Automation vs Robotics

- Automation –Machinery designed to carry out a specific task
  - Bottling machine
  - Dishwasher
  - Paint sprayer

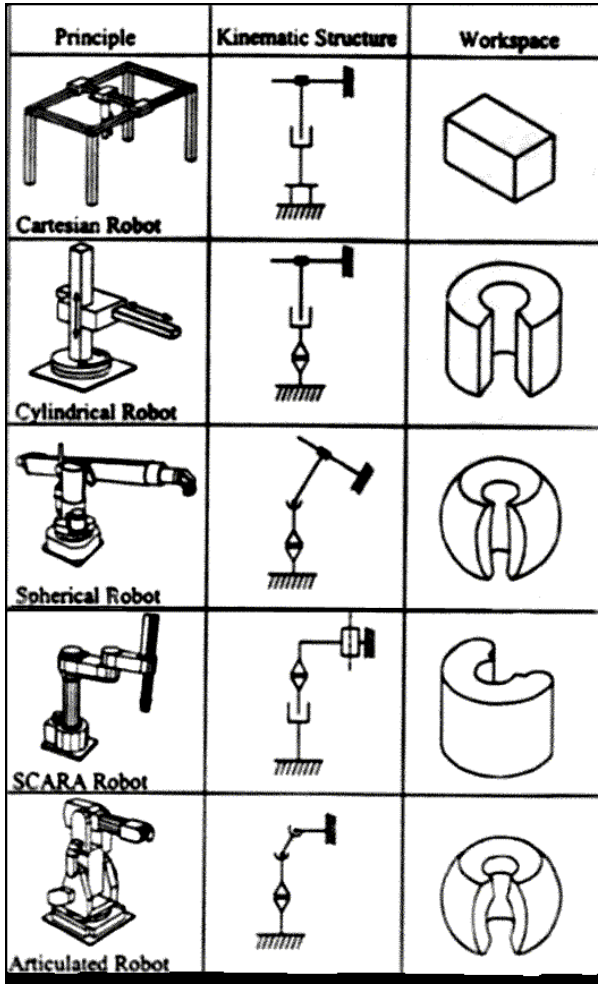
(These are always better than robots, because they can be optimally designed for a particular task).

- Robots – machinery designed

  to carry out a variety of tasks
  - Pick and place arms
  - Mobile robots
  - Computer Numerical Control machines

Measures of Performance

- Working volume
  - The space within which the robot operates.
  - Larger volume costs more but can increase the capabilities of a robot
- Speed and acceleration
  - Faster speed often reduces resolution or increases cost
  - Varies depending on position, load.
  - Speed can be limited by the task the robot performs (welding, cutting)
- Resolution
  - Often a speed tradeoff
  - The smallest step the robot can take



Performance (Cont.)

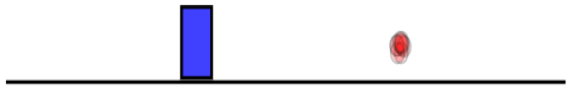
- Accuracy

  The difference between the actual position of the robot and the programmed position

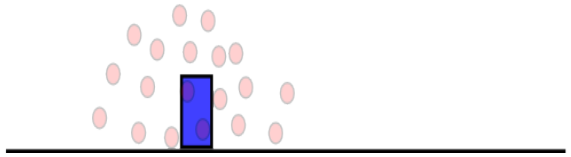
- Repeatability

  Will the robot always return to the same point under the same control conditions?

Low accuracy, high repeatability:



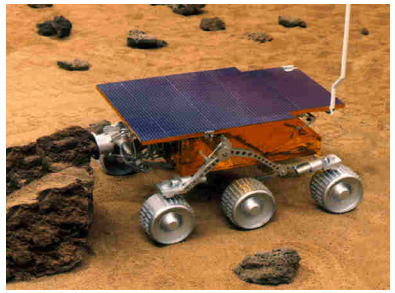
High accuracy, low repeatability:

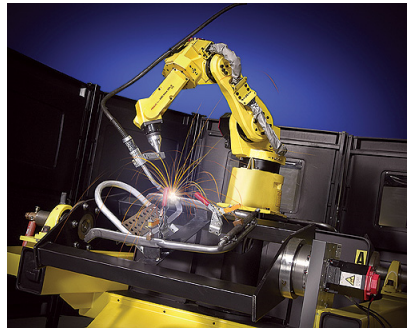


Robotic Task

Tasks which are:

- Dangerous
  - Space exploration
  - chemical spill cleanup
  - disarming bombs
  - disaster cleanup
- Boring and/or repetitive
  - Welding car frames
  - part pick and place
  - manufacturing parts.
- High precision or high speed
  - Electronics testing
  - Surgery
  - precision machining.





