RoboCup@Home Practical Course

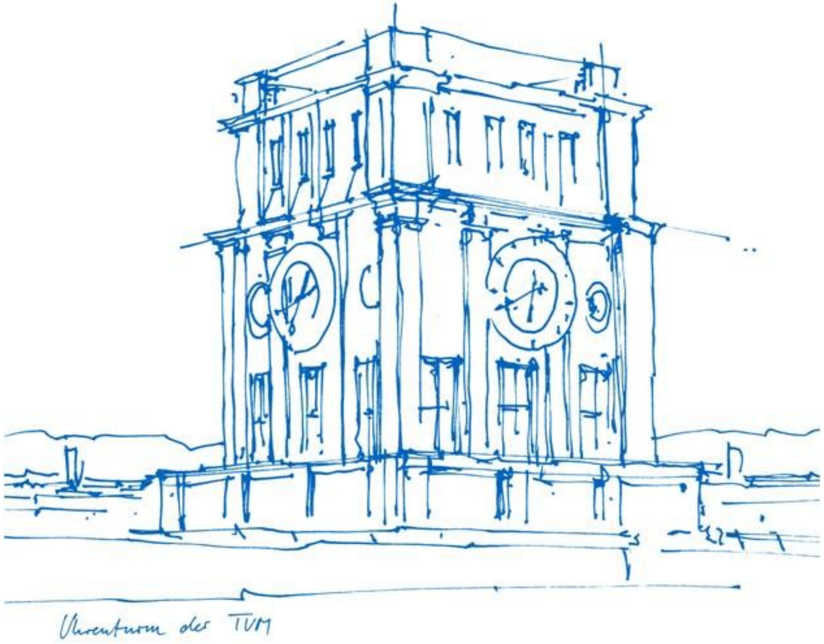
Rogelio Guadarrama

Technical University of Munich

Department of Electrical and Computer Engineering

