# CSC752 Autonomous Robotic Systems
# - Introduction into ROS (3) -

*Ubbo Visser*

Department of Computer Science
College of Arts and Sciences
University of Miami

September 2020

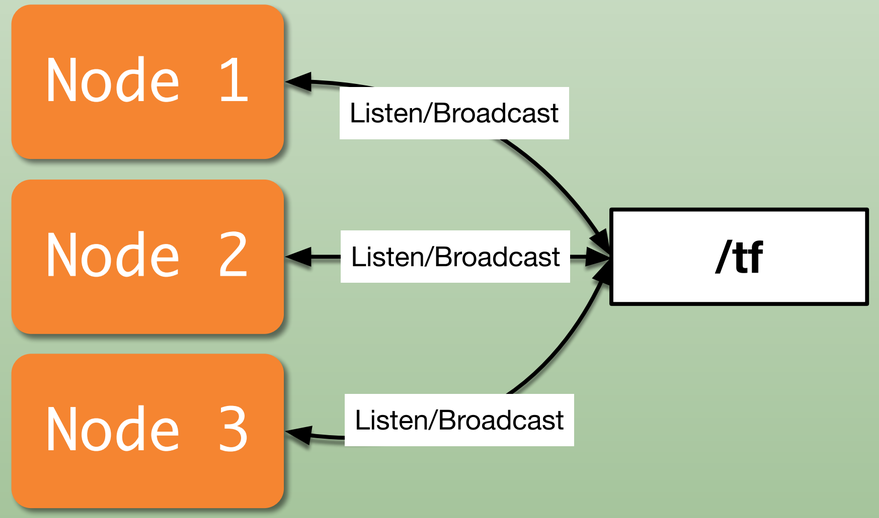
UNIVERSITY OF MIAMI
ROBOCANES

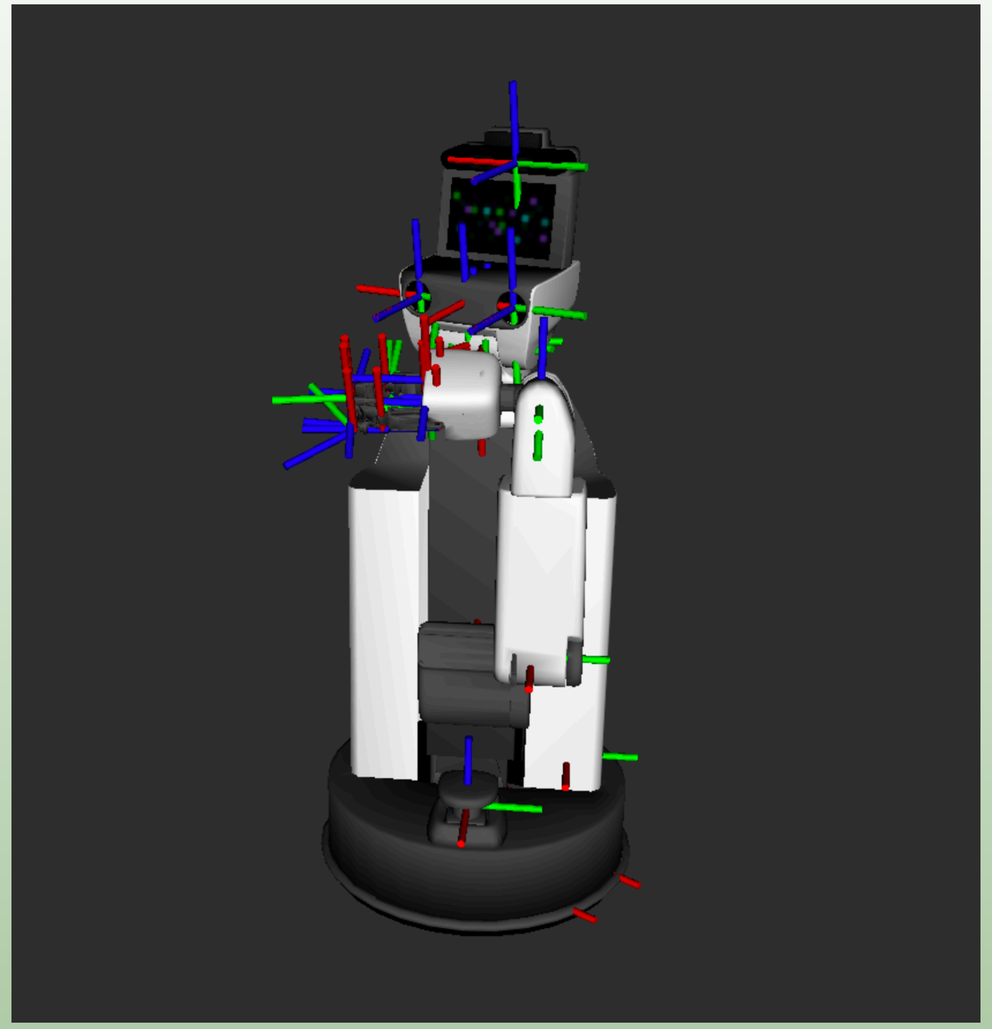
# OVERVIEW

- TF Transformation System
- rqt User Interface
- Robot models (URDF)
- Simulation descriptions (SDF)

# TF TRANSFORMATION SYSTEM

- Tool for keeping track of coordinate frames over time
- Maintains relationship between coordinate frames in a tree structure buffered in time
- Lets the user transform points, vectors, etc. between coordinate frames at desired time
- Implemented as publisher/subscriber model on the topics **/tf** and **/tf_static**



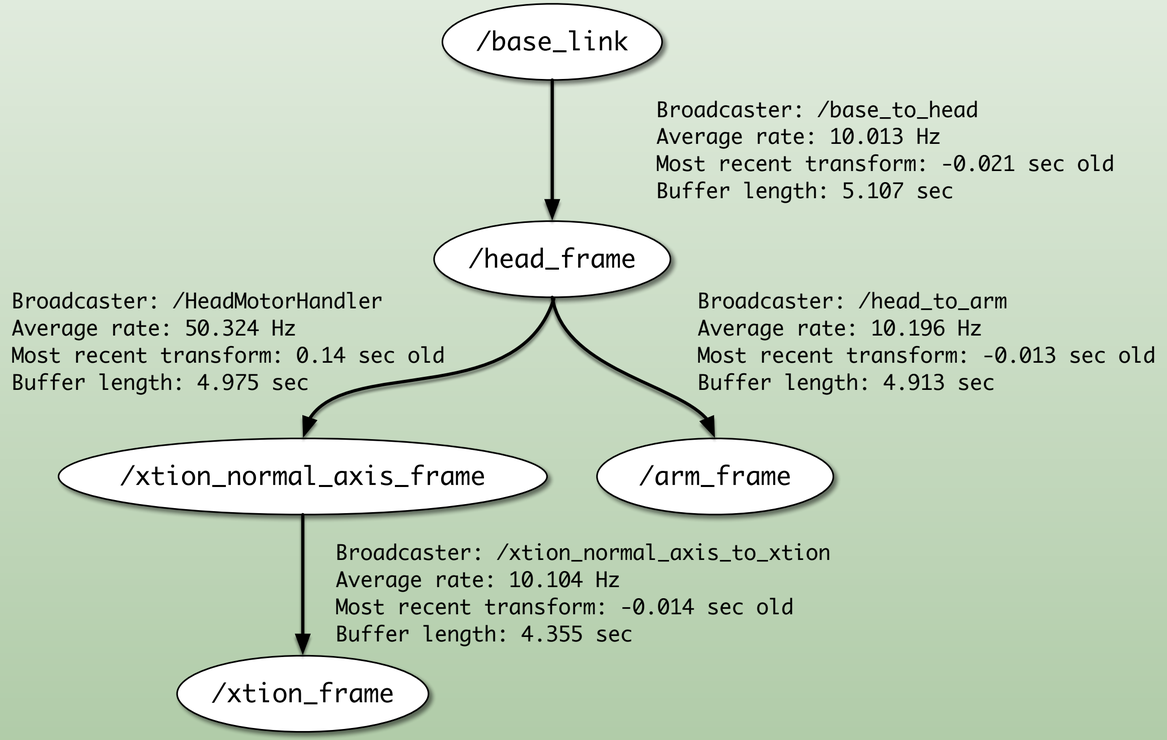
*Details at: http://wiki.ros.org/tf2*

# TF TRANSFORMATION SYSTEM - TRANSFORM TREE

- TF listeners use a buffer to listen to all broadcasted transforms
- Query for specific transforms from the transform tree