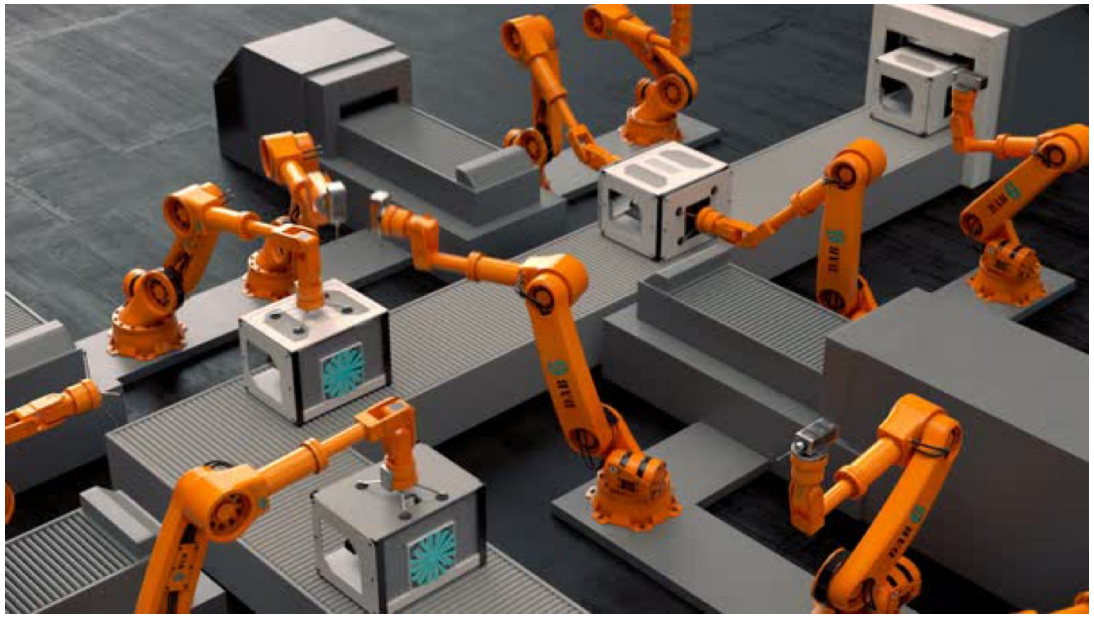
Industrial Robots

Material-Handling Robots

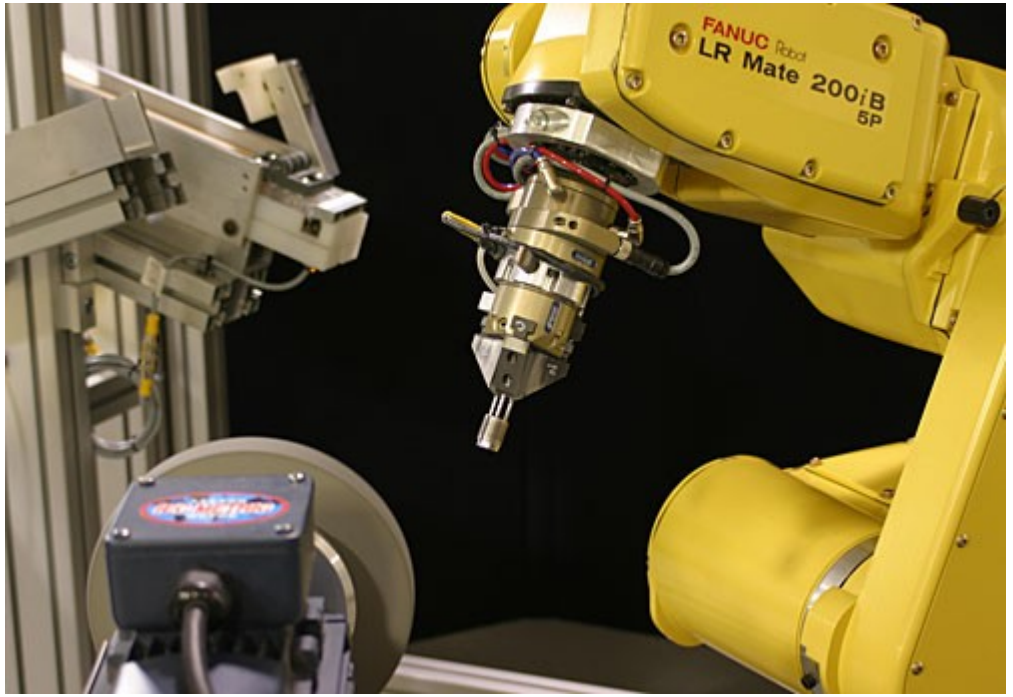
Processing Robots

Industrial Robots

Assembly Robots



Machining Robots



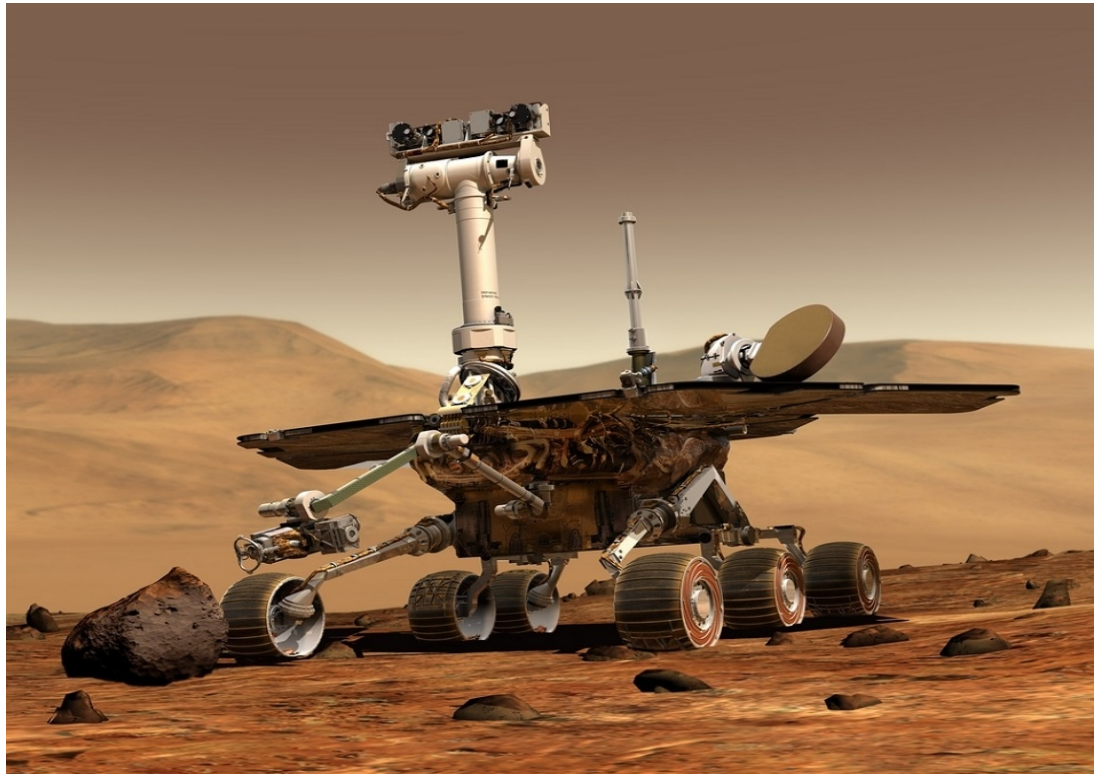
Domestic or household robots

- Medical robots

- Service robots

- Military robots

- Space robots



Miscellaneous Robots

- Agri-Bots

- Entertainment robots

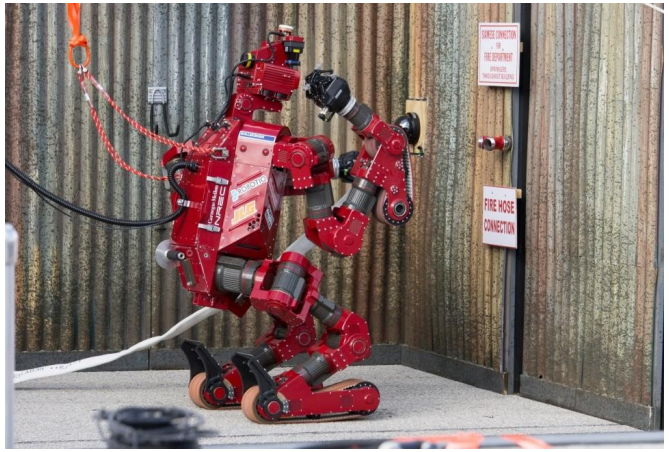
Current state of mobile robotics

Why should Robots be used?

