ROS INTRODUCTION FOR SCHUNK COMPONENTS

Christopher Parlitz
Schunk Modular Robotics

© Fraunhofer IPA

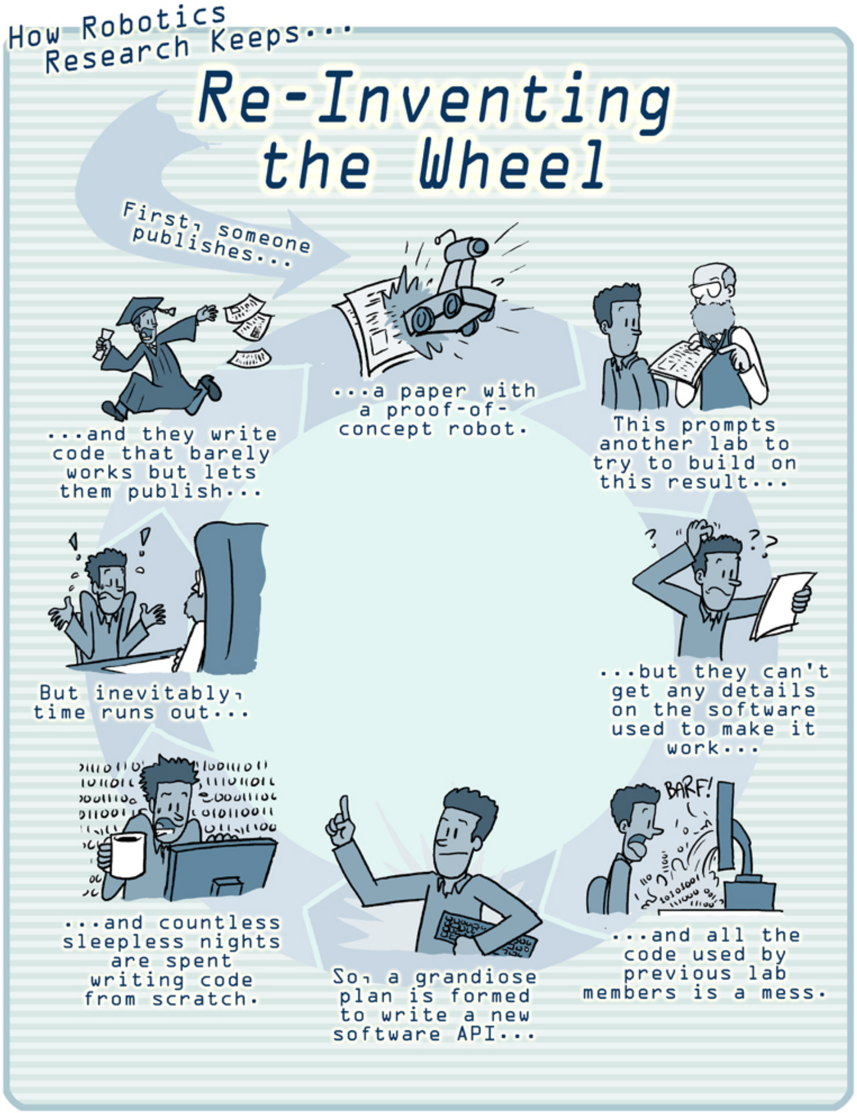
Introduction to ROS

© Fraunhofer IPA

Research in robotics

- Reinvention of the Wheel
- Little Commonality
- Short Lifespan
- Inability to Compare Results

→ ROS addresses these


How Robotics Research Keeps...
Re-Inventing the Wheel
First, someone publishes...
...a paper with a proof-of-concept robot.
This prompts another lab to try to build on this result...
...but they can't get any details on the software used to make it work...
BARF!
...and all the code used by previous lab members is a mess.
So, a grandiose plan is formed to write a new software API...
...and countless sleepless nights are spent writing code from scratch.
But inevitably, time runs out...
...and they write code that barely works but lets them publish...

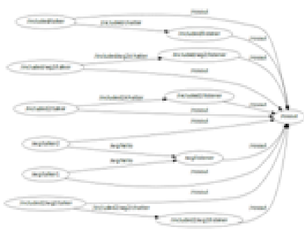
© Fraunhofer IPA

ROS – Robot Operating System

- ROS = **R**obot **O**perating **S**ystem

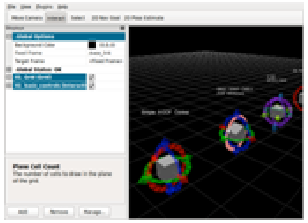
- „ROS is an open-source, meta-operating system for your robot.“ [ROS-wiki]

- ROS is a “robot framework” [ROS-wiki]



Plumbing

+

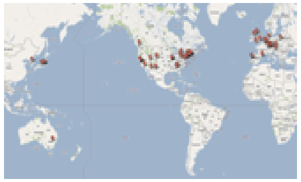


Tools

+

Capabilities

+



Ecosystem

www.ros.org/wiki/ROS/Introduction

© Fraunhofer IPA

ROS – Video

- 5 years of ROS

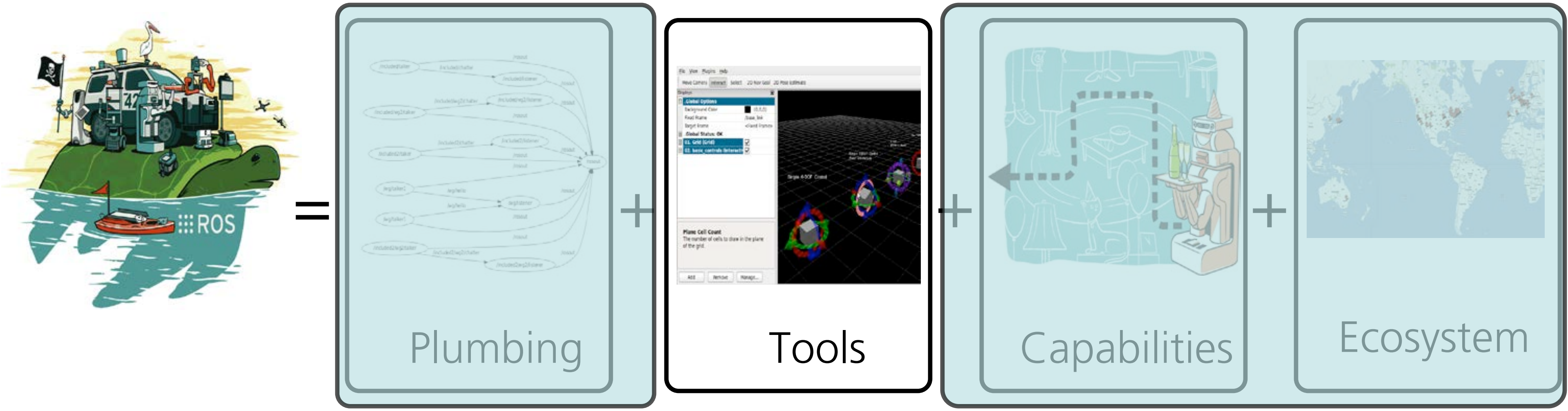
http://youtu.be/PGaXiLZD2KQ

ROS – Robot Operating System

- What is ROS?
- Provides
  - Hardware abstraction
  - Low-level device control
  - Communication layer with message-passing between processes
  - Recursive package management and build system
  - Runs primarily on Linux but is intended to be cross-platform compatible to MAC OS X and Windows
- Content
  - ROS core build and runtime system
  - ROS packages, a collection of robotic algorithms

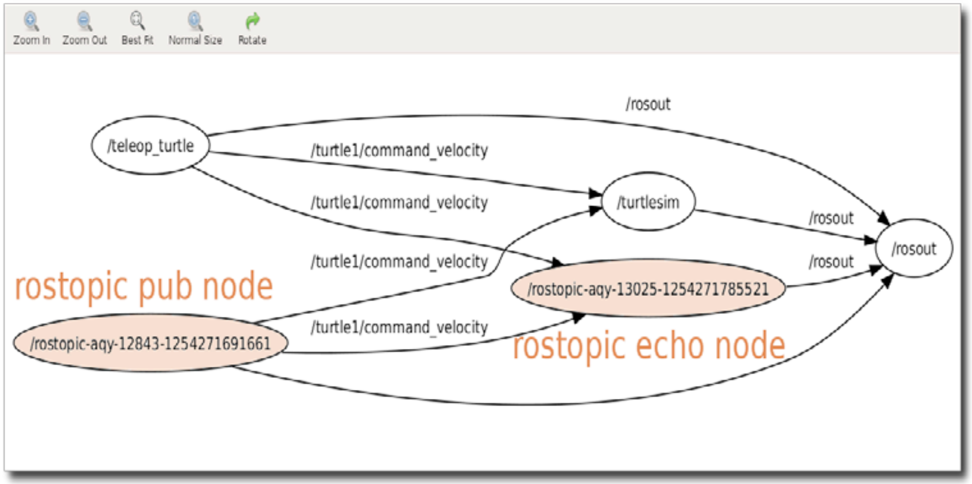
© Fraunhofer IPA

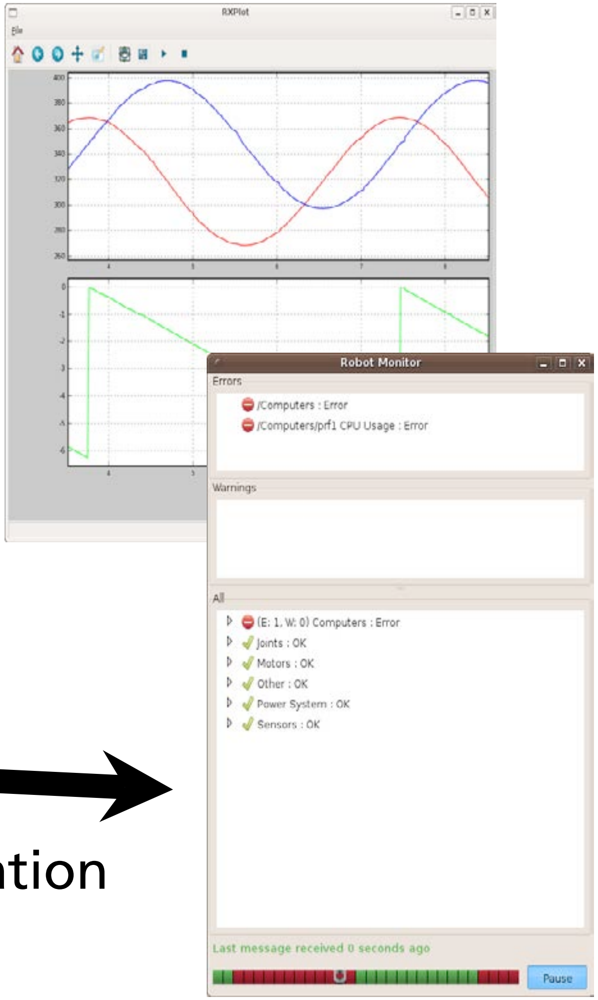
ROS – Tools



© Fraunhofer IPA

ROS – Tools


Zoom In
Zoom Out
Best Fit
Normal Size
Rotate
/rosout
/teleop_turtle
/turtle1/command_velocity
/turtle1/command_velocity
/turtlesim
/rosout
/rosout
/rosout
/turtle1/command_velocity
rostopic pub node
/rostopic-aqy-13025-1254271785521
/turtle1/command_velocity
/rostopic-aqy-12843-1254271691661
rostopic echo node