tf2_msgs/TFMessage.msg

```
geometry_msgs/TransformStamped[] transforms
  std_msgs/Header header
    uint32 seqtime stamp
string frame_id
string child_frame_id
geometry_msgs/Transform transform
    geometry_msgs/Vector3 translation
    geometry_msgs/Quaternion rotation
```




*Details at: http://docs.ros.org/jade/api/tf2_msgs/html/msg/TFMessage.html*

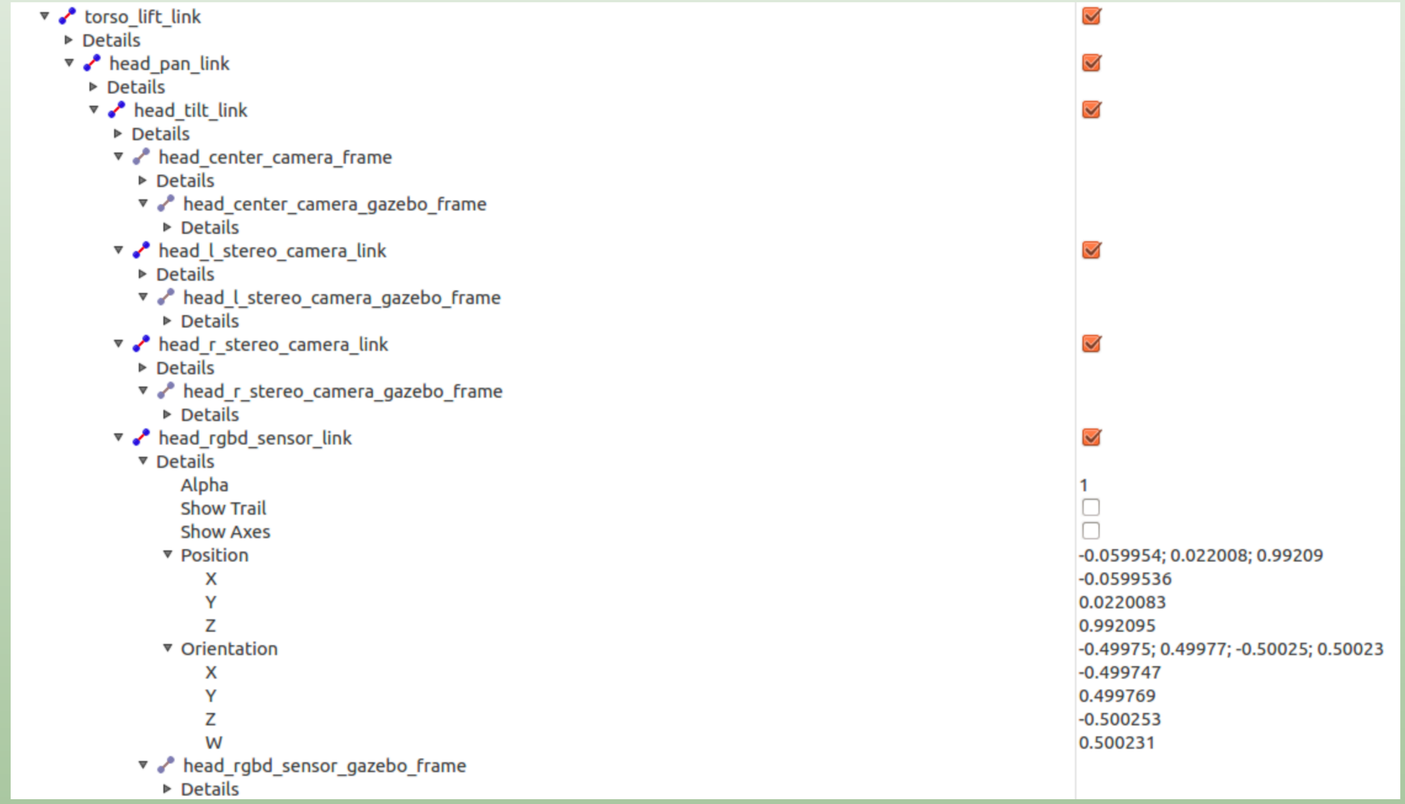
# TF TRANSFORMATION SYSTEM - TRANSFORM TREE

- TF listeners use a buffer to listen to all broadcasted transforms
- Query for specific transforms from the transform tree

tf2_msgs/TFMessage.msg

```
geometry_msgs/TransformStamped[] transforms
  std_msgs/Header header
    uint32 seqtime stamp
string frame_id
string child_frame_id
geometry_msgs/Transform transform
    geometry_msgs/Vector3 translation
    geometry_msgs/Quaternion rotation
```

Partial link tree from HSR_B, torso




*Details at: http://docs.ros.org/jade/api/tf2_msgs/html/msg/TFMessage.html*

# TF TRANSFORMATION SYSTEM - TOOLS

## Terminal

Get information about the current transform tree

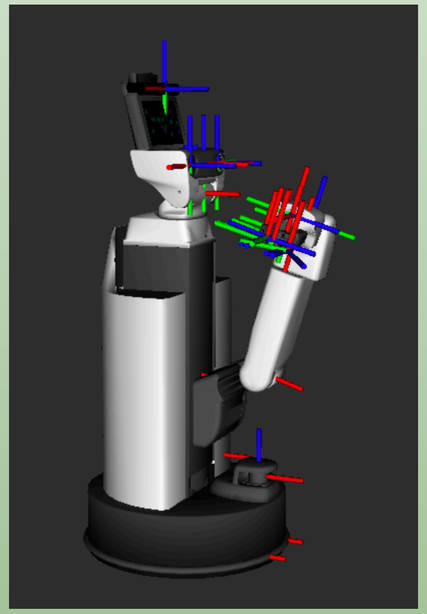
```
~$rosrun tf tf_monitor
```

Get information about the transform between two frames

```
~$rosrun tf tf_echo source_frame target_frame
```

