RoboCup@Home Practical Course

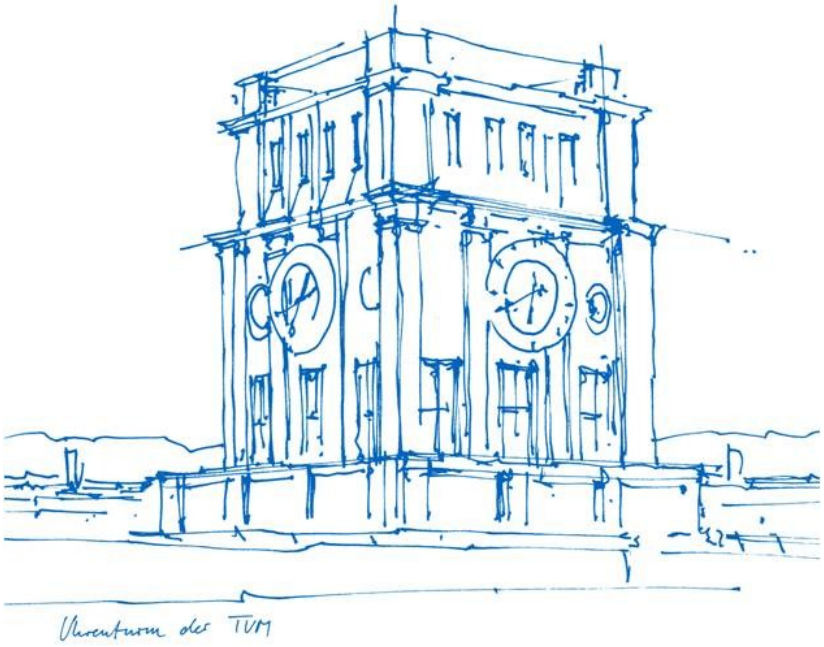
Rogelio Guadarrama

Technical University of Munich

Department of Electrical and Computer Engineering

Chair for Cognitive Systems

Munich, 30 Oct 2019



RoboCup@Home Practical course

WS 2019/20

Tutorials

M. Sc. Rogelio Guadarrama
M. Sc. Constantine Uhde
Prof. Dr. Gordon Cheng

Tutorial 2: Gazebo simulation and robot communication

Objectives for this tutorial

- Learn how to read data from the robot's sensors.
- Learn how to command actions to the robot's actuators.
- Learn how to develop applications for the TIAGo and the HSRB robots.
- Learn how to prepare simulations to test applications before deploying on the real robot.

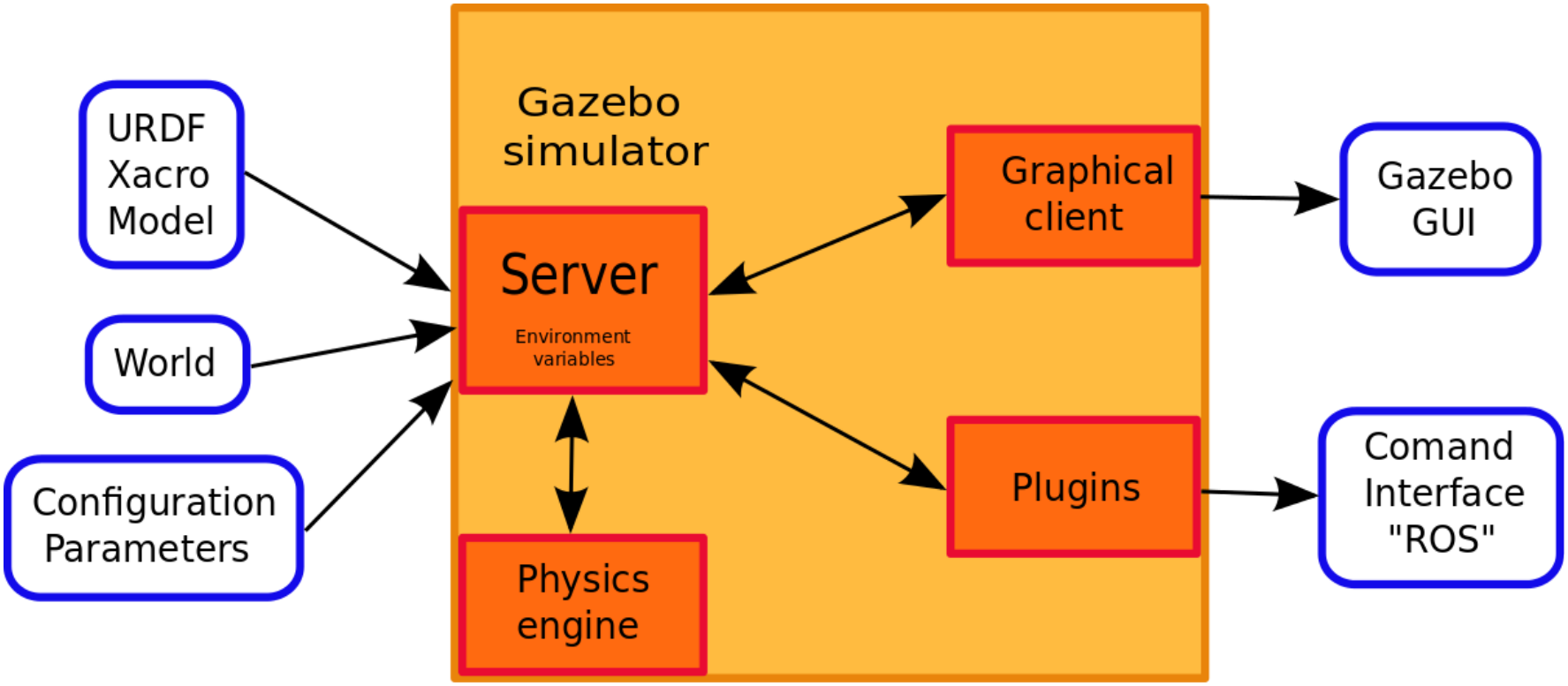
The Gazebo Simulator

With ROS-Kinetic:

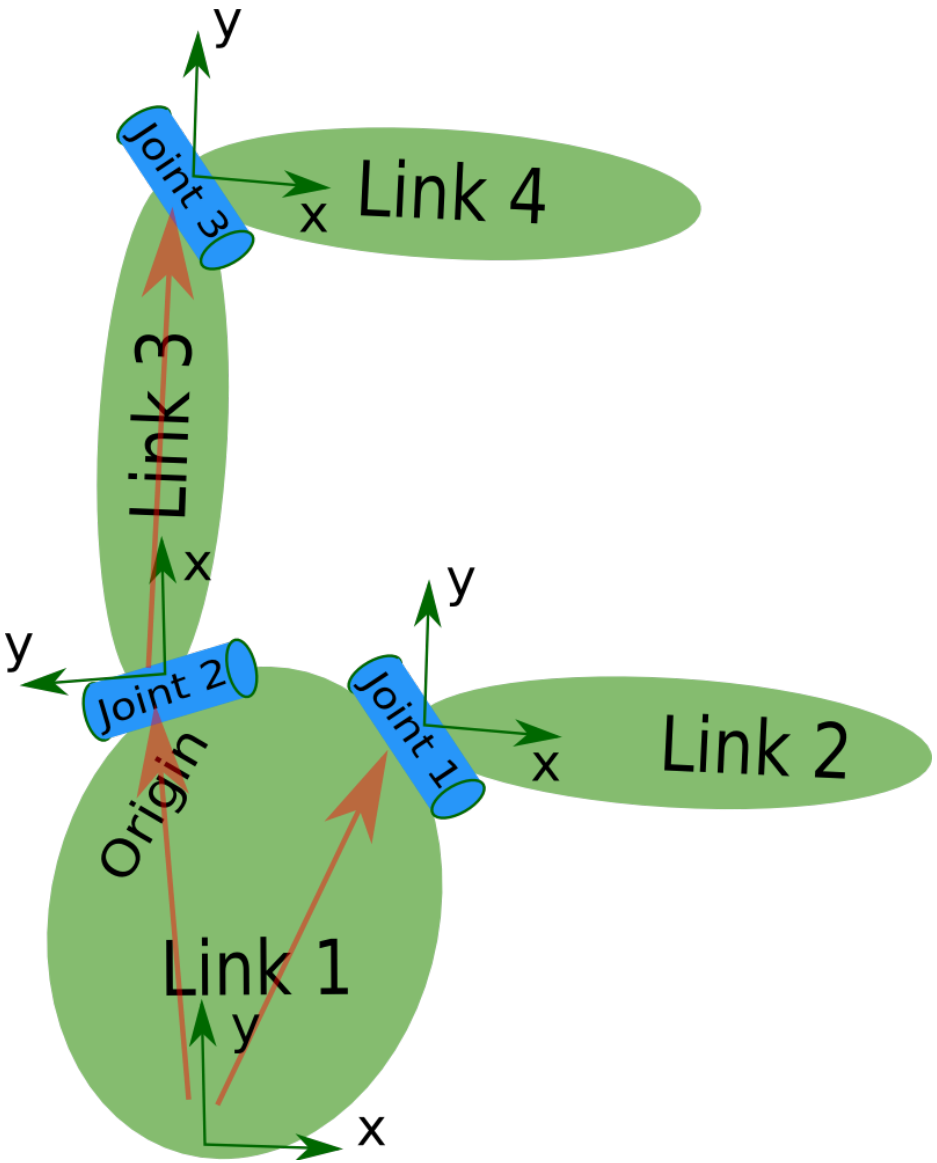
Gazebo multi-robot simulator, version 7.12
Copyright (C) 2012-2014
Open Source Robotics Foundation. Released under the Apache 2 License.