Errors

/Computers : Error

/Computers/prf1 CPU Usage : Error

Warnings

All

(E: 1, W: 0) Computers : Error

Joints : OK

Motors : OK

Other : OK

Power System : OK

Sensors : OK

Last message received 0 seconds ago

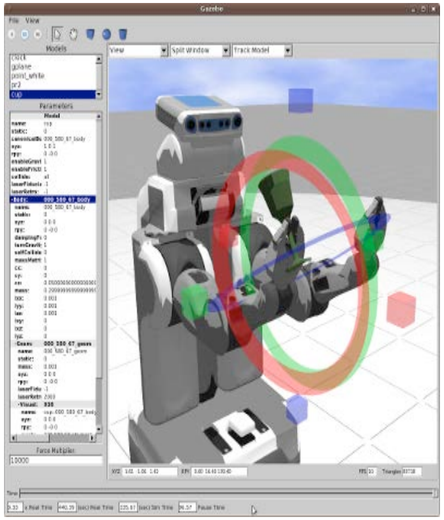
Pause

plotting

graph visualization

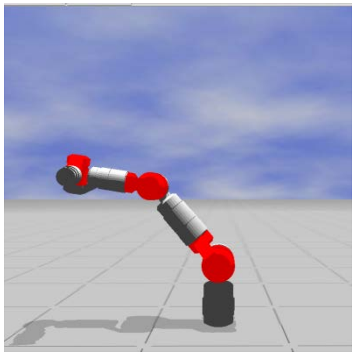
diagnostics

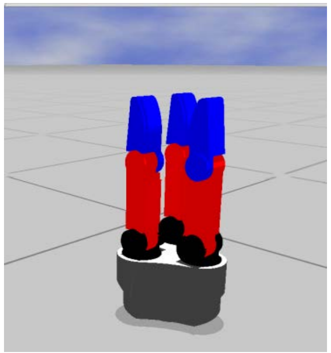
Simulation/visualization



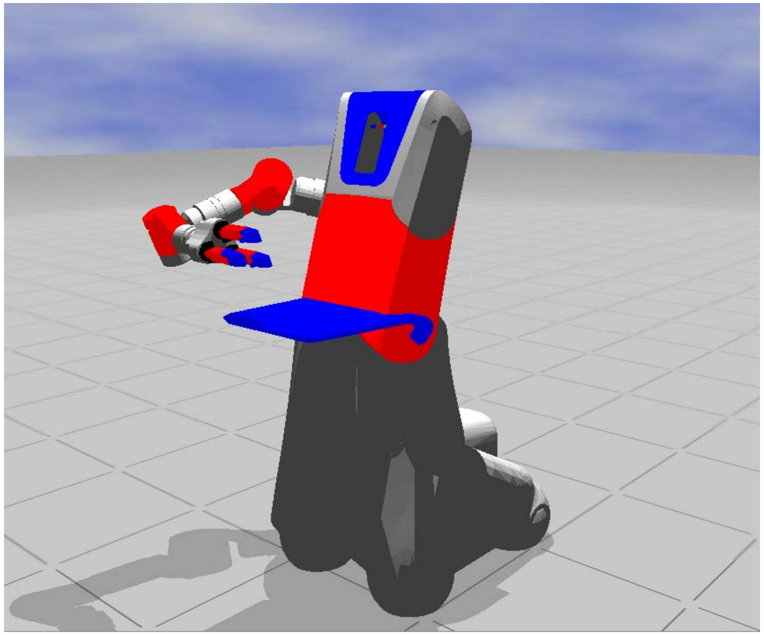
© Fraunhofer IPA

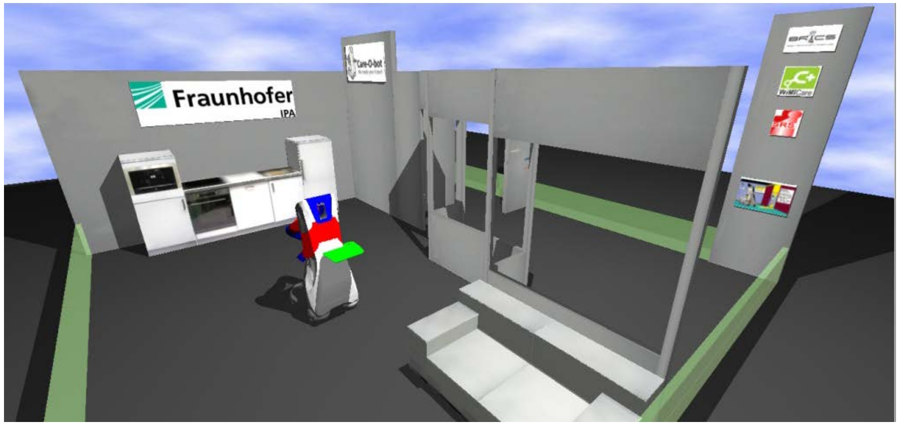
ROS – Tools Simulation





- Single simulated components
- Simulated sensors and actors, e.g.
  - Arm joints, hand
  - Cameras, laser scanners
- Model of the whole robot
- Kinematic and dynamic models of hardware components
- Environment model



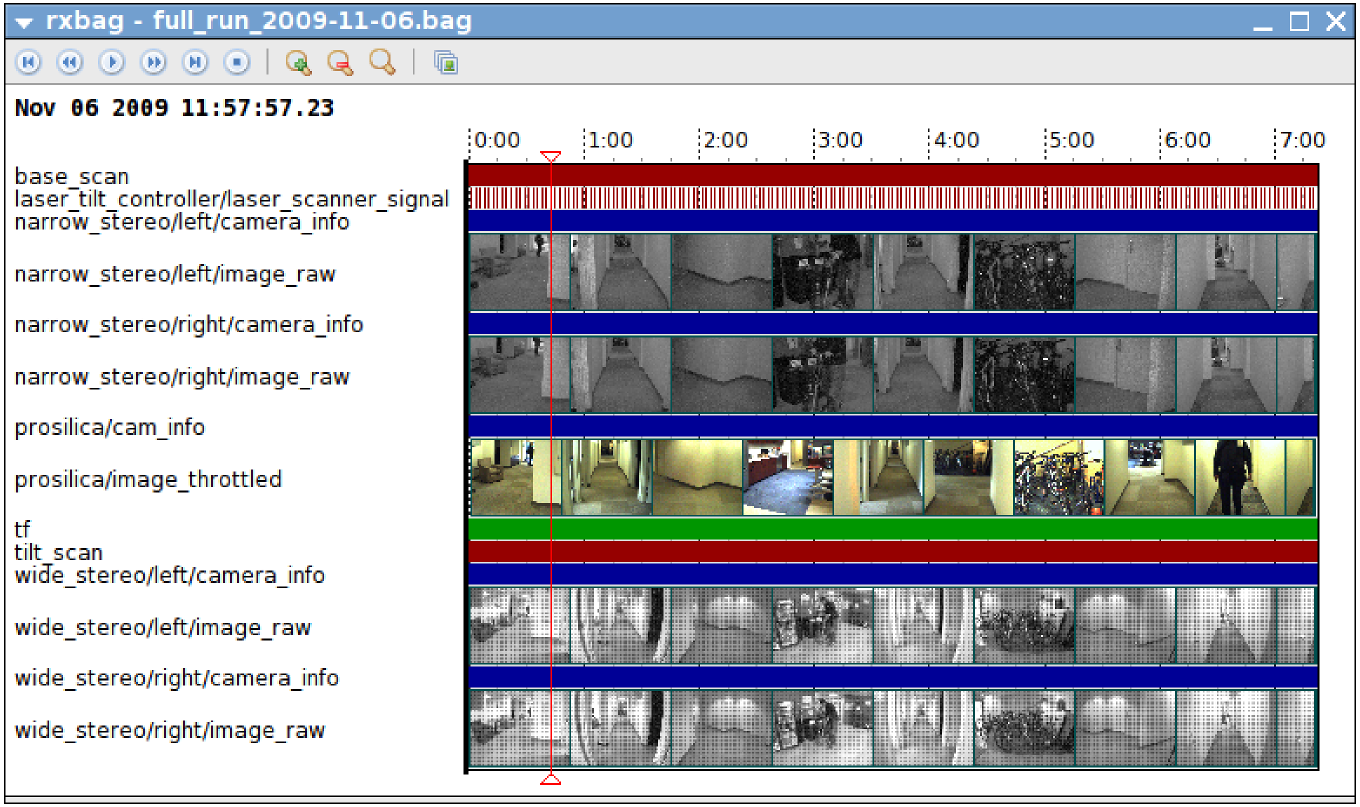


http://www.ros.org/wiki/simulator_gazebo

© Fraunhofer IPA

ROS – Tools Bag files

- Format for saving and playing back ROS message data
- Record and replay sensor data for developing and testing algorithms