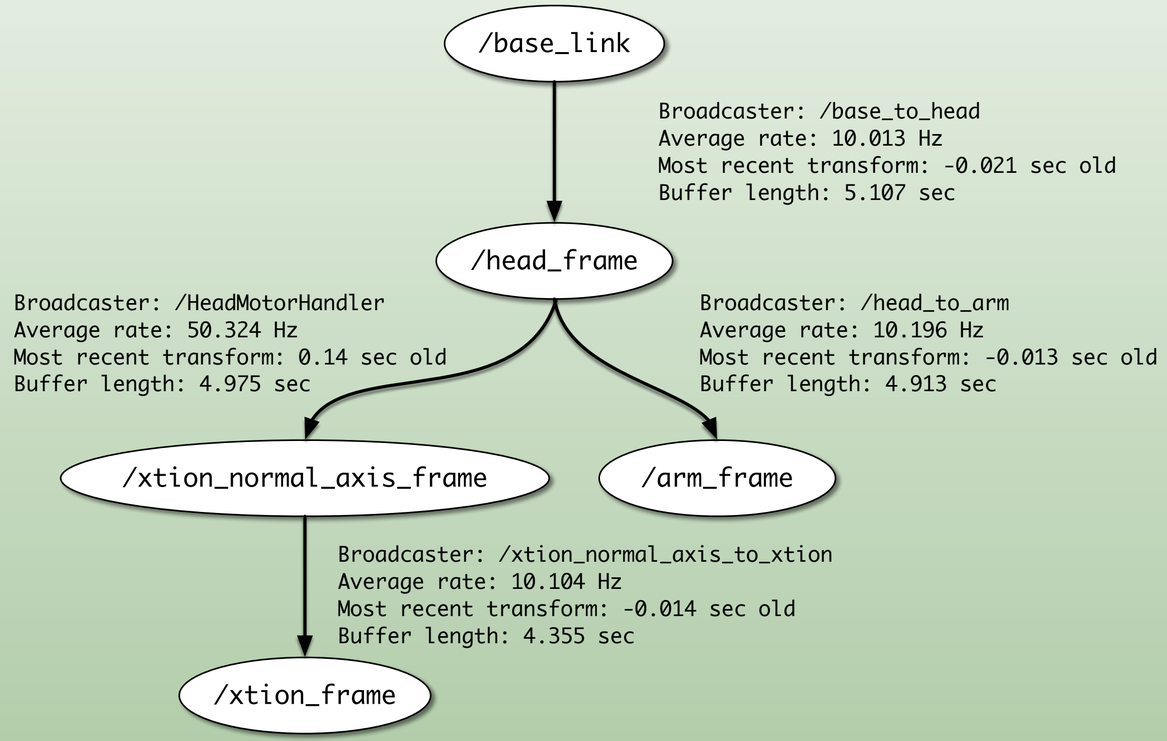
## RViz

3D visualization of the transforms

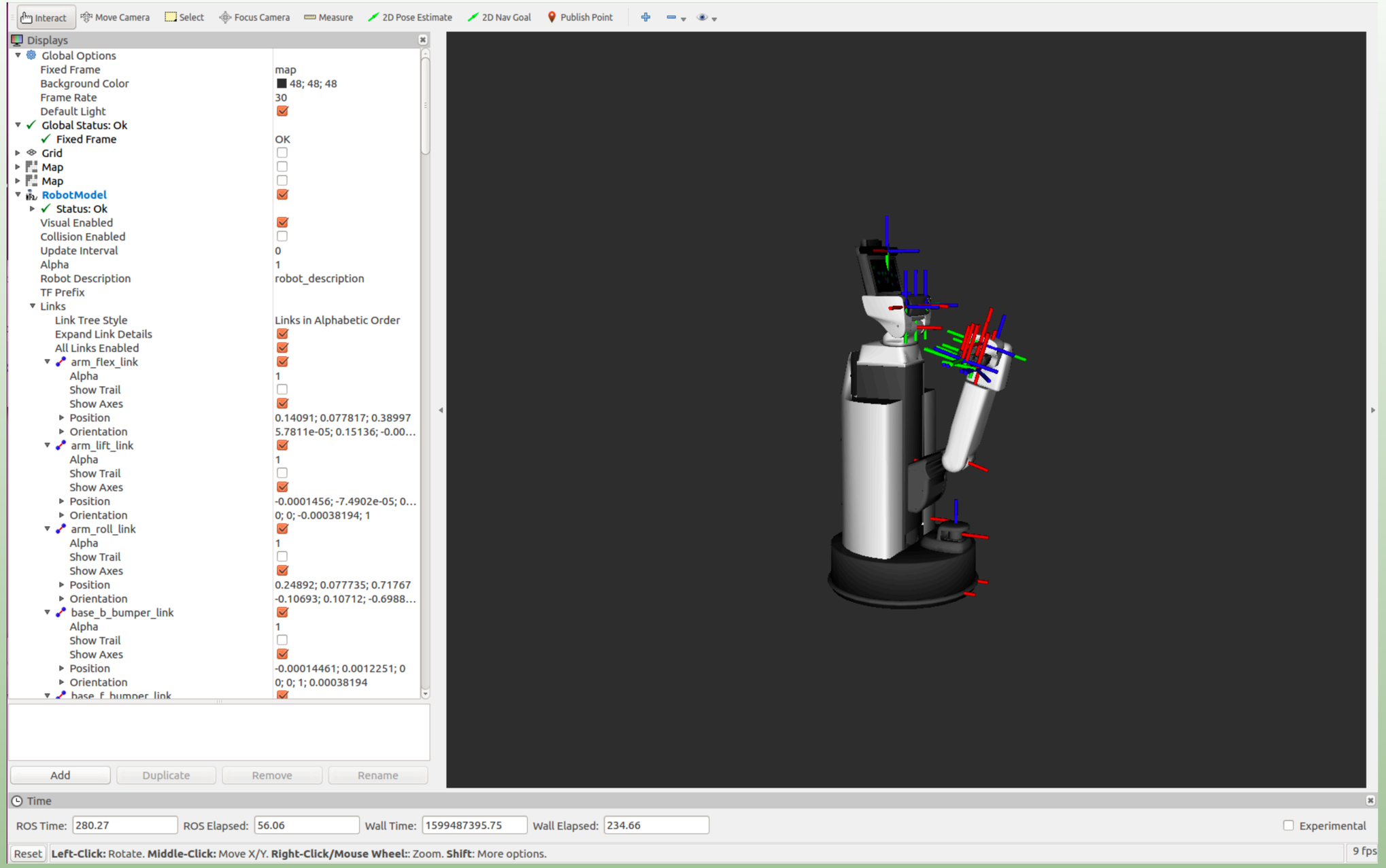


## View frames

Visual graph of the transform tree



# TF TRANSFORMATION SYSTEM RVIZ PLUGIN



# TF TRANSFORMATION SYSTEM - TRANSFORM LISTENER C++ API

```cpp
#include <ros/ros.h>
#include <tf2_ros/transform_listener.h>
#include <geometry_msgs/TransformStamped.h>

int main(int argc, char** argv)
{
  ros::init(argc, argv, "tf2_listener");
  ros::NodeHandle nodeHandle;

  tf2_ros::Buffer tfBuffer;
  tf2_ros::TransformListener tfListener(tfBuffer);

  ros::Rate rate(10.0);
  while (nodeHandle.ok())
  {
    geometry_msgs::TransformStamped transformStamped;
    try
    {
      transformStamped = ("base","odom", ros::Time(0));
    }
    catch (tf2::TransformException &exception)
    {
        ROS_WARN("%s", exception.what());
        os::Duration(1.0).sleep();
        continue;
    }
    rate.sleep();
  }
  return 0;
}
```

- Create a TF listener to fill up a buffer

```cpp
tf2_ros::Buffer tfBuffer;
tf2_ros::TransformListener tfListener(tfBuffer);
```

- Beware of scope!
- Lookup transformations use this:

```cpp
geometry_msgs::TransformStamped transformStamped =
    tfBuffer.lookupTransform(target_frame_id,
    source_frame_id, time);
```