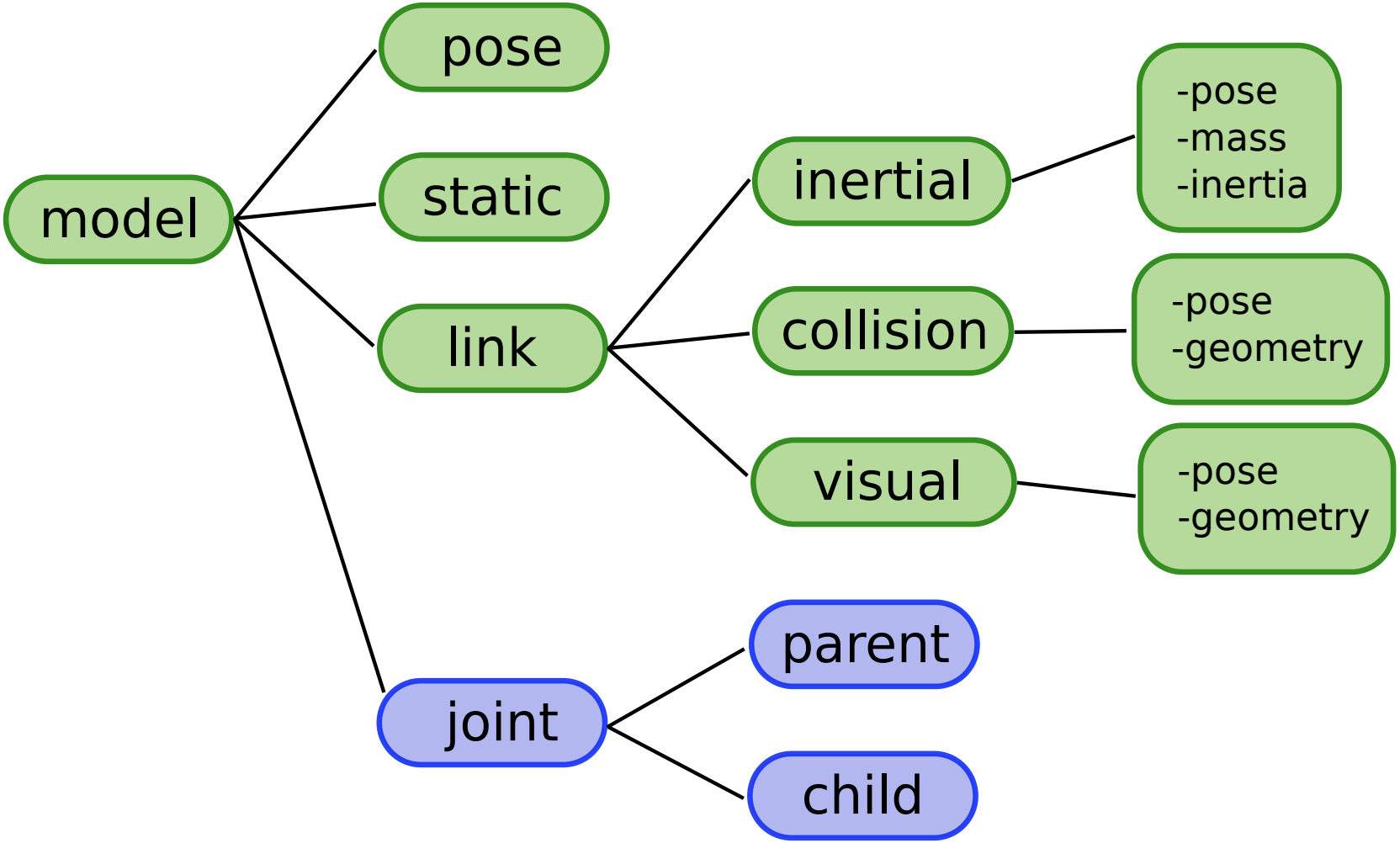
http://gazebosim.org

http://wiki.ros.org/gazebo


URDF
Xacro
Model
World
Configuration
Parameters
Gazebo
simulator
Server
Environment
variables
Physics
engine
Graphical
client
Plugins
Gazebo
GUI
Comand
Interface
"ROS"