> Dangerous
> Boring
> Unhealthy
> Expensive

Amazing Progress

> Autonomous cars
> DARPA Challenge with humanoid robots
> Navigation in human environments



CHIMP from CMU (Carnegie Mellon University)
http://www.darpa.mil

Overview

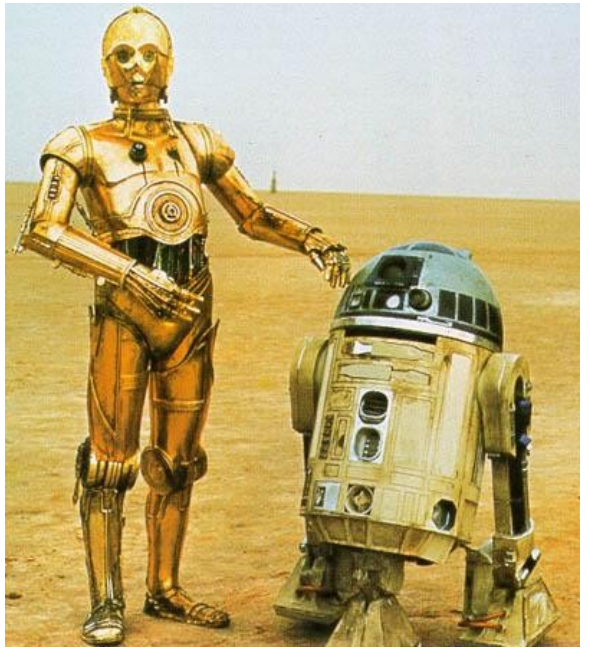
Great Expectations…

Serving? Working? Playing? Welding?
Manufacturing? Entertaining? Dancing? Singing?
Teaching? Mimicing? Rescuing?

Tasks for Mobile Robots

- Unsupervised Moving: **Locomotion**
- Obstacle Avoidance: **Perception**
- **Navigation**
  1. Localization
  2. Mapping
  3. Path Planning

Where am I and how to move where I should do my work?



What makes a Robot?

A Quick Intro to Mobile Autonomous Systems

CPU:

> Powerfull, small, low energy consumption

Sensors:

> IMU, Magnetometer, GPS, RGBD Camera, LIDAR, Stereo Vision

Actuators:

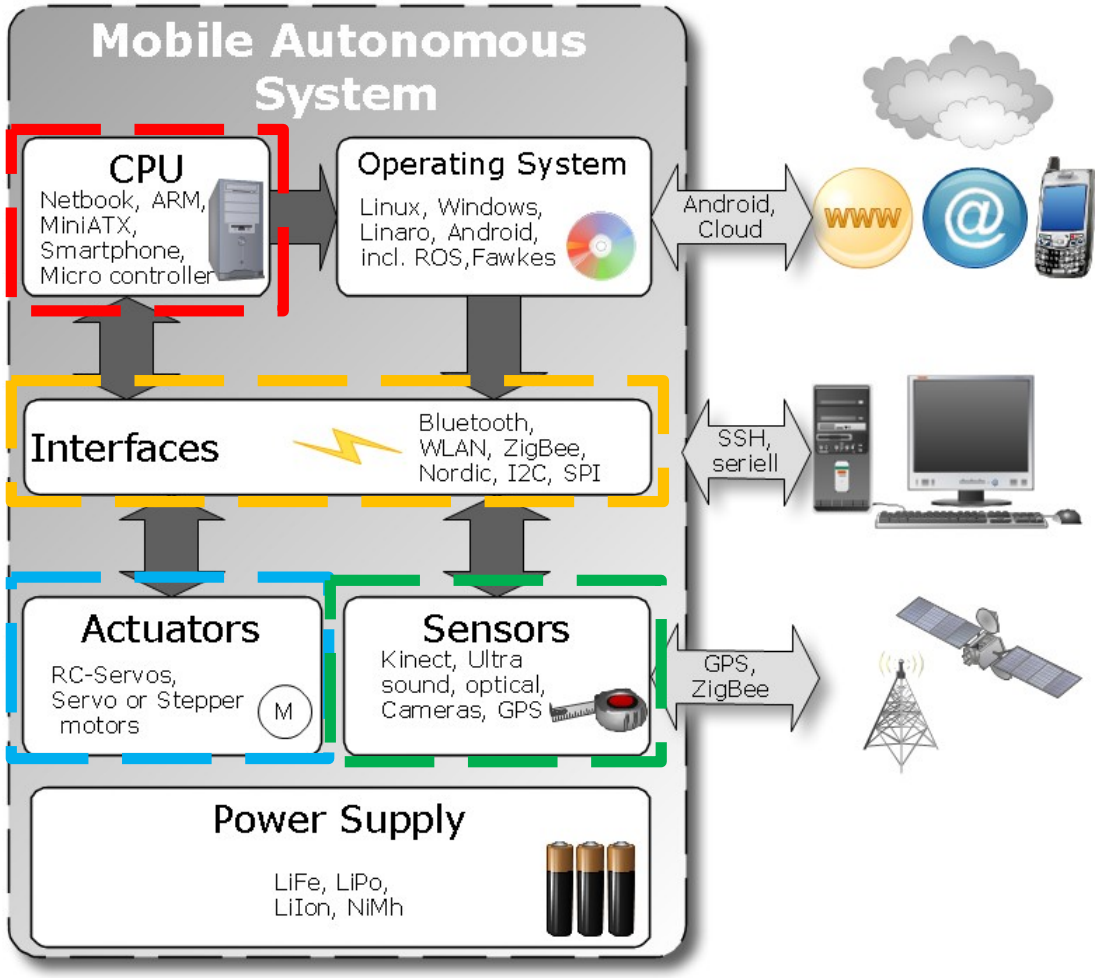
> RC Servos, Brushless Motors, Steppers

Interfaces:

> UART, I2C, SPI, Analog

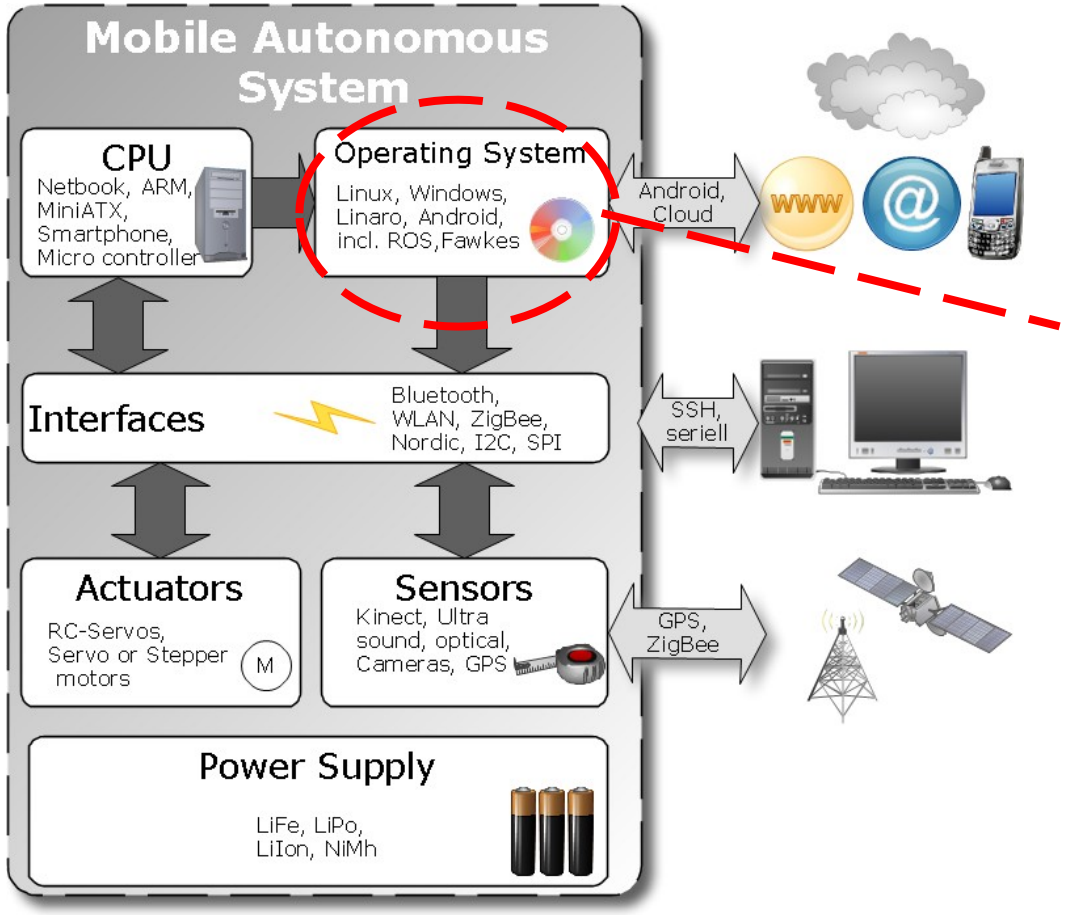
Power Supply:

> Standard



Why a special Robot Operating System?

Better: Middleware or Framework



Main Tasks for the OS

- Low Level device control
- Hardware abstraction
- Message Passing
- Parallel Execution of Tasks
- Distributed System
- High Level functionality like Collision Avoidance, Navigation, Motion Planning, Manipulation
- Artificial Intelligence
- Re-Use of code

What are the main tasks for our Robots?
What do we expect from a Robotics Framework?

- Communication
- Perception
- Collision Avoidance
- Motion Planning
- Navigation

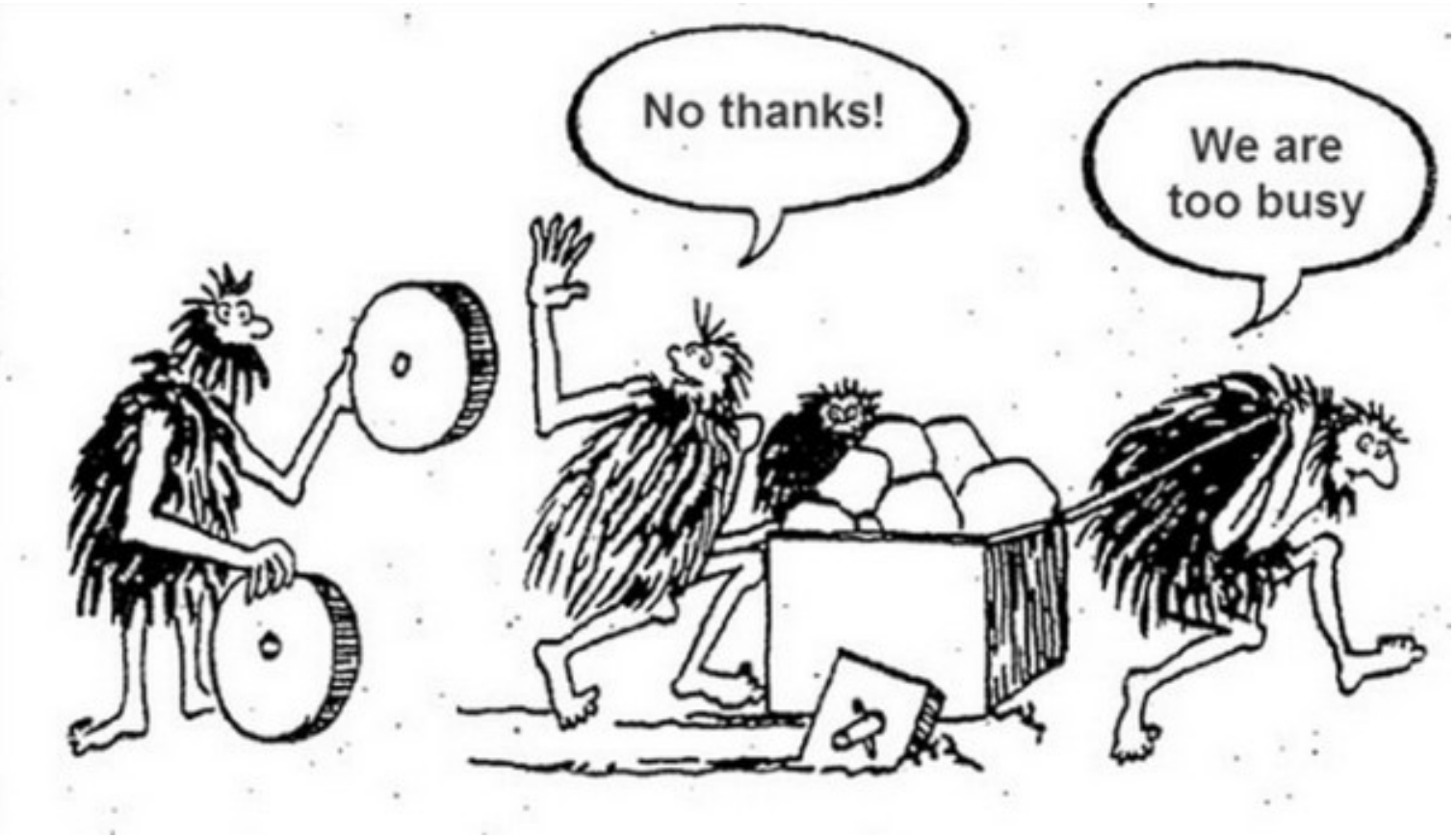
- Flexible Manipulation
- Industrial interface?