- For time: use **ros::Time(0)** to get latest available transform

*Details at:*
*http://wiki.ros.org/tf2/Tutorials/Writing%20a%20tf2%20listener%20(C++)*

# RQT USER INTERFACE

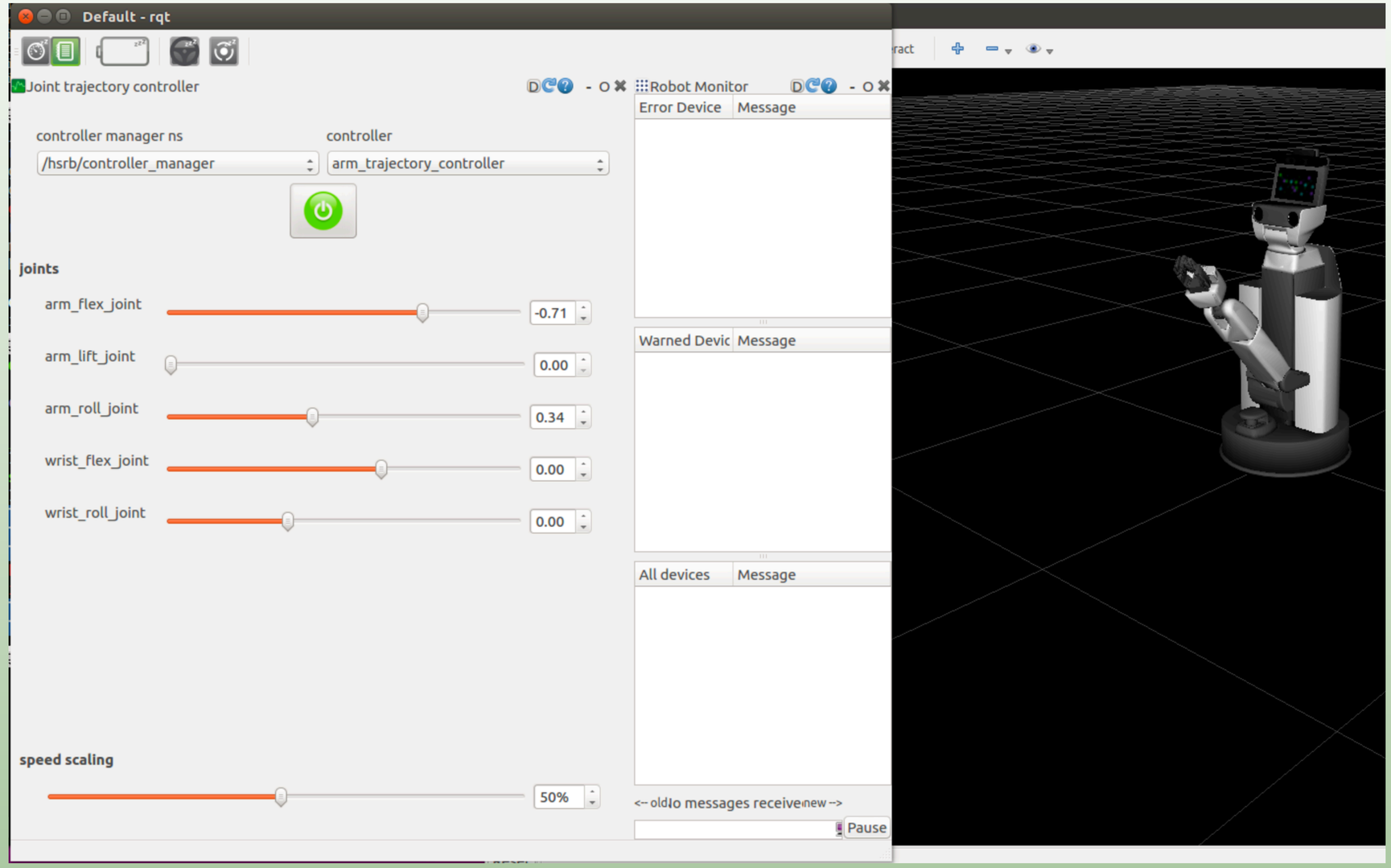
- User interface based on Qt
- Custom interfaces possible
- Use existing plugins
- Create your own plugins