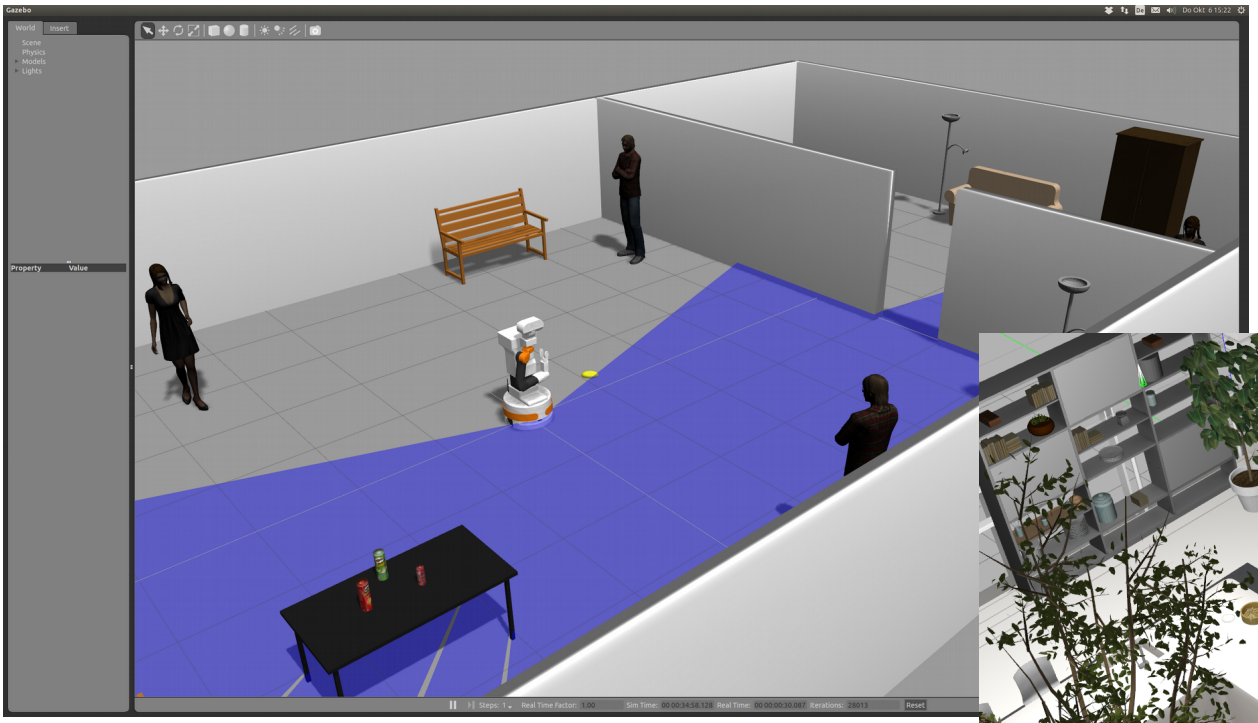
Robot Modeling

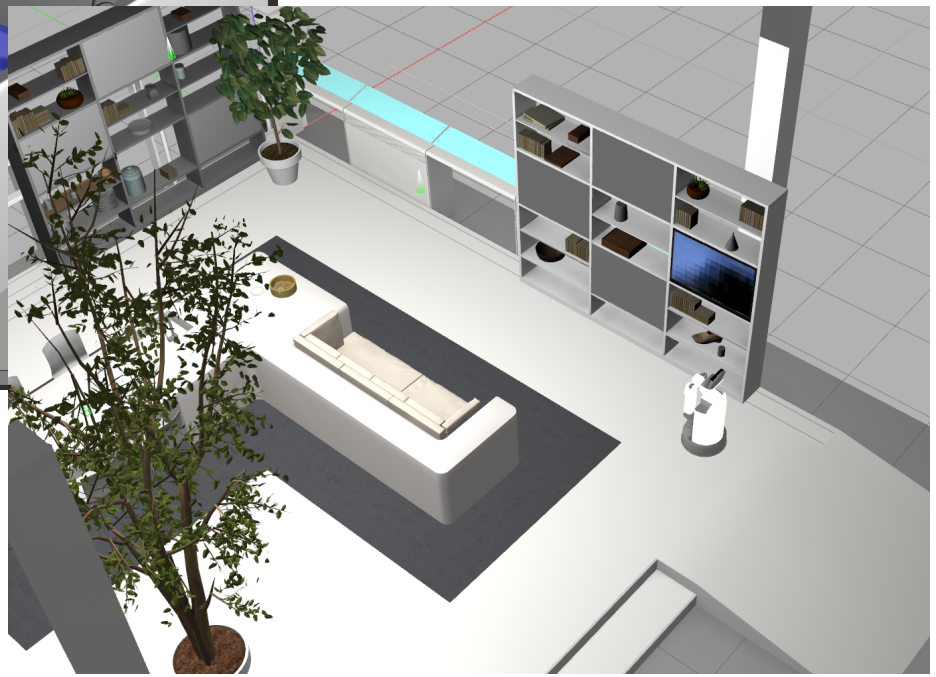




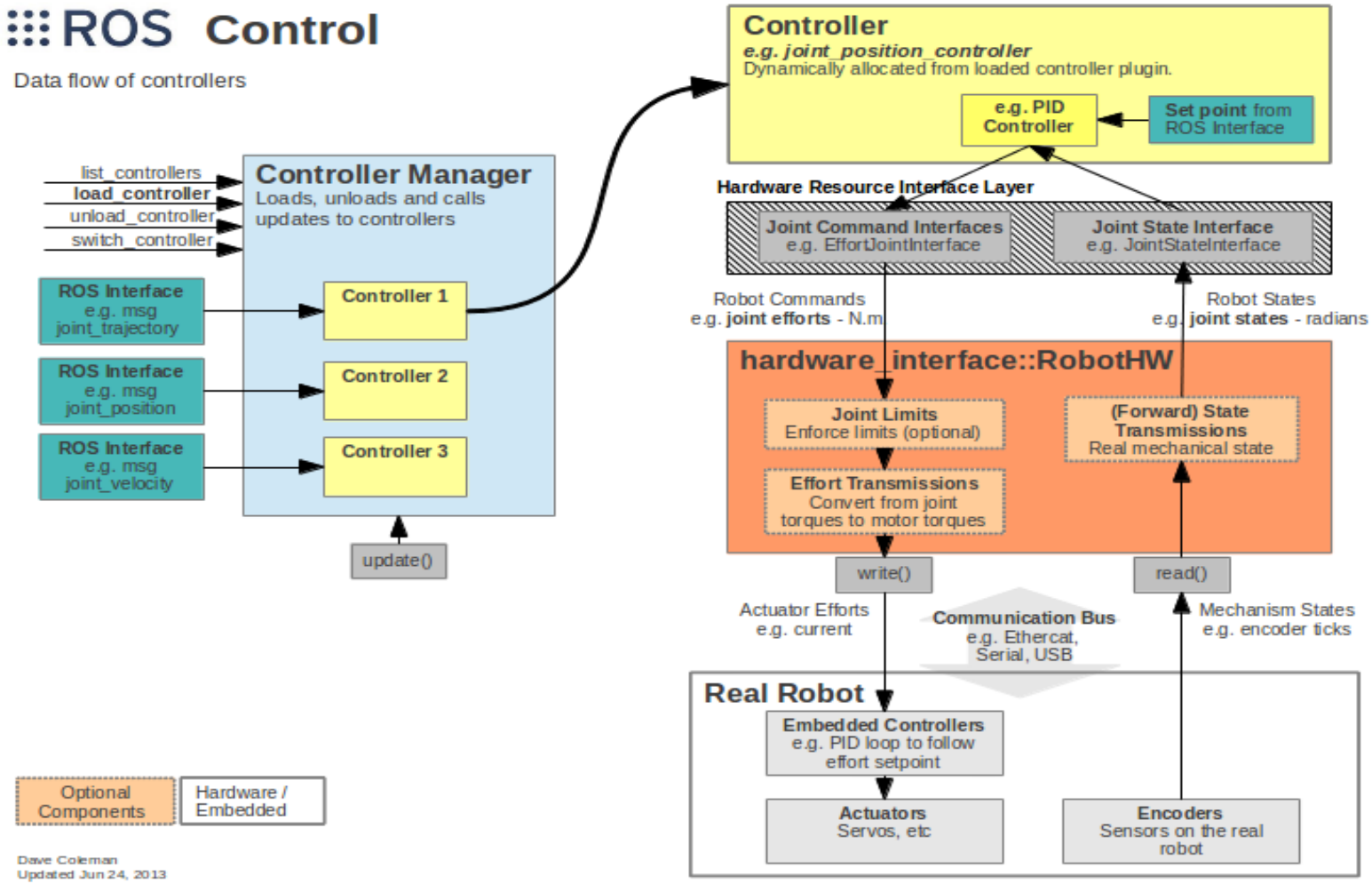
More information in:
http://sdformat.org/spec