- Less Programming Work
- Simulation capabilities
- Community Support
- Low cost

Perhaps I can help…

ROS.org

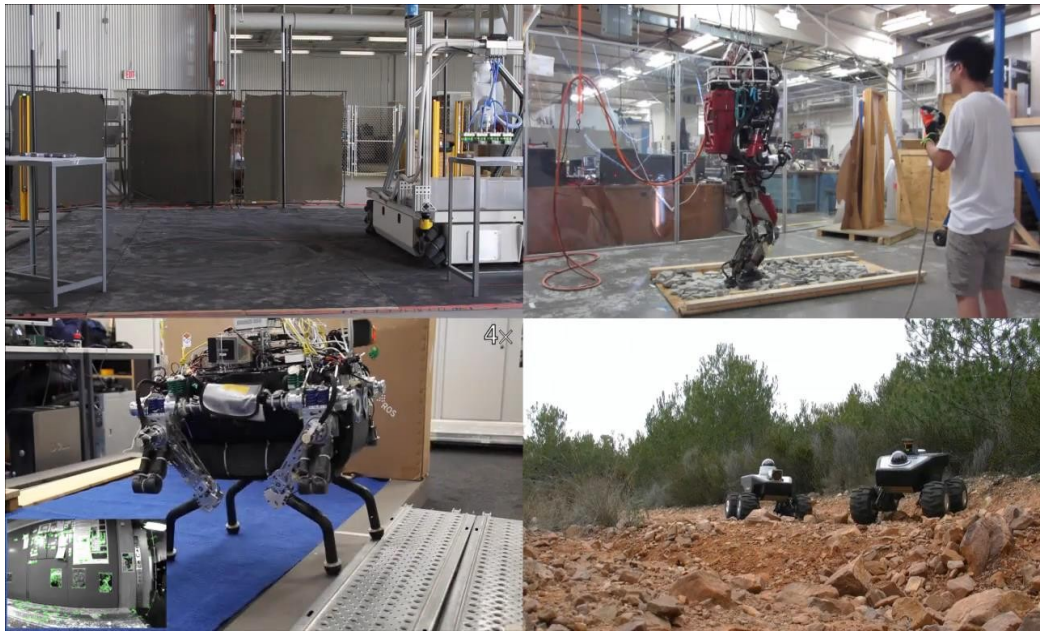
ROS Main Goal
Stop Reinventing the Wheel



The ROS goal is to provide a **standard for robotics software development, that you can use on any robot**.

History of ROS

- Originally developed in 2007 at the Stanford Artificial Intelligence Laboratory
- Since 2013 managed by OSRF
- Today used by many robots, universities and companies
- De facto standard for robot
- programming



What is ROS?

Put a smile on the robot‘s face

ROS

> Open-source, meta-operating system for robots.

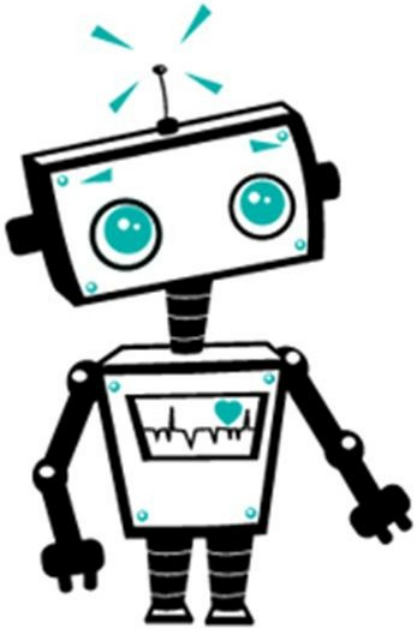
> Hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes and package management.

> Tools and libraries for obtaining, building, writing, and running code across multiple computers in C++, Python, Octave, LISP, and Java.

> Supports different Operating systems and Hardware platforms but works best under Ubuntu Linux.

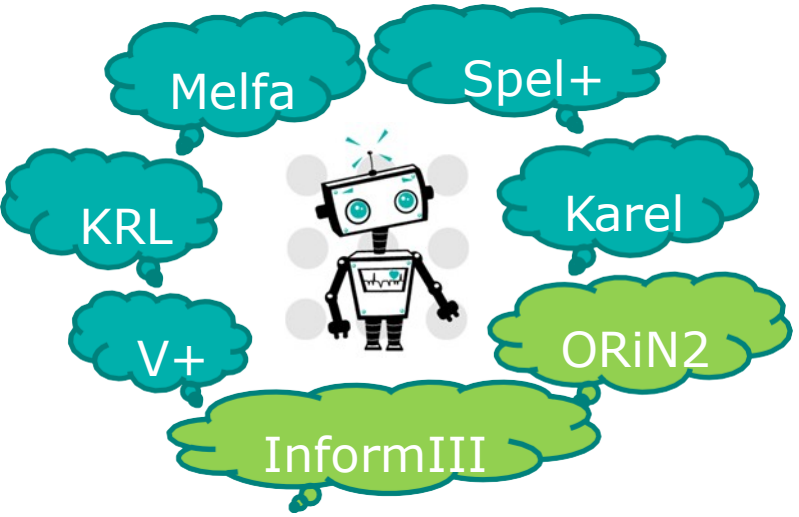
> Turns out to be more and more a standard for mobile robots.

> Spreading into the industrial environment with ROS industrial which is based on ROS



Why ROS industrial?

A quick look at industrial robots today…

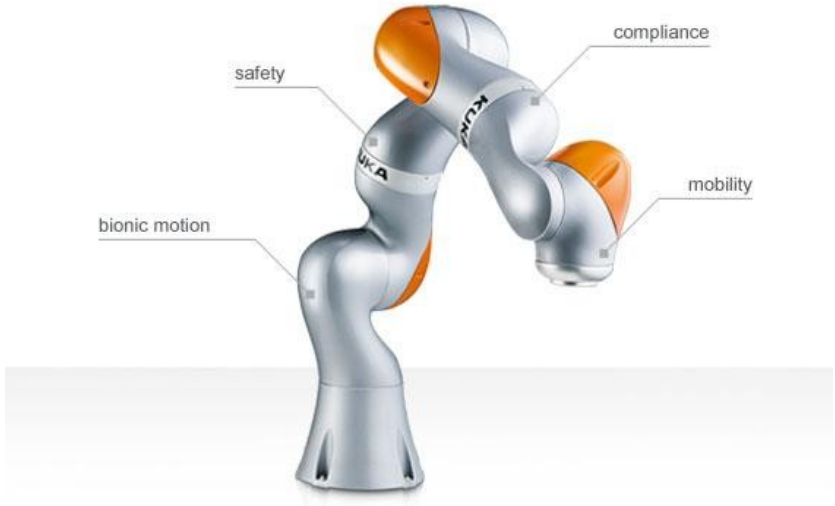


Only deterministic approach

> Active Motion planning by e.g. Robot vision possible but very expensive and not manufacturer independant
> No possibilities of planning components like collision avoidance with alternative movement
> Robot with internal sensors and high payload/weight ratio very expensive

