-38024.3398440	55174084.0000000	38020.0937500			-0.0000001	0.0001478	0.0000001						
	-40232.7890620	38020.0937500	59850288.0000000			-0.0000001	0.0000001	0.0001603						
Link5	3390847744.0000000	11710923.0000000	51480704.0000000	342217.1562500	1.9429050	0.0192512	0.0000665	0.0002923	1.3103730	-6.5368150	-126.9019700	0.0013104	-0.0065368	-0.1269020
	11710923.0000000	3346496512.0000000	11710947.0000000			0.0000665	0.0189994	0.0000665						
	51480704.0000000	11710947.0000000	328466528.0000000			0.0002923	0.0000665	0.0018648						
Link6	50388672.0000000	1558272.7500000	-3539348.0000000	58685.4296880	0.1698730	0.0001459	0.0000045	-0.0000102	-0.5237510	15.0388790	-24.0661260	-0.0005238	0.0150389	-0.0240661
	1558272.7500000	65878056.0000000	12389965.0000000			0.0000045	0.0001907	0.0000359						
	-3539348.0000000	12389965.0000000	54281604.0000000			-0.0000102	0.0000359	0.0001571						
Link7	1034856576.0000000	242925.1093750	-44354696.0000000	166373.2968750	1.4981100	0.0093184	0.0000022	-0.0003994	-1.9565280	-0.4413500	-82.1392670	-0.0019565	-0.0004414	-0.0821393
	242925.1093750	1054352448.0000000	-1441617.6250000			0.0000022	0.0094939	-0.0000130						
	-44354696.0000000	-1441617.6250000	166693856.0000000			-0.0003994	-0.0000130	0.0015010						
Link8	111444448.0000000	-226027.3593750	258395.2500000	147977.1250000	0.4192310	0.0003157	-0.0000006	0.0000007	-0.1057530	-0.4301010	16.0042320	-0.0001058	-0.0004301	0.0160042
	-226027.3593750	93393352.0000000	9533654.0000000			-0.0000006	0.0002646	0.0000270						
	258395.2500000	9533654.0000000	73619272.0000000			0.0000007	0.0000270	0.0002086						
Link9					0.3293260	#DIV/0!	#DIV/0!	#DIV/0!				0.0000000	0.0000000	0.0000000
						#DIV/0!	#DIV/0!	#DIV/0!						
						#DIV/0!	#DIV/0!	#DIV/0!						

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}$$

$$I_{ij} \stackrel{\text{def}}{=} \sum_k m_k (r_k^2 \delta_{ij} - r_{ki} r_{kj})$$



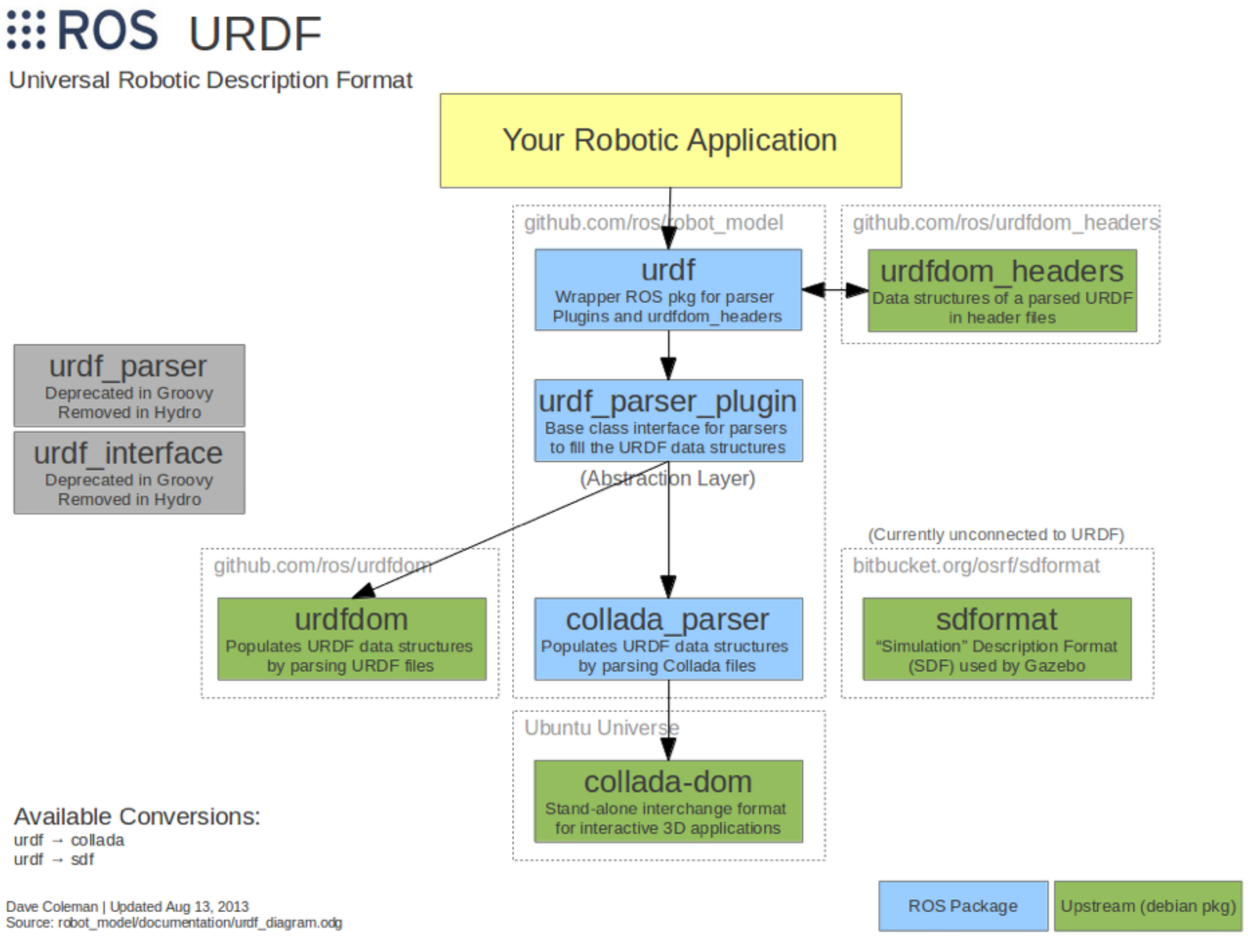
URDF 작성 및 확인

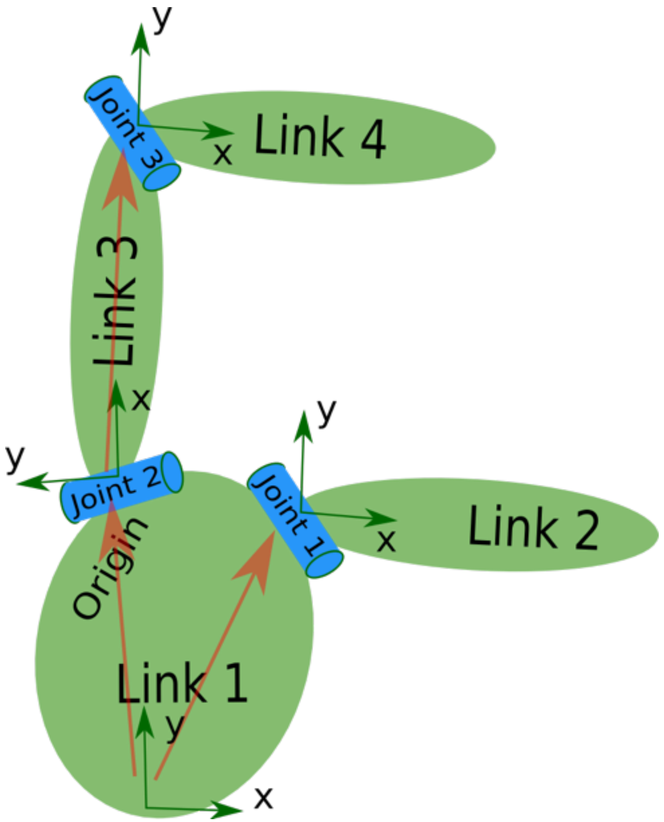


Adolfo Rodríguez Tsouroukdissian, “ros_control: An overview”, ROSCon 2014

URDF : URDF Tutorial - http://wiki.ros.org/urdf/Tutorials

- Build up your own robot models, sensors, scenes, etc.
- XML format for representing a robot model

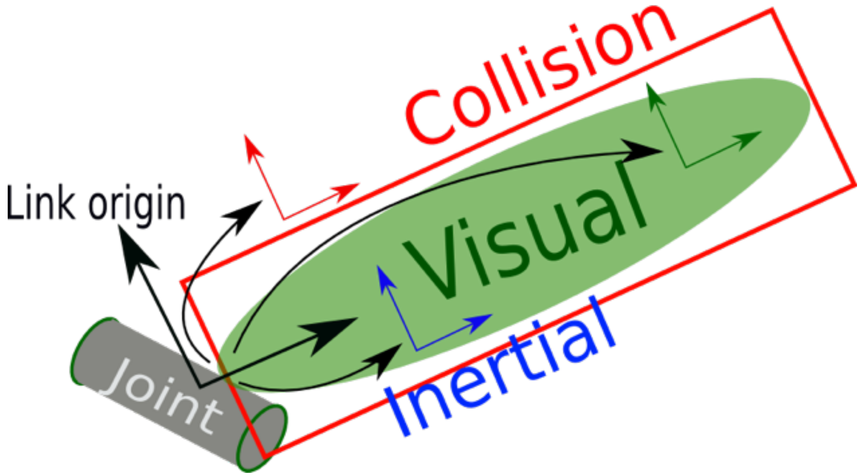





```
<robot name="test_robot">
  <link name="link1" />
  <link name="link2" />
  <link name="link3" />
  <link name="link4" />

  <joint name="joint1" type="continuous">
    <parent link="link1"/>
    <child link="link2"/>
  </joint>

  <joint name="joint2" type="continuous">
    <parent link="link1"/>
    <child link="link3"/>
  </joint>

  <joint name="joint3" type="continuous">
    <parent link="link3"/>
    <child link="link4"/>
  </joint>
</robot>
```

Link

http://wiki.ros.org/urdf/XML/link




```
<link name="my_link">
  <inertial>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="1.0" ixy="0" ixz="0" iyy="1.0" iyz="0" izz="1.0" />
  </inertial>

  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <box size="1 1 1" />
    </geometry>
    <material name="Cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
  </visual>

  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <cylinder radius="1" length="0.5"/>
    </geometry>
  </collision>
</link>
```

```
<inertial>
   <origin xyz="0 0 0.5" rpy="0 0 0"/>
   <mass value="1"/>
   <inertia ixx="1.0"  ixy="0"  ixz="0" iyy="1.0" iyz="0" izz="1.0" />
</inertial>
```

- Center of mass (meter)
- Mass (kg)
- Moments of inertia (kg * m^2)

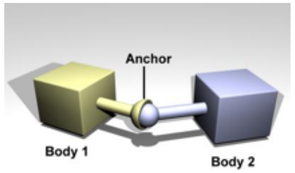
```
<visual>
   <origin xyz="0 0 0" rpy="0 0 0" />
   <geometry>
      <box size="1 1 1" />
      <mesh filename="package://safe_arm_meshes/meshes/link1.dae" scale="0.001 0.001 0.001" />
   </geometry>
   <material name="Cyan">
       <color rgba="0 1.0 1.0 1.0"/>
   </material>
</visual>

<collision>
   <origin xyz="0 0 0" rpy="0 0 0"/>
   <geometry>
      <cylinder radius="1" length="0.5"/>
   </geometry>
</collision>
```

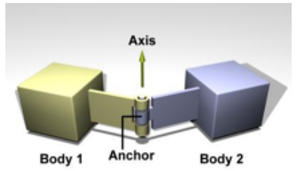
<geometry> *(required)*

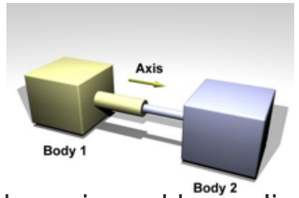
- The shape of the visual object. This can be *one* of the following:
- **<box>**
  - **size** attribute contains the three side lengths of the box. The origin of the box is in its center.
- **<cylinder>**
  - Specify the **radius** and **length**. The origin of the cylinder is in its center.
- **<sphere>**
  - Specify the **radius**. The origin of the sphere is in its center.
- **<mesh>**
  - A trimesh element specified by a **filename**, and an optional **scale** that scales the mesh's axis-aligned-bounding-box. The recommended format for best texture and color support is Collada .dae files, though .stl files are also supported. The mesh file is not transferred between machines referencing the same model. It must be a local file.

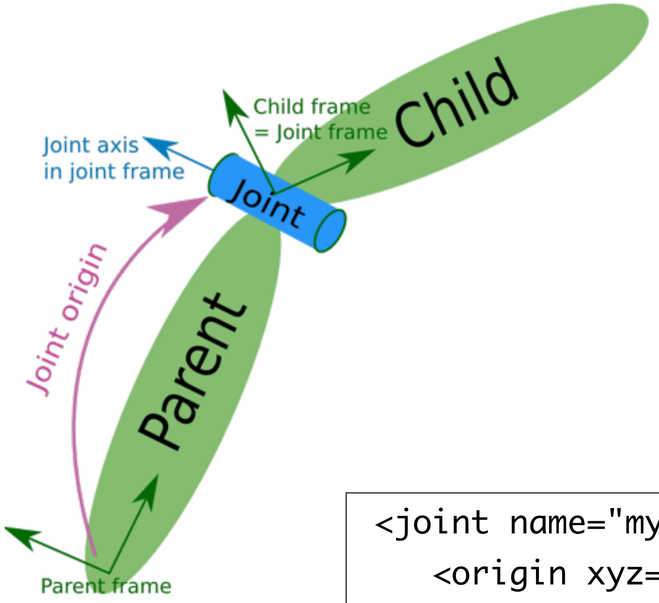
Joint

http://wiki.ros.org/urdf/XML/joint









- **revolute** - a hinge joint that rotates along the axis and has a limited range specified by the upper and lower limits.
- **continuous** - a continuous hinge joint that rotates around the axis and has no upper and lower limits
- **prismatic** - a sliding joint that slides along the axis, and has a limited range specified by the upper and lower limits.
- **fixed** - This is not really a joint because it cannot move. All degrees of freedom are locked. This type of joint does not require the axis, calibration, dynamics, limits or safety_controller.
- *floating* - This joint allows motion for all 6 degrees of freedom.
- *planar* - This joint allows motion in a plane perpendicular to the axis.

```
<joint name="my_joint" type="floating">
   <origin xyz="0 0 1" rpy="0 0 3.1416"/>
   <parent link="link1"/>
   <child link="link2"/>

   <calibration rising="0.0"/>
   <dynamics damping="0.0" friction="0.0"/>
   <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7" />
   <safety_controller k_velocity="10" k_position="15" soft_lower_limit="-2.0" soft_upper_limit="0.5" />
</joint>
```

Transmission

http://wiki.ros.org/urdf/XML/Transmission

```
<transmission name="${name}_tran">
   <type>transmission_interface/SimpleTransmission</type>
   <joint name="${name}">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
   </joint>

   <actuator name="${name}_motor">
      <hardwareInterface>EffortJointInterface</hardwareInterface>
      <mechanicalReduction>1</mechanicalReduction>
   </actuator>
</transmission>
```

joint: the joint the transmission is connected to

actuator: actuator the transmission is connected to

mechanicalReduction: a value for overall mechanical reduction at the joint/actuator transmission

Gazebo

http://wiki.ros.org/urdf/XML/Gazebo

The gazebo element is an extension to the URDF robot description format, used for simulation purposes in the Gazebo simulator.

```
<gazebo>
   <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
      <robotNamespace>/safe_arm</robotNamespace>
      <robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSimType>
   </plugin>
</gazebo>
```

The default behavior provides the following ros_control interfaces:

- hardware_interface::JointStateInterface
- hardware_interface::EffortJointInterface

Xacro http://wiki.ros.org/xacro

URDF 등의 XML을 작성할때 유용하게 사용할수 있는 패키지. Macro, Property, Math, Conditional 등의 기능을 지원함

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://ros.org/wiki/xacro" name="safe_arm">
    <xacro:include filename="$(find safe_arm_description)/urdf/visuals.xacro" />
    <xacro:include filename="$(find safe_arm_description)/urdf/safe_arm_v1_limits.xacro" />
    <xacro:include filename="$(find safe_arm_description)/urdf/safe_arm_v1_structure.urdf.xacro" />
    <xacro:include filename="$(find safe_arm_description)/urdf/safe_arm_v1.transmission.xacro" />
    <xacro:include filename="$(find safe_arm_description)/urdf/safe_arm_v1.gazebo.xacro" />

    <link name="world"/>
    <joint name="fixed" type="fixed">
        <parent link="world"/>
        <child link="base_link"/>
    </joint>

    <link name="base_link" />
    <link name="torso" />
    <joint name="base_joint" type="fixed">
        <origin xyz="0 0 1.0" rpy="0 0 0" />
        <parent link="base_link" />
        <child link="torso" />
    </joint>

    <!-- Attatch Left and Right Safe Arm to Torso Link -->
    <xacro:safe_arm side="l" reflect="1" parent="torso" />
    <xacro:safe_arm side="r" reflect="-1" parent="torso" />

</robot>
```

http://wiki.ros.org/urdf/Tutorials/Using%20Xacro%20to%20Clean%20Up%20a%20UR

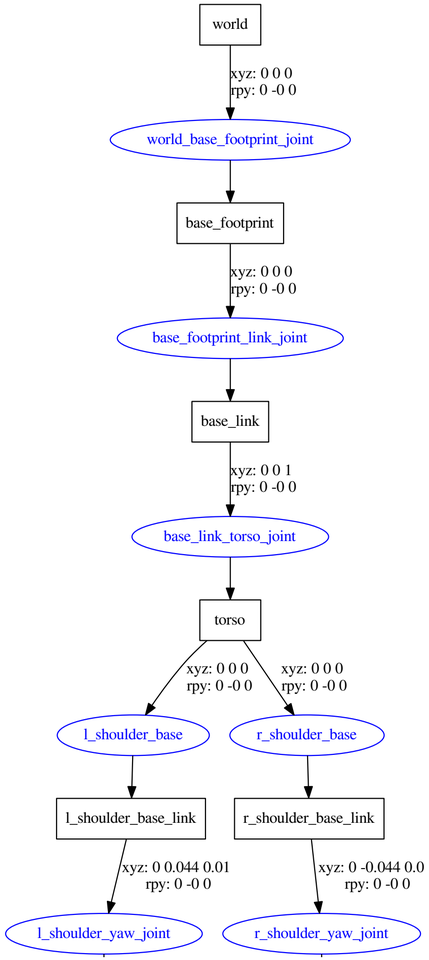
작성된 URDF에 대한 검증 작업

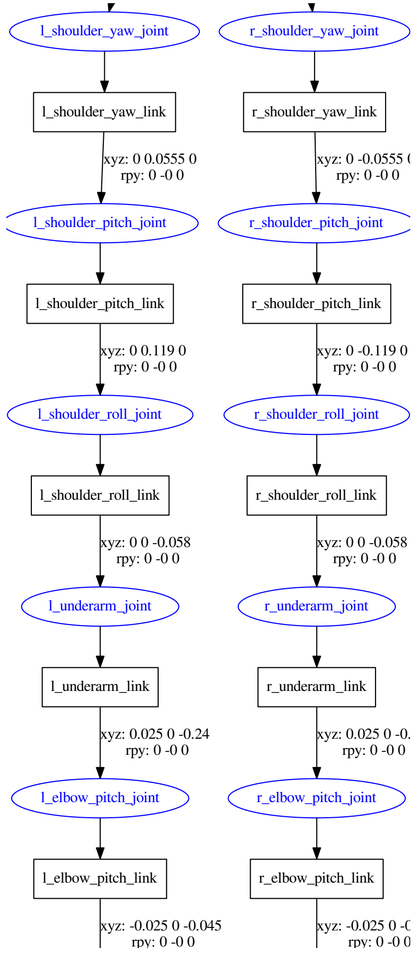
http://wiki.ros.org/urdf#Verification