Further information on Gazebo can be found in:
http://gazebosim.org/tutorials

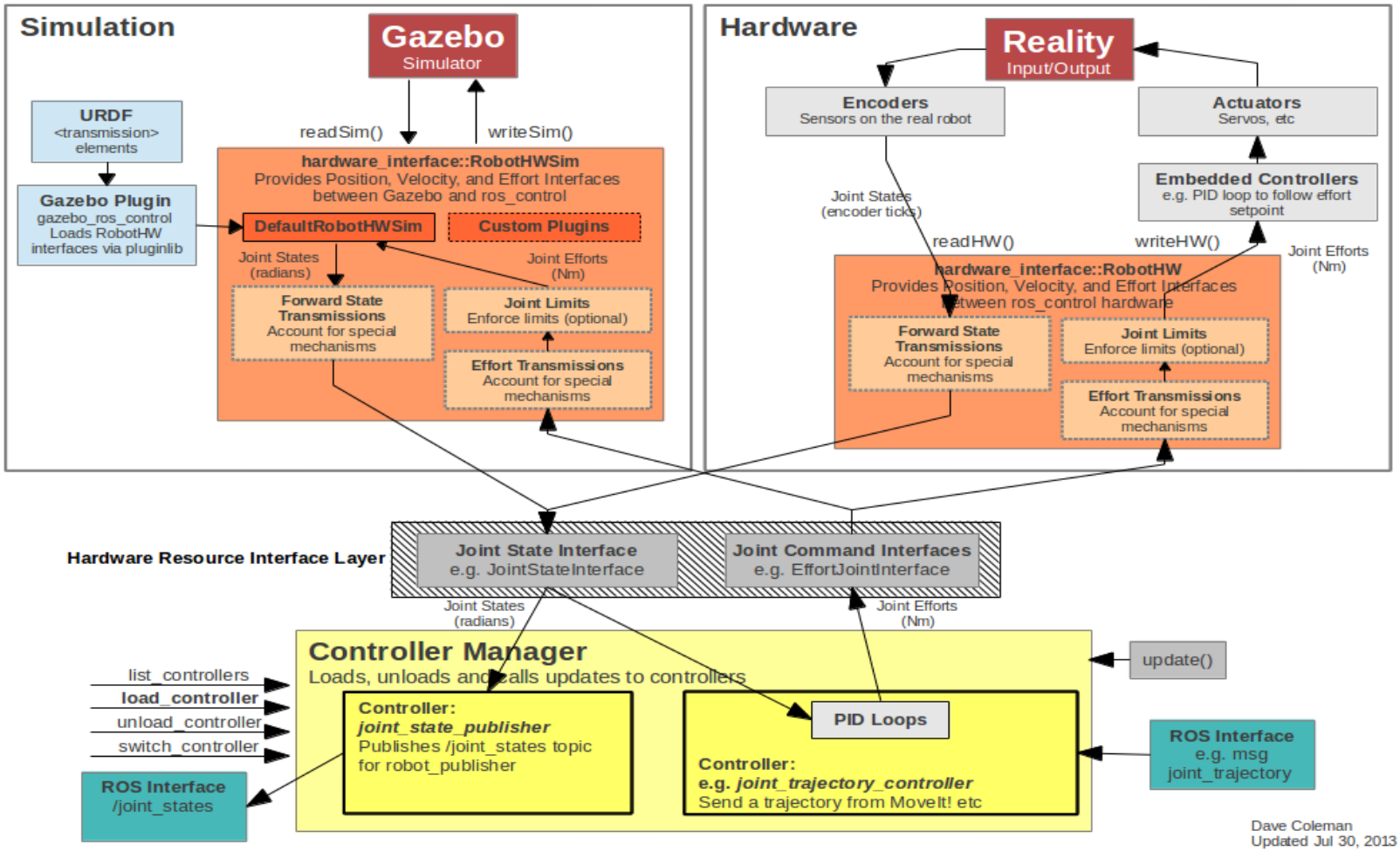




ROS Control


ROS Control
Data flow of controllers
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()
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
Optional Components
Hardware / Embedded
Dave Coleman
Updated Jun 24, 2013