Chair for Cognitive Systems

Munich, 31 October 2018



RoboCup@Home Practical course

WS 2018

Tutorials

Dr. Karinne Ramirez-Amaro

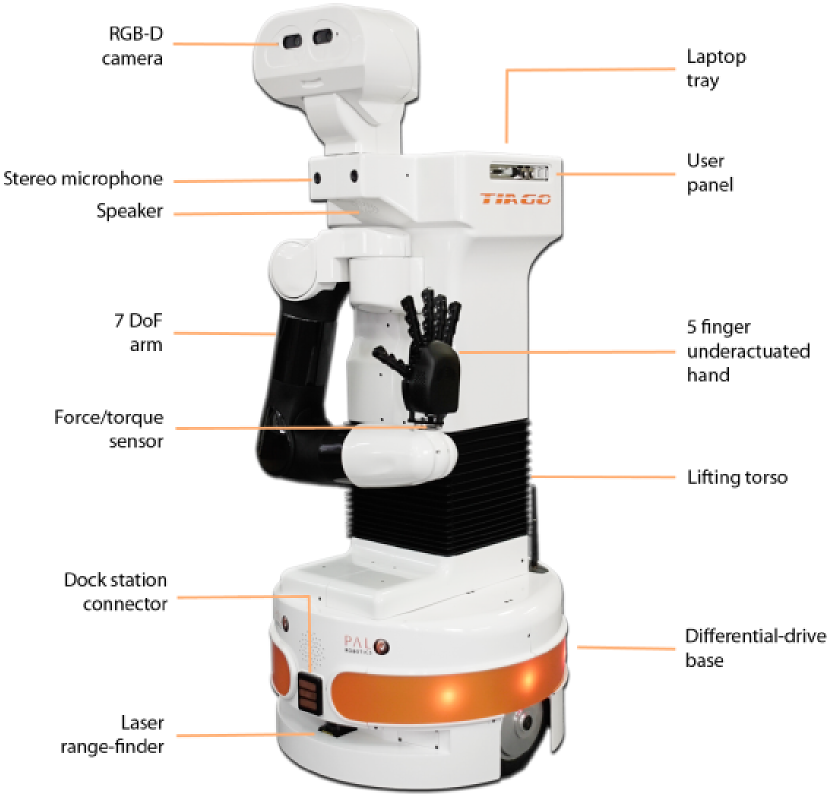
Dr. Emmanuel Dean