```
$ check_urdf my_robot.urdf
```

```
robot name is: safe_arm
---------- Successfully Parsed XML ---------------
root Link: world has 1 child(ren)
    child(1):  base_footprint
        child(1):  base_link
            child(1):  torso
                child(1):  l_shoulder_base_link
                    child(1):  l_shoulder_yaw_link
                        child(1):  l_shoulder_pitch_link
                            child(1):  l_shoulder_roll_link
                                child(1):  l_underarm_link
                                    child(1):  l_elbow_pitch_link
                                        child(1):  l_elbow_yaw_link
                                            child(1):  l_wrist_pitch_link
                                                child(1):  l_wrist_roll_link
                                                    child(1):  l_end_effector_link
                child(2):  r_shoulder_base_link
                    child(1):  r_shoulder_yaw_link
                        child(1):  r_shoulder_pitch_link
                            child(1):  r_shoulder_roll_link
                                child(1):  r_underarm_link
                                    child(1):  r_elbow_pitch_link
                                        child(1):  r_elbow_yaw_link
                                            child(1):  r_wrist_pitch_link
                                                child(1):  r_wrist_roll_link
                                                    child(1):  r_end_effector_link
```

```
$ urdf_to_graphiz my_robot.urdf
```





작성된 URDF에 대한 검증 작업

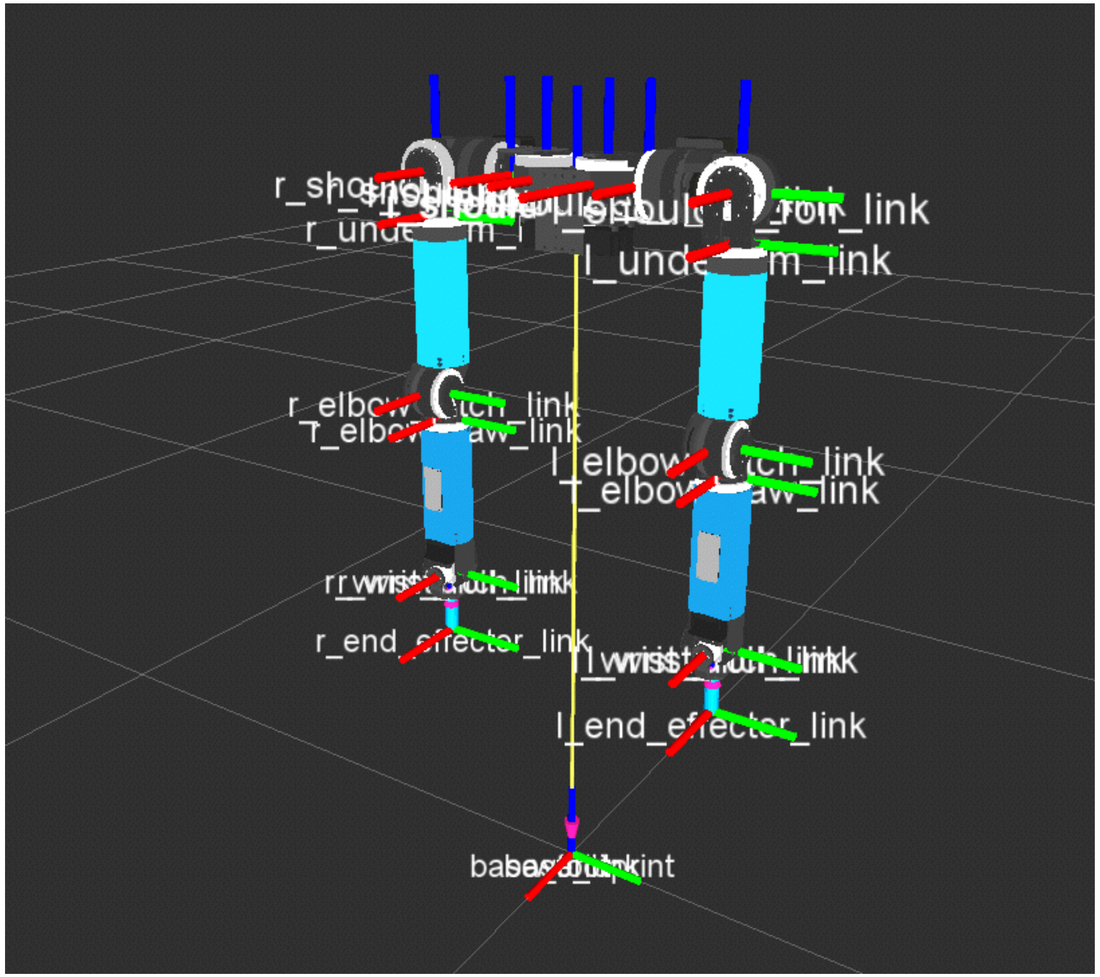
로봇의 동작 범위, 모델(Mesh) 등등 검증

```
$ roslaunch safe_arm_description safe_arm_publisher.launch
$ rosrun rqt_rviz rqt_rviz
```

```
<launch>
   <param name="robot_description" command="$(find xacro)/xacro.py $(find safe_arm_description)/urdf/
safe_arm_v1.urdf.xacro" />

   <node pkg="robot_state_publisher" type="state_publisher" name="robot_state_publisher" />
   <node pkg="joint_state_publisher" type="joint_state_publisher" name="joint_state_publisher">
      <param name="use_gui" value="1" />
   </node>

</launch>
```

robot_state_publisher

http://wiki.ros.org/robot_state_publisher

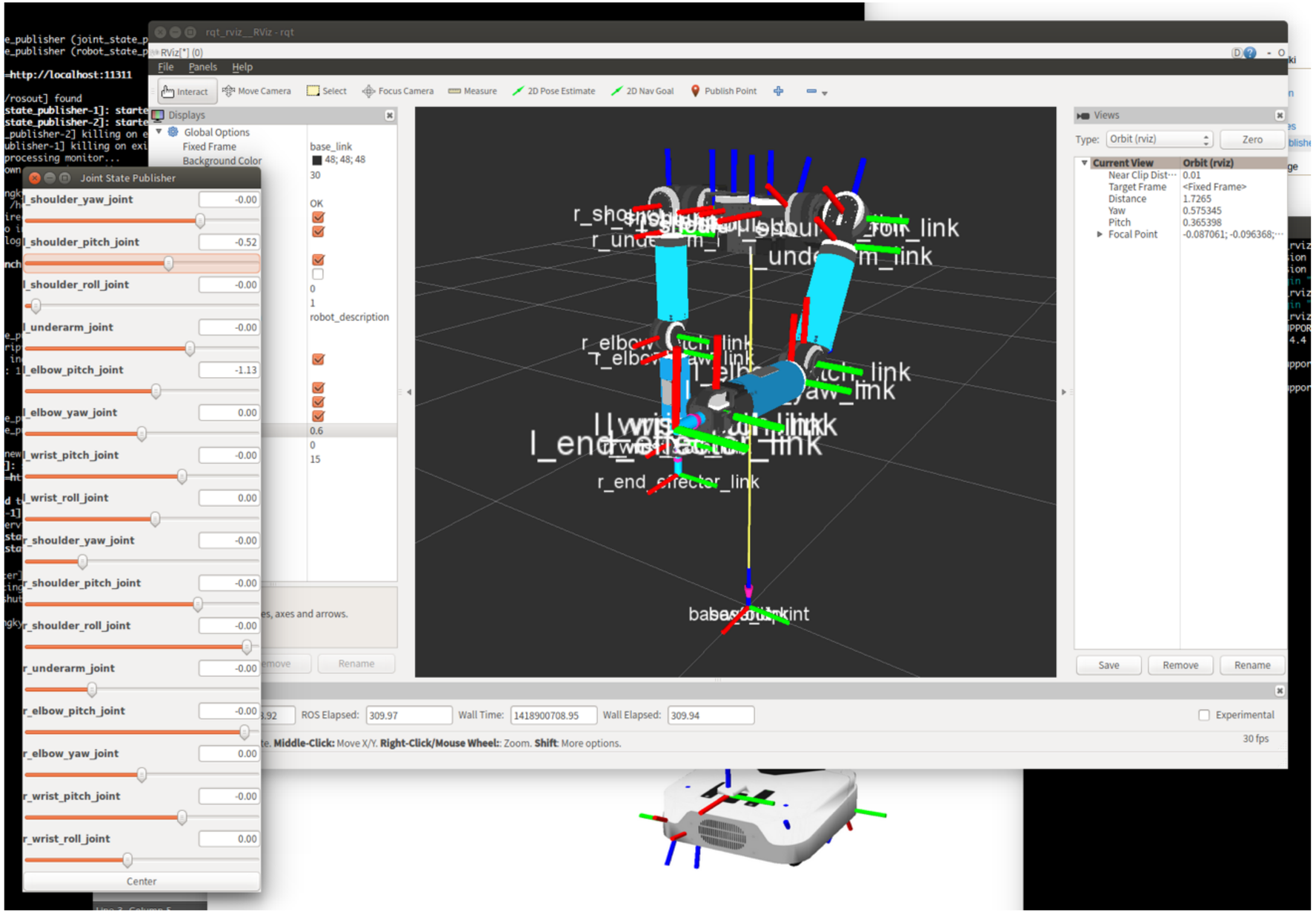


/joint_states 토픽을 구독하여, /tf로 출력

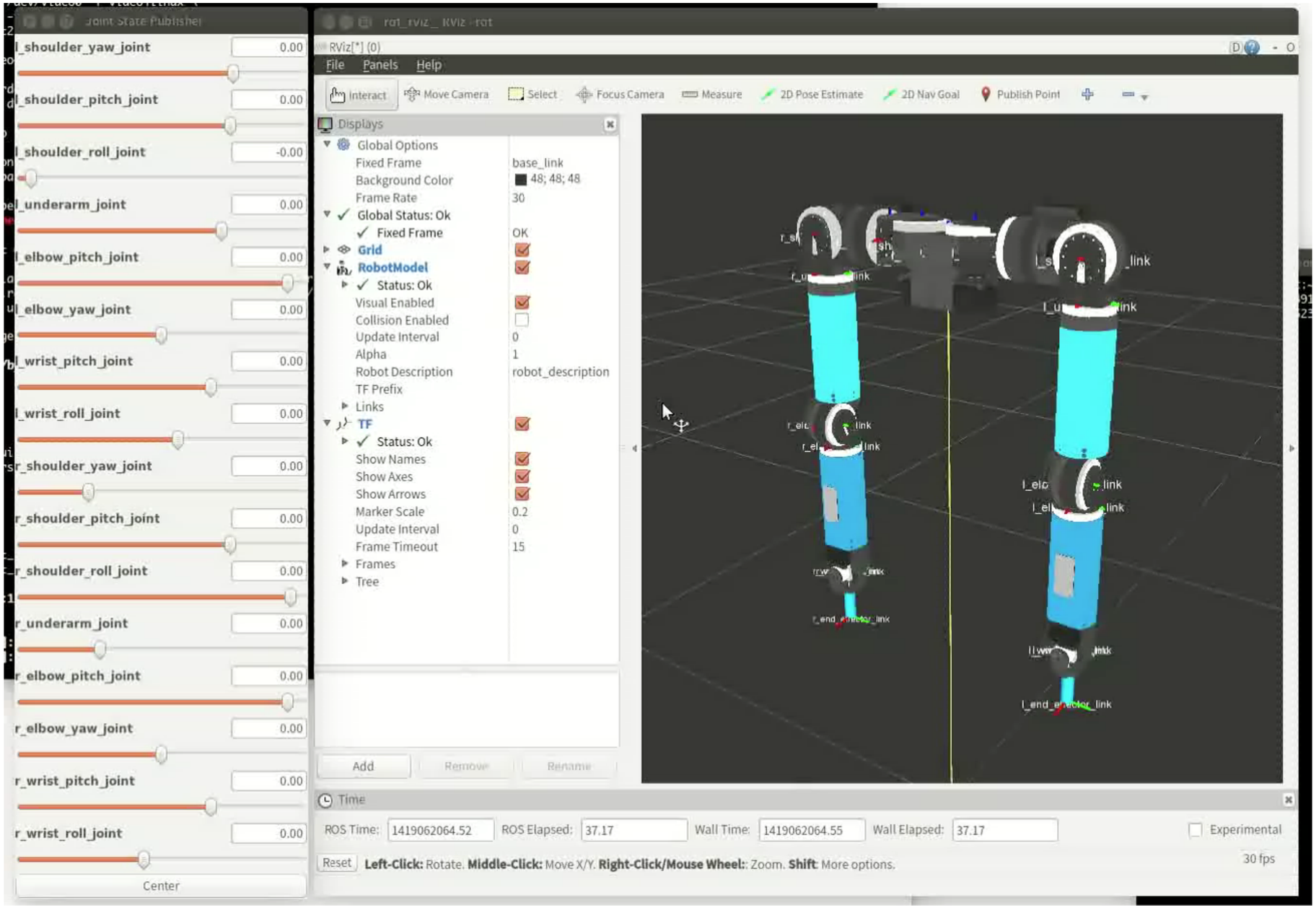
robot_description 파라미터에 명시된 로봇의 정보를 이용하여 Joint의 각도값, 링크의 3D 포즈, 로봇의 기구학적 트리 등에 대한 정보를 /tf로 Publish한다.

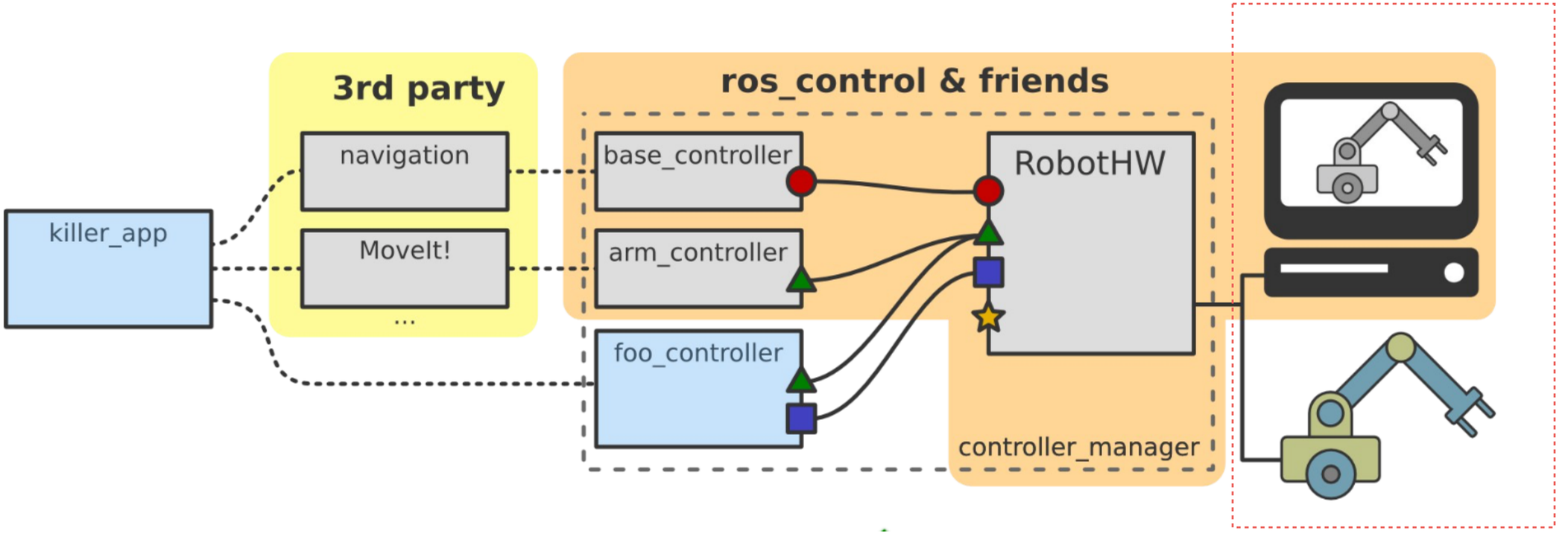
joint_state_publisher

http://wiki.ros.org/joint_state_publisher



robot_description 파라미터에 명시된 로봇의 Joint 정보를 이용하여 Joint의 각도값을 Publish한다.


Joint State Publisher
l_shoulder_yaw_joint 0.00
l_shoulder_pitch_joint 0.00
l_shoulder_roll_joint -0.00
l_underarm_joint 0.00
l_elbow_pitch_joint 0.00
l_elbow_yaw_joint 0.00
l_wrist_pitch_joint 0.00
l_wrist_roll_joint 0.00
r_shoulder_yaw_joint 0.00
r_shoulder_pitch_joint 0.00
r_shoulder_roll_joint 0.00
r_underarm_joint 0.00
r_elbow_pitch_joint 0.00
r_elbow_yaw_joint 0.00
r_wrist_pitch_joint 0.00
r_wrist_roll_joint 0.00
Center
RViz[*] (0)
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Displays
Global Options
Fixed Frame base_link
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid
RobotModel
Status: Ok
Visual Enabled
Collision Enabled
Update Interval 0
Alpha 1
Robot Description robot_description
TF Prefix
Links
TF
Status: Ok
Show Names
Show Axes
Show Arrows
Marker Scale 0.2
Update Interval 0
Frame Timeout 15
Frames
Tree
Add Remove Rename
Time
ROS Time: 1419062064.52 ROS Elapsed: 37.17 Wall Time: 1419062064.55 Wall Elapsed: 37.17
Experimental
Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
30 fps



Adolfo Rodríguez Tsouroukdissian, “ros_control: An overview”, ROSCon 2014

Gazebo를 이용한 시뮬레이션

Gazebo

3D 로봇 시뮬레이터로 다양한 종류의 물리엔진을 지원하여 동력한 시뮬레이션을 지원, 추가로 일반적으로 사용되는 센서(LRF, 카메라, Depth Sensor, 접촉센서, 힘-토크센서 등)을 지원함.

Dynamics Simulation

Access multiple high-performance physics engines including ODE, Bullet, Simbody, and DART.

Advanced 3D Graphics

Utilizing OGRE, Gazebo provides realistic rendering of environments including high-quality lighting, shadows, and textures.

Sensors and Noise

Generate sensor data, optionally with noise, from laser range finders, 2D/3D cameras, Kinect style sensors, contact sensors, force-torque, and more.

Plugins

Develop custom plugins for robot, sensor, and environmental control. Plugins provide direct access to Gazebo's API.

Robot Models

Many robots are provided including PR2, Pioneer2 DX, iRobot Create, and TurtleBot. Or build your own using SDF.

TCP/IP Transport

Run simulation on remote servers, and interface to Gazebo through socket-based message passing using Google Protobufs.

Cloud Simulation

Use CloudSim to run Gazebo on Amazon, Softlayer, or your own OpenStack instance.

Command Line Tools

Extensive command line tools facilitate simulation introspection and control.

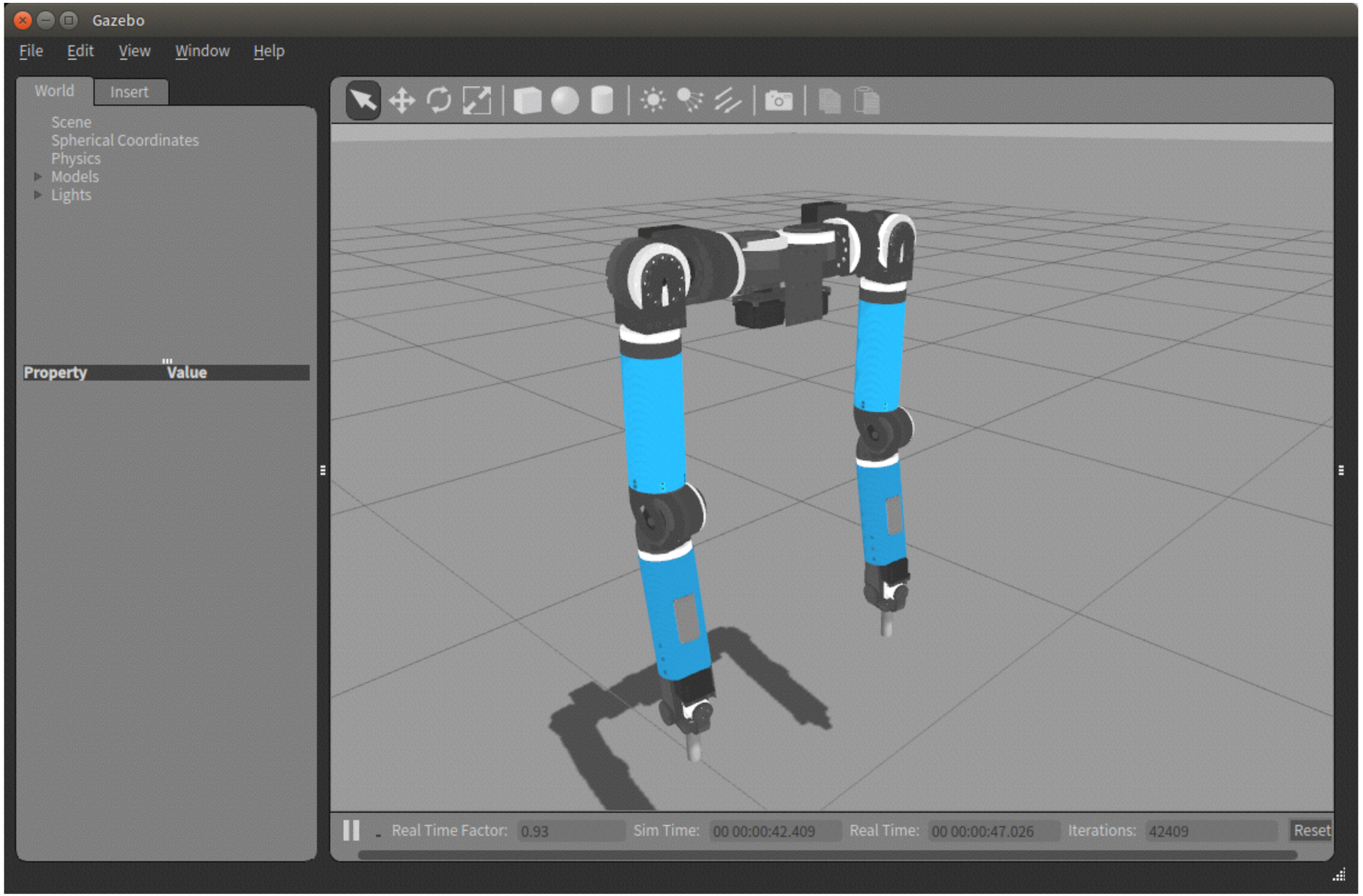
safe_arm gazebo 실행

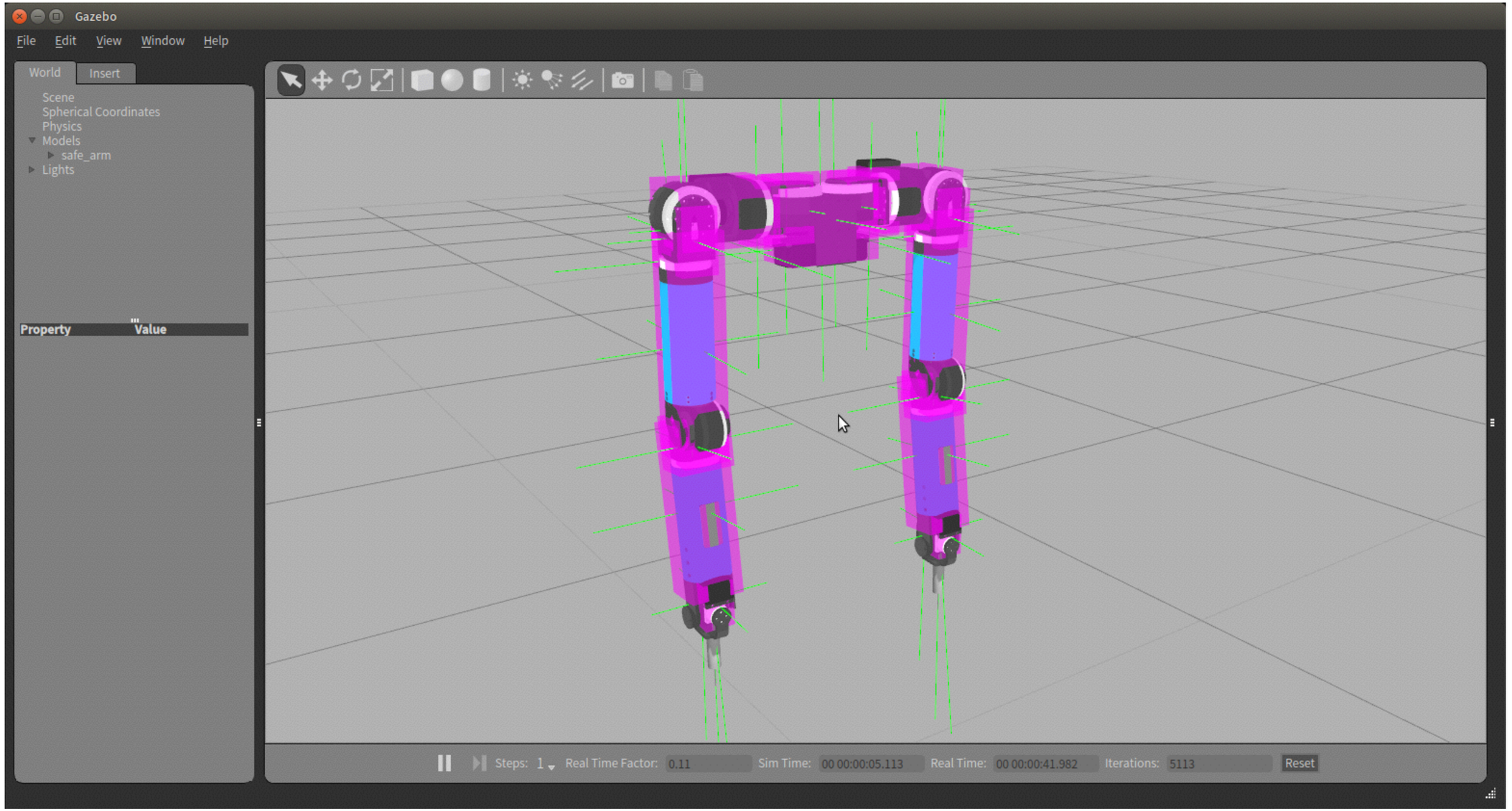
```
$ roslaunch safe_arm_bringup sim.launch
```

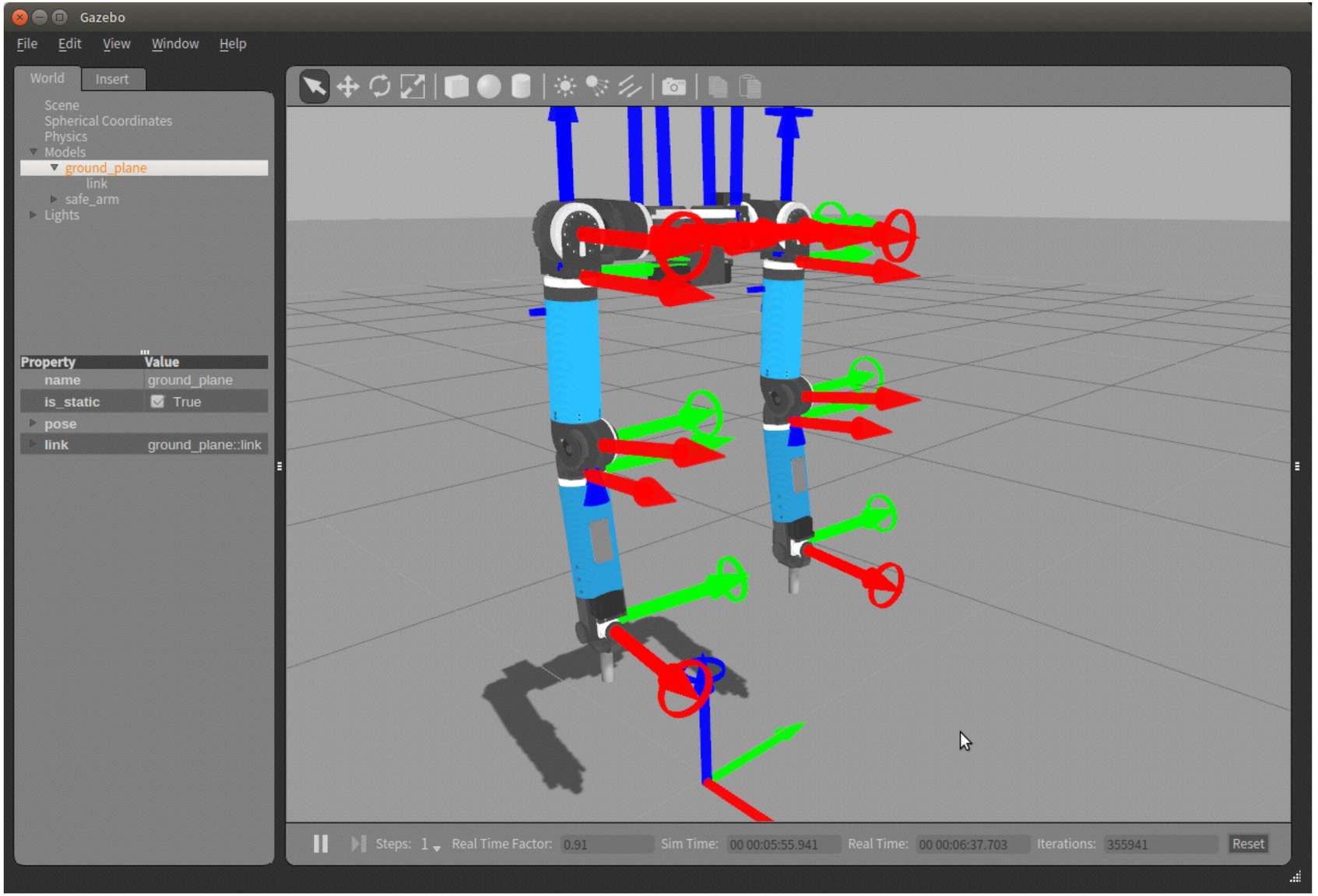
```
<launch>
    <arg name="paused" default="false"/>
    <arg name="use_sim_time" default="true"/>
    <arg name="gui" default="true"/>
    <arg name="headless" default="false"/>
    <arg name="debug" default="false"/>