GAZEBO + ROS + ros_control



Preparing the workspaces for this tutorial

Create one workspace for TIAGo.

Create the folder structure.

```
$ mkdir -p ~/ros/worspace/tiago_ws_YOURNAME/src/tiago
$ cd ~/ros/worspace/tiago_ws_YOURNAME/src/tiago
$ rosinstall . tiago_public.rosinstall
$ cd ~/ros/worspace/tiago_ws_YOURNAME
$ catkin_make −DCATKIN_ENABLE_TESTING=0
$ source devel/setup.bash
```

Create one workspace for the HSRB.

```
$ mkdir -p ~/ros/worspace/hsrb_ws_YOURNAME/src
$ cd ~/ros/worspace/hsrb_ws_YOURNAME
$ catkin_make
$ source devel/setup.bash
```

Exercise 1: Prepare a simulation scenario

Follow the steps in sections 3 and 4 of the document to:

- Know the simulation environment.
- Know the tools to handle models.
- Know how to build new models for simulation.

To deliver: The ics_gazebo package containing your world file and the models you build.

Exercise 1: Prepare a simulation scenario

Chose only **one** of the Robocup@Home or World Robot Summit challenges.
You must use the articulated Door model built in the tutorial and at least 20 objects Including furniture and manipulable objects.

Here are some examples:

- **Manipulation and Object Recognition:** The robot must reach a bookcase in which there are 10 objects at different shelves in the bookcase. The robot must then identify and grasp and identity 5 of those objects and put those into a new, easy-to-reach shelve that the team/robot may choose. Optionally, the robot may open a little door or drawer for additional points.

- **Navigation:** The robot must visit a set of way-points while avoiding obstacles on its path and finally follow a person outside the arena. There is a RoboCup@Home arena model on the model server, it consist on a series of rooms made of panels. If you choose this scenario, place a number of objects on it to prepare a navigation test.

- **General Purpose Service Robot:** Some of the tests may be performed in a common-life scenario for humans, a kitchen, a restaurant, a living room or a store. Prepare one of these possible scenarios including furniture and objects to handle.