Dr. Pablo Lanillos

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

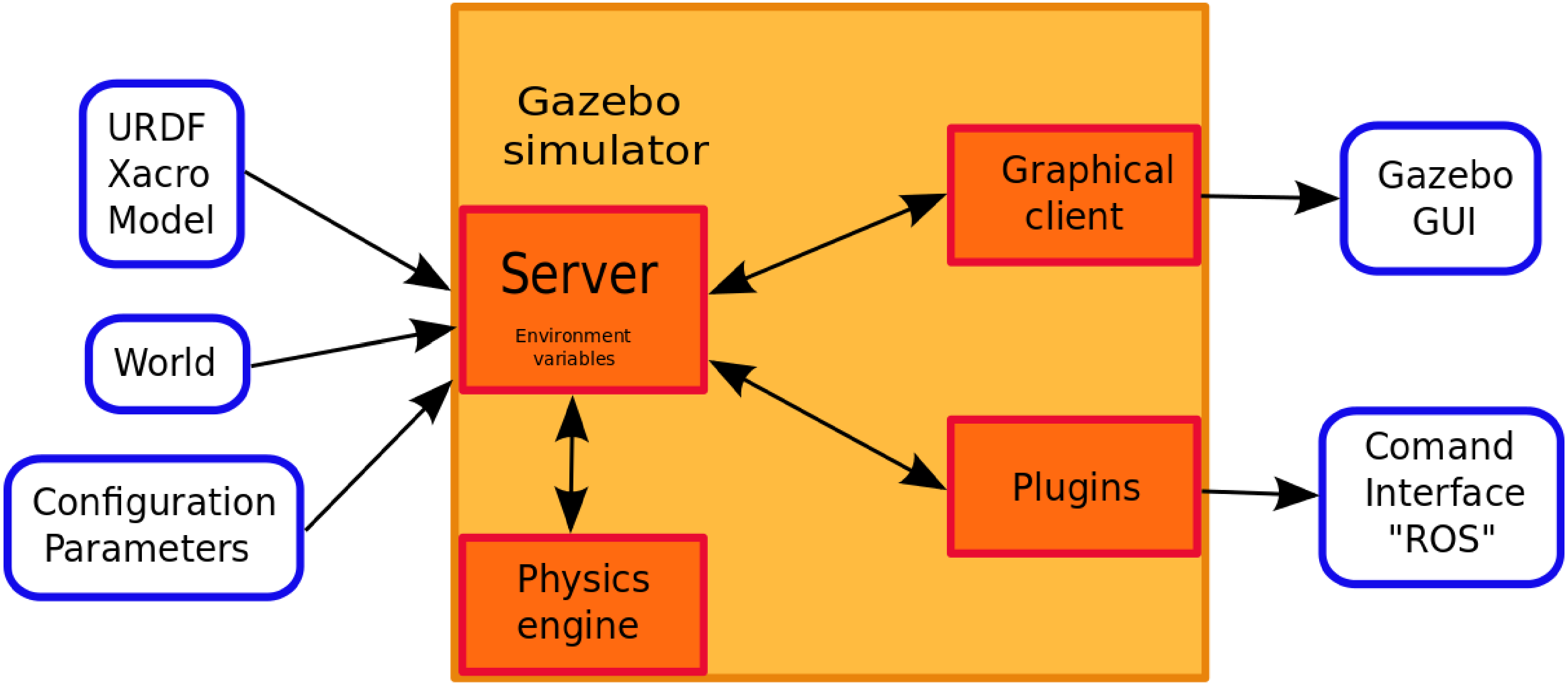
The Gazebo Simulator

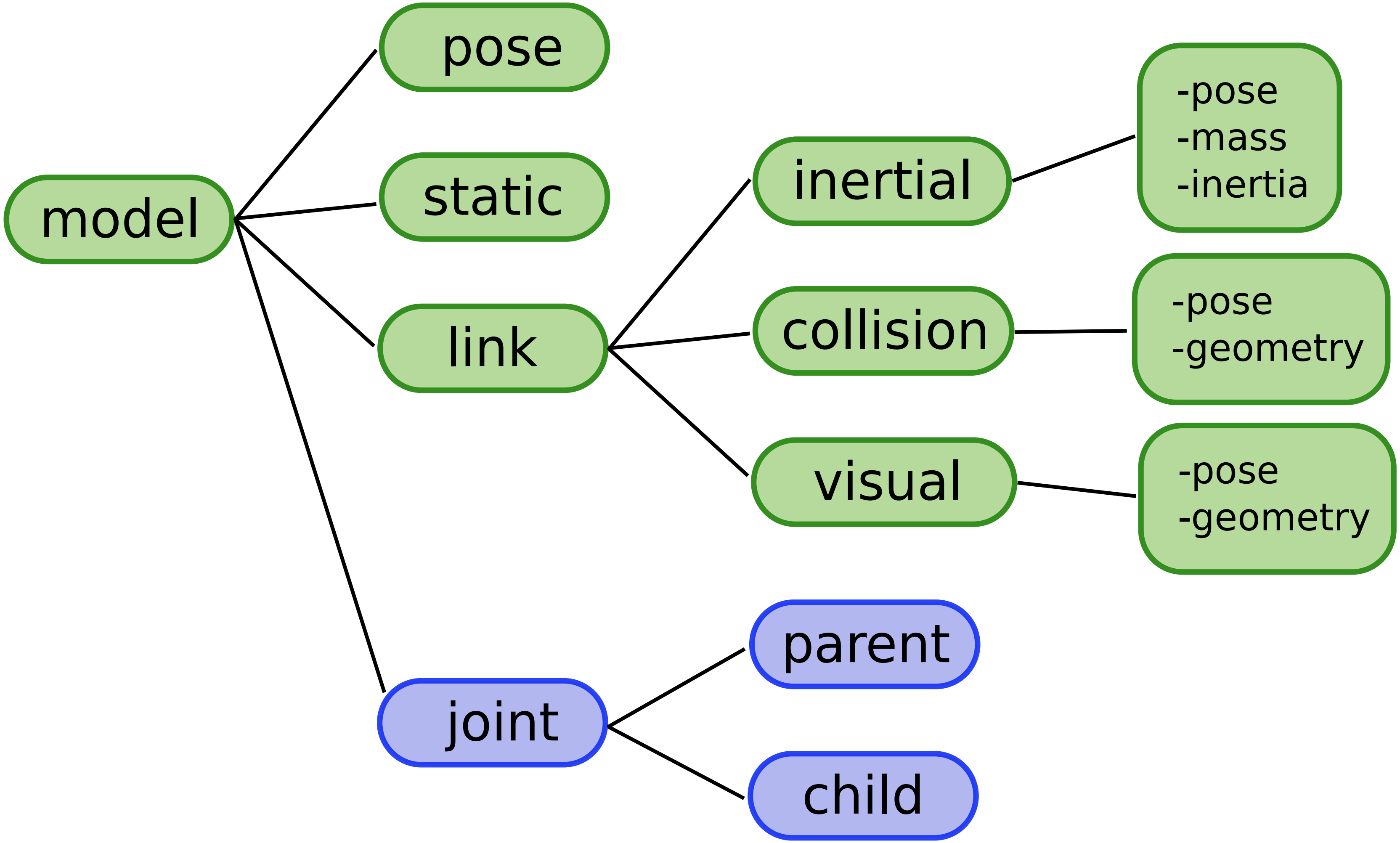
With ROS-Kinetic:

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

http://gazebosim.org

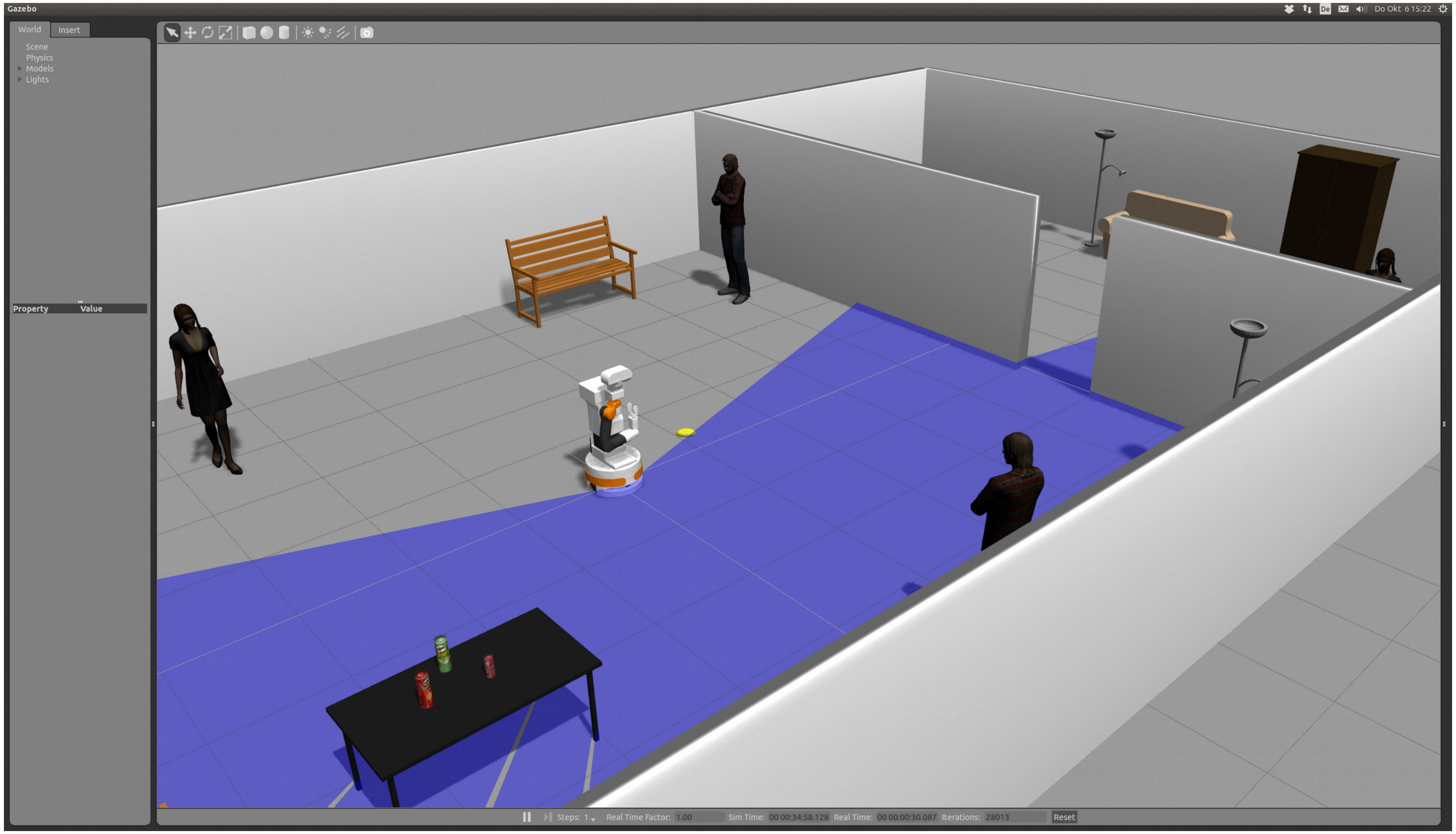
http://wiki.ros.org/gazebo