    <include file="$(find safe_arm_bringup)/launch/empty_world.launch">
        <arg name="debug" value="$(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>

    <param name="robot_description" command="$(find xacro)/xacro.py $(find safe_arm_description)/urdf/safe_arm_v1.urdf.xacro" />

    <node name="urdf_spawner" pkg="gazebo_ros" type="spawn_model" respawn="false" output="screen"
            args="-urdf -model safe_arm -param robot_description"/>
</launch>
```

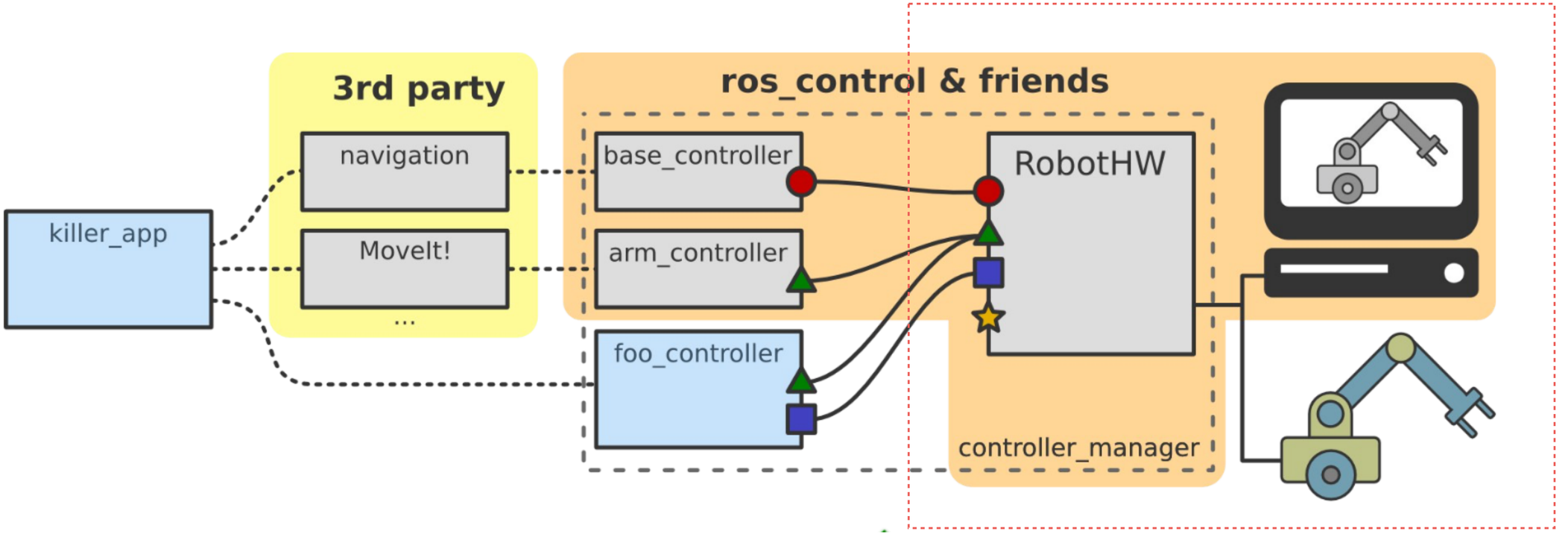
empty.world 기존 empty_world에서 물리엔진 선택 부분 추가

```
<launch>
  <!-- these are the arguments you can pass this launch file, for example paused:=true -->
  <arg name="paused" default="false"/>
  <arg name="use_sim_time" default="true"/>
  <arg name="gui" default="true"/>
  <arg name="headless" default="true"/>
  <arg name="debug" default="false"/>
  <arg name="verbose" default="false"/>
  <arg name="world_name" default="worlds/empty.world"/> <!-- Note: the world_name is with respect to GAZEBO_RESOURCE_PATH environmental variable -->

  <!-- set use_sim_time flag -->
  <group if="$(arg use_sim_time)">
    <param name="/use_sim_time" value="true" />
  </group>

  <!-- set command arguments -->
  <arg unless="$(arg paused)" name="command_arg1" value=""/>
  <arg     if="$(arg paused)" name="command_arg1" value="-u"/>
  <arg unless="$(arg headless)" name="command_arg2" value=""/>
  <arg     if="$(arg headless)" name="command_arg2" value="-r"/>
  <arg unless="$(arg verbose)" name="command_arg3" value=""/>
  <arg     if="$(arg verbose)" name="command_arg3" value="--verbose"/>
  <arg unless="$(arg debug)" name="script_type" value="gzserver"/>
  <arg     if="$(arg debug)" name="script_type" value="debug"/>

  <!-- start gazebo server-->
  <node name="gazebo" pkg="gazebo_ros" type="$(arg script_type)" respawn="false" output="screen"
           args="$(arg command_arg1) $(arg command_arg2) $(arg command_arg3) $(arg world_name) -e simbody" />

  <!-- start gazebo client -->
  <group if="$(arg gui)">
    <node name="gazebo_gui" pkg="gazebo_ros" type="gzclient" respawn="false" output="screen"/>
  </group>