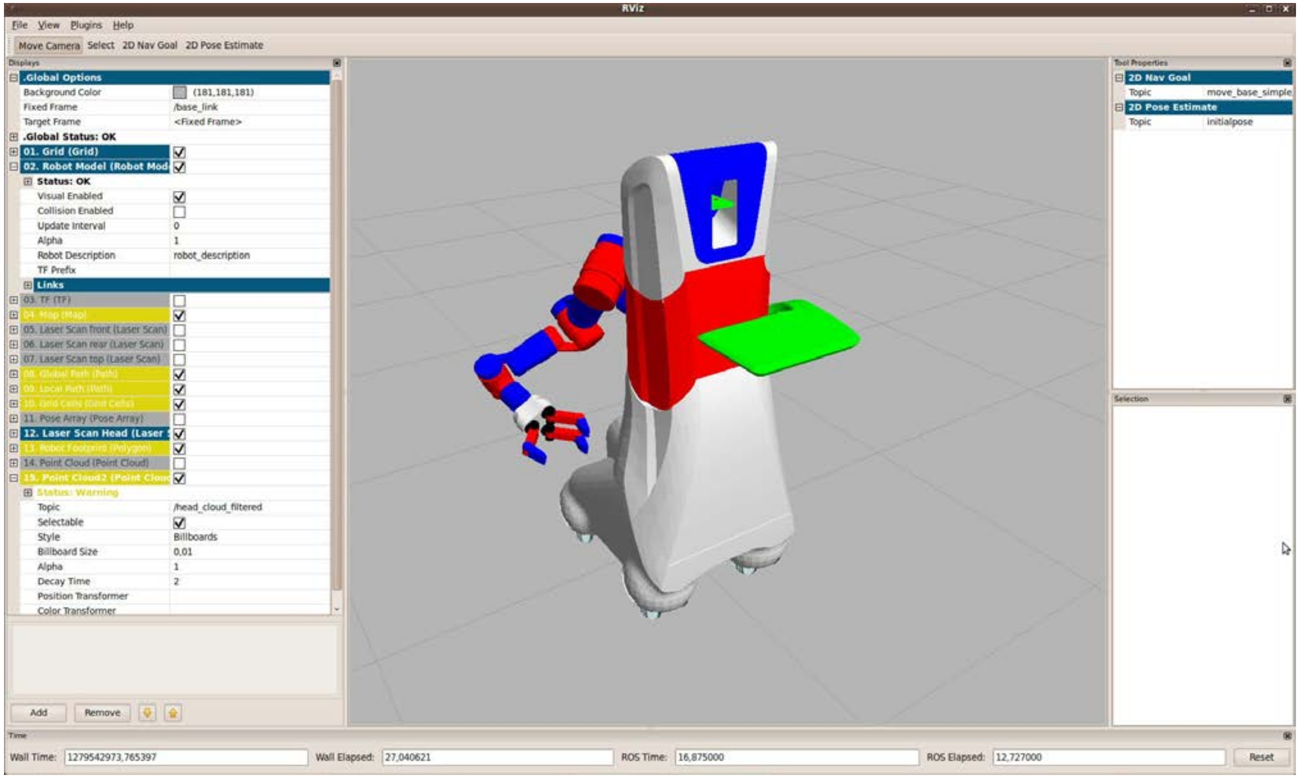
rxbag - full_run_2009-11-06.bag
Nov 06 2009 11:57:57.23
0:00
1:00
2:00
3:00
4:00
5:00
6:00
7:00
base_scan
laser_tilt_controller/laser_scanner_signal
narrow_stereo/left/camera_info
narrow_stereo/left/image_raw
narrow_stereo/right/camera_info
narrow_stereo/right/image_raw
prosilica/cam_info
prosilica/image_throttled
tf
tilt_scan
wide_stereo/left/camera_info
wide_stereo/left/image_raw
wide_stereo/right/camera_info
wide_stereo/right/image_raw

http://www.ros.org/wiki/rosbag

© Fraunhofer IPA

ROS – Tools Visualization

- Robot Model
- Sensor data (laser scanner, point cloud)
- 2D and 3D maps
- Markers



.Global Options	
Background Color	(181,181,181)
Fixed Frame	/base_link
Target Frame	<Fixed Frame>
.Global Status: OK	
01. Grid (Grid)	☑
02. Robot Model (Robot Mod	☑
Status: OK	
Visual Enabled	☑
Collision Enabled	□
Update Interval	0
Alpha	1
Robot Description	robot_description
TF Prefix	
Links	
03. TF (TF)	□
04. Map (Map)	☑
05. Laser Scan front (Laser Scan)	□
06. Laser Scan rear (Laser Scan)	□
07. Laser Scan top (Laser Scan)	□
08. Global Path (Path)	☑
09. Local Path (Path)	☑
10. Grid Cells (Grid Cells)	☑
11. Pose Array (Pose Array)	□
12. Laser Scan Head (Laser S	☑
13. Robot Footprint (Polygon)	☑
14. Point Cloud (Point Cloud)	□
15. Point Cloud2 (Point Cloud	☑
Status: Warning	
Topic	/head_cloud_filtered
Selectable	☑
Style	Billboards
Billboard Size	0,01
Alpha	1
Decay Time	2
Position Transformer	
Color Transformer	

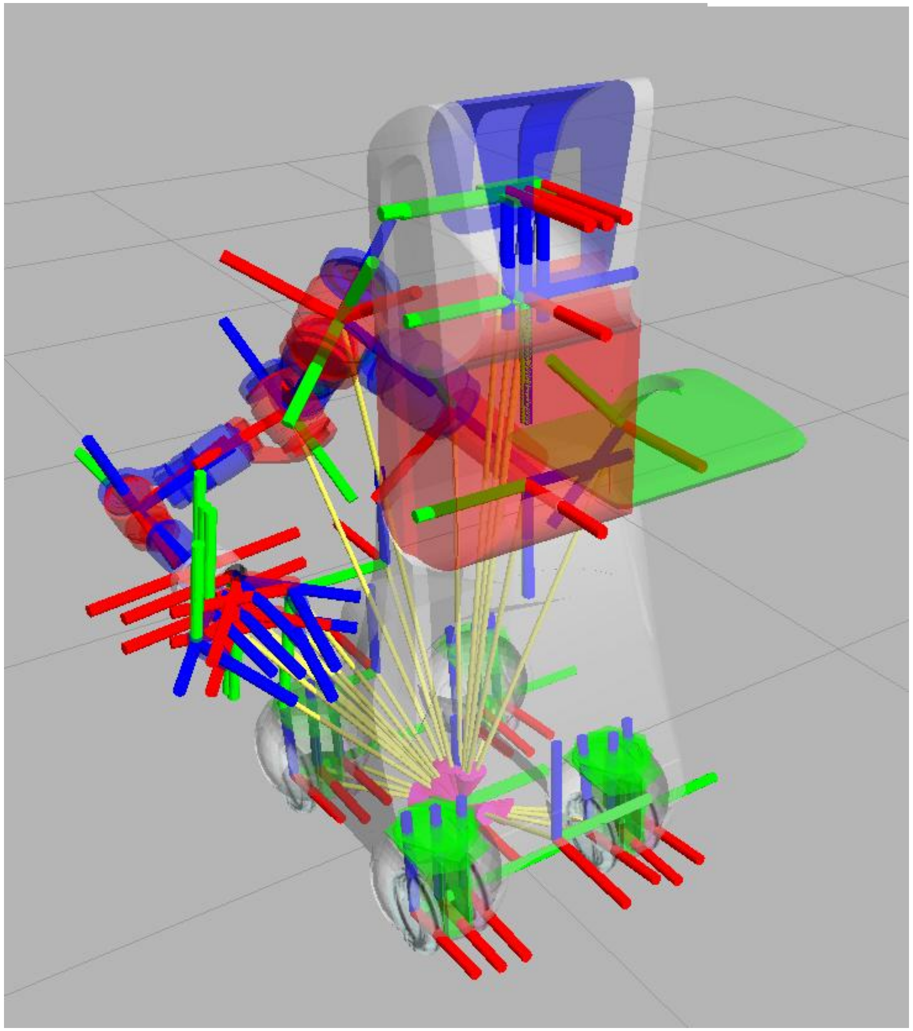
2D Nav Goal	
Topic	move_base_simple
2D Pose Estimate	
Topic	initialpose

Wall Time: 1279542973,765397 Wall Elapsed: 27,040621 ROS Time: 16,875000 ROS Elapsed: 12,727000

http://www.ros.org/wiki/rviz

ROS – Tools transformations library (tf)

- Tree of coordinate systems
- Defined by urdf (Robot Description Language)
- Transformations between all coordinate systems available
- Generated automatically out of /joint_states topic