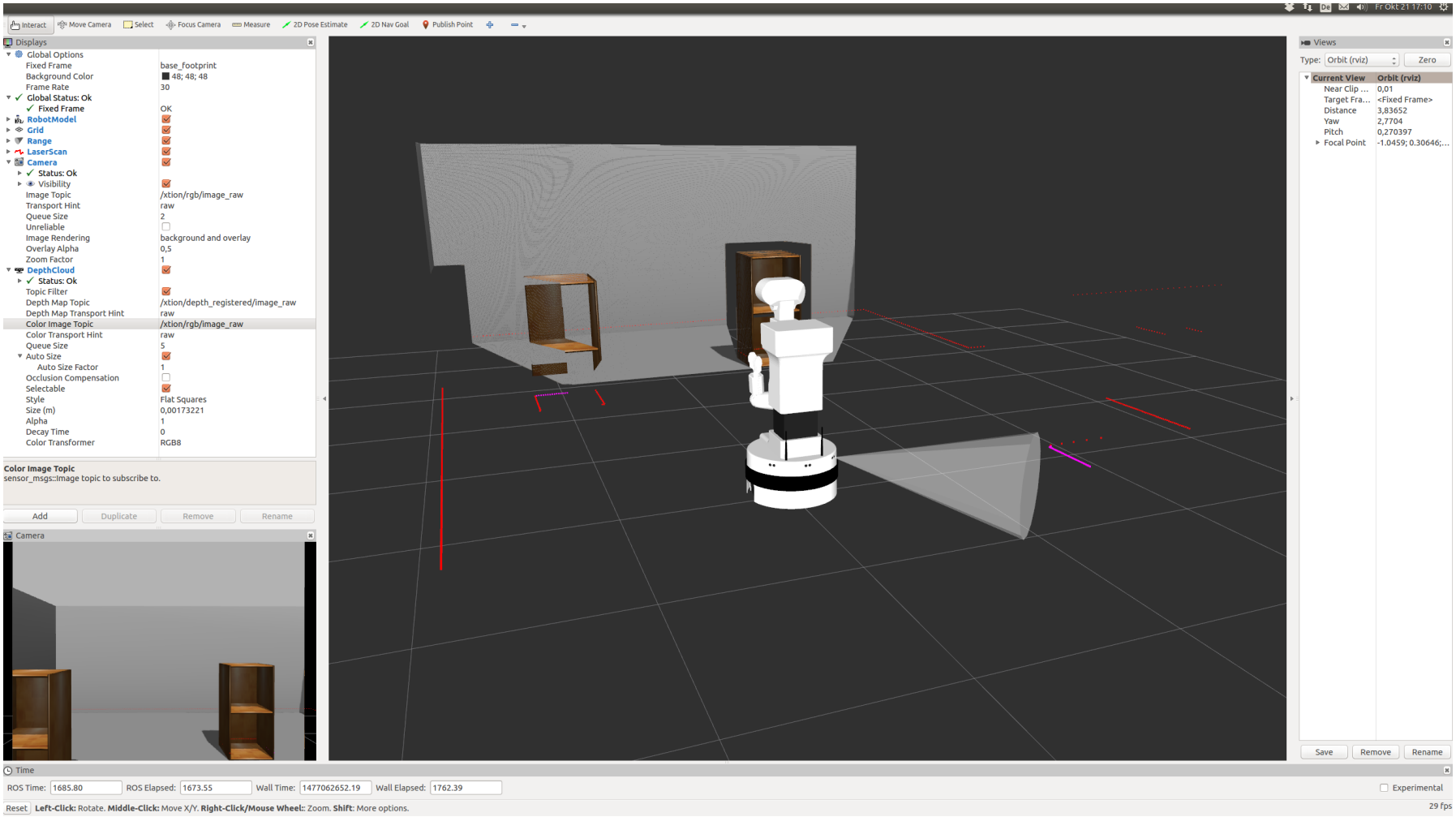
Exercise 2: Robot in rviz

Use the steps in section 5.1 of the document to load the model in rviz.

- Know the ROS Topics used to receive information from the robot.
- Load the robot in rviz and save the configuration.

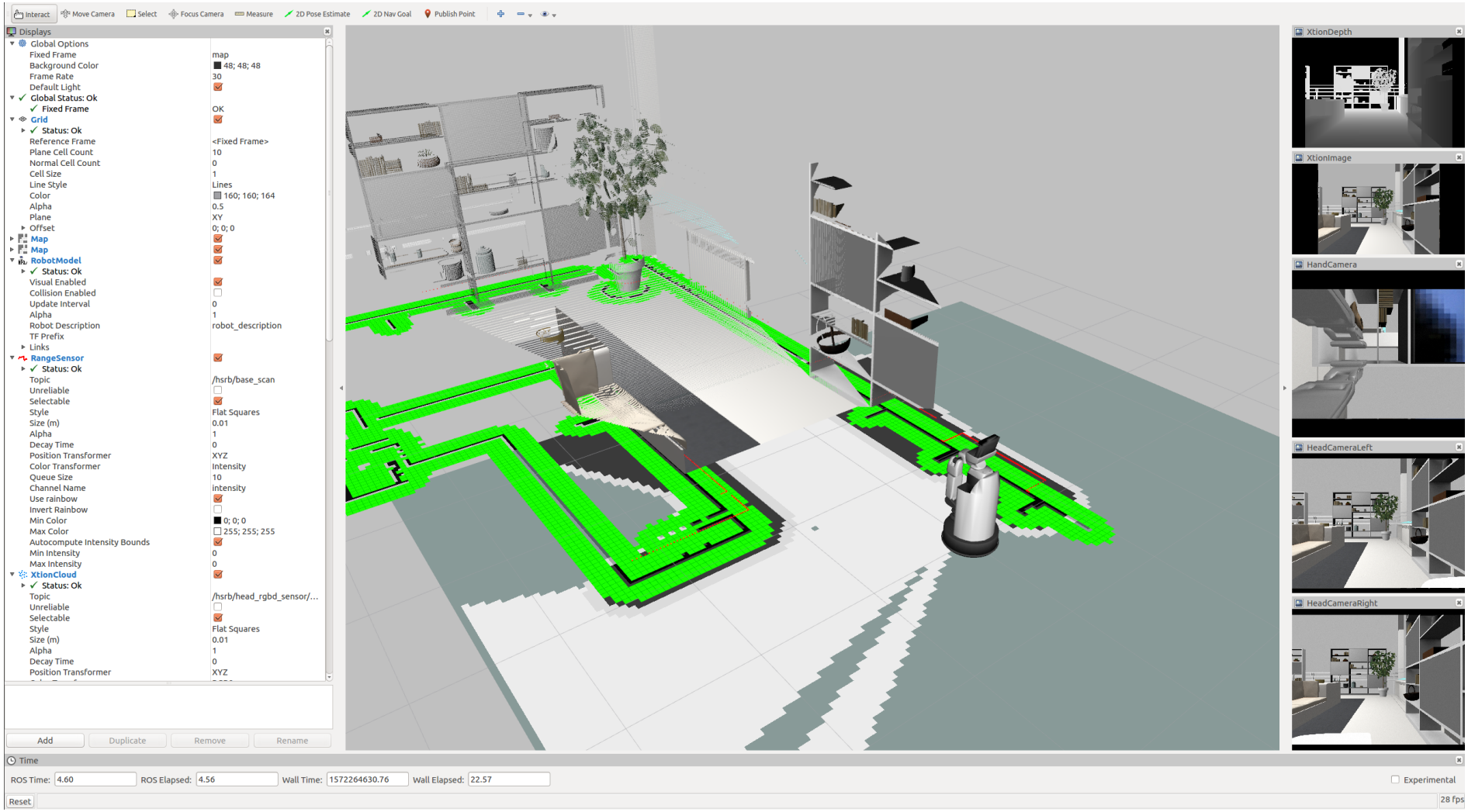
To deliver: The ics_gazebo package containing your ROBOT.rviz file. (name the file tiago.rviz or hsrb.rviz)

Exercise 2: Robot in rviz



To deliver: The ics_gazebo package containing your ROBOT.rviz file. (name the file tiago.rviz or hsrb.rviz)

Exercise 2: Robot in rviz



To deliver: The ics_gazebo package containing your ROBOT.rviz file. (name the file tiago.rviz or hsrb.rviz)

Exercise 3: Default controllers

Use the steps in section 5.2 of the document to command a robot using the default controllers.