</launch>
```

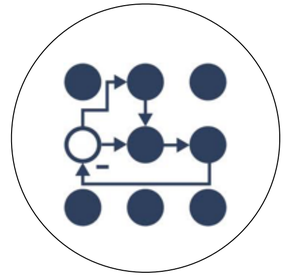

Gazebo
File
Edit
View
Window
Help
World
Insert
Scene
Spherical Coordinates
Physics
Models
Lights
Property
Value
Real Time Factor: 0.93
Sim Time: 00 00:00:42.409
Real Time: 00 00:00:47.026
Iterations: 42409
Reset


Gazebo
File
Edit
View
Window
Help
World
Insert
Scene
Spherical Coordinates
Physics
Models
safe_arm
Lights
Property
Value
Steps: 1
Real Time Factor: 0.11
Sim Time: 00 00:00:05.113
Real Time: 00 00:00:41.982
Iterations: 5113
Reset


Gazebo
File
Edit
View
Window
Help
World
Insert
Scene
Spherical Coordinates
Physics
Models
ground_plane
link
safe_arm
Lights
Property
Value
name
ground_plane
is_static
True
pose
link
ground_plane::link
Steps:
1
Real Time Factor:
0.91
Sim Time:
00 00:05:55.941
Real Time:
00 00:06:37.703
Iterations:
355941
Reset

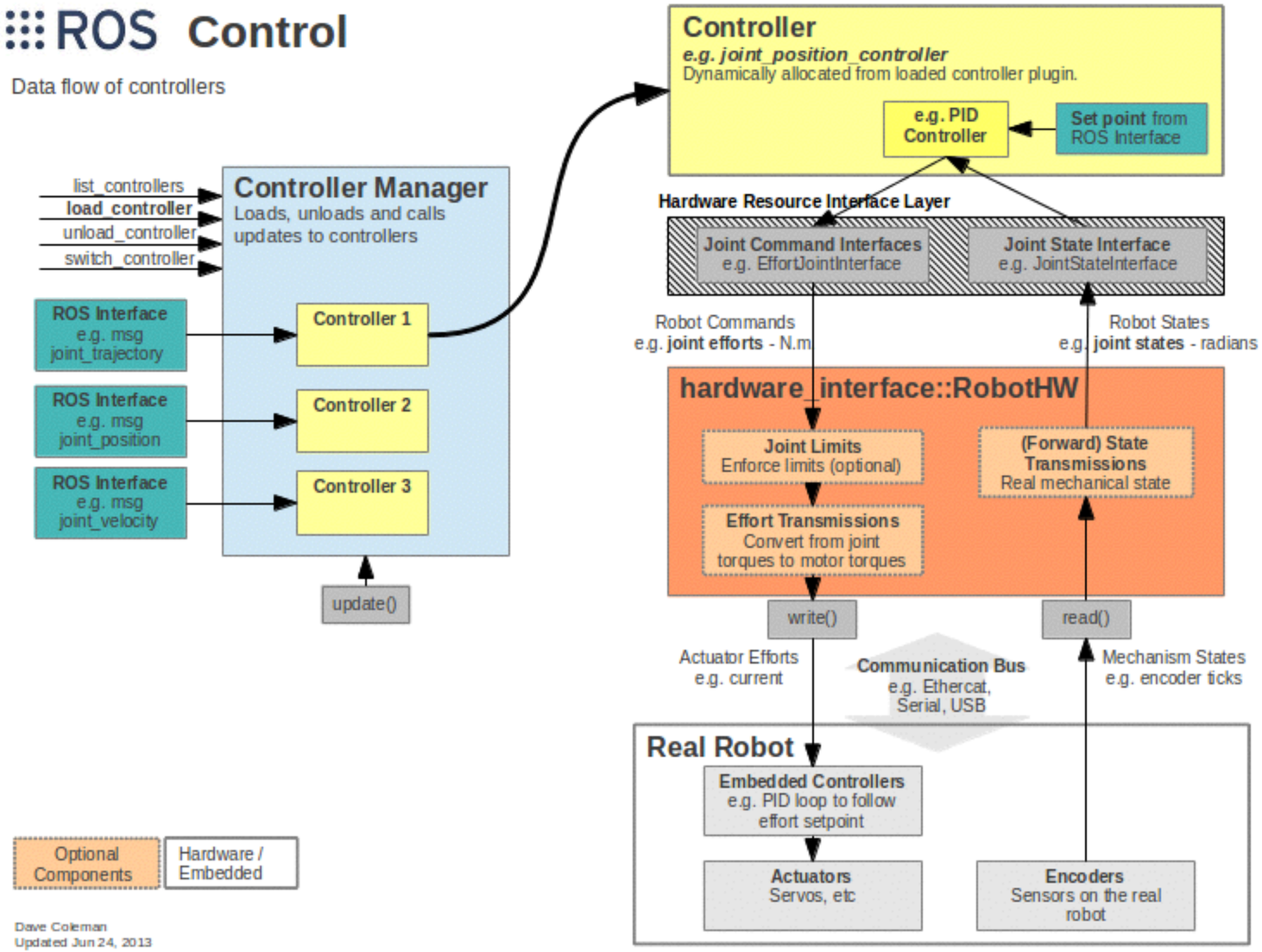


Adolfo Rodríguez Tsouroukdissian, “ros_control: An overview”, ROSCon 2014



Gazebo + ros_control

ros_control

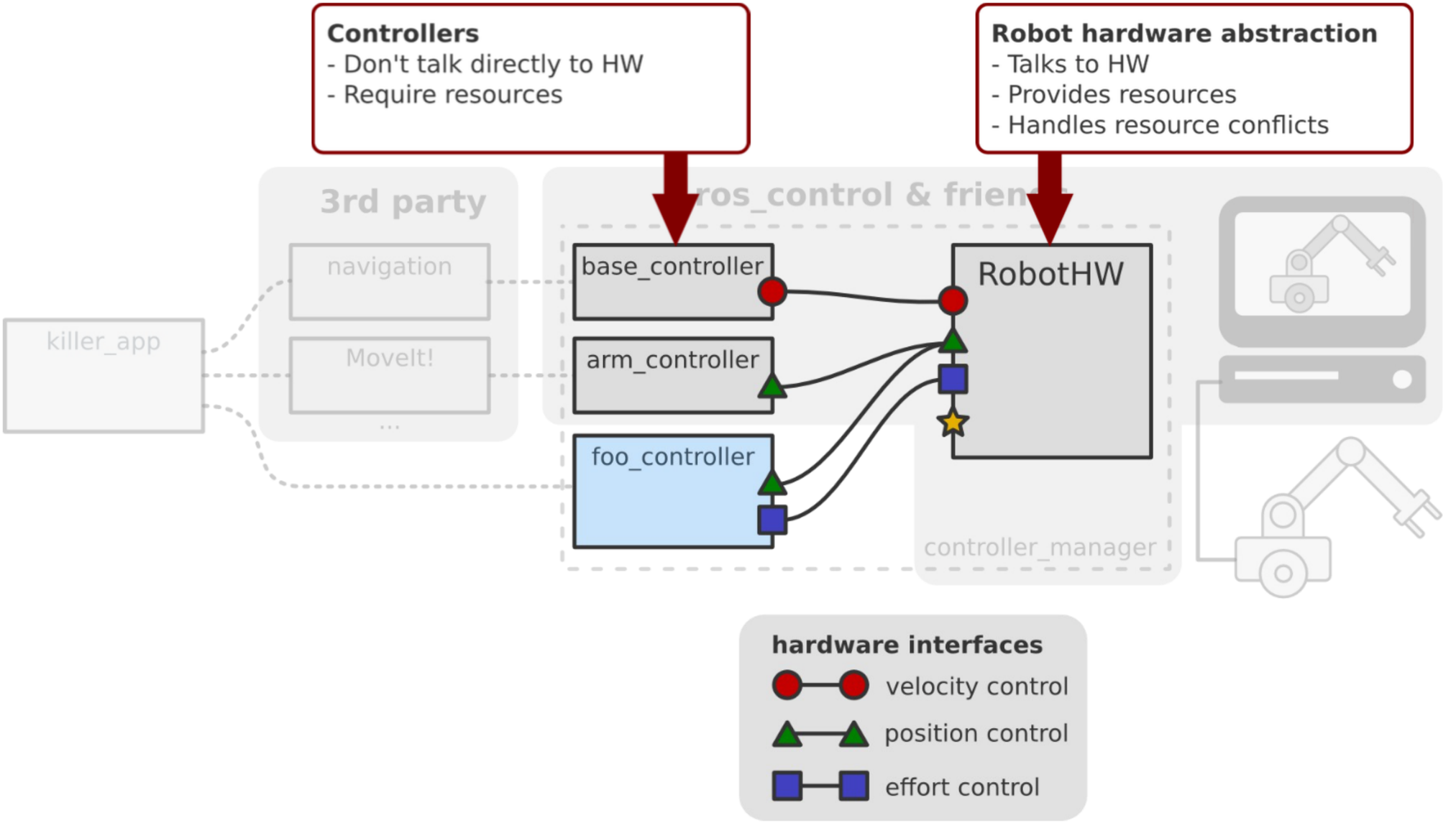

ROS Control
Data flow of controllers
Controller
e.g. joint_position_controller
Dynamically allocated from loaded controller plugin.
e.g. PID Controller
Set point from ROS Interface
Hardware Resource Interface Layer
Joint Command Interfaces
e.g. EffortJointInterface
Joint State Interface
e.g. JointStateInterface
Robot Commands
e.g. joint efforts - N.m
Robot States
e.g. joint states - radians
hardware_interface::RobotHW
Joint Limits
Enforce limits (optional)
Effort Transmissions
Convert from joint torques to motor torques
(Forward) State Transmissions
Real mechanical state
write()
read()
Actuator Efforts
e.g. current
Communication Bus
e.g. Ethercat, Serial, USB
Mechanism States
e.g. encoder ticks
Real Robot
Embedded Controllers
e.g. PID loop to follow effort setpoint
Actuators
Servos, etc
Encoders
Sensors on the real robot
list_controllers
load_controller
unload_controller
switch_controller
Controller Manager
Loads, unloads and calls updates to controllers
ROS Interface
e.g. msg
joint_trajectory
ROS Interface
e.g. msg
joint_position
ROS Interface
e.g. msg
joint_velocity
Controller 1
Controller 2
Controller 3
update()
Optional Components
Hardware / Embedded
Dave Coleman
Updated Jun 24, 2013

Goal

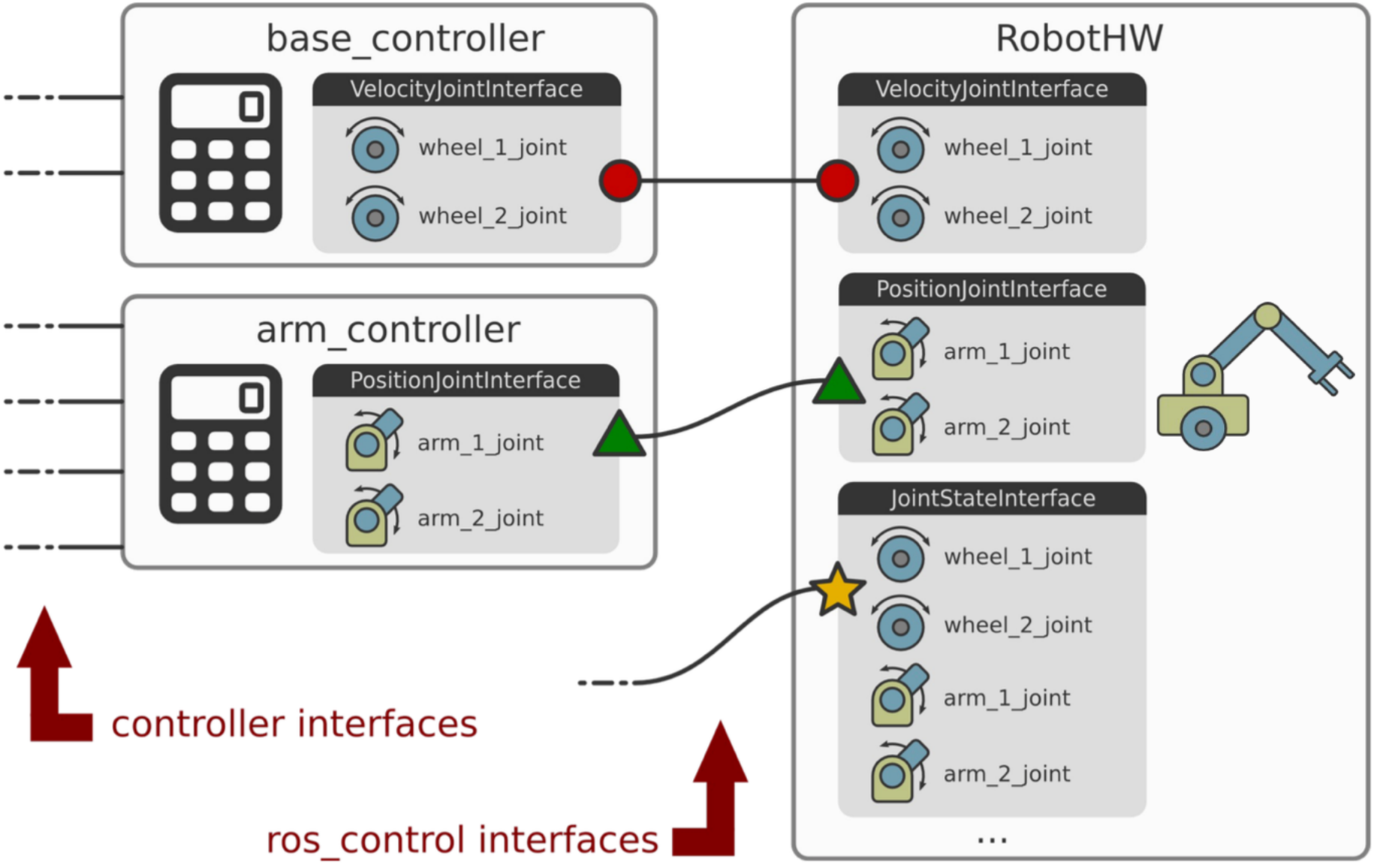
- Lower **entry barrier** for exposing HW to ROS
- Promote **reuse** of control code
- Provide **ready-to-use** tools
- **Real-time ready** implementation

History

- **pr2_controller_manager** (2009)
  - developed mainly by **Willow Garage** (WG)
  - **PR2**-specific
- **ros_control** (late 2012)
  - started by **hiDOF**, in collaboration with **WG**
  - continued by **PAL Robotics** and **community**
  - **robot-agnostic** version of the pr2_controller_manager



Adolfo Rodríguez Tsouroukdissian, “ros_control: An overview”, ROSCon 2014



Adolfo Rodríguez Tsouroukdissian, “ros_control: An overview”, ROSCon 2014

Controllers

- effort_controllers
  - joint_effort_controller
  - joint_position_controller
  - joint_velocity_controller
- joint_state_controller
  - joint_state_controller
- position_controllers
  - joint_position_controller
- velocity_controllers
  - joint_velocity_controllers

Hardware Interfaces

- Joint Command Interfaces
  - Effort Joint Interface
  - Velocity Joint Interface
  - Position Joint Interface
- Joint State Interfaces
- Actuator State Interfaces
- Actuator Command Interfaces
  - Effort Actuator Interface
  - Velocity Actuator Interface
  - Position Actuator Interface
- Force-torque sensor Interface
- IMU sensor Interface

!! You can of course create your own

safe_arm_v1.transmission.xacro

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

   <xacro:macro name="insert_transmission" params="name">
      <transmission name="${name}_tran">
         <type>transmission_interface/SimpleTransmission</type>
         <joint name="${name}">
            <hardwareInterface>EffortJointInterface</hardwareInterface>
         </joint>

         <actuator name="${name}_motor">
            <hardwareInterface>EffortJointInterface</hardwareInterface>
            <mechanicalReduction>1</mechanicalReduction>
         </actuator>
      </transmission>
   </xacro:macro>

</robot>
```

<xacro:insert_transmission name="${side}_shoulder_yaw_joint" />

safe_arm_joint_state.yaml

```
safe_arm:
  joint_state_controller:
    type: joint_state_controller/JointStateController
    publish_rate: 100
```

sensor_msgs/JointState Message

std_msgs/Header header
string[] name
float64[] position
float64[] velocity
float64[] effort

safe_arm_joint_state_controller.launch

```
<launch>
   <rosparam file="$(find safe_arm_control)/config/safe_arm_joint_state.yaml" command="load"/>

   <node name="joint_controller_spawner" pkg="controller_manager" type="spawner" respawn="false"
      output="screen" ns="/safe_arm" args="joint_state_controller" />

   <node name="robot_state_publisher" pkg="robot_state_publisher" type="robot_state_publisher"
      respawn="false" output="screen">
      <remap from="/joint_states" to="/safe_arm/joint_states" />
   </node>
</launch>
```

safe_arm_l_control.yaml

```
safe_arm:
  l_arm_joint_controller:
    type: "effort_controllers/JointTrajectoryController"
    joints:
      - l_shoulder_yaw_joint
      - l_shoulder_pitch_joint
      - l_shoulder_roll_joint
      - l_underarm_joint
      - l_elbow_pitch_joint
      - l_elbow_yaw_joint
      - l_wrist_pitch_joint
      - l_wrist_roll_joint

    gains:
      l_shoulder_yaw_joint:
        {p: 1500.0, i: 0.01, d: 0.1, i_clamp: 1000}
      l_shoulder_pitch_joint:
        {p: 1500.0, i: 0.01, d: 0.1, i_clamp: 1000}
      l_shoulder_roll_joint:
        {p: 1500.0, i: 0.01, d: 0.1, i_clamp: 1000}
      l_underarm_joint:
        {p: 1500.0, i: 0.01, d: 0.1, i_clamp: 1000}
      l_elbow_pitch_joint:
        {p: 1500.0, i: 0.01, d: 0.1, i_clamp: 1000}
      l_elbow_yaw_joint:
        {p: 1500.0, i: 0.01, d: 0.1, i_clamp: 1000}
      l_wrist_pitch_joint:
        {p: 1500.0, i: 0.01, d: 0.1, i_clamp: 1000}
      l_wrist_roll_joint:
        {p: 1500.0, i: 0.01, d: 0.1, i_clamp: 1000}
```

safe_arm_l_controller.launch

```
<launch>
    <rosparam file="$(find safe_arm_control)/config/safe_arm_l_control.yaml"
command="load"/>

    <node name="l_controller_spawner" pkg="controller_manager" type="spawner"
respawn="false"
        output="screen" ns="/safe_arm" args="l_arm_joint_controller"/>
</launch>
```

safe_arm_r_control.yaml

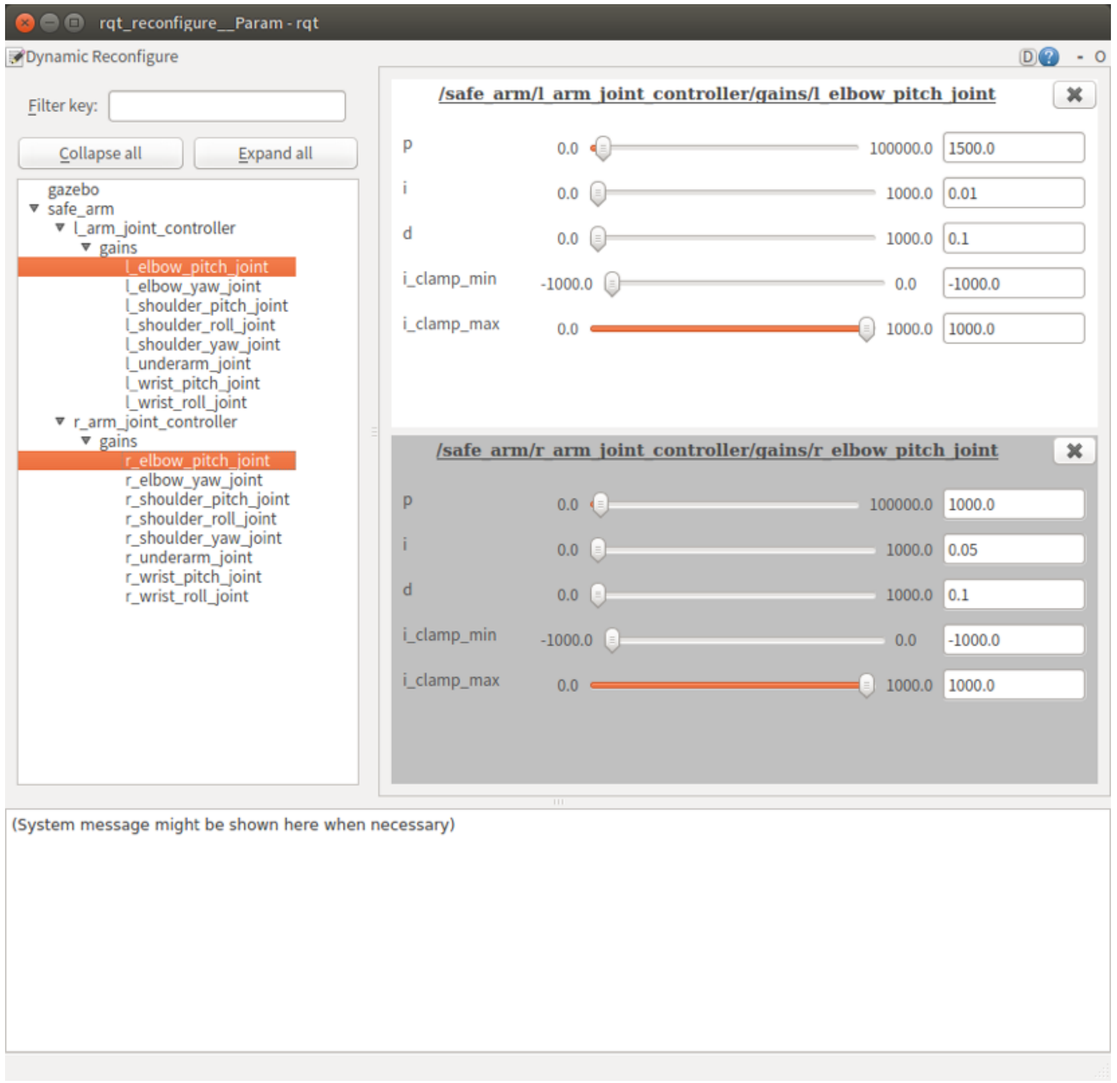
```
safe_arm:
  r_arm_joint_controller:
    type: "effort_controllers/JointTrajectoryController"
    joints:
      - r_shoulder_yaw_joint
      - r_shoulder_pitch_joint
      - r_shoulder_roll_joint
      - r_underarm_joint
      - r_elbow_pitch_joint
      - r_elbow_yaw_joint
      - r_wrist_pitch_joint
      - r_wrist_roll_joint

    gains:
      r_shoulder_yaw_joint:
        {p: 1000.0, i: 0.05, d: 0.1, i_clamp: 1000}
      r_shoulder_pitch_joint:
        {p: 1000.0, i: 0.05, d: 0.1, i_clamp: 1000}
      r_shoulder_roll_joint:
        {p: 1000.0, i: 0.05, d: 0.1, i_clamp: 1000}
      r_underarm_joint:
        {p: 1000.0, i: 0.05, d: 0.1, i_clamp: 1000}
      r_elbow_pitch_joint:
        {p: 1000.0, i: 0.05, d: 0.1, i_clamp: 1000}
      r_elbow_yaw_joint:
        {p: 1000.0, i: 0.05, d: 0.1, i_clamp: 1000}
      r_wrist_pitch_joint:
        {p: 1000.0, i: 0.05, d: 0.1, i_clamp: 1000}
      r_wrist_roll_joint:
        {p: 1000.0, i: 0.05, d: 0.1, i_clamp: 1000}
```