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



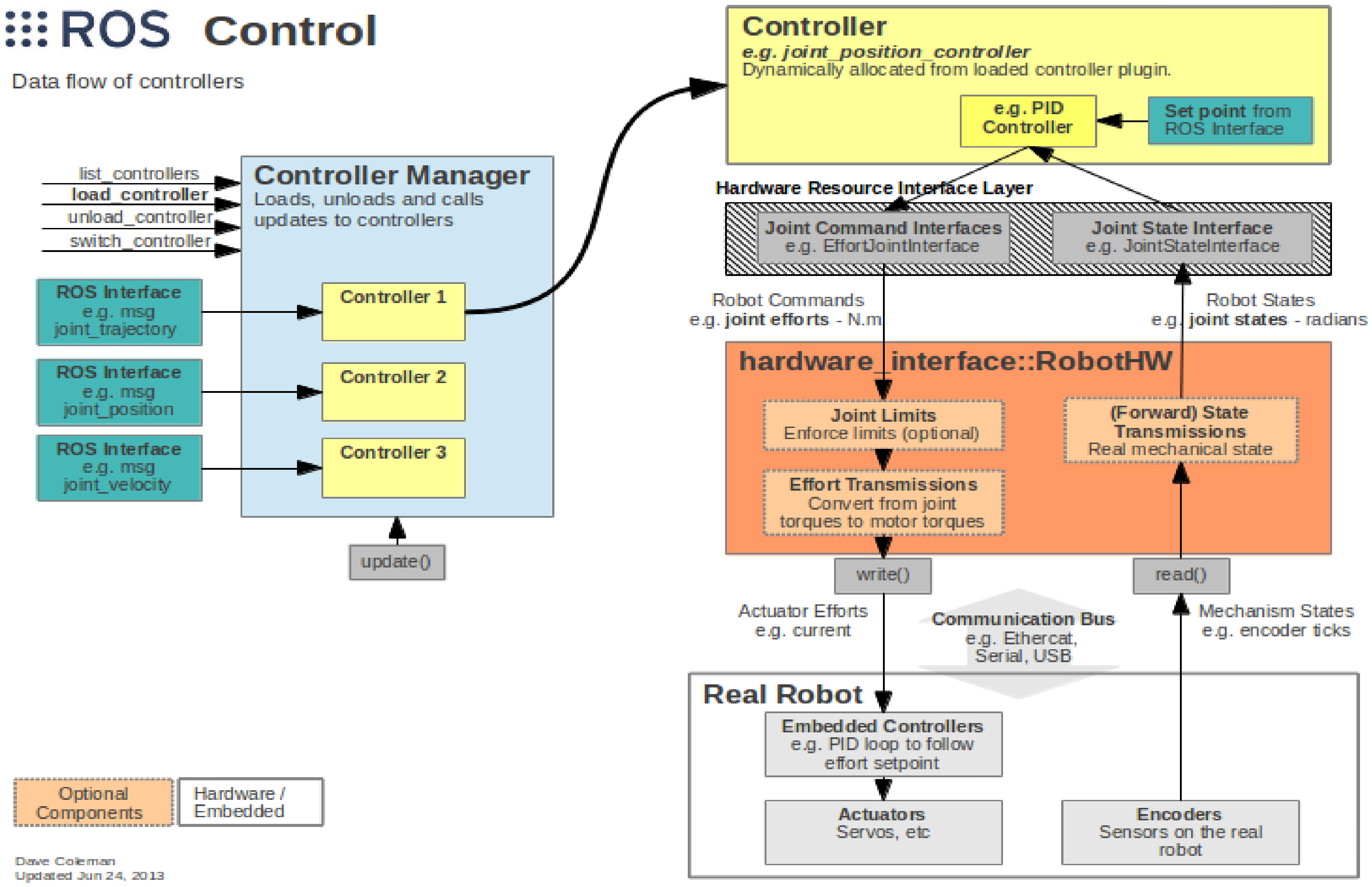
More information in:
http://sdformat.org/spec