Difficult Programming

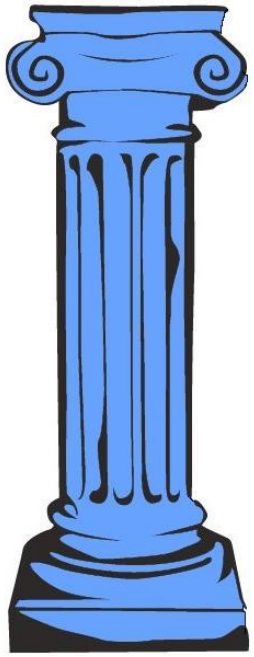
> A lot of different languages
> Often no interface to high level languages like C++
> No toolbox for standard problems
> Perception difficult and not too often



Kuka Light weight robot

Level of Concepts

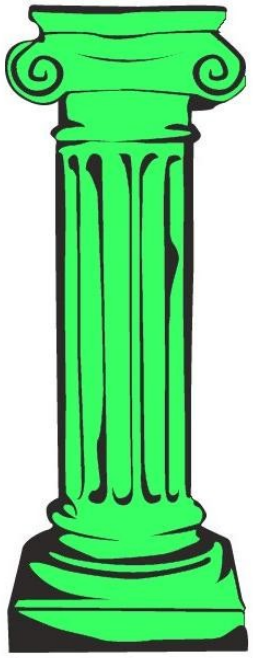
Higher-Level Concepts



Filesystem

www.wadeco.de

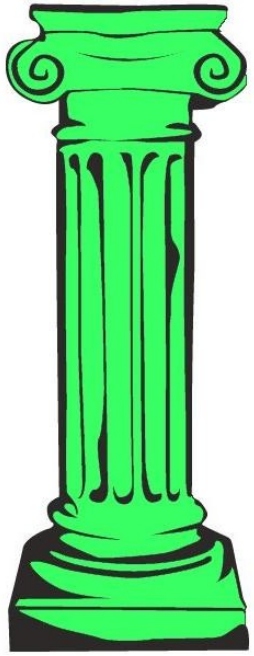
Computation Graph



Community

The first level …

… Community

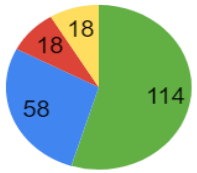


www.wadeco.de

ROS Community Worldwide

ROS Users of the World

- GREEN - School
- BLUE - Company
- RED - Research Institute
- YELLOW - Other
- (white - unknown)



Add to or edit the map by changing the yaml files in this repository, or by emailing the map maintainer.

> ROS has grown to include a large community of users worldwide
> The Community is the major pillar of ROS

ROS Community Resources

ros.org

ROS Website

ROS Answers

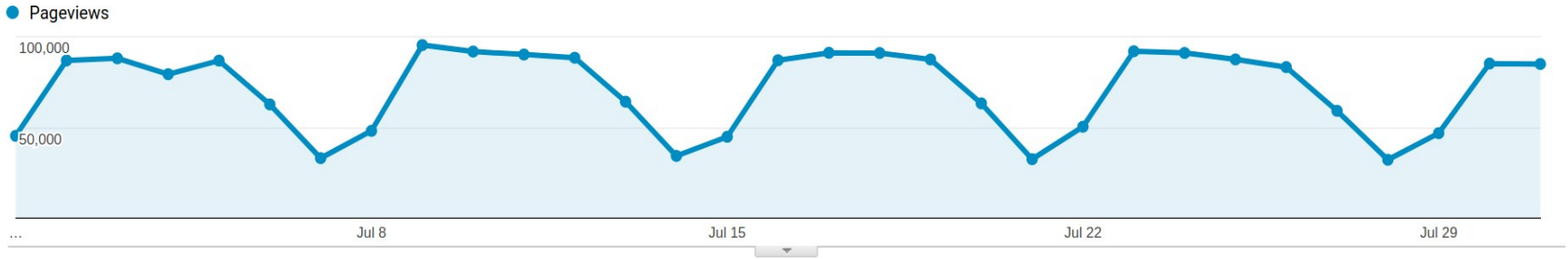
ROS Wiki

News

ROS News

http://www.ros.org

wiki.ros.org Visitors: July 2018



Pageviews	Unique Pageviews	Avg. Time on Page	Bounce Rate	% Exit
2,204,869	1,682,363	00:02:54	45.81%	29.18%