→ http://www.ros.org/wiki/tf

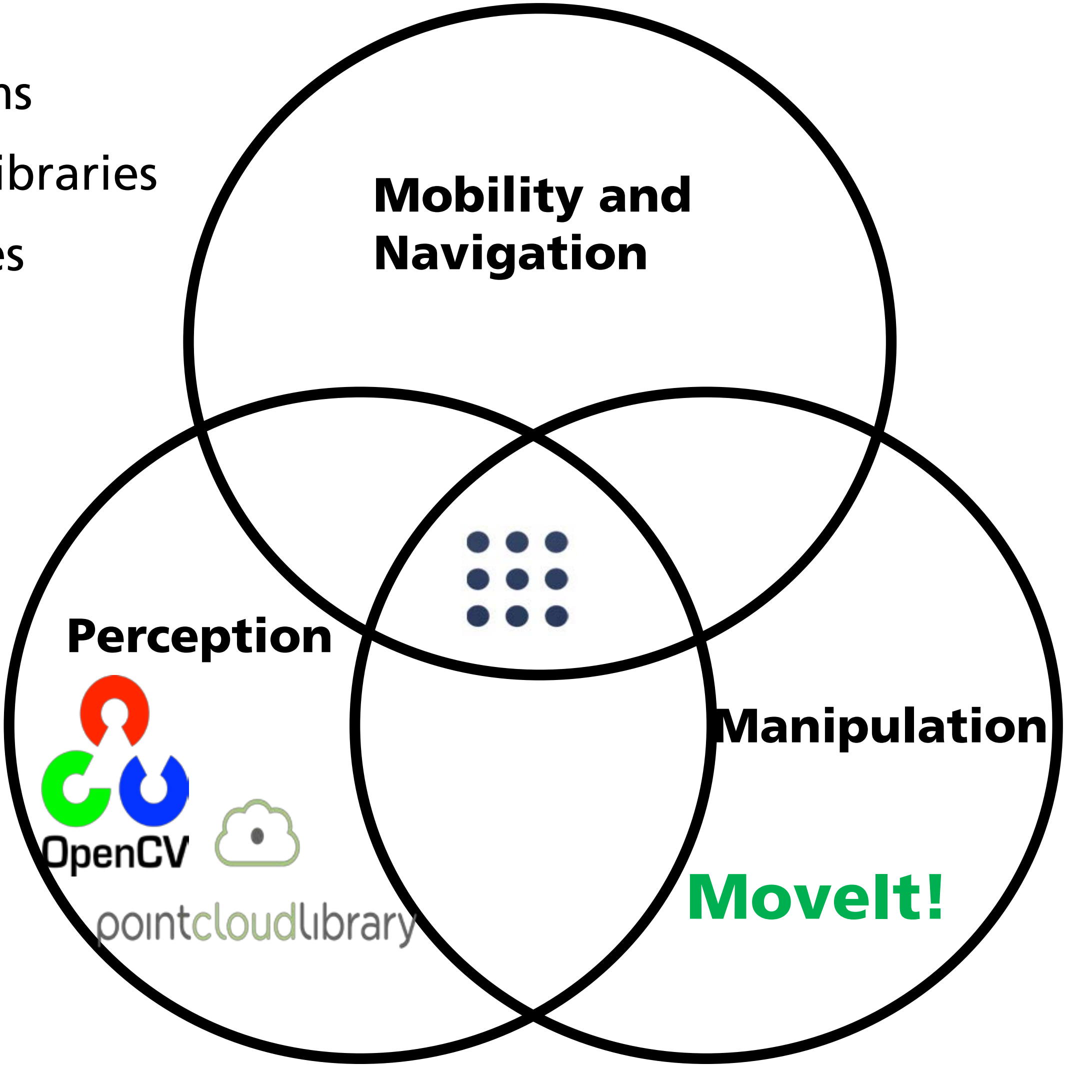
ROS – Capabilities

=
Plumbing
+
Tools
+
Capabilities
+
Ecosystem

© Fraunhofer IPA

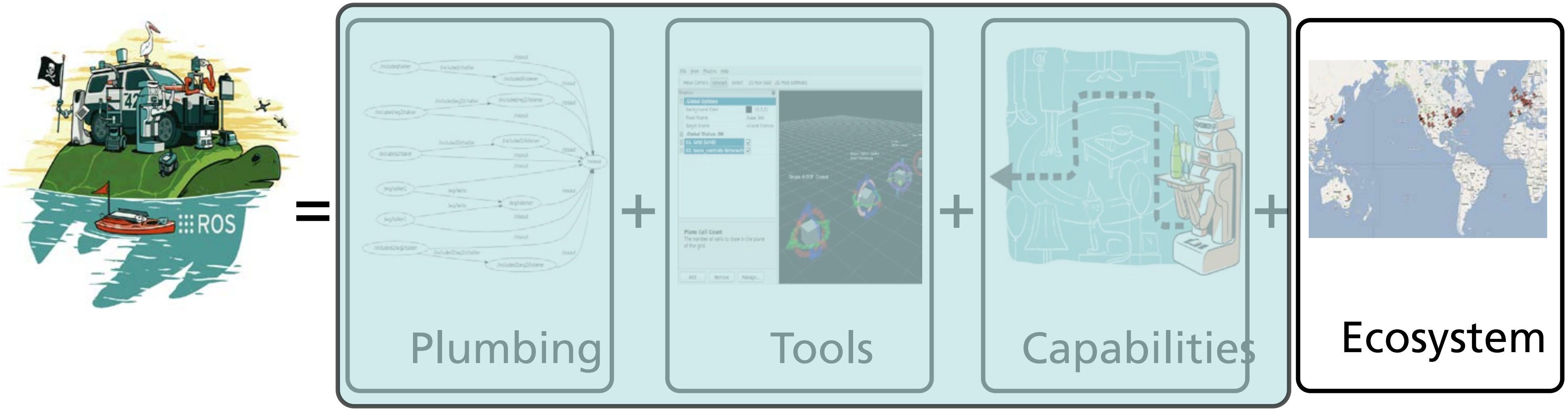
ROS – Capabilities

- State of the art algorithms
- Integration of available libraries
- Wide range of capabilities
    - Navigation
    - Perception
    - manipulation



© Fraunhofer IPA

ROS – Community/Ecosystem

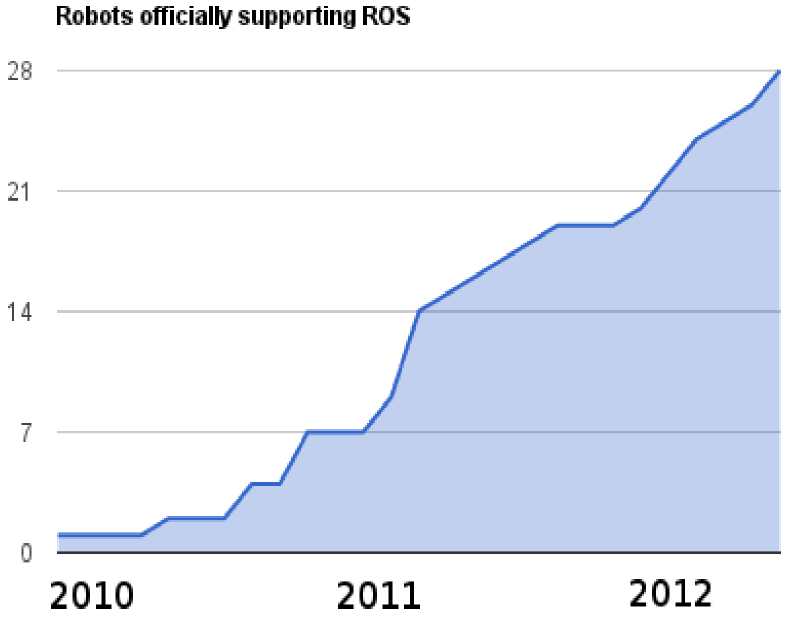


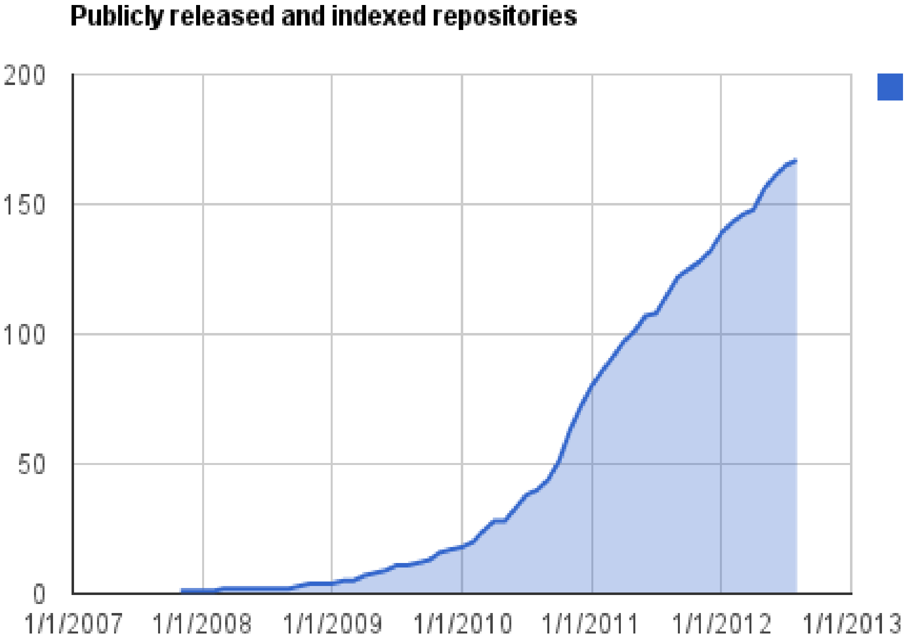
= Plumbing + Tools + Capabilities + Ecosystem

© Fraunhofer IPA

ROS – Community/Ecosystem

- Fast growing community
- De facto standard for service robotics

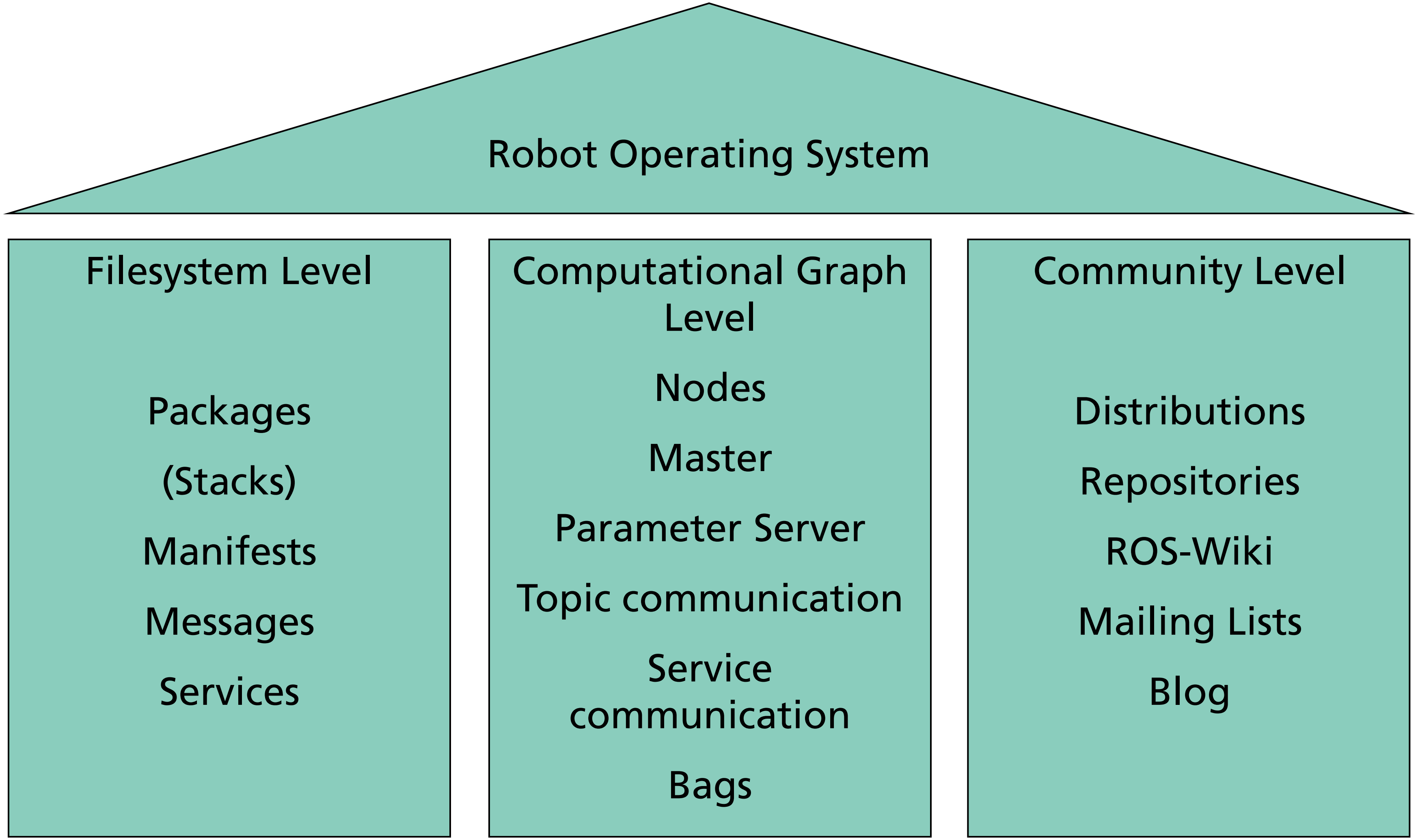




© Fraunhofer IPA

ROS – Technical details

Three levels of ROS concepts



Robot Operating System

Filesystem Level

Packages

(Stacks)

Manifests

Messages

Services

Computational Graph Level

Nodes

Master

Parameter Server

Topic communication

Service communication

Bags

Community Level

Distributions

Repositories

ROS-Wiki

Mailing Lists

Blog

http://ros.org/wiki/ROS/Concepts

© Fraunhofer IPA

ROS – Filesystem Level

- Packages
  - Main unit for organizing software
  - Typically one functionality, e.g. localisation or path planning
  - Contains: runtime processes (nodes), libraries, datasets, configuration files, …
- Stacks
  - Collection of packages
  - Aggregate functionality, e.g. navigation stack
  - Releases and versioning
- Stack- and Package- Manifests (*.xml)
  - Provide Metadata about a package/stack, e.g. license information and dependencies to other packages/stacks

→ http://ros.org/wiki/ROS/Concepts

ROS – Filesystem Level

- Messages types (*.msg)
  - Message descriptions, define data structures used for message communication
  - Language independent

TargetPoses.msg

```
Header header
Std_msgs/String name
Geometry_msgs/Pose2D[] poses
```

Pose2D.msg

```
Float64 x
Float64 y
Float64 theta
```

- Services types (*.srv)
  - Service descriptions, define request and response data structures used for service communication
  - Language independent