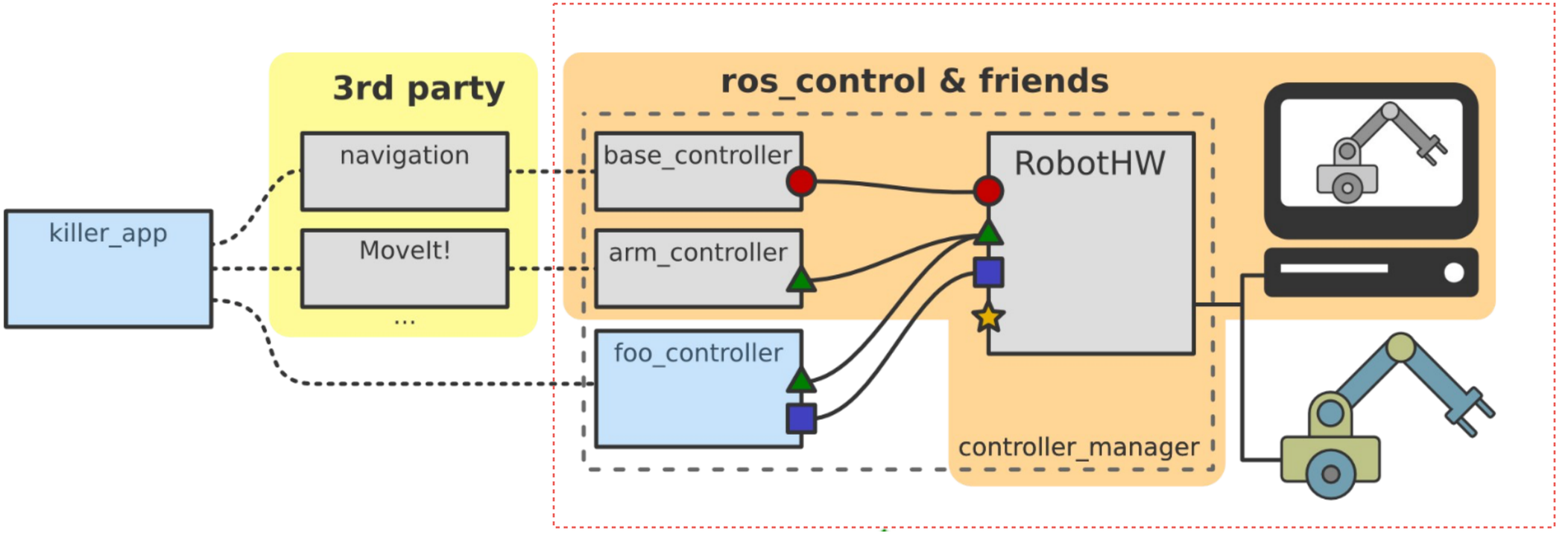
safe_arm_r_controller.launch

```
<launch>
    <rosparam file="$(find safe_arm_control)/config/safe_arm_r_control.yaml"
command="load"/>

  <node name="r_controller_spawner" pkg="controller_manager" type="spawner"
respawn="false"
    output="screen" ns="/safe_arm" args="r_arm_joint_controller"/>
</launch>
```

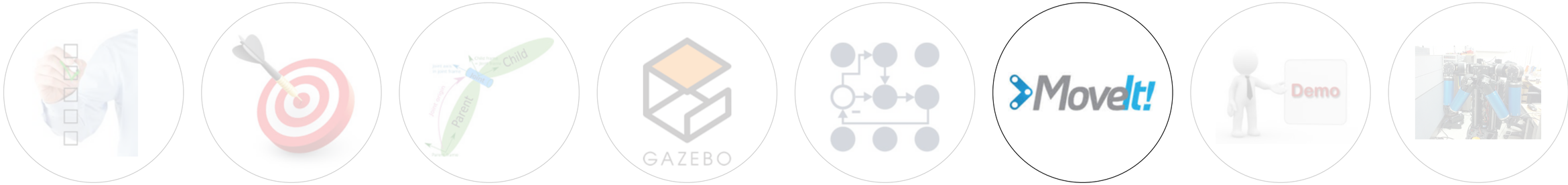
safe_arm_bringup/bringup.launch

```
<launch>
   <include file="$(find safe_arm_bringup)/launch/safe_arm_sim.launch" />

   <include file="$(find safe_arm_control)/launch/safe_arm_joint_state_controller.launch" />
   <include file="$(find safe_arm_control)/launch/safe_arm_l_controller.launch" />
   <include file="$(find safe_arm_control)/launch/safe_arm_r_controller.launch" />
</launch>
```


rqt_reconfigure__Param - rqt
Dynamic Reconfigure
Filter key:
Collapse all
Expand all
gazebo
safe_arm
l_arm_joint_controller
gains
l_elbow_pitch_joint
l_elbow_yaw_joint
l_shoulder_pitch_joint
l_shoulder_roll_joint
l_shoulder_yaw_joint
l_underarm_joint
l_wrist_pitch_joint
l_wrist_roll_joint
r_arm_joint_controller
gains
r_elbow_pitch_joint
r_elbow_yaw_joint
r_shoulder_pitch_joint
r_shoulder_roll_joint
r_shoulder_yaw_joint
r_underarm_joint
r_wrist_pitch_joint
r_wrist_roll_joint
/safe_arm/l_arm_joint_controller/gains/l_elbow_pitch_joint
p 0.0 100000.0 1500.0
i 0.0 1000.0 0.01
d 0.0 1000.0 0.1
i_clamp_min -1000.0 0.0 -1000.0
i_clamp_max 0.0 1000.0 1000.0
/safe_arm/r_arm_joint_controller/gains/r_elbow_pitch_joint
p 0.0 100000.0 1000.0
i 0.0 1000.0 0.05
d 0.0 1000.0 0.1
i_clamp_min -1000.0 0.0 -1000.0
i_clamp_max 0.0 1000.0 1000.0
(System message might be shown here when necessary)



Adolfo Rodríguez Tsouroukdissian, “ros_control: An overview”, ROSCon 2014



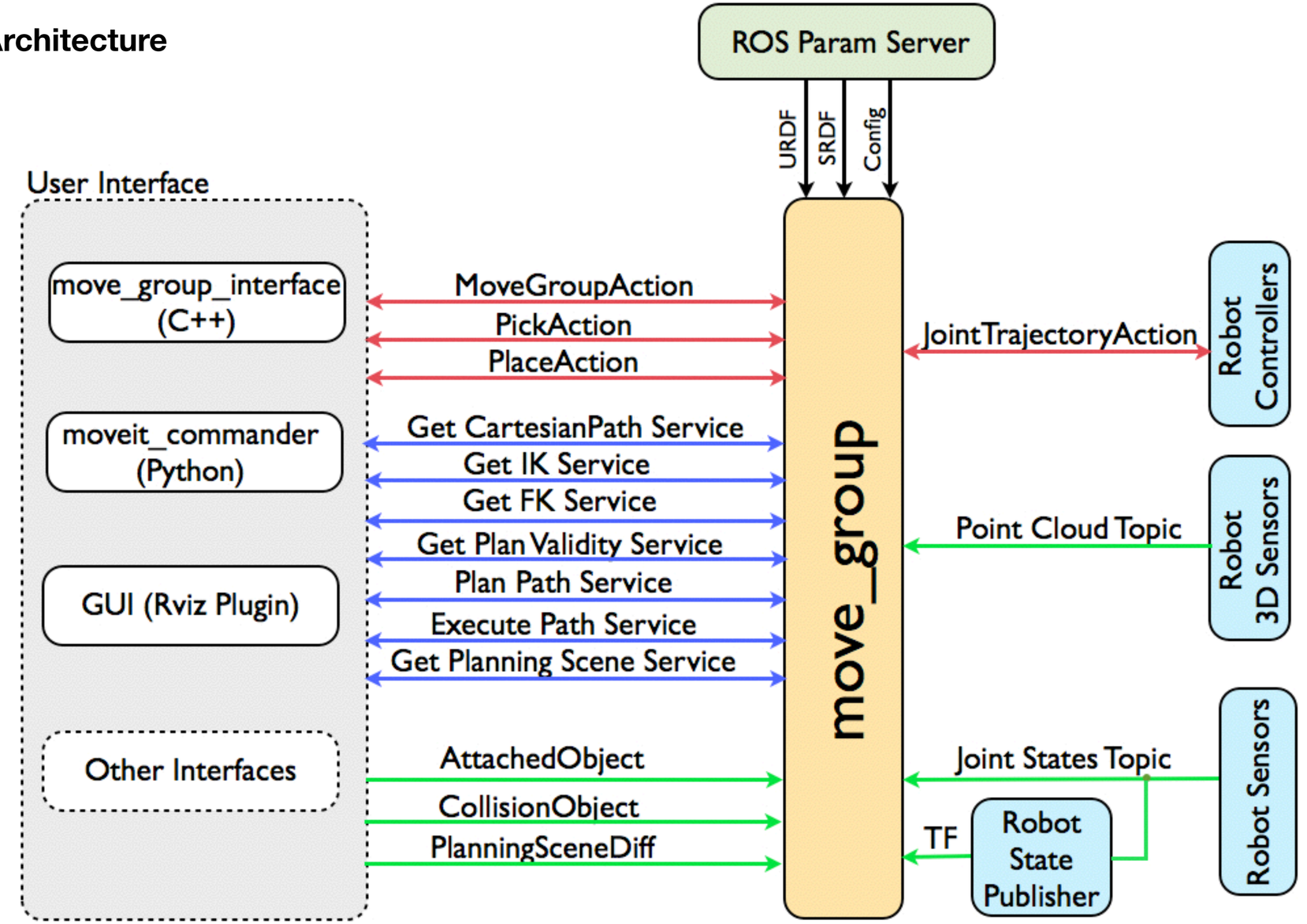
MoveIt 연동

MoveIt

MoveIt! is state of the art software for mobile manipulation, incorporating the latest advances in motion planning, manipulation, 3D perception, kinematics, control and navigation

http://www.youtube.com/watch?v=dblCGZzeUqs

System Architecture

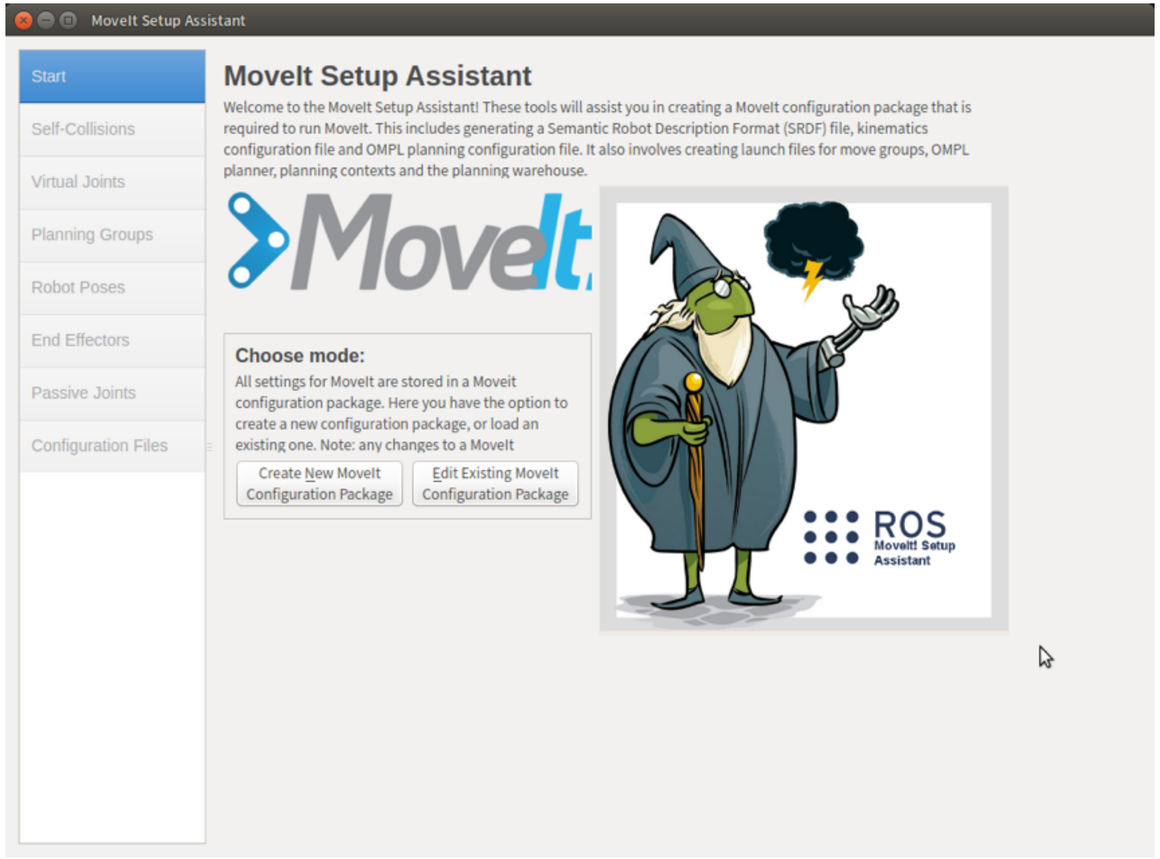


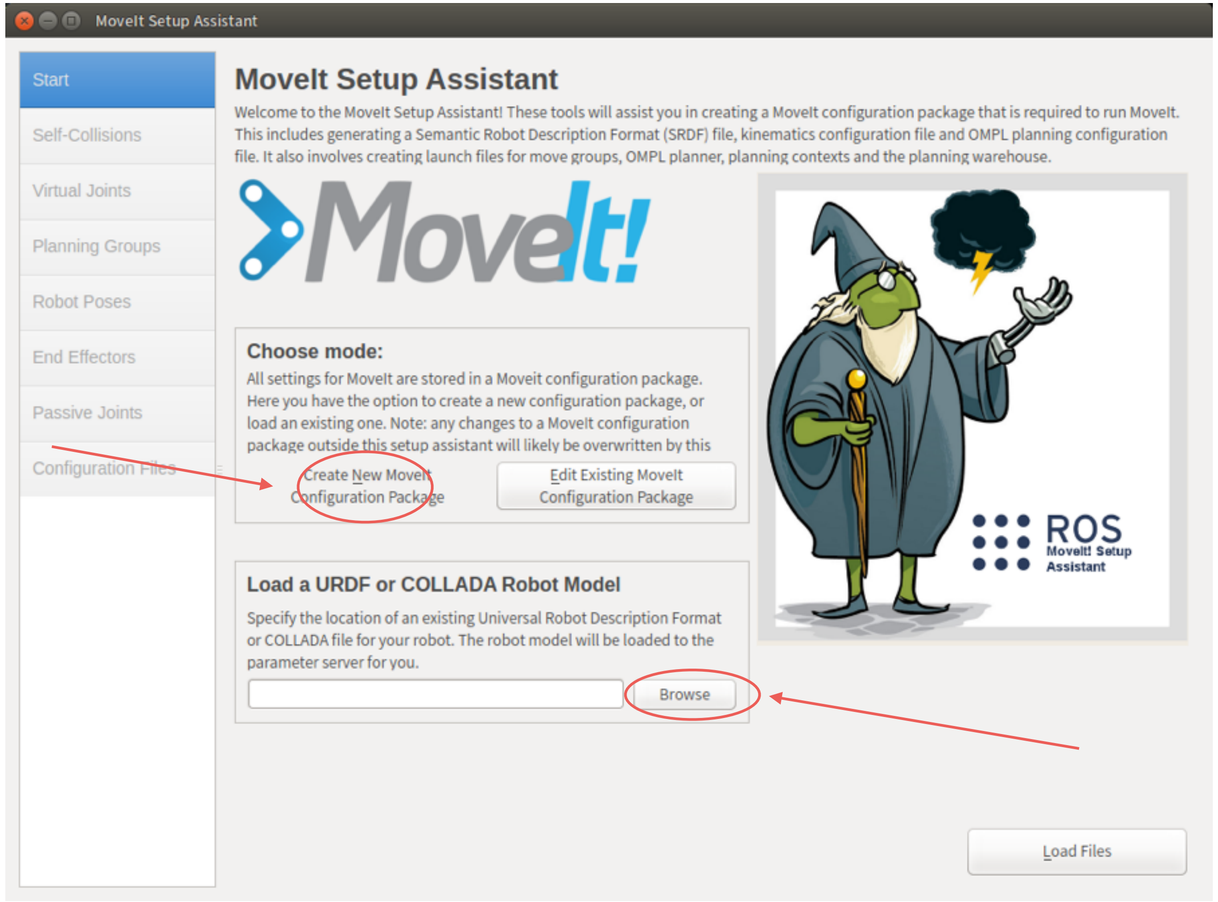
MoveIt Setup Assistant

http://docs.ros.org/hydro/api/moveit_setup_assistant/html/doc/tutorial.html

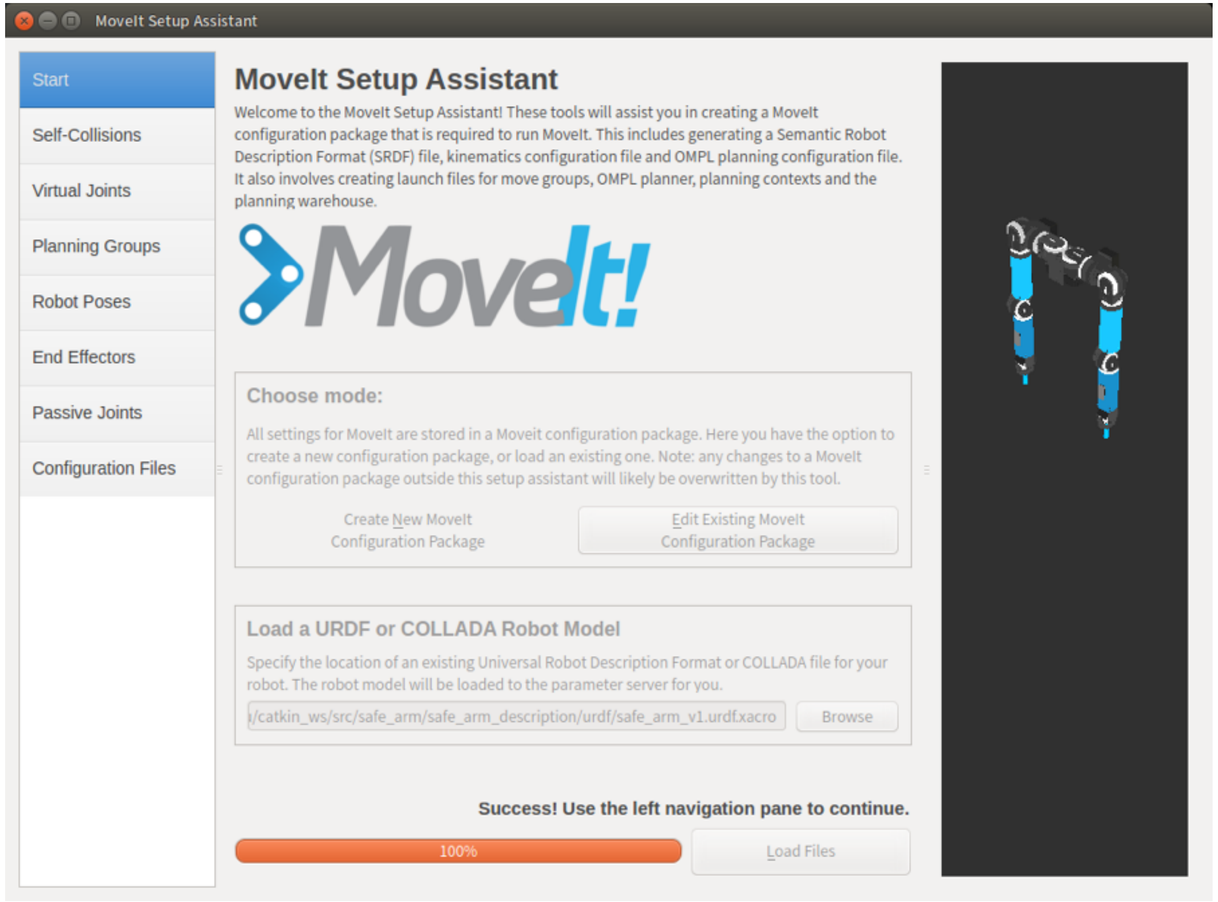
MoveIt와 로봇과의 연동을 위한 환경설정 GUI
결과물로 각종 MoveIt 패키지와 SRDF(Semantic Robot Description Format) 생성

```
$ roslaunch moveit_setup_assistant setup_assistant.launch
```