Site Content

Page

Page Title

Site Search

Search Term

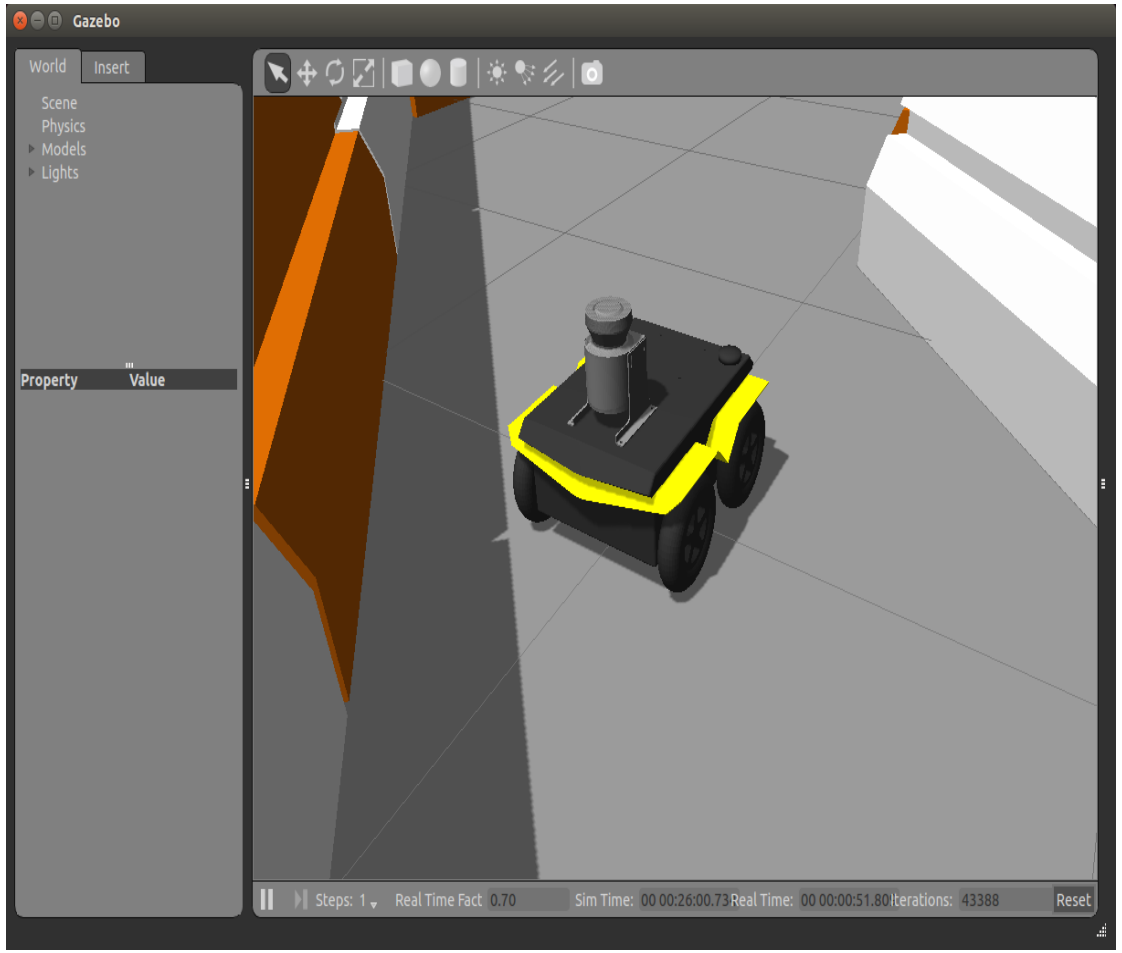
Events

Site: wiki.ros.org
Source: Google Analytics
Annual Growth: 21%

	Page	Pageviews	% Pageviews
1.	/ROS/Tutorials	103,620	4.70%
2.	/	62,152	2.82%
3.	/kinetic/Installation/Ubuntu	55,985	2.54%
4.	/ROS/Installation	43,803	1.99%
5.	/kinetic/Installation	37,048	1.68%
6.	/cn/ROS/Tutorials	32,001	1.45%
7.	/ROS/Tutorials/InstallingandConfiguringROSEnvironment	29,558	1.34%
8.	/ROS/Tutorials/CreatingPackage	25,635	1.16%
9.	/ROS/Tutorials/WritingPublisherSubscriber(c++)	21,841	0.99%
10.	/ROS/Tutorials/NavigatingTheFilesystem	17,345	0.79%

answers.ros.org: activity

- Total questions:
  - 42,360 (17% increase)
- Answered questions:
  - 29,269 (18% increase)
- New questions in July:
  - 648 (20.9 / day, 27% increase)



Source: answers.ros.org
Sampled: 2018-09-28

Research use

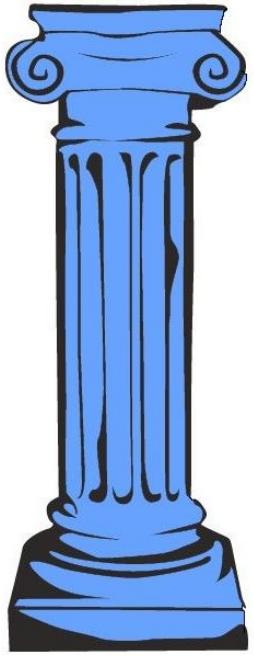
Total number of papers citing “ROS: an open- source Robot Operating System” (Quigley et al., 2009):

4806 (29% increase)

ROS based ATLAS Robot

Source: Google Scholar 2018-09-28

ROS Industrial Consortium



www.wadeco.de

The next level …

…Filesystem

- Neo : [aiming at a helicopter] Can you fly that thing?
- Trinity : Not yet. [picks the phone, calls Tank]
- Trinity : Tank, I need a pilot program for a B-212 helicopter. [Tank loads the program in Trinity’s brain]
- Trinity:[to Neo] Let’s go.

Credits : Movie MATRIX

ROS Filesystem Package

- Software in ROS is organized into **packages**
  - smallest build part in ROS
  - consists anything what makes it a useful module, e.g.:
    - hardware drivers,
    - algorithms,
    - visualization tools,
    - libraries …

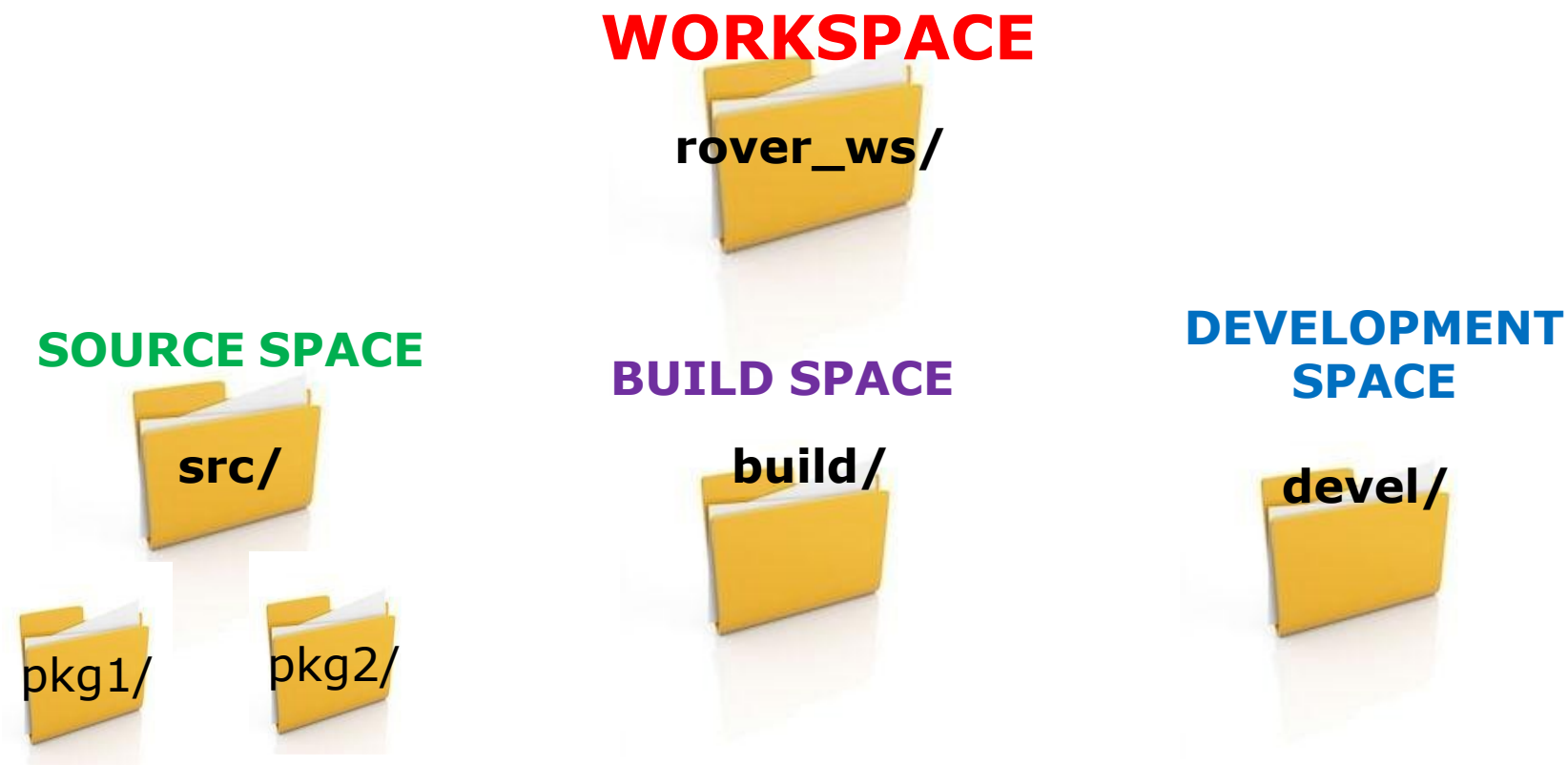
Install Options

Debian Packages:	Source Repositories:
• automatic installation • stable versions • prebuilt binaries	• “latest” code • manual compilation • allows code adjustments