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

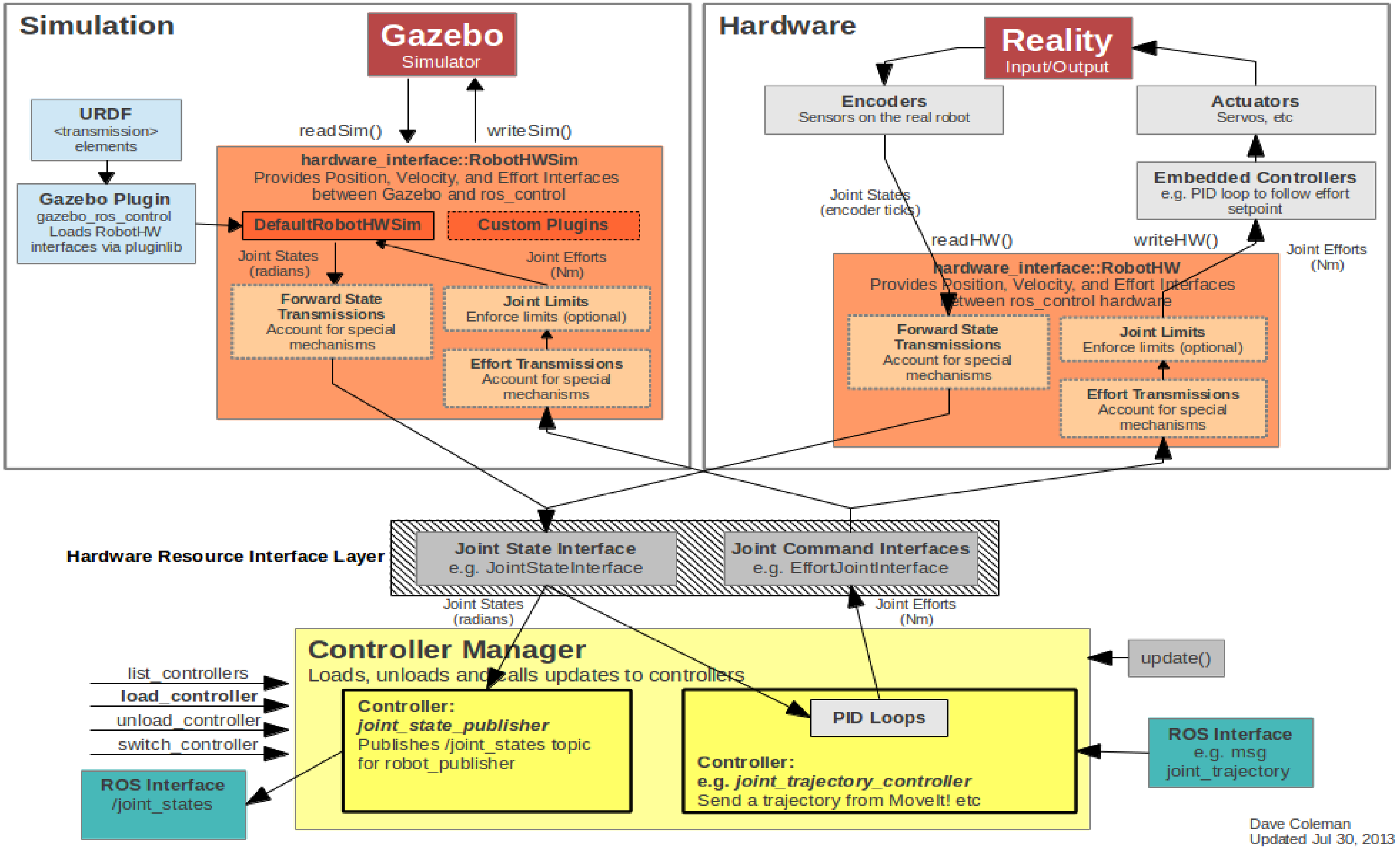


Steps: 1 Real Time Factor: 1.00 Sim Time: 00 00:34:58.128 Real Time: 00 00:00:30.087 Iterations: 28013

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



Simulation

Gazebo
Simulator

readSim()

writeSim()

URDF
<transmission>
elements

Gazebo Plugin
gazebo_ros_control
Loads RobotHW
interfaces via pluginlib

hardware_interface::RobotHWSim
Provides Position, Velocity, and Effort Interfaces
between Gazebo and ros_control

DefaultRobotHWSim

Custom Plugins

Joint States
(radians)

Joint Efforts
(Nm)

Forward State
Transmissions
Account for special
mechanisms

Joint Limits
Enforce limits (optional)