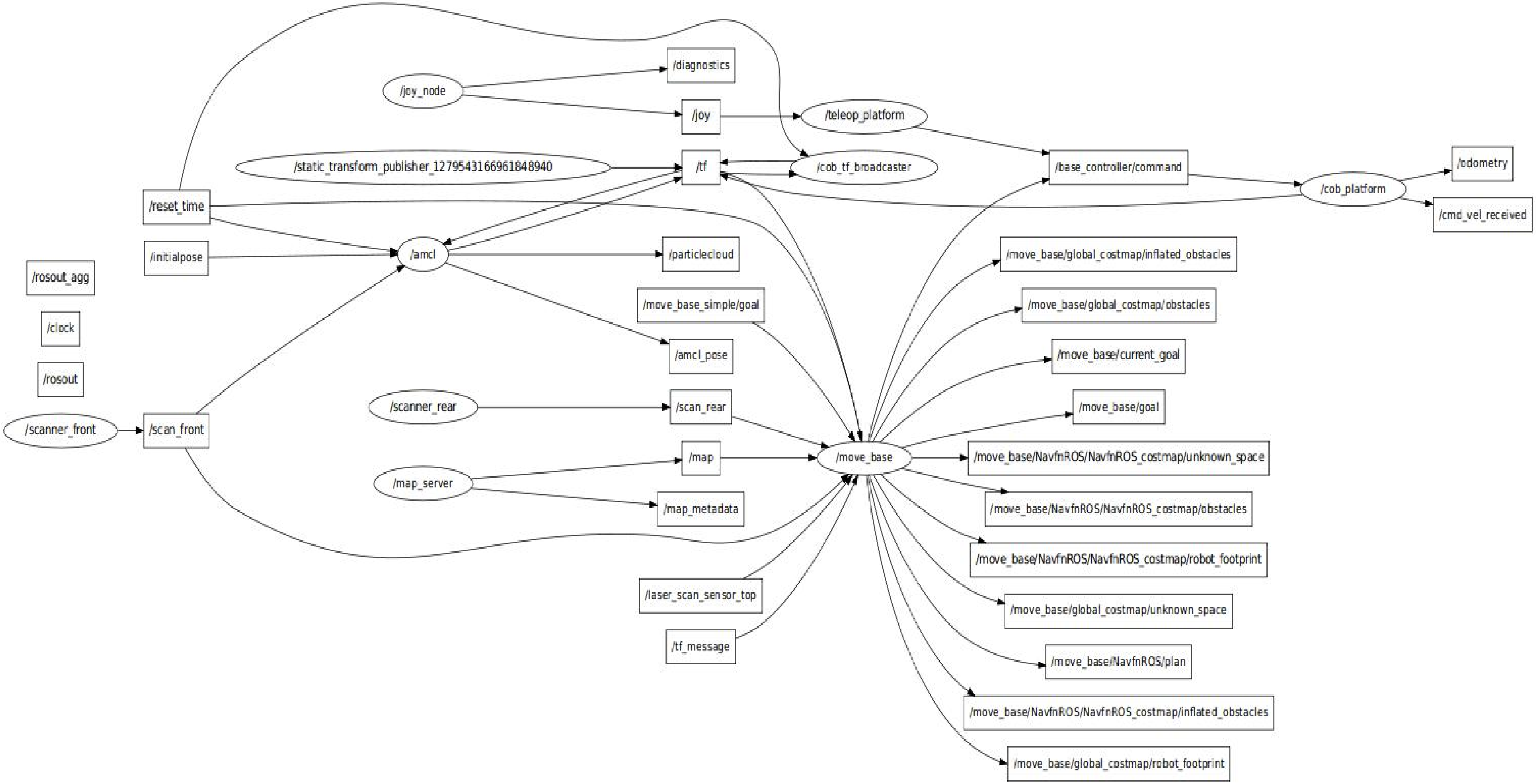
GetPose.srv

```
std_msgs/String name
--
Geometry_msgs/Pose2D pose
```

http://ros.org/wiki/msg, http://ros.org/wiki/srv

© Fraunhofer IPA

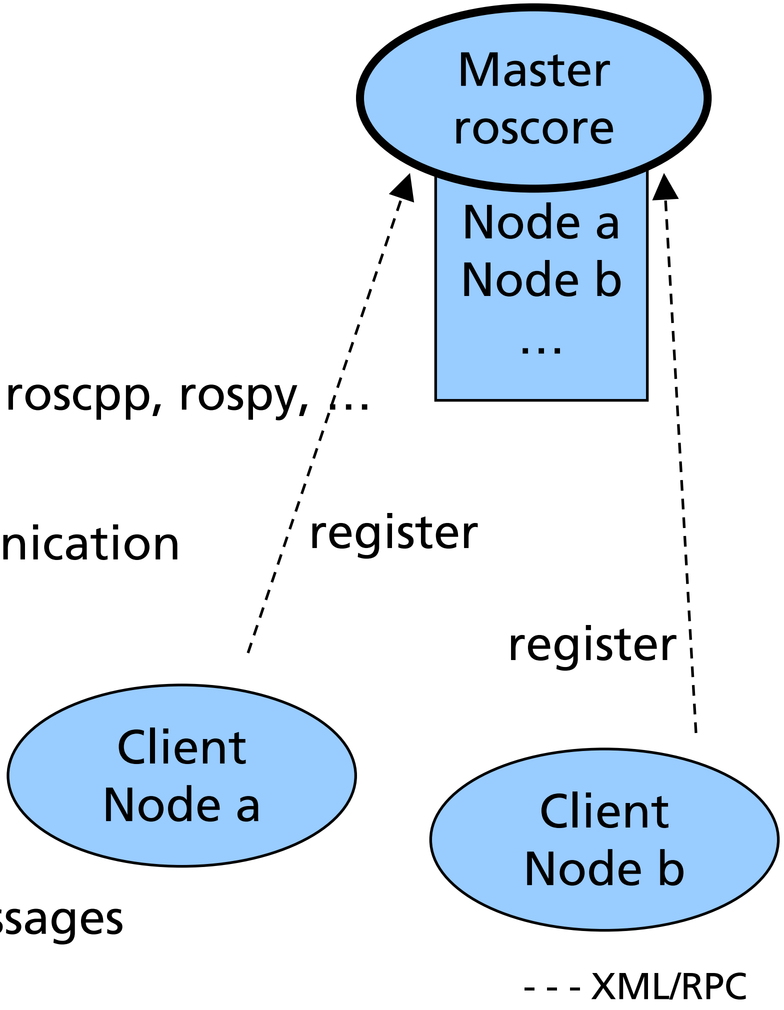
ROS – Computational Graph Level


/diagnostics
/joy_node
/joy
/teleop_platform
/static_transform_publisher_1279543166961848940
/tf
/cob_tf_broadcaster
/base_controller/command
/odometry
/cob_platform
/cmd_vel_received
/reset_time
/initialpose
/amcl
/particlecloud
/rosout_agg
/clock
/rosout
/move_base_simple/goal
/amcl_pose
/move_base/global_costmap/inflated_obstacles
/move_base/global_costmap/obstacles
/move_base/current_goal
/move_base/goal
/scanner_rear
/scan_rear
/scanner_front
/scan_front
/map
/move_base
/move_base/NavfnROS/NavfnROS_costmap/unknown_space
/map_server
/map_metadata
/move_base/NavfnROS/NavfnROS_costmap/obstacles
/move_base/NavfnROS/NavfnROS_costmap/robot_footprint
/laser_scan_sensor_top
/move_base/global_costmap/unknown_space
/tf_message
/move_base/NavfnROS/plan
/move_base/NavfnROS/NavfnROS_costmap/inflated_obstacles
/move_base/global_costmap/robot_footprint

© Fraunhofer IPA

ROS – Computational Graph Level

- Nodes
  - Processes to perform computation
  - Usually many nodes at runtime
  - Written by a ROS client library, e.g. roscpp, rospy, …
- Master
  - Coordinating processes and communication
  - Name registration and lockup
- Parameter Server
  - Central location for storing data
- Messages
  - Nodes communicate by passing messages



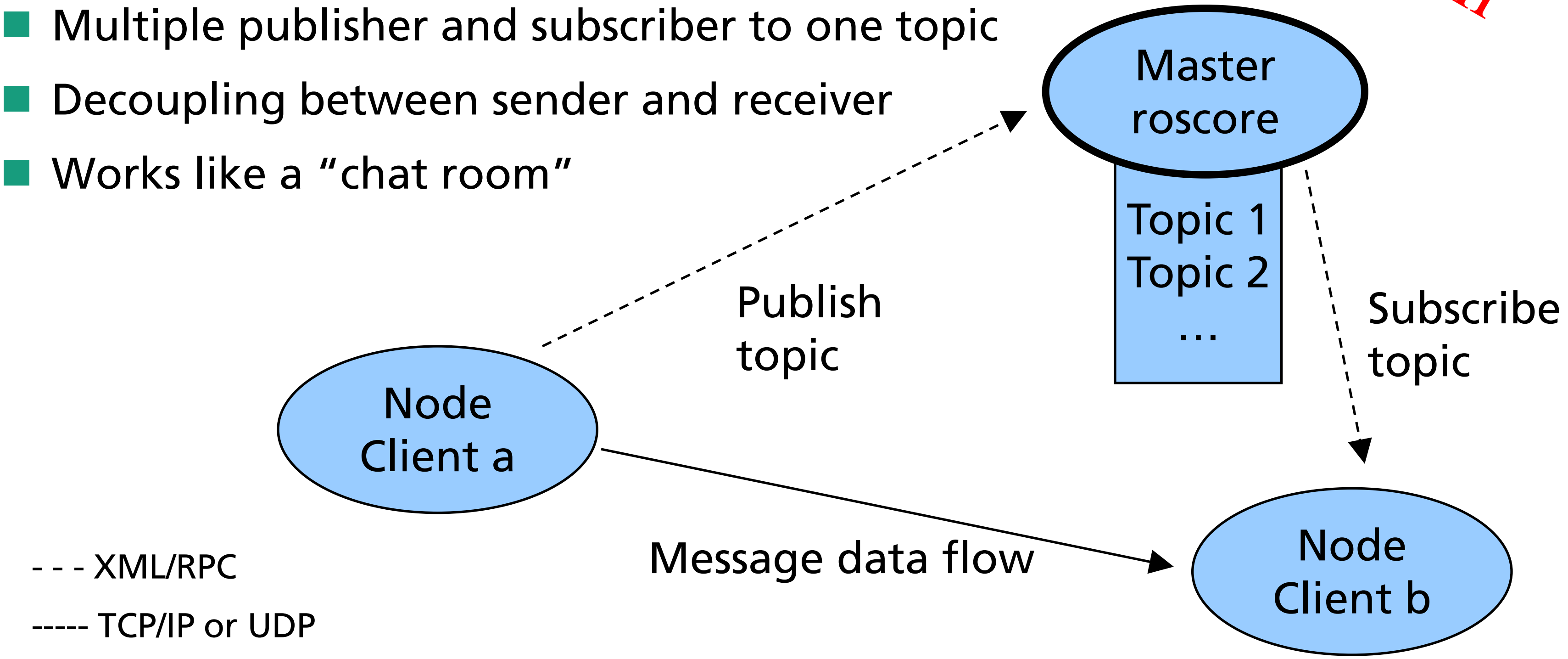
http://www.ros.org/wiki/ROS/Technical%20Overview