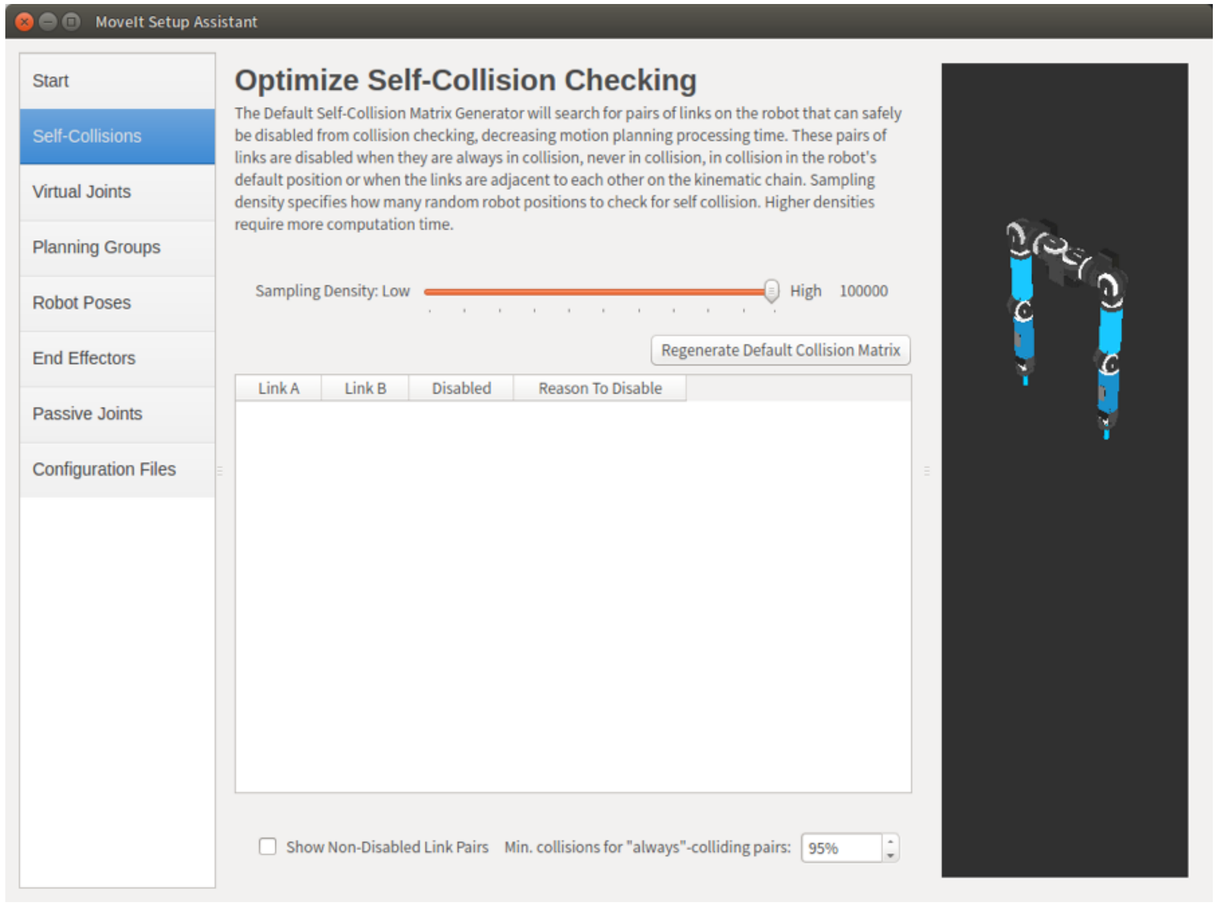




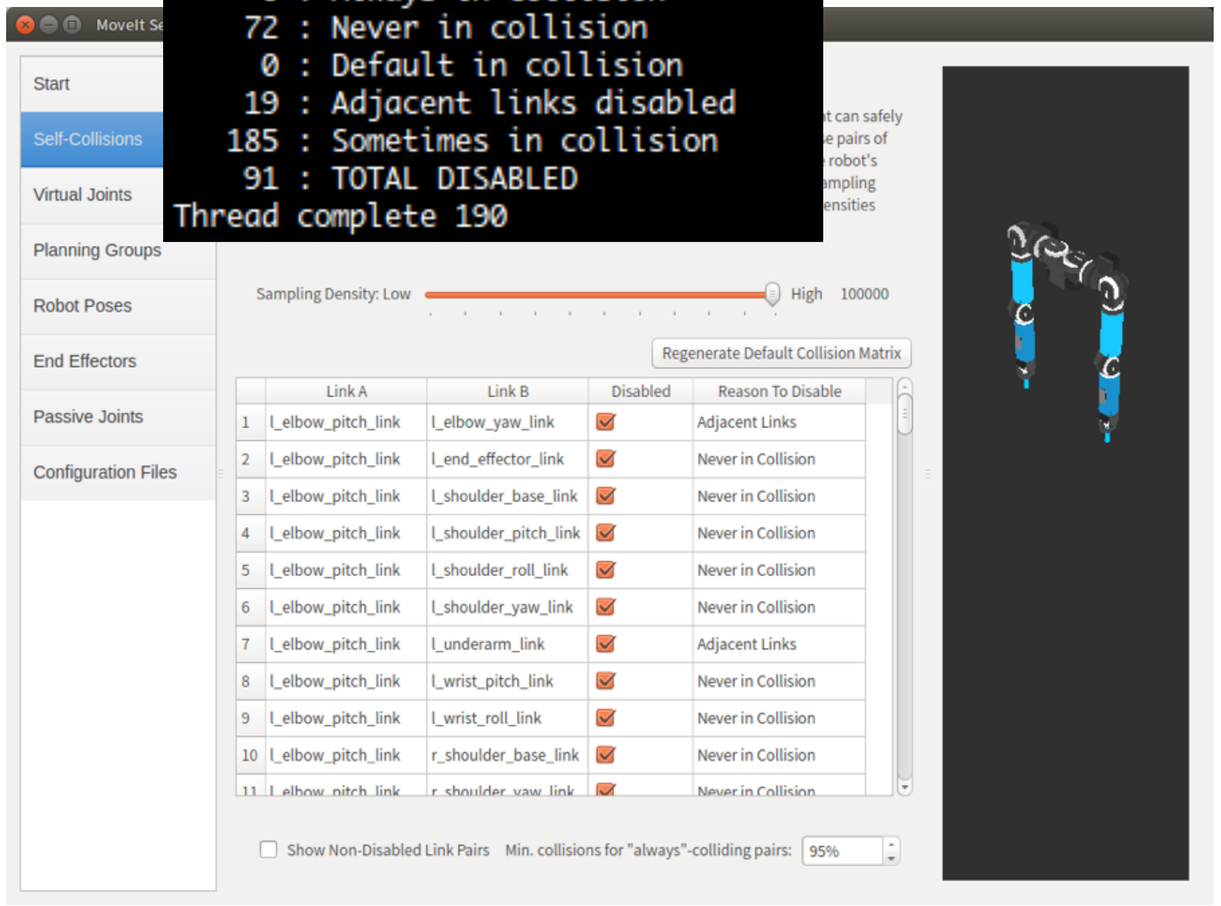
MoveIt과 연동할 로봇의 URDF를 불러옴. XACRO로 되어 있어도 무방



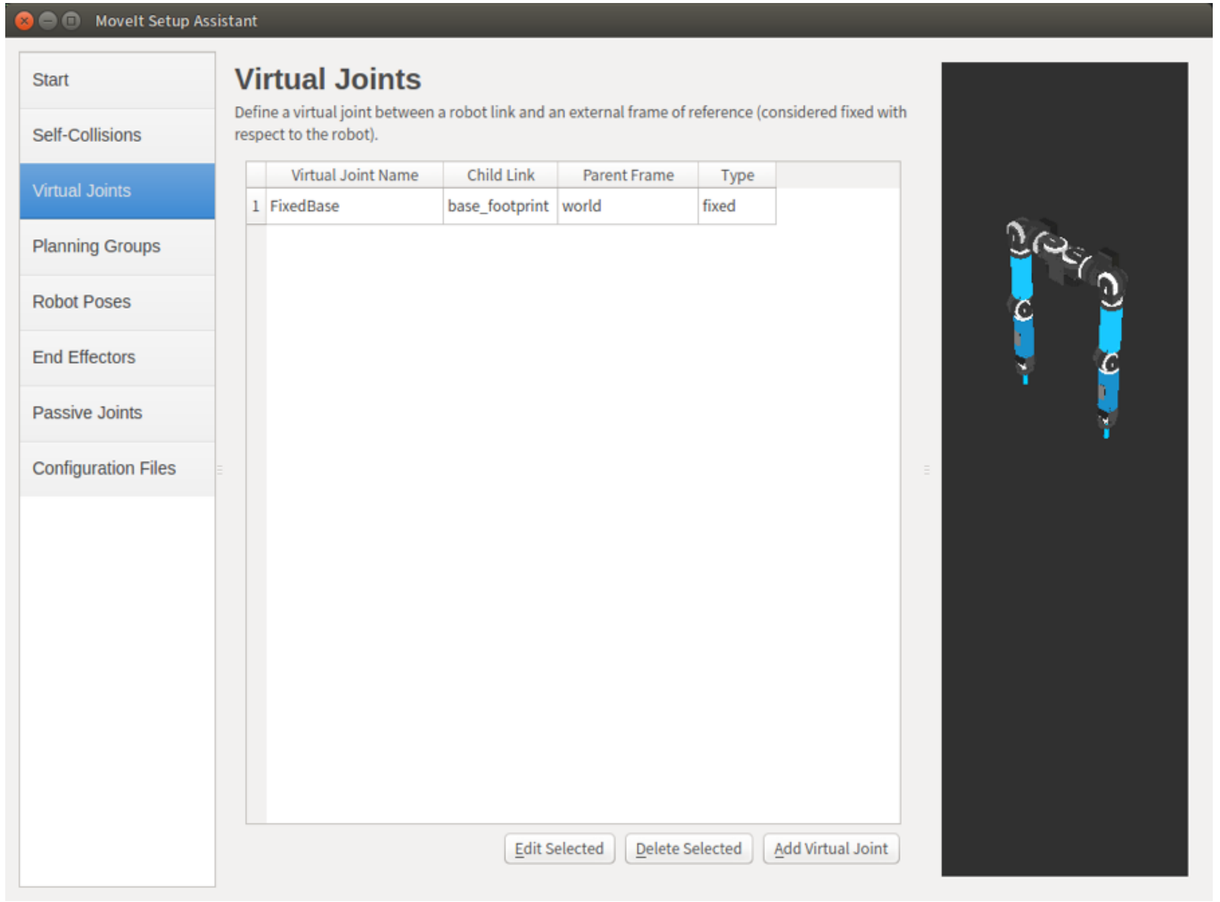
정상적으로 불러온 것을 확인



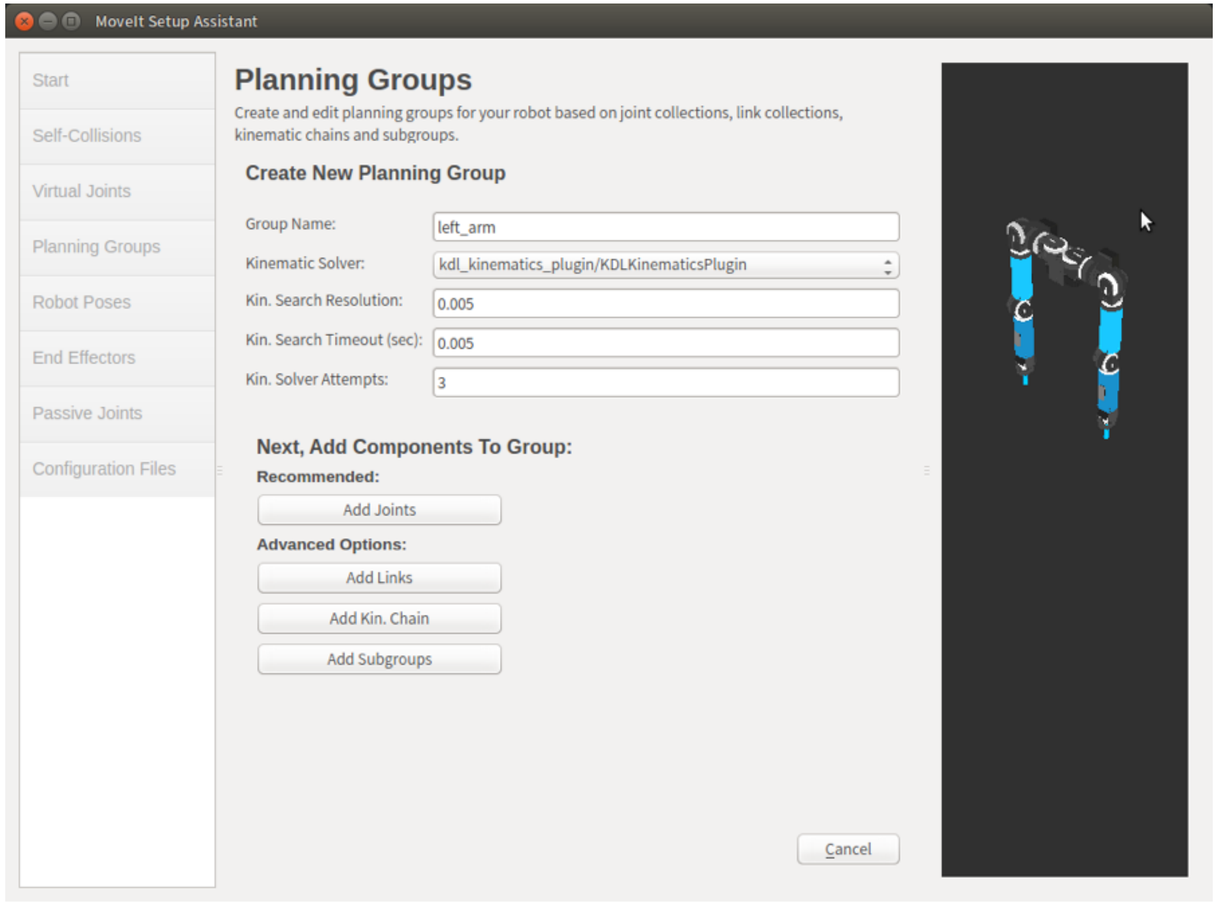

```
Statistics:
    24 : Total Links
   276 : Total possible collisions
     0 : Always in collision
    72 : Never in collision
     0 : Default in collision
    19 : Adjacent links disabled
   185 : Sometimes in collision
    91 : TOTAL DISABLED
Thread complete 190
```



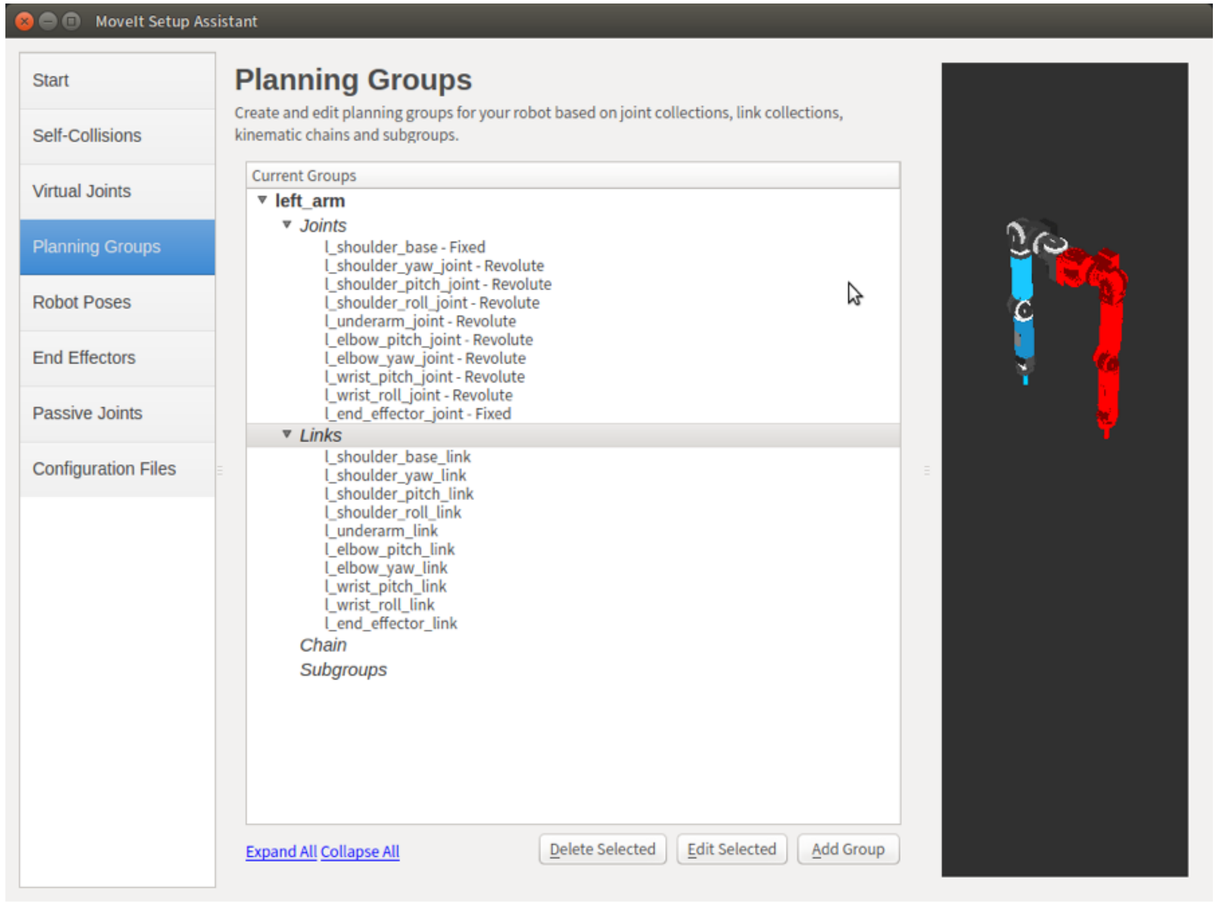
Sampling Density를 최대값으로 올리고, 모델 링크간의 충돌에 대한
매트릭스 생성