Run RQT

```
~$rosrun rqt_gui rqt_gui
```

Alternative

```
~$rqt
```



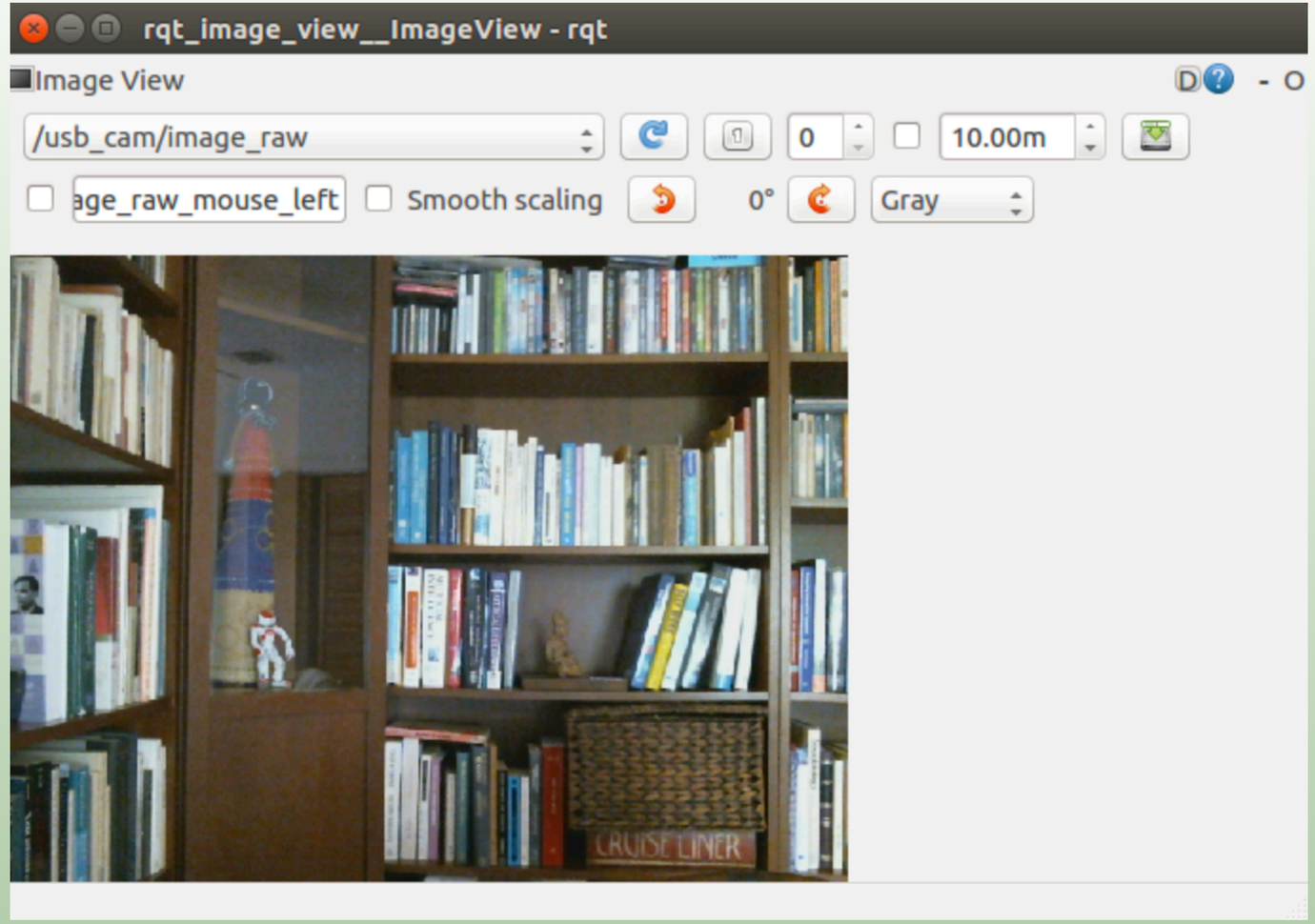

*Details at: http://wiki.ros.org/rqt/Plugins*

# RQT USER INTERFACE IMAGE VIEW

- Visualizing images

Run rqt_image_view

```
~$rosrun rqt_image_view rqt_image_view
```



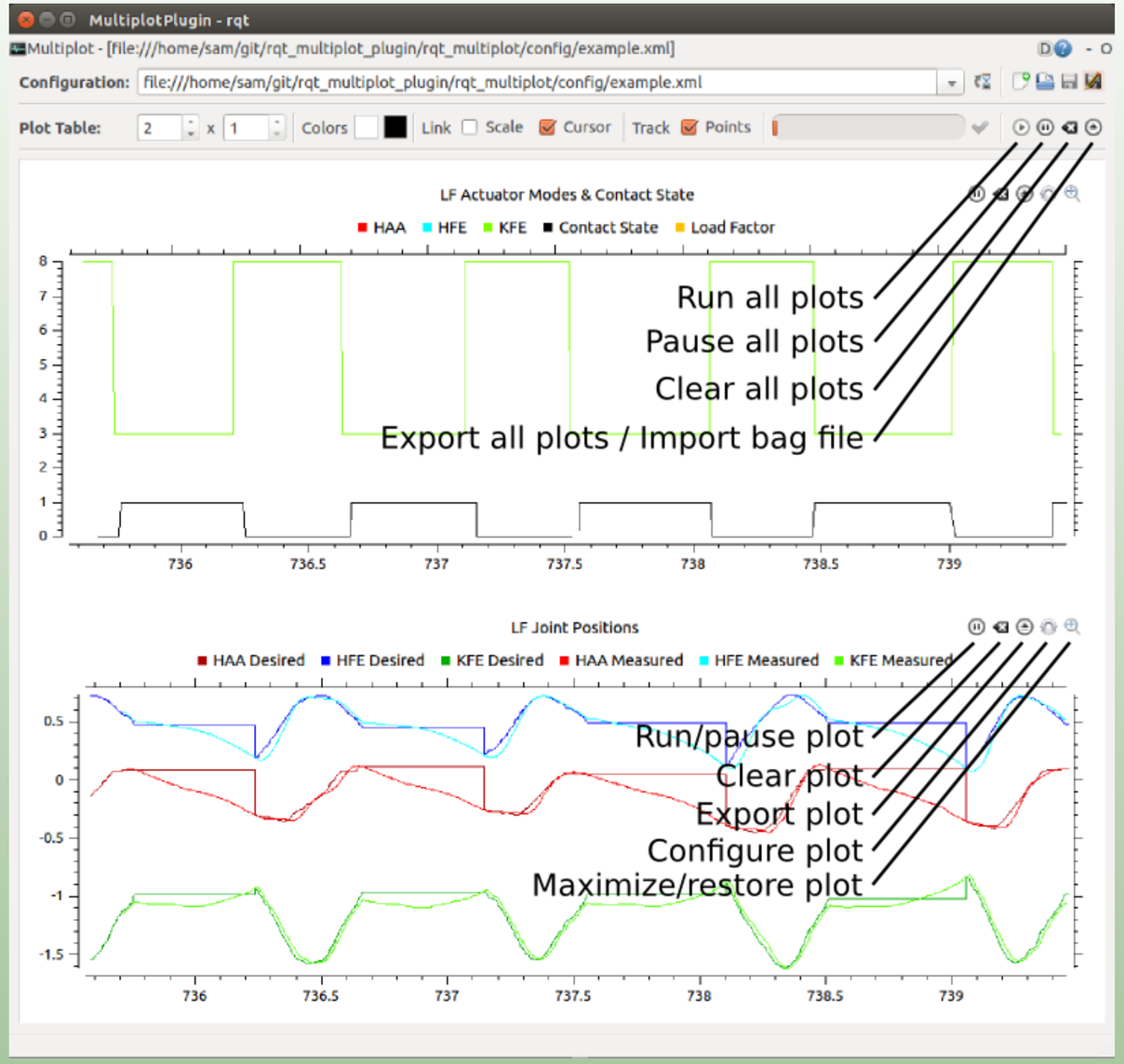

*Details at: http://wiki.ros.org/rqt_image_view*

# RQT USER INTERFACE RQT_MULTIPLOT

- Visualizing numeric values in 2D plots

Run rqt_multiplot

```
~$rosrun rqt_multiplot rqt_multiplot
```



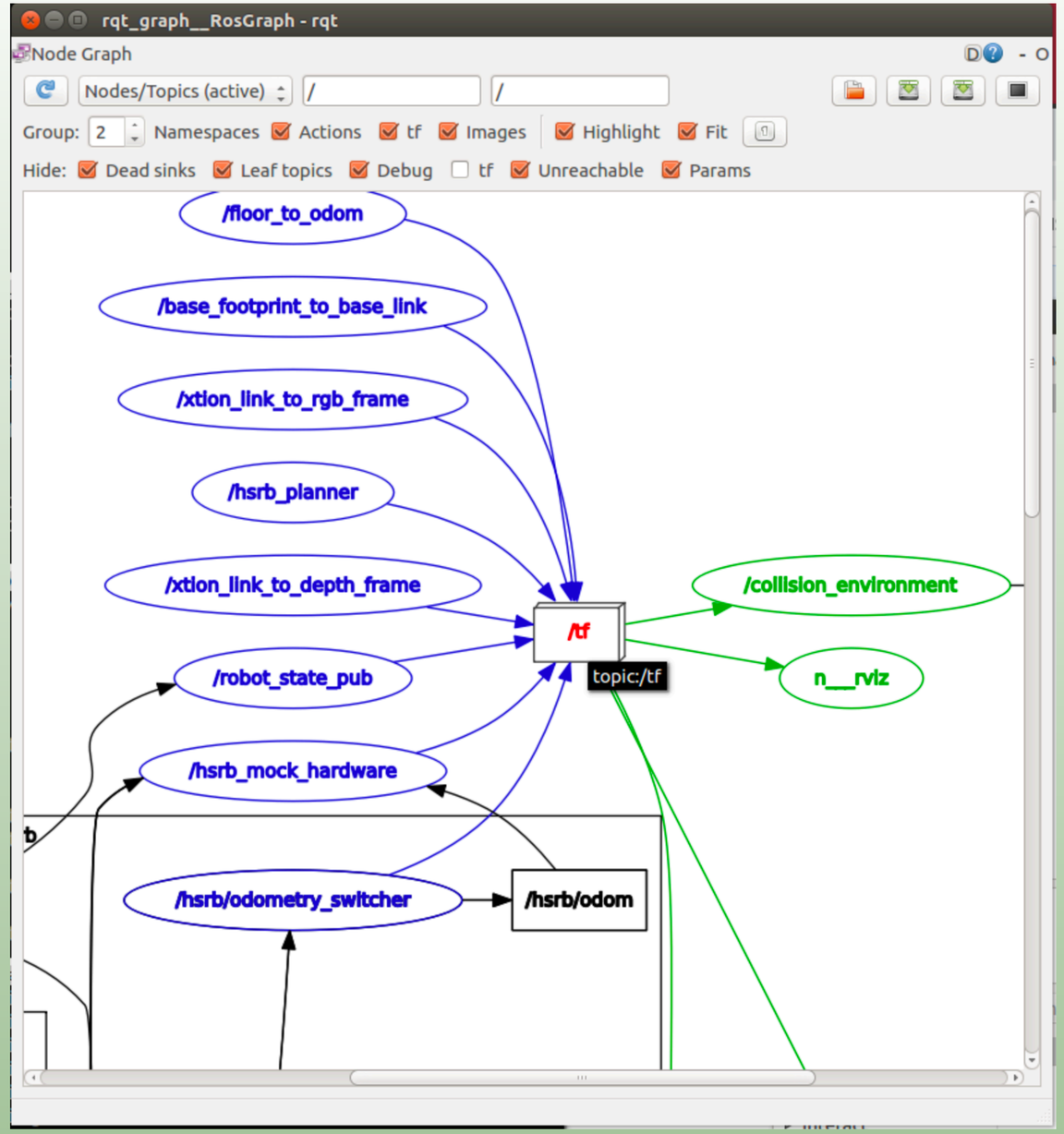

*Details at: http://wiki.ros.org/rqt_multiplot*

# RQT USER INTERFACE RQT_GRAPH

- Visualizing the ROS computation graph

Run rqt_graph

```
~$rosrun rqt_graph rqt_graph
```



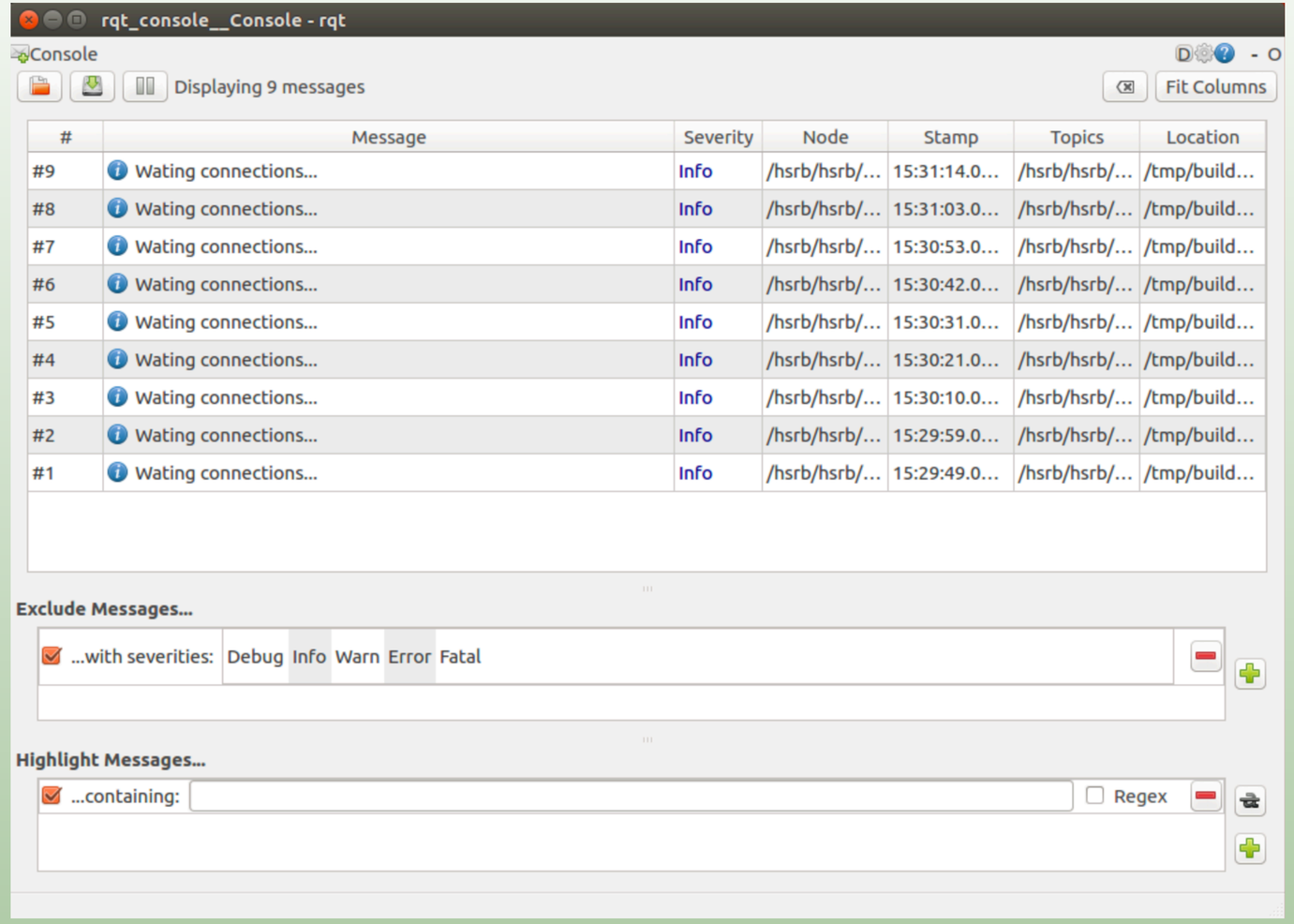

*Details at: http://wiki.ros.org/rqt_graph*