© Fraunhofer IPA

ROS – Communication concepts – topics

Many-to-many, one-way communication

- Topics (asynchronous streaming)
- Nodes can publish and subscribe to topics
- Multiple publisher and subscriber to one topic
- Decoupling between sender and receiver
- Works like a “chat room”



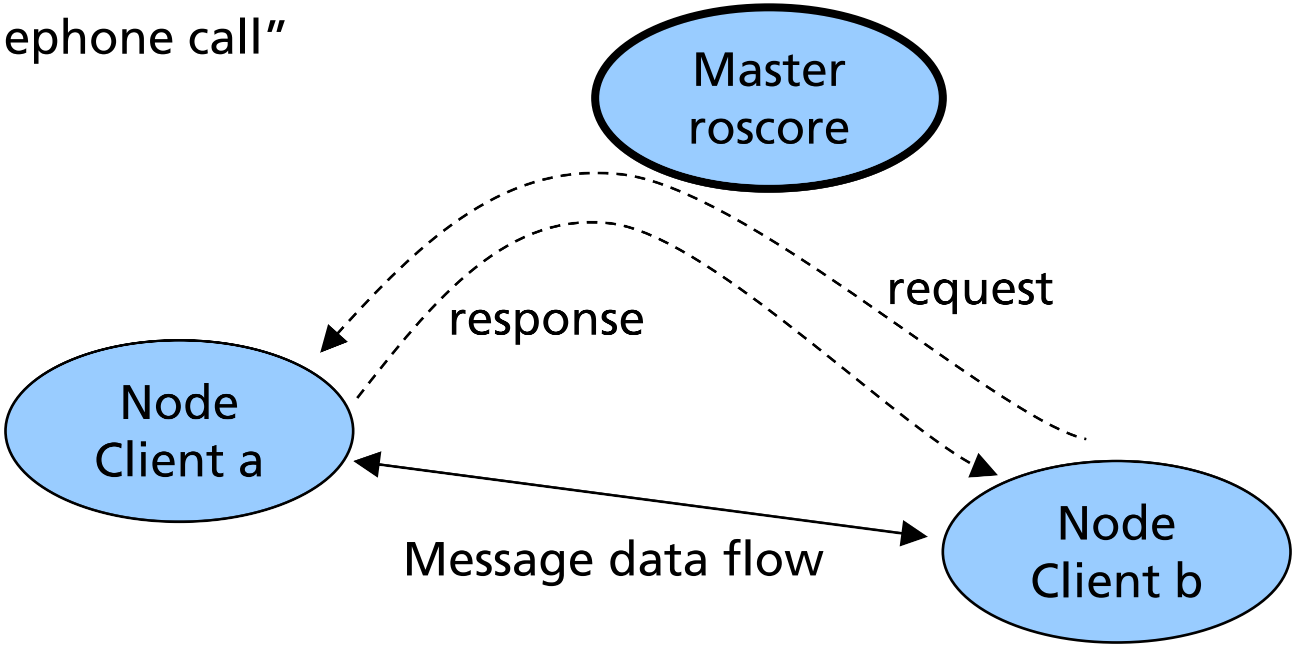
http://www.ros.org/wiki/Topics

© Fraunhofer IPA

ROS – Communication concepts – services

One-to-one,
two-way
communication

- Services (synchronous communication)
- Request and reply interaction
- Dedicated connection between two nodes
- Works like a “telephone call”



- - - XML/RPC

----- TTCP/IP or UDP

http://www.ros.org/wiki/Services

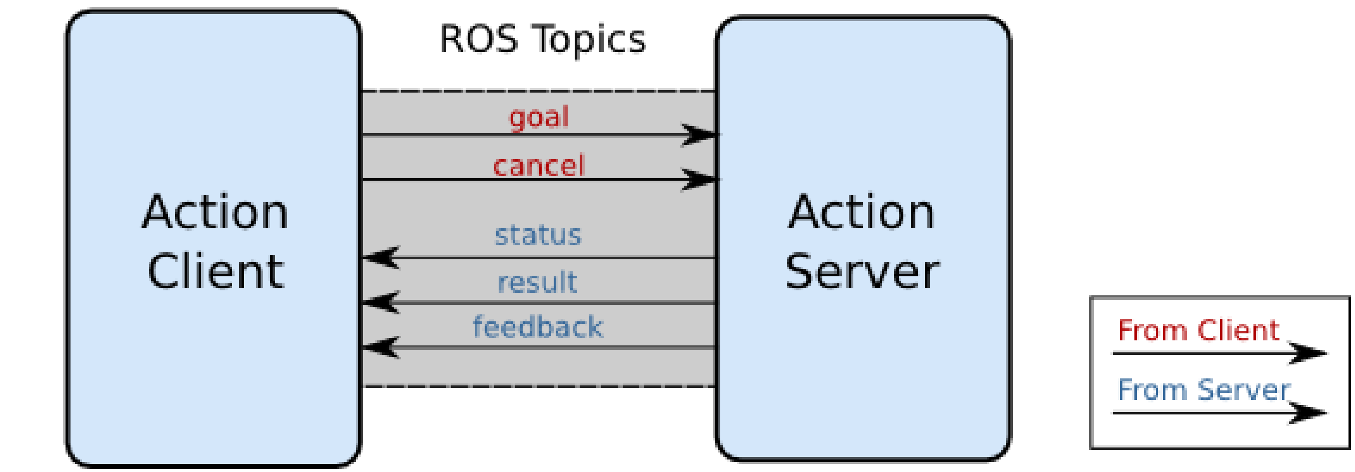
ROS – Communication concepts – actionlib

- Goal description similar to message and service definitions
- State-machine running on server and client

DetectBottle.action

```
#goal
std_msgs/String drink_name
---
#result
Geometry_msgs/Pose3D pose
---
#feedback
Int16 status
```

Action Interface



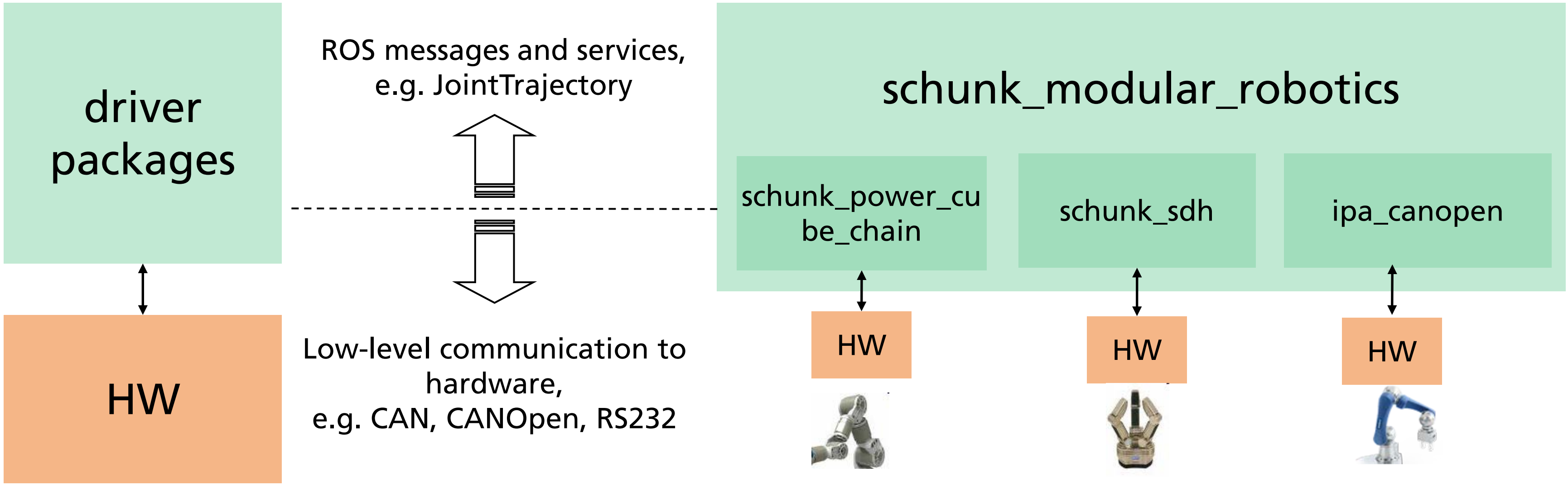
http://www.ros.org/wiki/actionlib

© Fraunhofer IPA

ROS support of Schunk components

ROS for Schunk components

- Devices supported in ROS
  - SDH (Schunk Dextrous Hand, SDH library)
  - LWA (Lightweight Arm, M5API)
  - Powerball (Powerball arm, CanOpen)