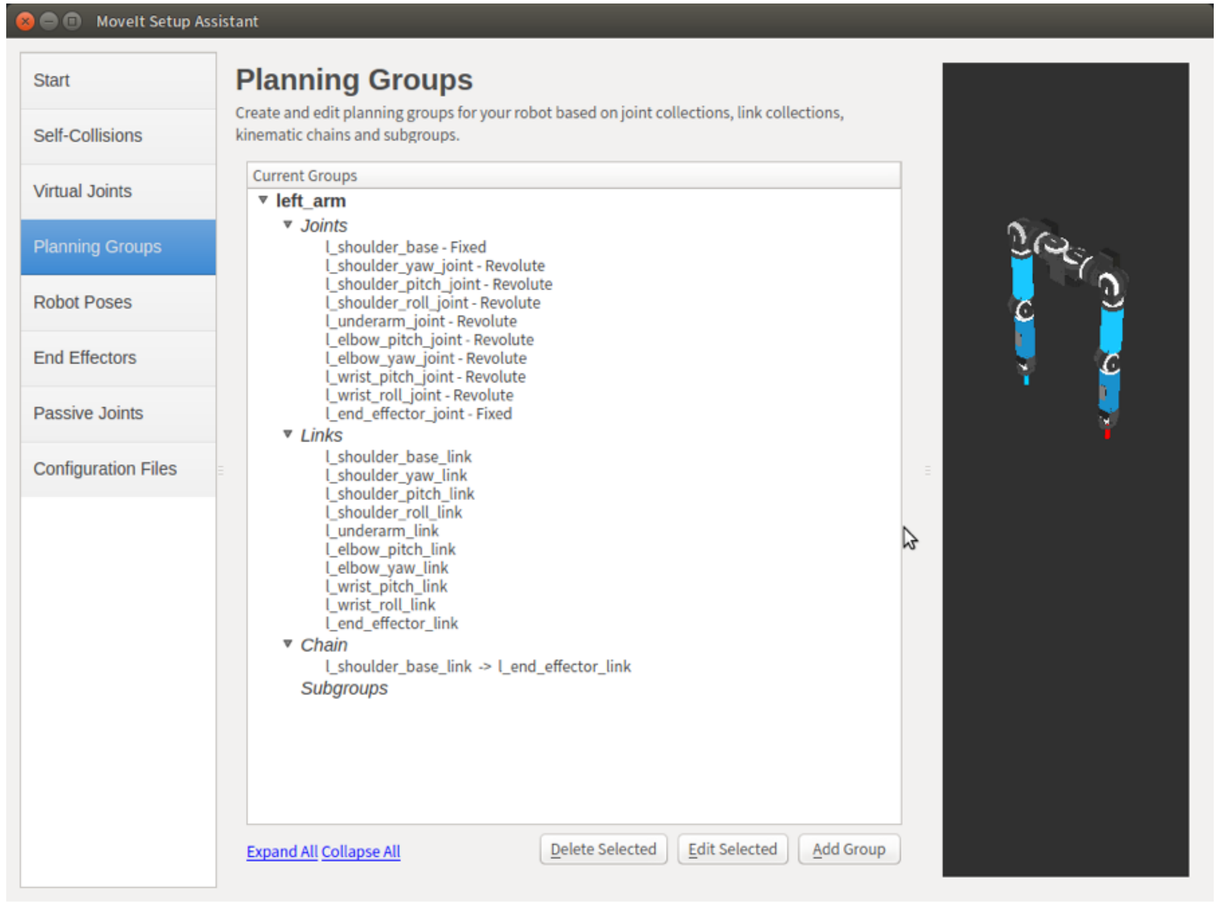


외부환경과 로봇과의 연결을 위한 가상 조인트(고정) 생성

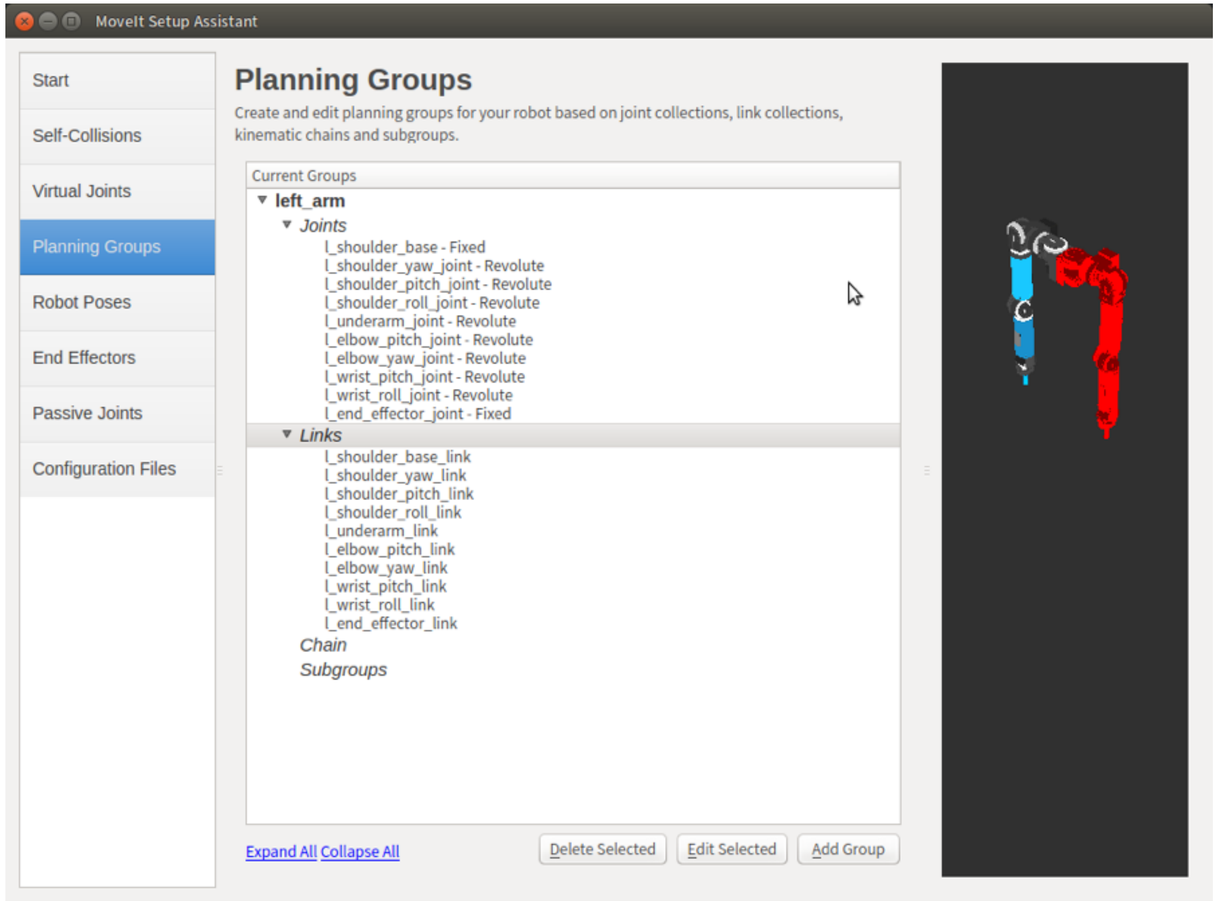


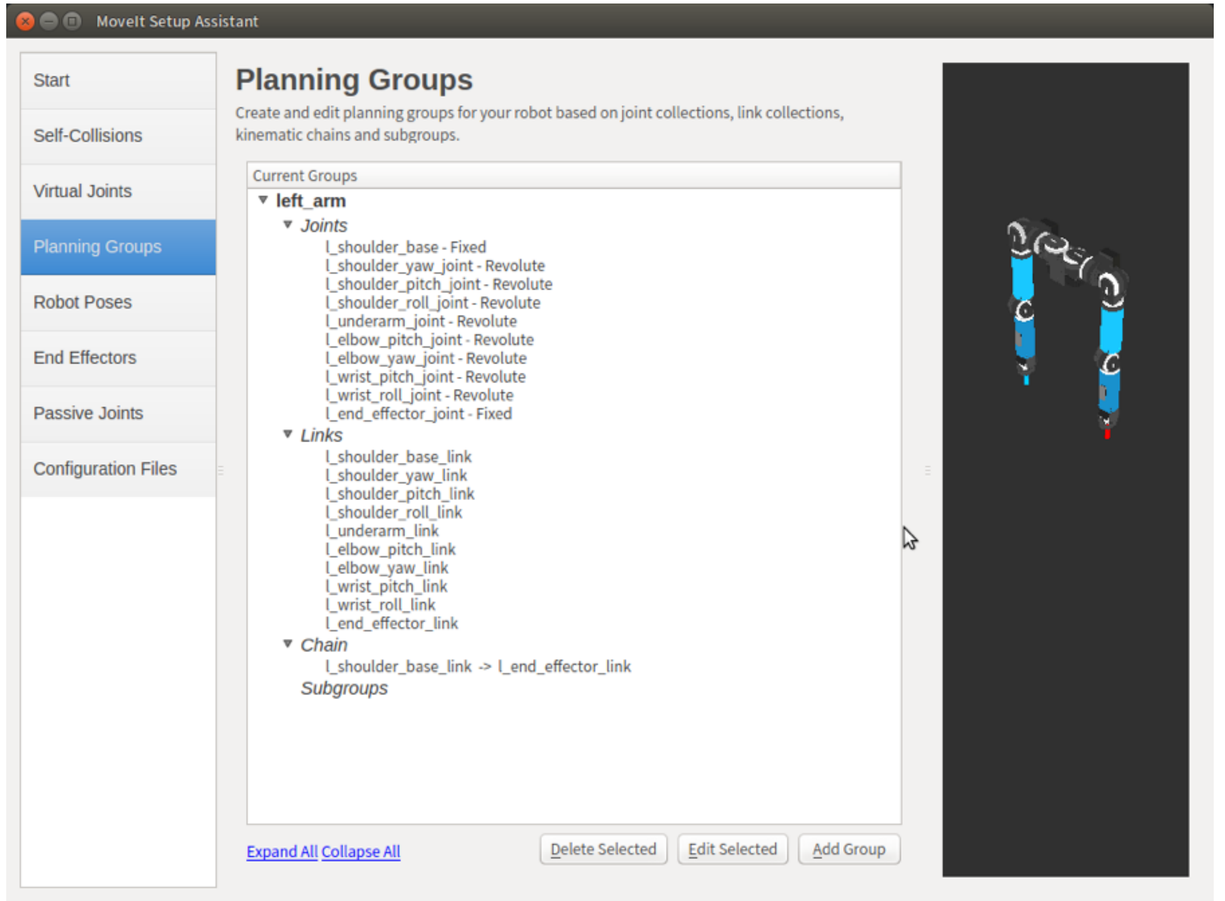
Planning 그룹 설정. safe_arm의 경우, 왼쪽과 오른쪽 팔 두개의 Planning 그룹을 생성함



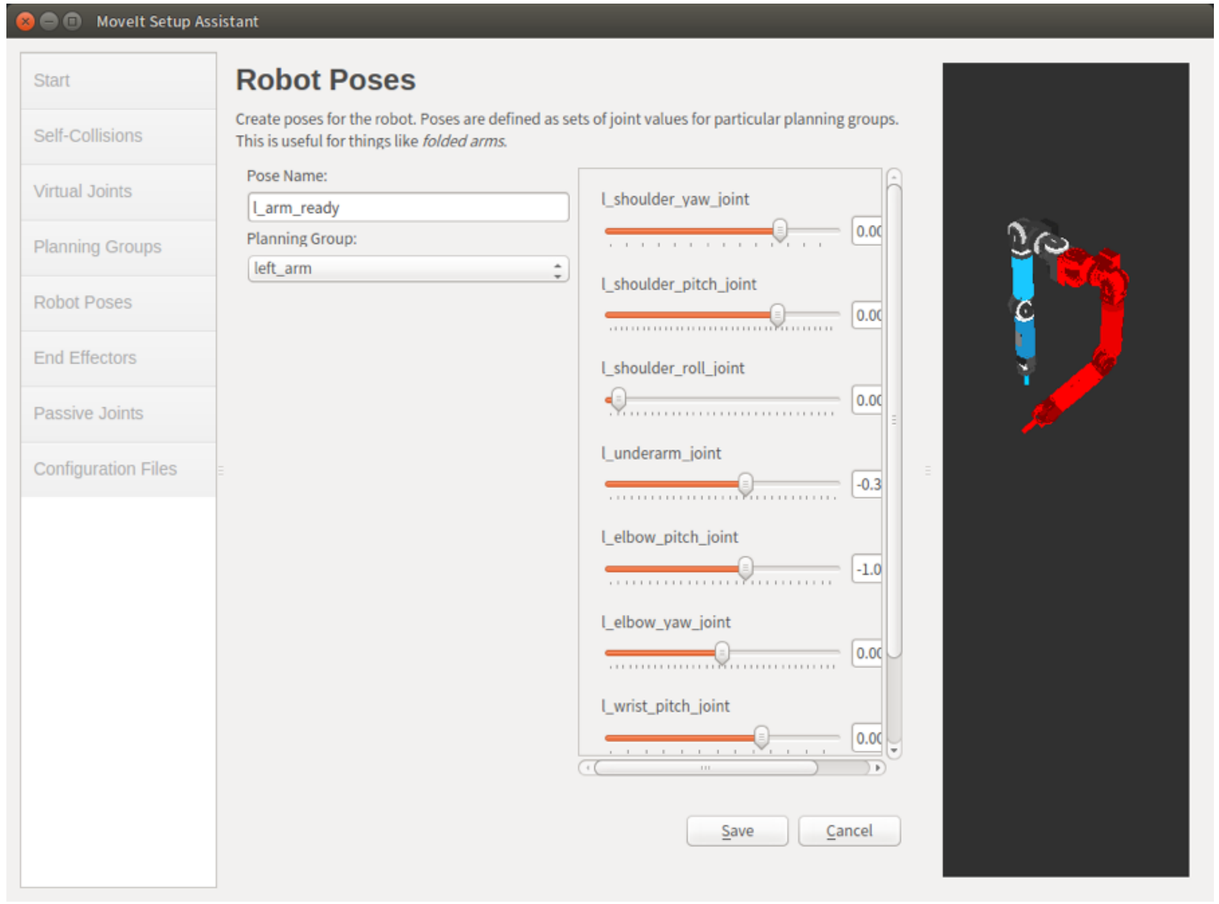


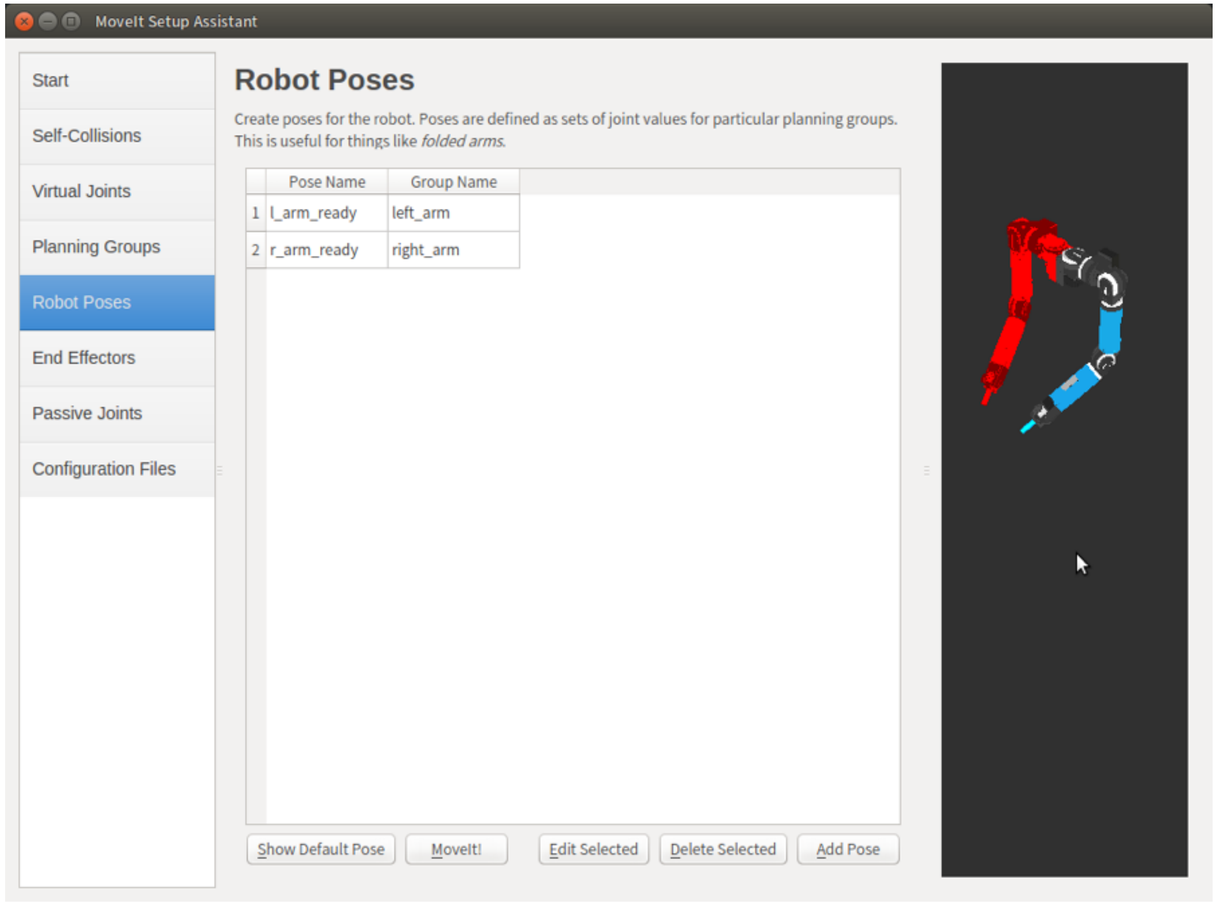
왼쪽팔에 대한 조인트 그룹, 링크 그룹, Kinematic Chain을 설정해 준다.