# RQT USER INTERFACE RQT_CONSOLE

- Displaying and filtering ROS messages

Run rqt_console

```
~$rosrun rqt_console rqt_console
```



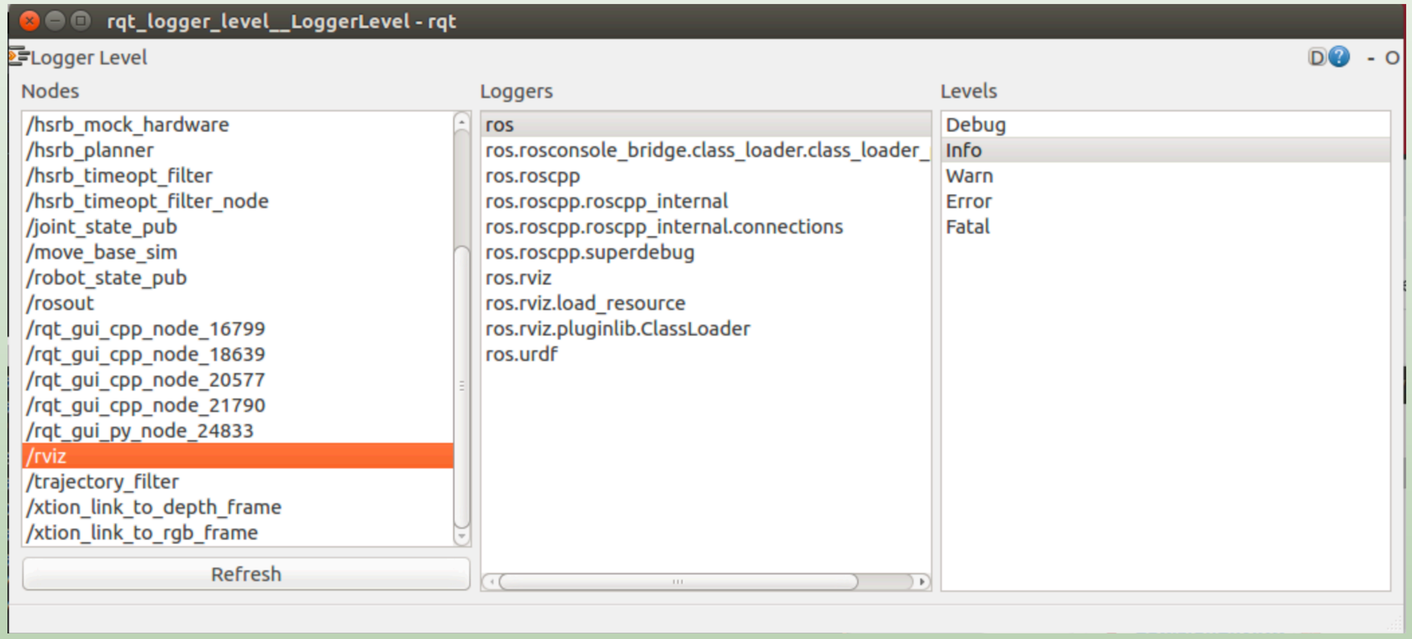

*Details at: http://wiki.ros.org/rqt_console*

# RQT USER INTERFACE RQT_LOGGER_LEVEL

- Configuring the logger level of ROS nodes

Run rqt_logger_level

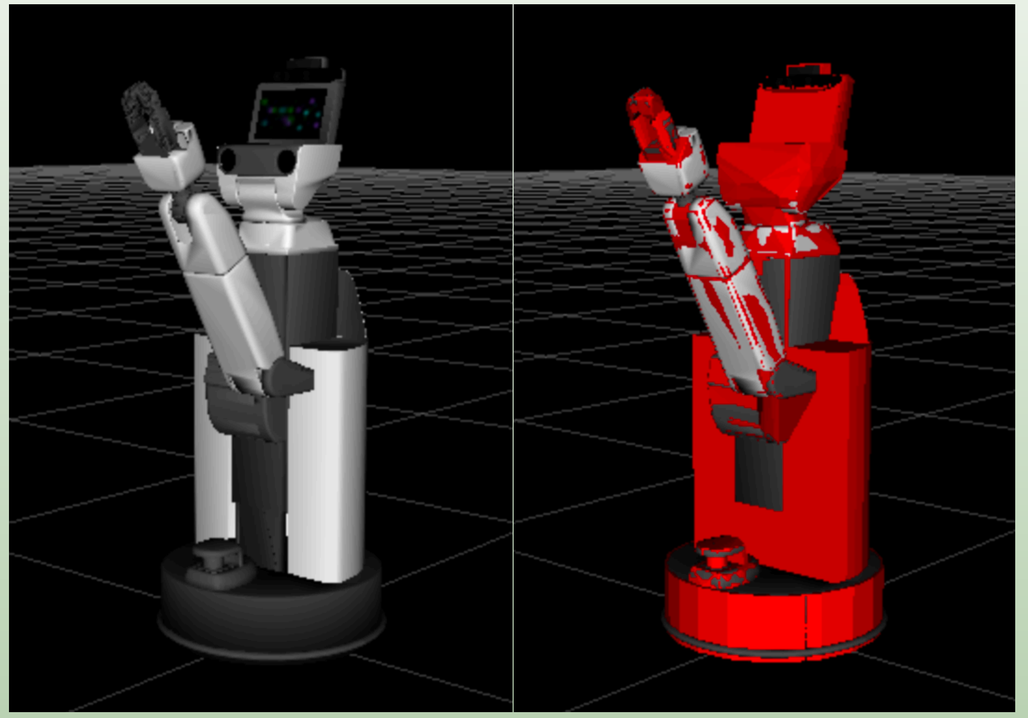
```
~$rosrun rqt_logger_level rqt_logger_level
```




*Details at: http://wiki.ros.org/rqt_logger_level*

# ROBOT MODELS - URDF

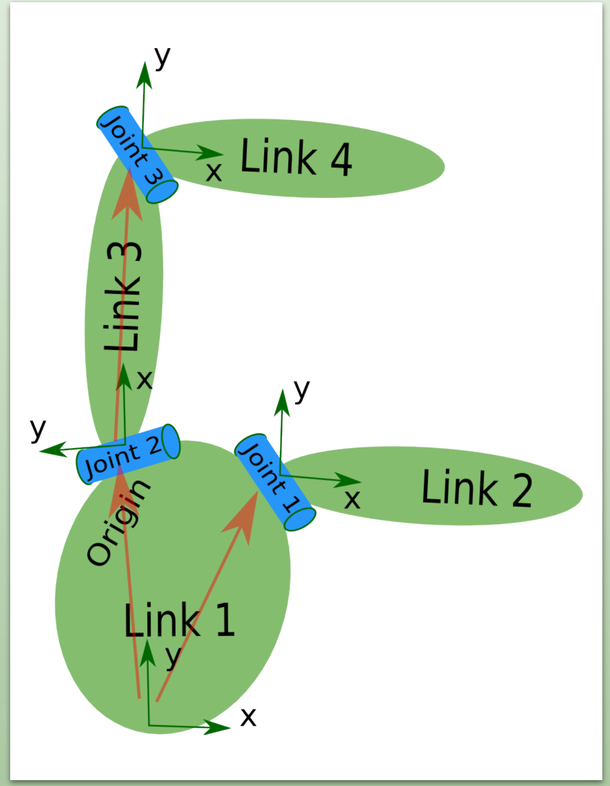
- Unified Robot Description Format (URDF)
- Defines robot model in XML format
  - Kinematic description
  - Dynamic description
  - Visual representation (left figure of HSR)
  - Collision model (right figure of HSR)
- URDF generation can be scripted using XACRO



*Details at:*
*http://wiki.ros.org/urdf, http://wiki.ros.org/xacro*

# ROBOT MODELS - URDF

- Description consists of a set of **link** elements and a set of **joint** elements
- Joints connect the link elements together




```
<robot name="hsr">
  <link> ... </link>
  <link> ... </link>
  <link> ... </link>

  <joint>  ....  </joint>
  <joint>  ....  </joint>
  <joint>  ....  </joint>
</robot>
```

```
<link name="my_link">
  <inertial>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="100"  ixy="0"  ixz="0" iyy="100"
    iyz="0" izz="100" />
  </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>
```

*Details at:*
*http://wiki.ros.org/urdf/XML/model*

# ROBOT MODELS - USAGE IN ROS

- The robot description (URDF) is stored on the parameter server (typically) under **/robot_description**
- You can visualize the robot model in RViz with the RobotModel plugin