Effort Transmissions
Account for special
mechanisms

Hardware

Reality
Input/Output

Encoders
Sensors on the real robot

Actuators
Servos, etc

Embedded Controllers
e.g. PID loop to follow effort
setpoint

Joint States
(encoder ticks)

readHW()

writeHW()

Joint Efforts
(Nm)

hardware_interface::RobotHW
Provides Position, Velocity, and Effort Interfaces
between ros_control hardware

Forward State
Transmissions
Account for special
mechanisms

Joint Limits
Enforce limits (optional)

Effort Transmissions
Account for special
mechanisms

Hardware Resource Interface Layer

Joint State Interface
e.g. JointStateInterface

Joint Command Interfaces
e.g. EffortJointInterface

Joint States
(radians)

Joint Efforts
(Nm)

Controller Manager
Loads, unloads and calls updates to controllers

list_controllers

load_controller

unload_controller

switch_controller

update()

Controller:
joint_state_publisher
Publishes /joint_states topic
for robot_publisher

PID Loops

Controller:
e.g. *joint_trajectory_controller*
Send a trajectory from MoveIt! etc

ROS Interface
/joint_states

ROS Interface
e.g. msg
joint_trajectory

Dave Coleman
Updated Jul 30, 2013

Preparing the workspace for TIAGo robot

Detailed instructions in http://wiki.ros.org/Robots/TIAGo/Tutorials

Create the folder structure.

```
$ mkdir -p ~/ros/worspace/tiago_ws/src
```

Copy the tiago_public.rosinstall file into the folder

~/ros/worspace/tiago_ws

Preparing the workspace for TIAGo robot

Now install the packages for the new workspace

```
$ cd ~/ros/worspace/tiago_ws

$ rosinstall src tiago_public.rosinstall
```

Then compile the workspace

```
$ cd ~/ros/worspace/roboCupHome_tutorial_YOURNAME/

$ source devel/setup.bash
```

Use the -DCATKIN_ENABLE_TESTING=0 flag the first time you compile it !!!

```
$ catkin_make -DCATKIN_ENABLE_TESTING=0
```

Preparing the workspace for TIAGo robot

Once the compiler reaches a 100%, test the installation:

```
$ roslaunch tiago_gazebo tiago_gazebo.launch

public_sim:=true robot:=steel world:=tutorial_office
```

Exercise 1: Prepare a simulation scenario

Use 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 Tutorial_NAME.world file and all the needed models to use it.

Exercise 1: Prepare a simulation scenario

Chose only **one** of these options:

- **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: TIAGo 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 TIAGo in rviz and save the configuration.

To deliver: The TIAGo.rviz file.

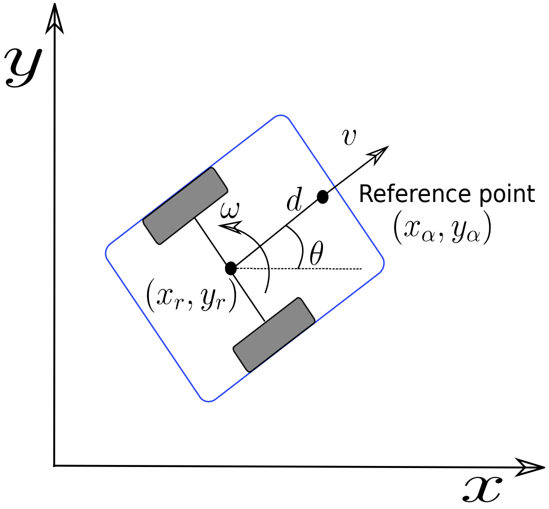
Exercise 3: Default controllers

Use the steps in section 5.2 of the document to command the TIAGo 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

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



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

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

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

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

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 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 4 folders named T2_E1, T2_E2, T2_E3 and T2_E4

These folders must contain:

T2_E1: A world file for gazebo and model folders if needed.

T2_E2: A rviz configuration file.

T2_E3: The modified turtle_viz package and instructions to run.

T2_E4: The controllers_tutorials package modified.

NOTE: Be clear and precise in your instructions and HowTos to
run your programs.

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 !!