driver packages

ROS messages and services, e.g. JointTrajectory

schunk_modular_robotics

schunk_power_cube_chain

schunk_sdh

ipa_canopen

HW

Low-level communication to hardware, e.g. CAN, CANOpen, RS232

HW

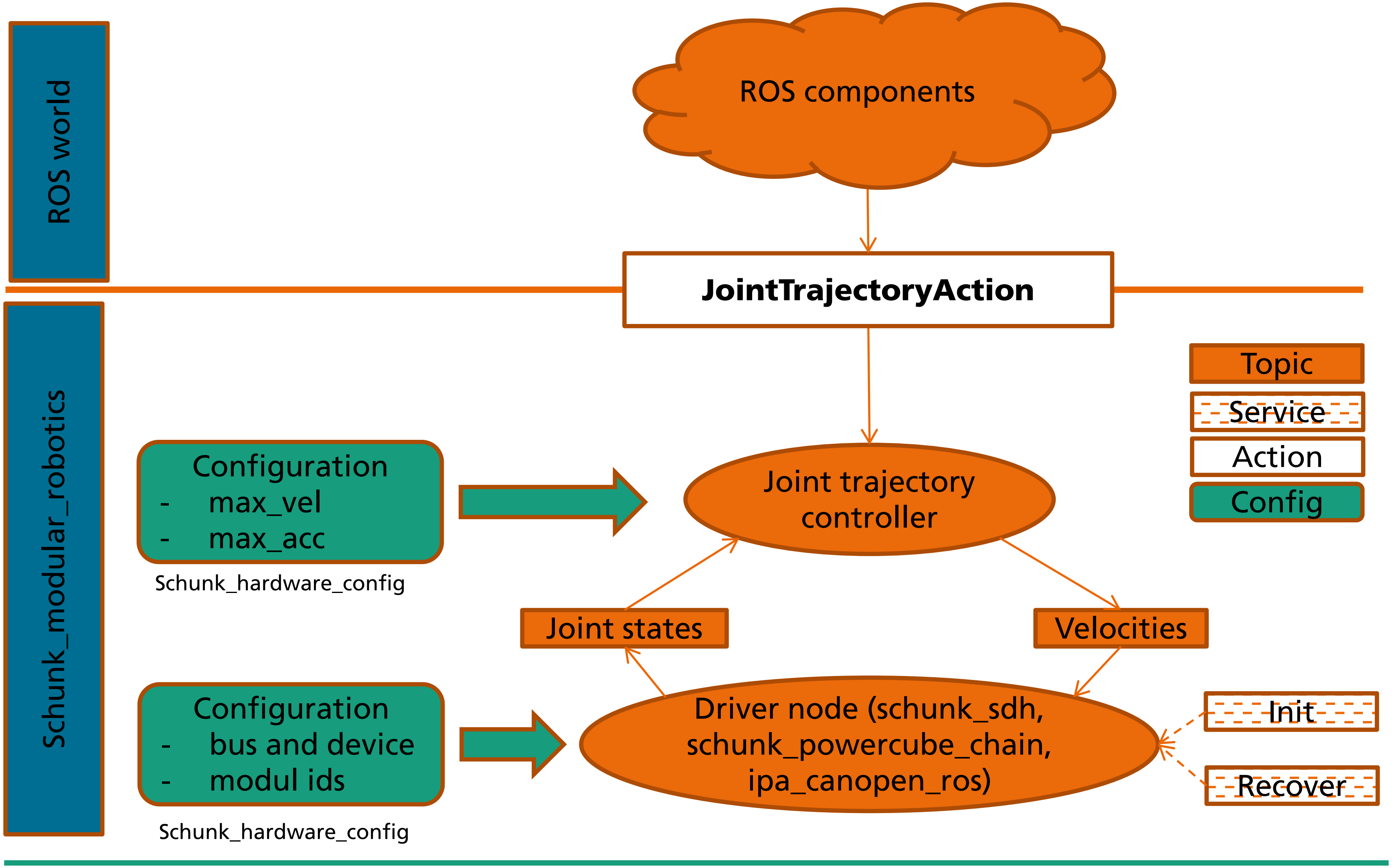
HW

HW

→ http://www.ros.org/wiki/schunk_modular_robotics

© Fraunhofer IPA

SW Architecture



© Fraunhofer IPA

Installation and Operation Instructions

Installing ROS for Schunk components for Fuerte

- Install ROS Fuerte http://www.ros.org/wiki/fuerte/Installation/Ubuntu
- Download Schunk repositories to your ROS_PACKAGE_PATH
  - *mkdir <<YOUR ROS_PACKAGE_PATH>>, e.g. ~/git/schunk_robots*
  - *cd <<YOUR ROS_PACKAGE_PATH>>*
  - *rosinstall . https://raw.github.com/ipa320/schunk_robots/electric_dev/fuerte.rosinstall*
  - *echo "export <<YOUR ROS_PACKAGE_PATH>>:$ROS_PACKAGE_PATH" >~/.bashrc*
  - *Source ~/.bashrc*
- Build Schunk packages
  - *rosdep install schunk_robots*
  - *rosmake schunk_robots*
- Configure your hardware (Example for powerball)
  - *roscd schunk_hardware_config/powerball/config*
  - Modify powerball_modules.yaml, see slide 40
- Run driver (Example for lwa with M5API)
  - *roslaunch schunk_bringup powerball_solo.launch*
- Move with dashboard (Example for lwa with M5API)
  - *roslaunch schunk_bringup dashboard_powerball.launch*

IPA CANopen Installation - Prerequisites

- 32Bit *nix operating system
- CMake (to manage the build process), in Ubuntu use the command

```
sudo apt-get install cmake
```

- Git (to download the sources from github) in Ubuntu use the command

```
sudo apt-get install git
```

- A C++ compiler with good support for the C++11 Standard

IPA CANopen Installation – CANopen library

- To install the C++ library independently from ROS

```
-   git clone git://github.com/ipa320/ipa_canopen.git
-   cd ipa_canopen/ipa_canopen_core
-   mkdir build
-   cd build
-   cmake ..
-   make
```

- To use the two command line tools, change the directory

```
cd ipa_canopen/ipa_canopen_core/build/tools
```

© Fraunhofer IPA

IPA CANopen Installation – ROS package

- To install the IPA CANopen ROS package

```
-   git clone
    git://gitub.com/ipa320/ipa_canopen.git
-   rosmake ipa_canopen_ros
```

- Test if the installation was successful
  - In one terminal

```
roscore
```

  - In another terminal

```
Rosrun ipa_canopen_ros canopen_ros
```

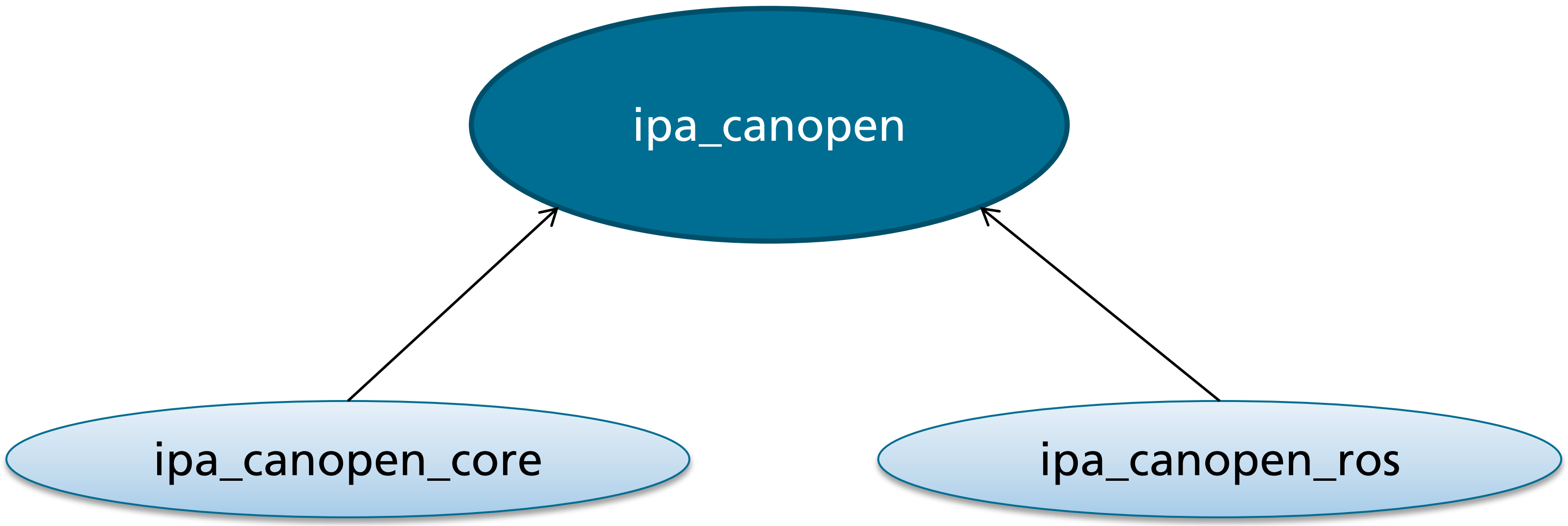
  - You get the following message

```
Missing parameter on server; shutting down
```

Technical details CANOpen

CanOpen Architecture

- C++ framwork
- Enables communication with CANopen motor devices
- Uses the pcan device driver
- Consits of two parts



→ http://www.ros.org/wiki/ipa_canopen

ipa_canopen_core

- ROS independent library
- Provides two command line tools (in ipa_canopen/ipa_canopen_core/buidl/tools)
  - Homing (takes two arguments)
    - Name of the devicefile
    - CANdevice ID of the module
    - E.g.

```
./homing /dev/pcan32 12
```

  - Move device (takes five argumtents)
    - Name of the devicefile
    - CAN device ID of the module
    - Synchronization time
    - Target velocity in rad/msec
    - Target acceleration in rad/msec²
    - E.g.

```
./move_device /dev/pcan32 12 10 0.05 0.01
```

→ http://www.ros.org/wiki/ipa_canopen_core

ipa_canopen_ros

- Wrapper to control CANopen motor devices in ROS
- Two ways to launch the node
  - Direct

```
rosrun ipa_canopen_ros ipa_canopen
```

  - Via a launchfile, e.g.

```
roslaunch schunk_bringup powerball_solo.launch
```

- In order to use the node you need
  - A trajectory controller
  - A rudimentary robot model (urdf)
- For a list of services, subcribed and published topics and necessary parameters on the parameter server follow the link below

http://www.ros.org/wiki/ipa_canopen_ros

Moving the Schunk LWA 4.6 (Powerball) arm

Driving the Schunk LWA 4.6 (Powerball) arm

- Make sure you have the following two repositories on your pc
  - Ipa320/schunk_robots.git
  - Ipa320/schunk_modular_robotics.git
- If not

```
-   git clone git://github.com/ipa320/schunk_robots.git
-   git clone git://github.com/ipa320/schunk_modular_robotics.git
```

- To launch the CANopen driver together with a trajectory controller

```
roslaunch schunk_bringup powerball_solo.launch
```

- To launch the Powerball-arm in Gazebo

```
roslaunch schunk_bringup_sim powerball.launch
```

- To move the arm with a graphical command GUI

```
roslaunch schunk_bringup dashboard_powerball.launch
```

- To configure the necessary yaml-files: check the next slides

© Fraunhofer IPA

Configuring the Schunk LWA 4.6 (Powerball) arm I

- Change directory

```
roscd schunk_hardware_config/powerball/config
```

- Modify powerball_modules.yaml e.g.

```
Devices:
        - name: /dev/pcan32
          baudrate: 500K
          sync_interval: 10
Chains: [“arm_controller“]
```

- Modify powerball_trajectory_controller.yaml e.g.

```
joint_names: [“arm_1_joint”, “arm_2_joint”, “arm_3_joint”,
“arm_4_joint”, “arm_5_joint”, “arm_6_joint”]
module_ids: [3, 4, 5, 6, 7, 8]
Devices: [“/dev/pcan32”, “/dev/pcan32”, “/dev/pcan32”,
“/dev/pcan32”, “/dev/pcan32”, “/dev/pcan32”]
```

Configuring the Schunk LWA 4.6 (Powerball) arm II

- Change directory

```
roscd schunk_hardware_config/powerball/config
```

- Modify powerball_CANopen.yaml e.g.

```
# canopen parameters
can_module: PCAN
can_baudrate: 1000
max_accelerations: [0.8, 0.8, 0.8, 0.8, 0.8, 0.8]
OperationMode: position

# trajectory controll parameters
ptp_vel: 0.4 # rad/sec
ptp_acc: 0.1 # rad/sec²
max_error: 0.2 # rad
frequency: 100
```