husky.urdf.xacro

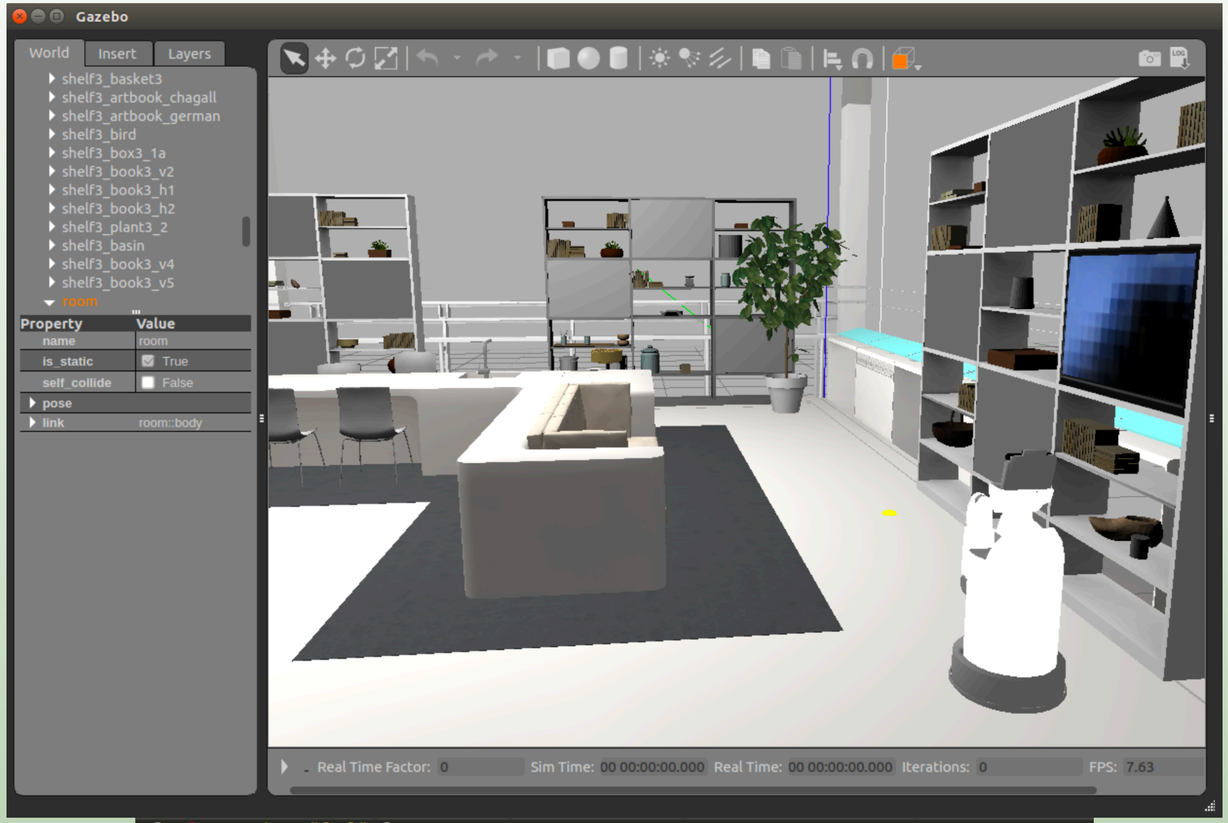
```
<robot name="husky" xmlns:xacro="http://ros.org/wiki/xacro">

  <xacro:arg name="laser_enabled" default="false" />
  <xacro:arg name="laser_xyz" default="$(optenv HUSKY_LMS1XX_XYZ 0.2206 0.0 0.00635)" />
  <xacro:arg name="laser_rpy" default="$(optenv HUSKY_LMS1XX_RPY 0.0 0.0 0.0)" />

  <xacro:arg name="kinect_enabled" default="false" />
  <xacro:arg name="kinect_xyz" default="$(optenv HUSKY_KINECT_XYZ 0 0 0)" />
  <xacro:arg name="kinect_rpy" default="$(optenv HUSKY_KINECT_RPY 0 0.18 3.14)" />

  <xacro:arg name="realsense_enabled" default="false" />
  <xacro:arg name="realsense_xyz" default="$(optenv HUSKY_REALSENSE_XYZ 0 0 0)" />
  <xacro:arg name="realsense_rpy" default="$(optenv HUSKY_REALSENSE_RPY 0 0 0)" />
  <xacro:arg name="realsense_mount" default="$(optenv HUSKY_REALSENSE_MOUNT_FRAME
  sensor_arch_mount_link)" />

  <xacro:property name="husky_front_bumper_extend" value="$(optenv
  HUSKY_FRONT_BUMPER_EXTEND 0)" />
  <xacro:property name="husky_rear_bumper_extend" value="$(optenv HUSKY_REAR_BUMPER_EXTEND
  0)" />

  <xacro:arg name="robot_namespace" default="/" />
  <xacro:arg name="urdf_extras" default="empty.urdf" />
```

spawn_husky.launch

```
<launch>

  <arg name="multimaster" default="false"/>

  <arg name="robot_namespace" default="/"/>

  <arg name="x" default="0.0"/>
  <arg name="y" default="0.0"/>
  <arg name="z" default="0.0"/>
  <arg name="yaw" default="0.0"/>

  <arg name="laser_enabled" default="$(optenv HUSKY_LMS1XX_ENABLED false)"/>
  <arg name="kinect_enabled" default="$(optenv HUSKY_KINECT_ENABLED false)"/>
  <arg name="realsense_enabled" default="$(optenv HUSKY_REALSENSE_ENABLED false)"/>
  <arg name="urdf_extras" default="$(optenv HUSKY_URDF_EXTRAS)"/>

  <group ns="$(arg robot_namespace)">

    <group if="$(arg multimaster)">
      <include file="$(find husky_description)/launch/description.launch" >
        <arg name="robot_namespace" value="$(arg robot_namespace)"/>
        <arg name="laser_enabled" default="$(arg laser_enabled)"/>
        <arg name="kinect_enabled" default="$(arg kinect_enabled)"/>
        <arg name="realsense_enabled" default="$(arg realsense_enabled)"/>
        <arg name="urdf_extras" default="$(arg urdf_extras)"/>
      </include>

      <include file="$(find multimaster_launch)/launch/multimaster_gazebo_robot.launch">
        <arg name="gazebo_interface" value="$(find husky_control)/config/
        gazebo_interface.yaml" />
        <arg name="robot_namespace" value="$(arg robot_namespace)"/>
      </include>

      <!-- For multimaster bringup, need to load the controller config -->
      <rosparam command="load" file="$(find husky_control)/config/control.yaml" />
    </group>
```

# SIMULATION DESCRIPTION FORMAT (SDF)

- Defines the following in XML syntax
  - Environments (incl. gravity, lights etc)
  - Objects (both static and dynamic)
  - Sensors
  - Robots
- SDF is standard for Gazebo
- Gazebo converts a URDF file to SDF automatically




```xml
<?xml version="1.0" ?>
<sdf version="1.4">
  <model name="shelf3_book3_v4">
    <link name="body">
      <pose>0 0 0 0 0 0</pose>
      <inertial>
        <pose>   0.001798   0.085315   0.005927 0 0 0</pose>
        <mass>0.5</mass>
        <inertia>
          <ixx>0.01</ixx>
          <ixy>0</ixy>
          <ixz>0</ixz>
          <iyy>0.01</iyy>
          <iyz>0</iyz>
          <izz>0.01</izz>
        </inertia>
      </inertial>
      <collision name="collision">
```

*Details at:*
*http://sdformat.org/*

# FURTHER REFERENCES

- ROS Wiki
  - http://wiki.ros.org/
- Installation
  - http://wiki.ros.org/ROS/Installation
- Tutorials
  - http://wiki.ros.org/ROS/Tutorials
- Packages
  - https://www.ros.org/browse/list.php
- ROS Cheat Sheet
  - https://www.clearpathrobotics.com/ros-robot-operating-system-cheat-sheet/
  - https://kapeli.com/cheat_sheets/ROS.docset/Contents/Resources/Documents/index
- ROS Best Practices
  - https://github.com/leggedrobotics/ros_best_practices/wiki
- ROS Package Templates
  - https://github.com/leggedrobotics/ros_best_practices/tree/master/ros_package_template

# ACKNOWLEDGEMENT

Material is based on ROS Wiki and ETH Zürich ROS Introduction (https://rsl.ethz.ch/)