- Know the ROS Topics used to send commands to the robot.
- Know the controller_manager package.
- Control the robot by publishing to a topic.
- Implement a simple control law for the base.

Exercise 3: Default controllers

Chose one of the robots to be controlled.

Adapt the turtle_viz package from Tutorial 1 to control the position of the mobile base of the robot. (Only modify the turtle class and the control node)

HSRB robot

If you choose the HSRB robot, the control design shall be the same as in Tutorial 1.

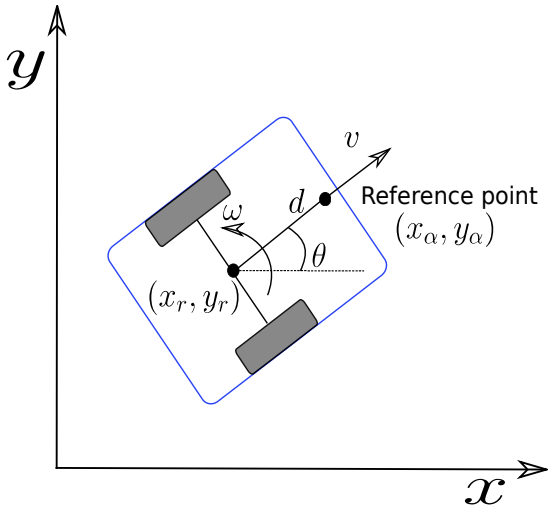
Modify the turtle_vis package to read the real robot position and to command the mobile base.

To deliver: The modified turtle_viz package and instructions to run the code.

Exercise 3: Default controllers

TIAGo robot

Tiago has a differential mobile base (non holonomic). Therefore, the controller must be adapted to such architecture.



$$\begin{bmatrix} \dot{x}_\alpha \\ \dot{y}_\alpha \end{bmatrix} = \begin{bmatrix} \cos\theta & -d\sin\theta \\ \sin\theta & d\cos\theta \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$X = \begin{bmatrix} x_\alpha \\ y_\alpha \end{bmatrix} = \begin{bmatrix} x_r + d\cos\theta \\ y_r + d\sin\theta \end{bmatrix} \qquad T = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$e = X_d - X \qquad \dot{X}_d = Ke \qquad K \in \mathbb{R}^2$$

$$T = \begin{bmatrix} \cos\theta & -d\sin\theta \\ \sin\theta & d\cos\theta \end{bmatrix}^{-1} \dot{X}_d$$

To deliver: The modified turtle_viz package and instructions to run the code.

Exercise 4: Create a controller plugin

Use the steps in section 5.3 to create a new controller plugin for the torso joint

- Know the controller base class.
- Know all the files needed to create a new controller.
- Know how to create a new controller plugin.

Exercise 4: Create a controller plugin

Adapt the files to on the controllers_tutorials package on the template to create a new controller for the torso joint.

To deliver: The modified controllers_tutorials package and instructions to run the code.

What to deliver?

One compressed folder named
“Name_lastName_roboCupHome_tutorial2”

Containing inside the modified packages:

ics_gazebo, turtle_viz, controllers_tutorials

and a read-me file containing the instructions to compile and run your code.

Example

```
Readme.txt

%%%%%%%%%%%%%%%%%%%%%%%%%% Exercise 1 %%%%%%%%%%%%%%%%%%%%%%%%%%

Robot: HSRB

Packages: ics_gazebo

Compilation:

$ catkin_make

Execution:

$ roslaunch ics_gazebo hsrb.launch world_suffix:=tutoroal2


%%%%%%%%%%%%%%%%%%%%%%%%%% Exercise 2 %%%%%%%%%%%%%%%%%%%%%%%%%%

Robot: HSRB

Packages: ics_gazebo
.
.
.
.
```

Important Remarks

- **DO NOT** copy all the Workspace into the zip file.
- Copy only the packages needed for the solution.
- Be clear when writing your read me file.
- Specify which robot you used in the first line of every section.
- Include the list of the packages needed for every solution.
- Include all the commands to compile, execute and operate your solutions.
- Follow the naming conventions specified in this Document.
- Make sure that your solution compiles and runs in a single try.
- Deliver your solution before the deadline.

Important Remarks

- **DO NOT** copy all the Workspace into the zip file.
- Copy only the packages needed for the solution.

Thursday **5th of November** before 23:59 hrs.

utions.

What about other robots?

Other robots work with similar frameworks. Check the documentation of PR2 robot in (Needed for next tutorial):

- http://wiki.ros.org/Robots/PR2
- http://wiki.ros.org/pr2_simulator/Tutorials

Enjoy the week !!