© Fraunhofer IPA

Configuring the Schunk LWA 4.6 (Powerball) arm III

- Change directory

```
roscd schunk_default_config/powerball/config
```

- Modify powerball_joint_configurations.yaml e.g.

```
joint_names:
["arm_1_joint","arm_2_joint","arm_3_joint","arm_4_joint","arm_5_joint","arm_6_j
oint"]

# back side positions
home: [[0,0,0,0,0,0]]
folded: [[0.32108866388214263, 0.6484189832579226, 2.06286710514828, -
1.2376313006847157, 5.658013215042093, -7.150174321779446e-05]]
wave_left: [[0.321033880484058, 0.49950722659008573, -0.4061025056033145, -
0.2370251233291425, 5.300248440143207e-06, 9.462633828505318e-06]]
wave_right: [[0.4741062629069983, -0.7912476227793528, 0.0041526706870680385, -
2.4662076334003302e-05, 2.4489075676648042e-05, 8.393716051102729e-06]]

# trajectories
wave_left-wave_right-home: [wave_left,wave_right,home]
```

© Fraunhofer IPA

Schunk Demo

Save new positions for the ROS parameter server:

```
rosrun schunk_demo save_position.py -p <position>
```

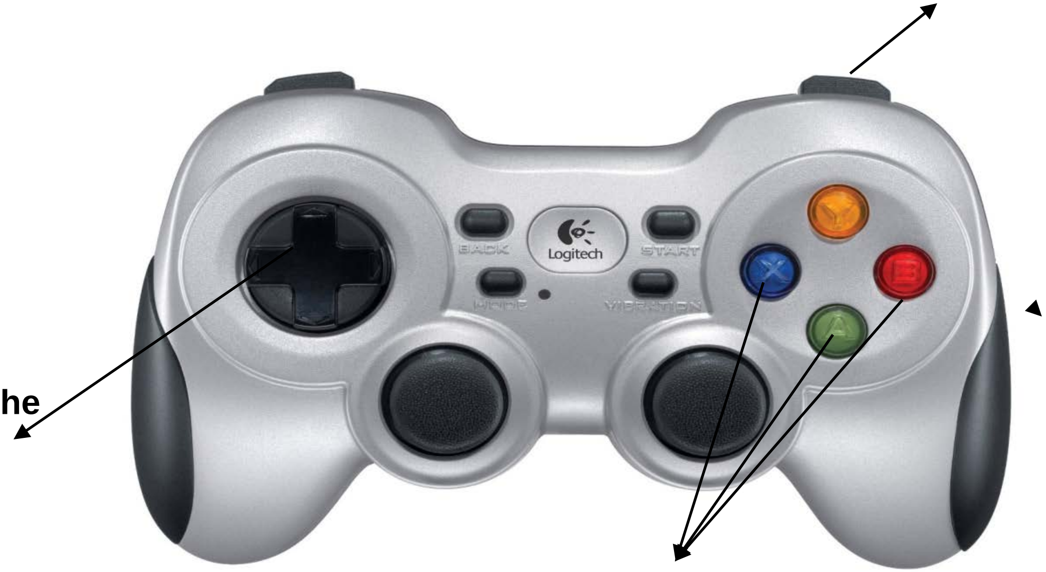
- home
- folded
- waveright
- waveleft

Moving through all the predefined positions:

```
rosrun schunk_demo demo_arm.py
```

Moving with the joystick

The joystick node is automatically loaded from the schunk_bringup.



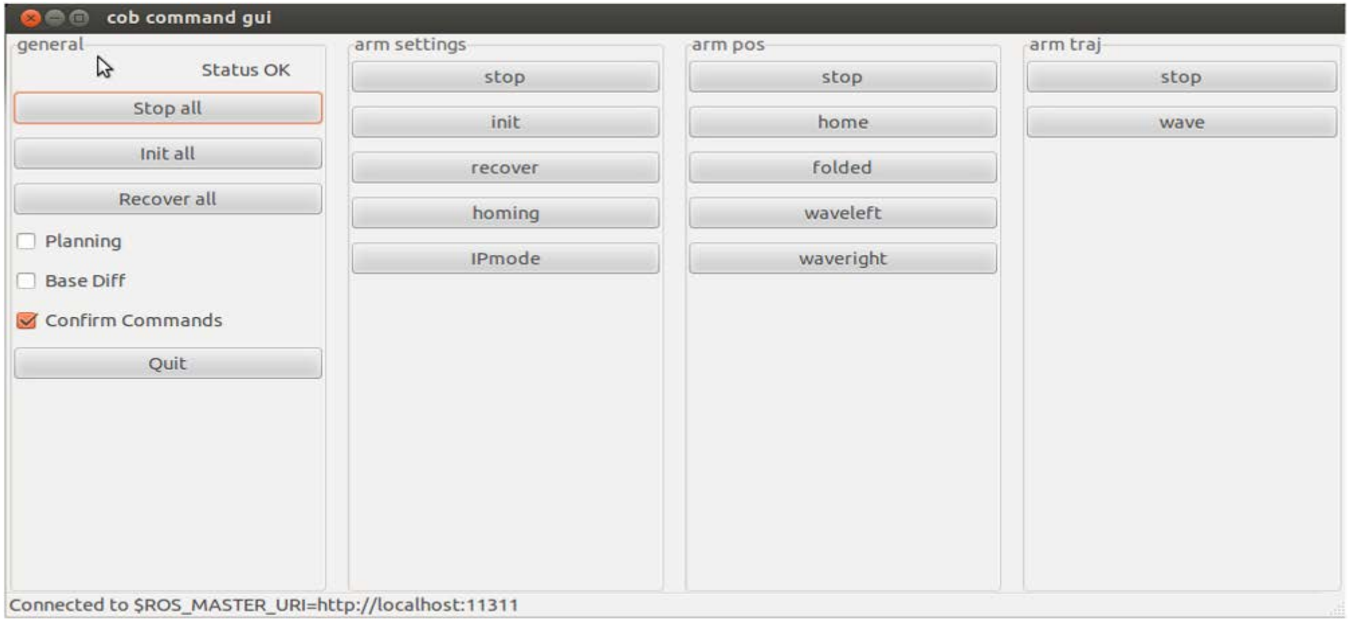
Enable Movement

Move using the directionals

Select Joint

ROS for Schunk components – command gui

- Tool to easily move arm or hand
- Buttons for initializing and recover
- Buttons to move components in joint space
- *"roslaunch schunk_bringup dashboard_lwa.launch"*



Status OK

□ Planning

□ Base Diff

☑ Confirm Commands

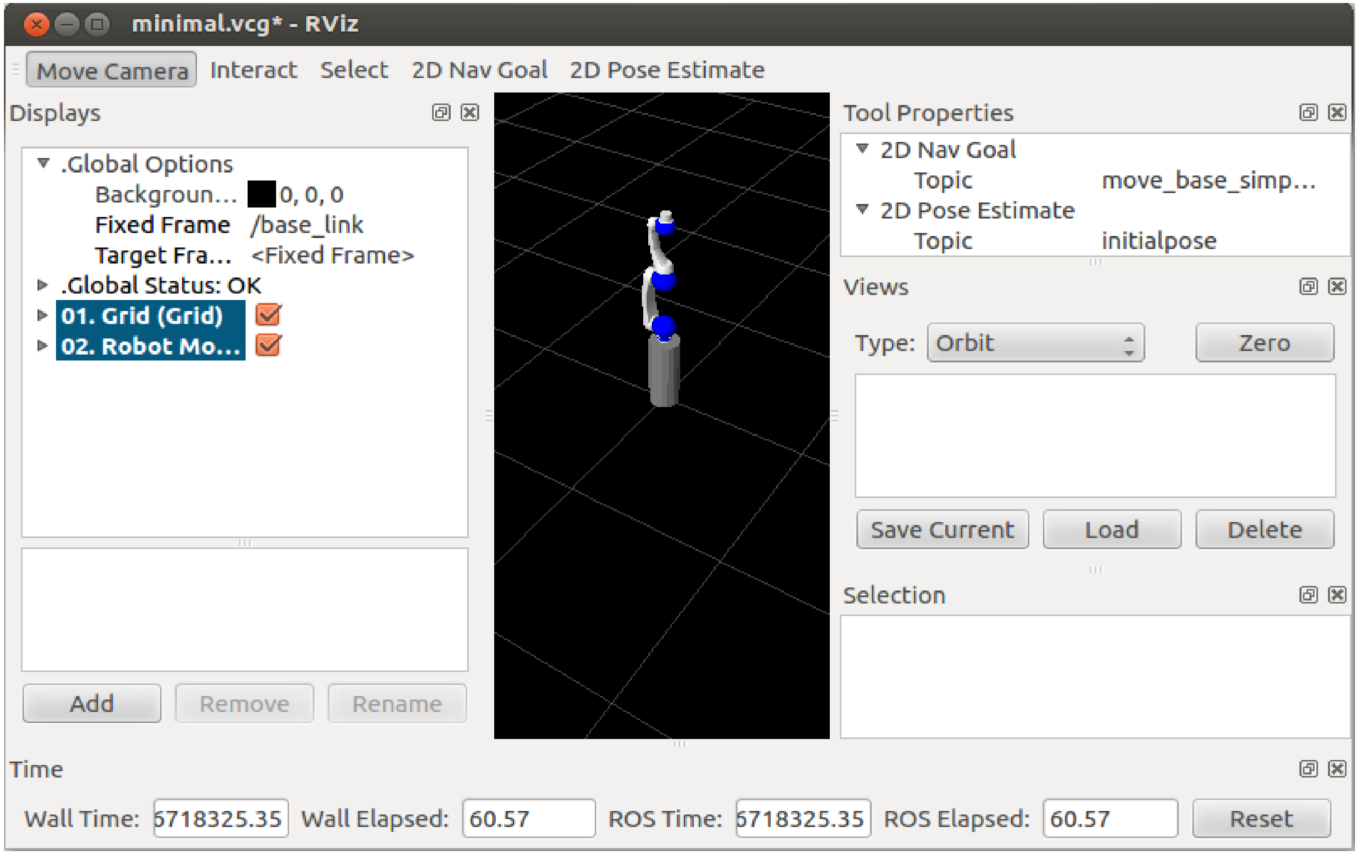
Connected to $ROS_MASTER_URI=http://localhost:11311

http://www.ros.org/wiki/cob_command_gui

© Fraunhofer IPA

ROS for Schunk components – rviz

- Tool to visualize robot model
- *"roslaunch schunk_bringup rviz.launch"*



minimal.vcg* - RViz

Displays

- .Global Options
  - Backgroun... ■ 0, 0, 0
  - Fixed Frame /base_link
  - Target Fra... <Fixed Frame>
- .Global Status: OK
- 01. Grid (Grid) ☑
- 02. Robot Mo... ☑

Tool Properties

- 2D Nav Goal
  - Topic move_base_simp...
- 2D Pose Estimate
  - Topic initialpose

Views

Type: Orbit

Selection

Time

Wall Time: 6718325.35 Wall Elapsed: 60.57 ROS Time: 6718325.35 ROS Elapsed: 60.57

http://www.ros.org/wiki/rviz

© Fraunhofer IPA

Thank you for your attention

ROS → SCHUNK®

Contact:

Christopher Parlitz
Christopher.Parlitz@de.schunk.com

www.schunk-modular-robotics.com
www.ros.org/wiki/schunk_modular_robotics