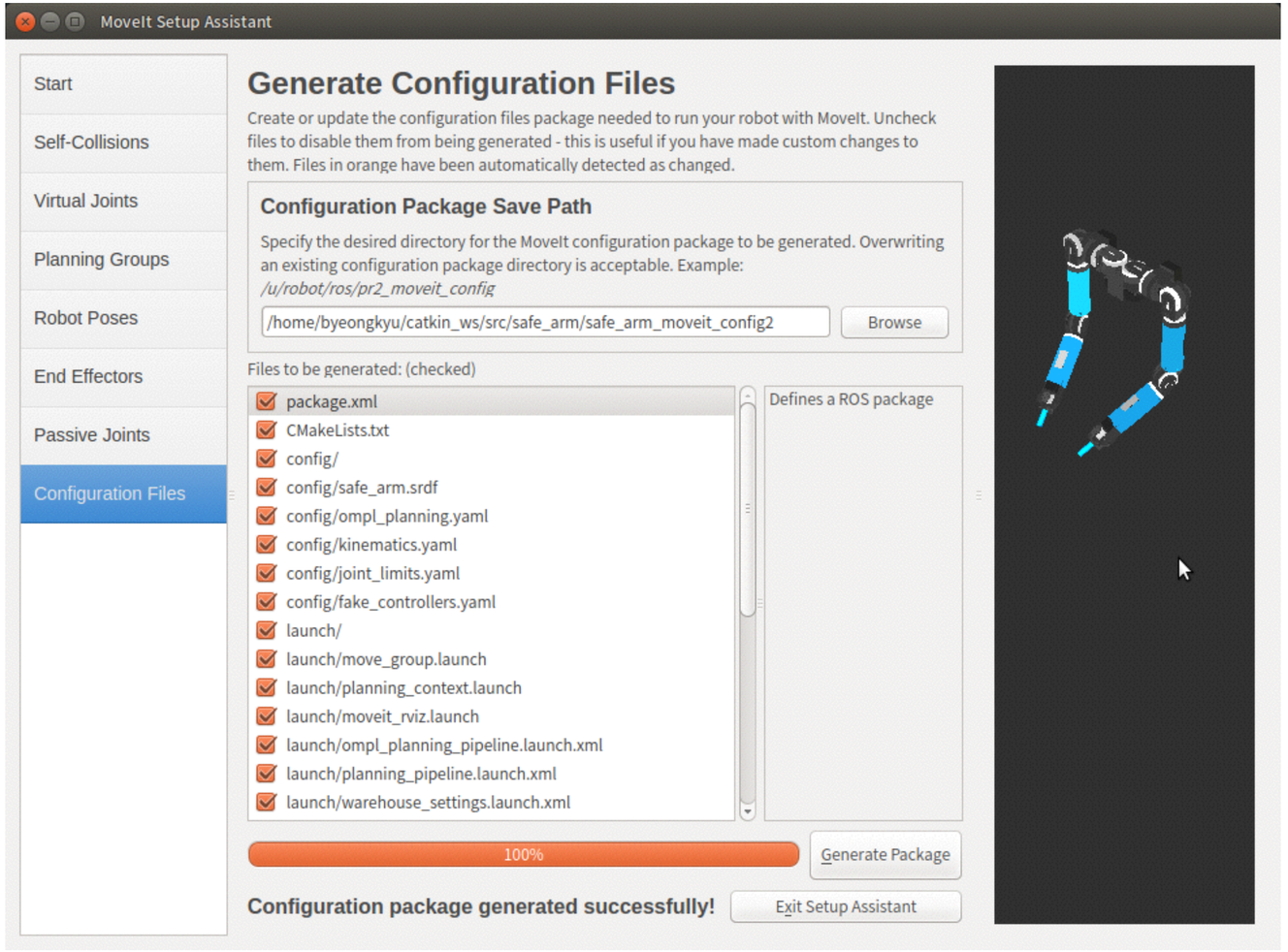


왼쪽팔에 대한 조인트 그룹, 링크 그룹, Kinematic Chain을 설정해 준다.
오른쪽 팔에 대해서도 같은 방법으로 설정





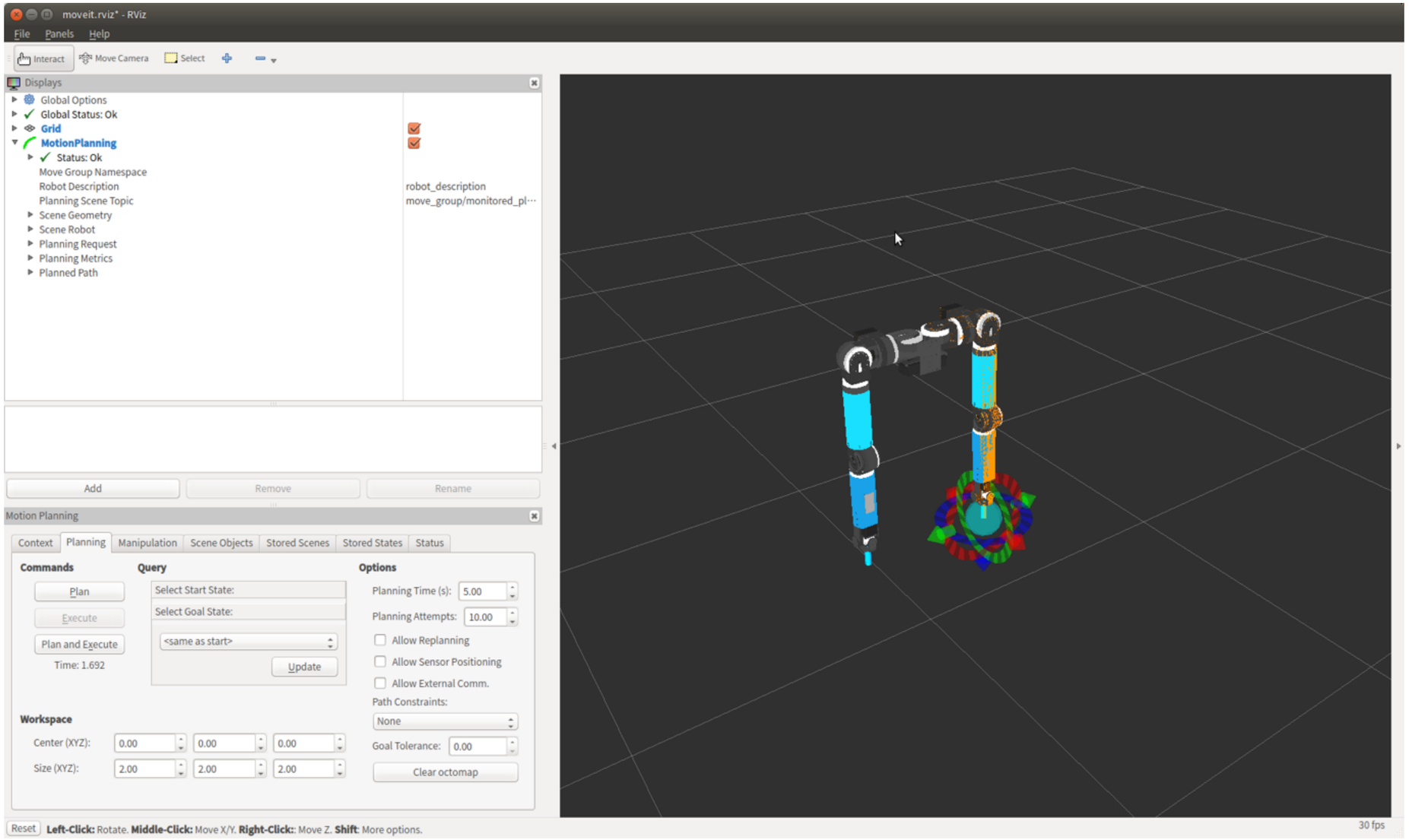
로봇의 기본 자세 (Homing Position, Ready, etc)를 만들어준다. (선택사항)

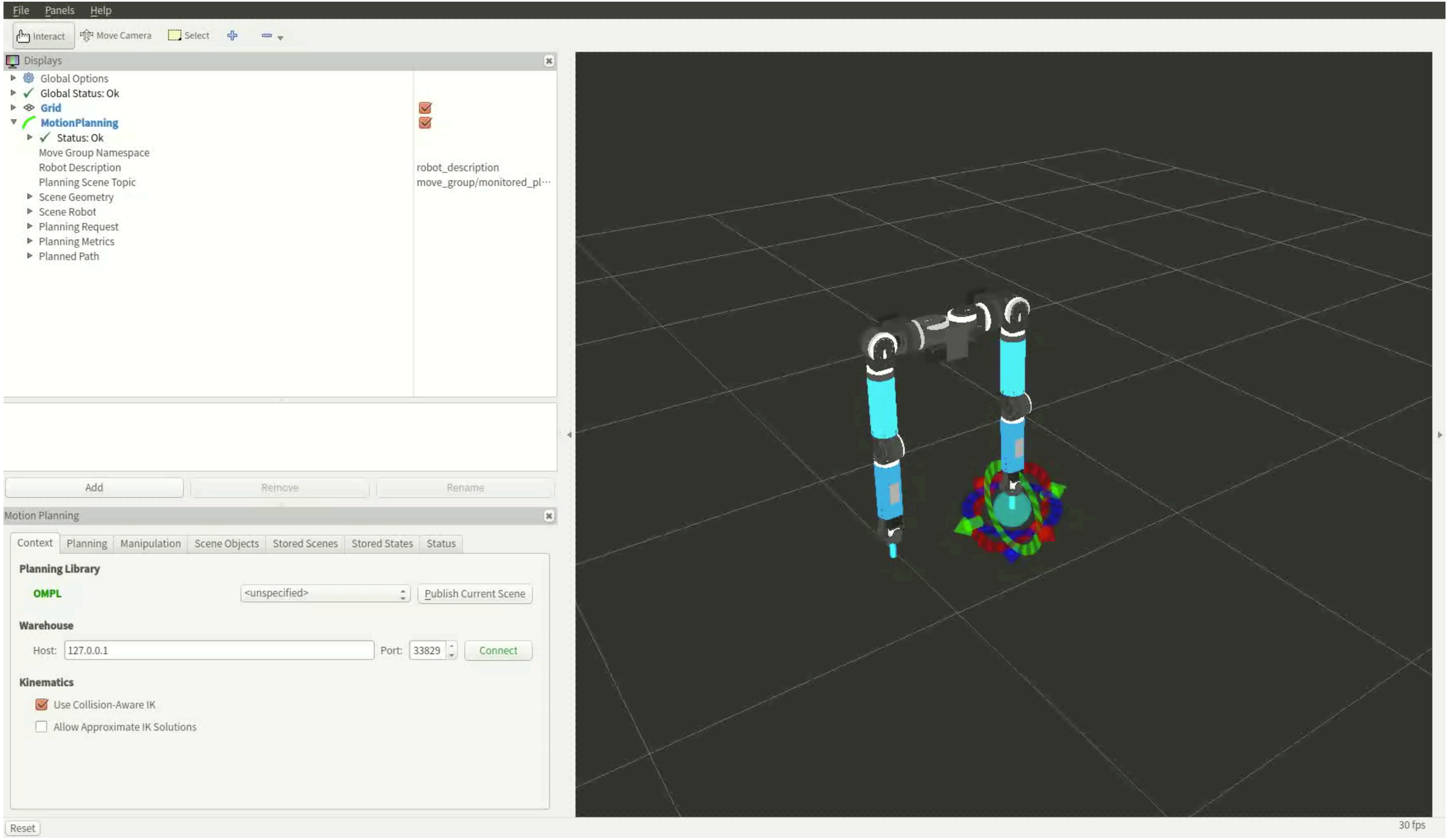


마지막으로 catkin workspace 내에 패키지가 저장될 곳을 선택한 후, 패키지 생성 완료

MoveIt Setup에 대한 검증

```
$ roslaunch safe_arm_moveit_config demo.launch
```




File
Panels
Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
move_group/monitored_pl···
Scene Geometry
Scene Robot
Planning Request
Planning Metrics
Planned Path
Add
Remove
Rename
Motion Planning
Context
Planning
Manipulation
Scene Objects
Stored Scenes
Stored States
Status
Planning Library
OMPL
<unspecified>
Publish Current Scene
Warehouse
Host:
127.0.0.1
Port:
33829
Connect
Kinematics
Use Collision-Aware IK
Allow Approximate IK Solutions
Reset
30 fps

MoveIt과 Gazebo와의 연동을 위한 추가 작업

MoveIt Config 패키지 저장 장소가 ~/catkin_ws/src/safe_arm_moveit_config라고 가정

MoveIt Controller Manager 설정

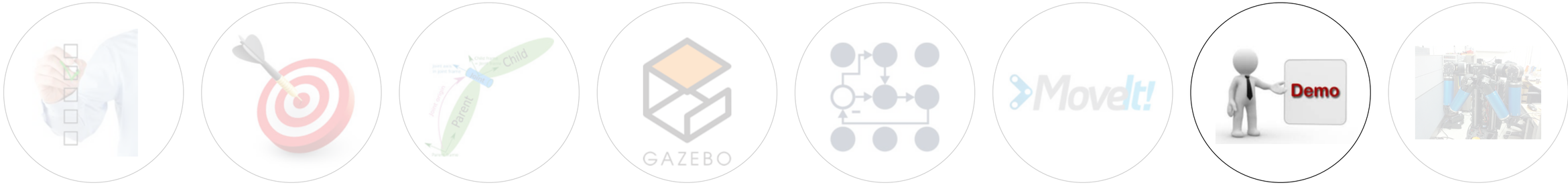
safe_arm_moveit_config/config/controllers.yaml 파일 생성

```
controller_list:
  - name: "safe_arm/l_arm_joint_controller"
    action_ns: follow_joint_trajectory
    type: FollowJointTrajectory
    joints:
      - l_shoulder_yaw_joint
      - l_shoulder_pitch_joint
      - l_shoulder_roll_joint
      - l_underarm_joint
      - l_elbow_pitch_joint
      - l_elbow_yaw_joint
      - l_wrist_pitch_joint
      - l_wrist_roll_joint
  - name: "safe_arm/r_arm_joint_controller"
    action_ns: follow_joint_trajectory
    type: FollowJointTrajectory
    joints:
      - r_shoulder_yaw_joint
      - r_shoulder_pitch_joint
      - r_shoulder_roll_joint
      - r_underarm_joint
      - r_elbow_pitch_joint
      - r_elbow_yaw_joint
      - r_wrist_pitch_joint
      - r_wrist_roll_joint
```

ros_control 설정시 보았던 controller의 이름과 Joint 리스트를 적어줌.

safe_arm_moveit_config/config/safe_arm_moveit_controller_manager.launch.xml 파일 수정

```
<launch>
    <!-- Set the param that trajectory_execution_manager needs to find the controller plugin -->
    <arg name="moveit_controller_manager" default="moveit_simple_controller_manager/MoveItSimpleControllerManager" />
    <param name="moveit_controller_manager" value="$(arg moveit_controller_manager)"/>

    <!-- load controller_list -->
    <arg name="use_controller_manager" default="true" />
    <param name="use_controller_manager" value="$(arg use_controller_manager)" />

    <!-- Load joint controller configurations from YAML file to parameter server -->
    <rosparam file="$(find safe_arm_moveit_config2)/config/controllers.yaml"/>
</launch>
```



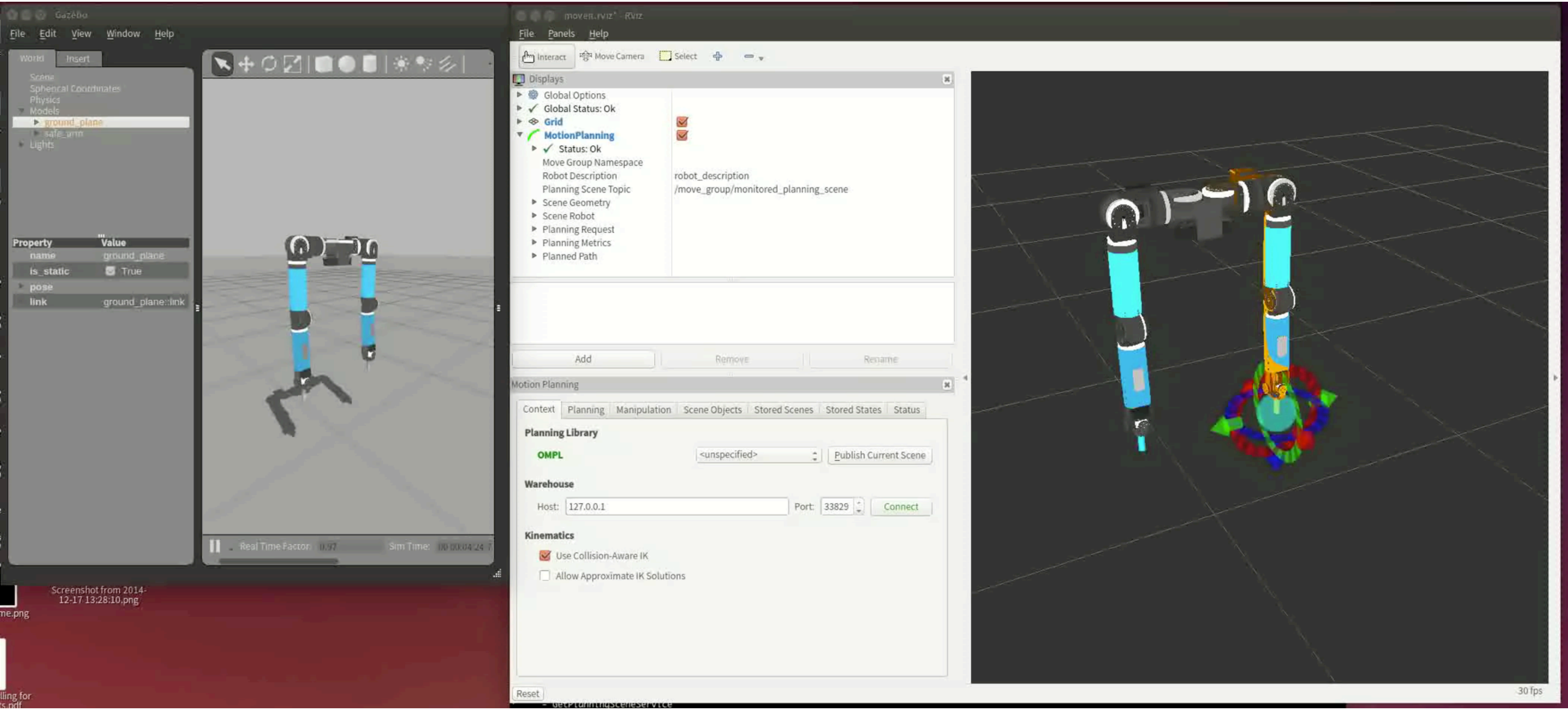
실행

Demo

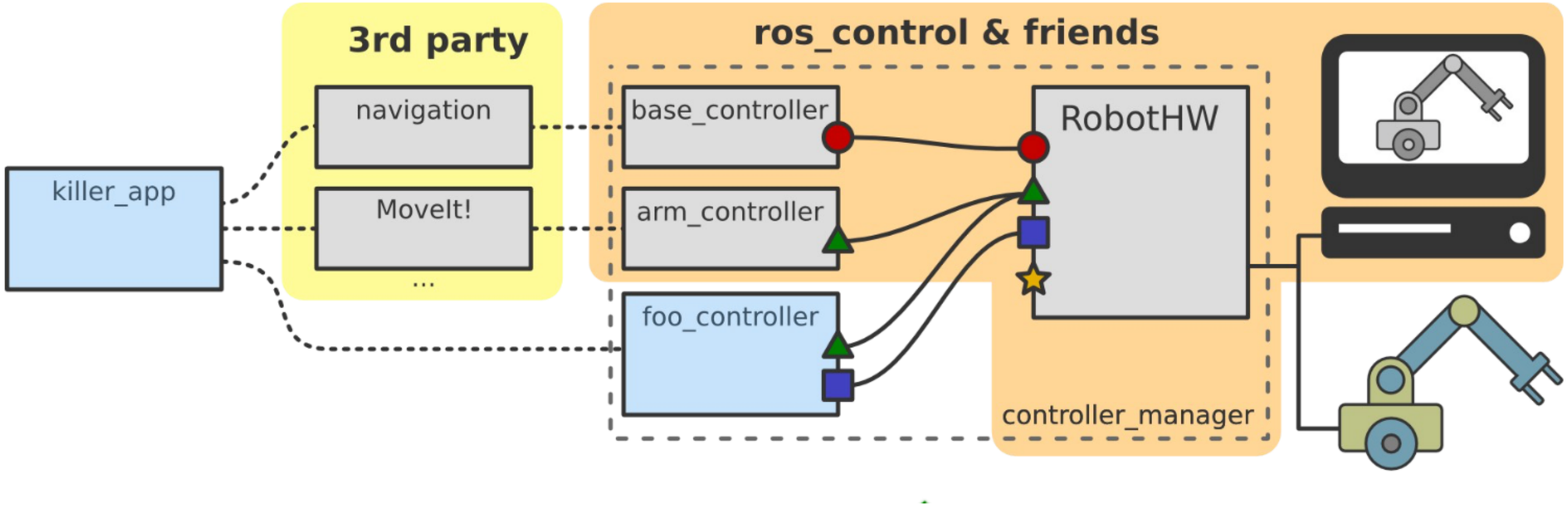
```
$ roslaunch safe_arm_bringup bringup.launch
```

safe_arm_bringup/launch/bringup.launch

```
<launch>
    <include file="$(find safe_arm_bringup)/launch/safe_arm_sim.launch" />

    <include file="$(find safe_arm_control)/launch/safe_arm_joint_state_controller.launch" />
    <include file="$(find safe_arm_control)/launch/safe_arm_l_controller.launch" />
    <include file="$(find safe_arm_control)/launch/safe_arm_r_controller.launch" />

    <include file="$(find safe_arm_moveit_config)/launch/move_group.launch" />
    <include file="$(find safe_arm_moveit_config)/launch/moveit_rviz.launch">
        <arg name="config" value="true" />
    </include>

    <include file="$(find safe_arm_moveit_config)/launch/default_warehouse_db.launch" />
</launch>
```


Gazebo
File Edit View Window Help
World Insert
Scene
Spherical Coordinates
Physics
Models
ground_plane
safe_arm
Lights
Property Value
name ground_plane
is_static True
pose
link ground_plane::link
Real Time Factor: 0.97
Sim Time: 00 00:04:24.7
Screenshot from 2014-12-17 13:28:10.png
me.png
lling for
s.pdf
move.rviz* - RViz
File Panels Help
Interact
Move Camera
Select
Displays
Global Options
Global Status: Ok
Grid
MotionPlanning
Status: Ok
Move Group Namespace
Robot Description
robot_description
Planning Scene Topic
/move_group/monitored_planning_scene
Scene Geometry
Scene Robot
Planning Request
Planning Metrics
Planned Path
Add
Remove
Rename
Motion Planning
Context
Planning
Manipulation
Scene Objects
Stored Scenes
Stored States
Status
Planning Library
OMPL
<unspecified>
Publish Current Scene
Warehouse
Host: 127.0.0.1
Port: 33829
Connect
Kinematics
Use Collision-Aware IK
Allow Approximate IK Solutions
Reset
30 fps

Real Robot?



Adolfo Rodríguez Tsouroukdissian, “ros_control: An overview”, ROSCon 2014

Q&A?

byeongkyu@gmail.com
http://ahnbk.com
http://facebook.com/ahnbk