ROS Filesystem

Structure of „workspaces“

> A catkin workspace is a folder where you modify, build, and install catkin packages.
>
> - Specific workspace layout:



ROS Filesystem

Structure of “workspace” in detail

- Source Space:
  - Contains the source code of catkin packages.
  - Space where you can extract/checkout/clone/create source code for the packages you want to build

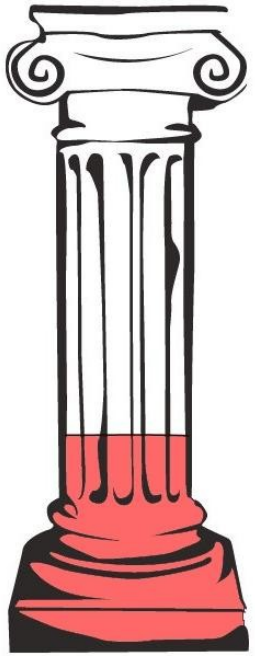
- Build Space:
  - Space where CMake is invoked to build the catkin packages
  - CMake and catkin keep their cache information and other intermediate files here

- Devel Space:
  - Space where **built targets** are placed prior to being installed

- to compile a package, also python based ones:

catkin_make

The next level …

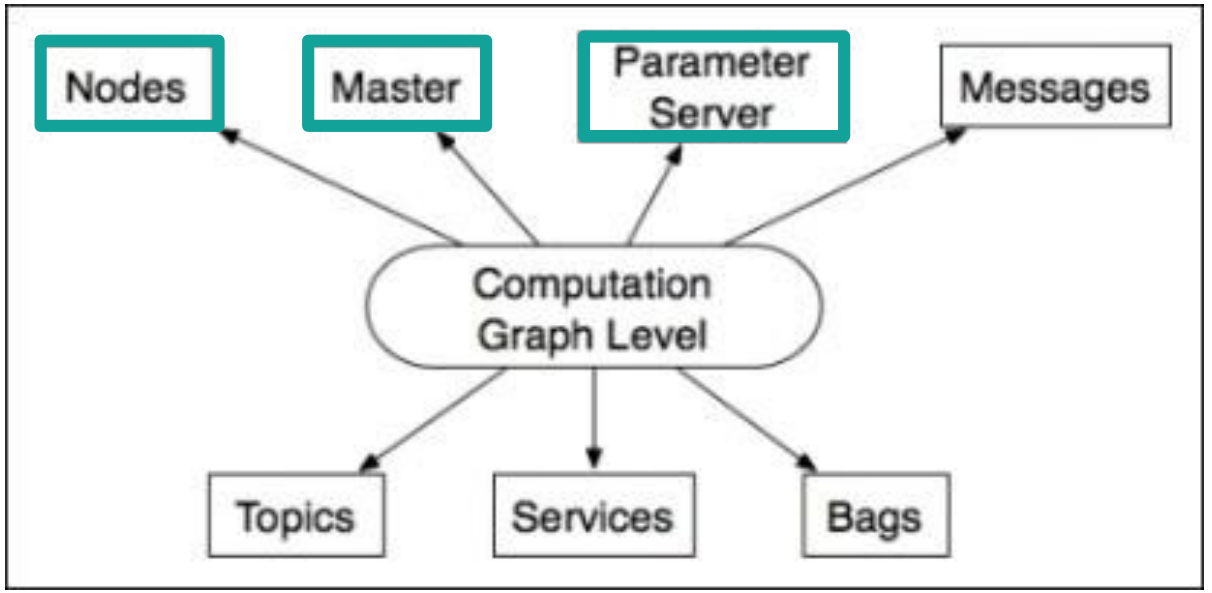


www.wadeco.de

… Computation Graph

ROS Computation Graph Concept

> The *Computation Graph* is the peer-to-peer network of ROS processes that are processing data together.

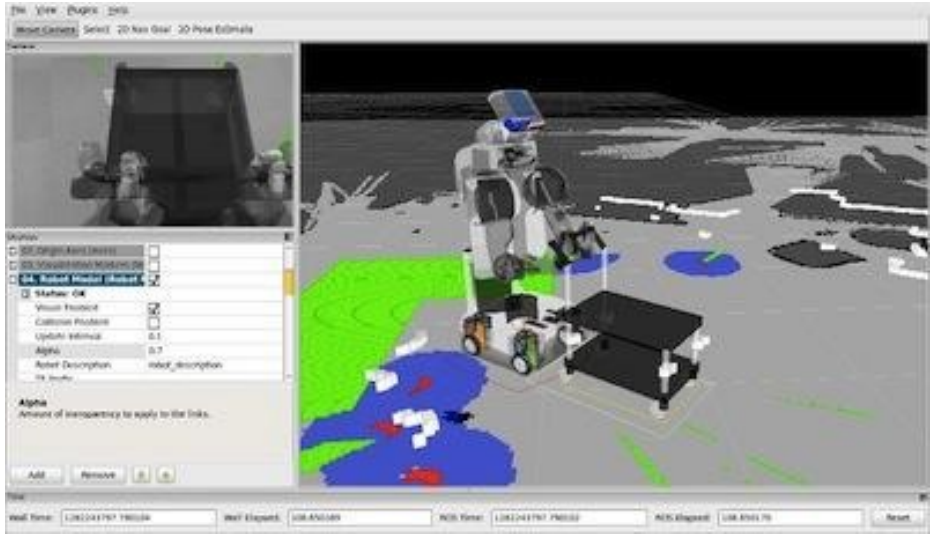


Learning ROS for Robotics Programming; Enrique Fernández, Aaron Martinez

> Reflects the hole communication in the ROS system

ROS Computation Graph RViz

- 3D visualization tool of ROS
- Can be used to visualize any type of sensor data or algorithm results, e.g.:
  - images,
  - laser scan data,
  - imu data,
  - transformations,
  - maps,
  - robot models,
  - planned paths,
  - etc.

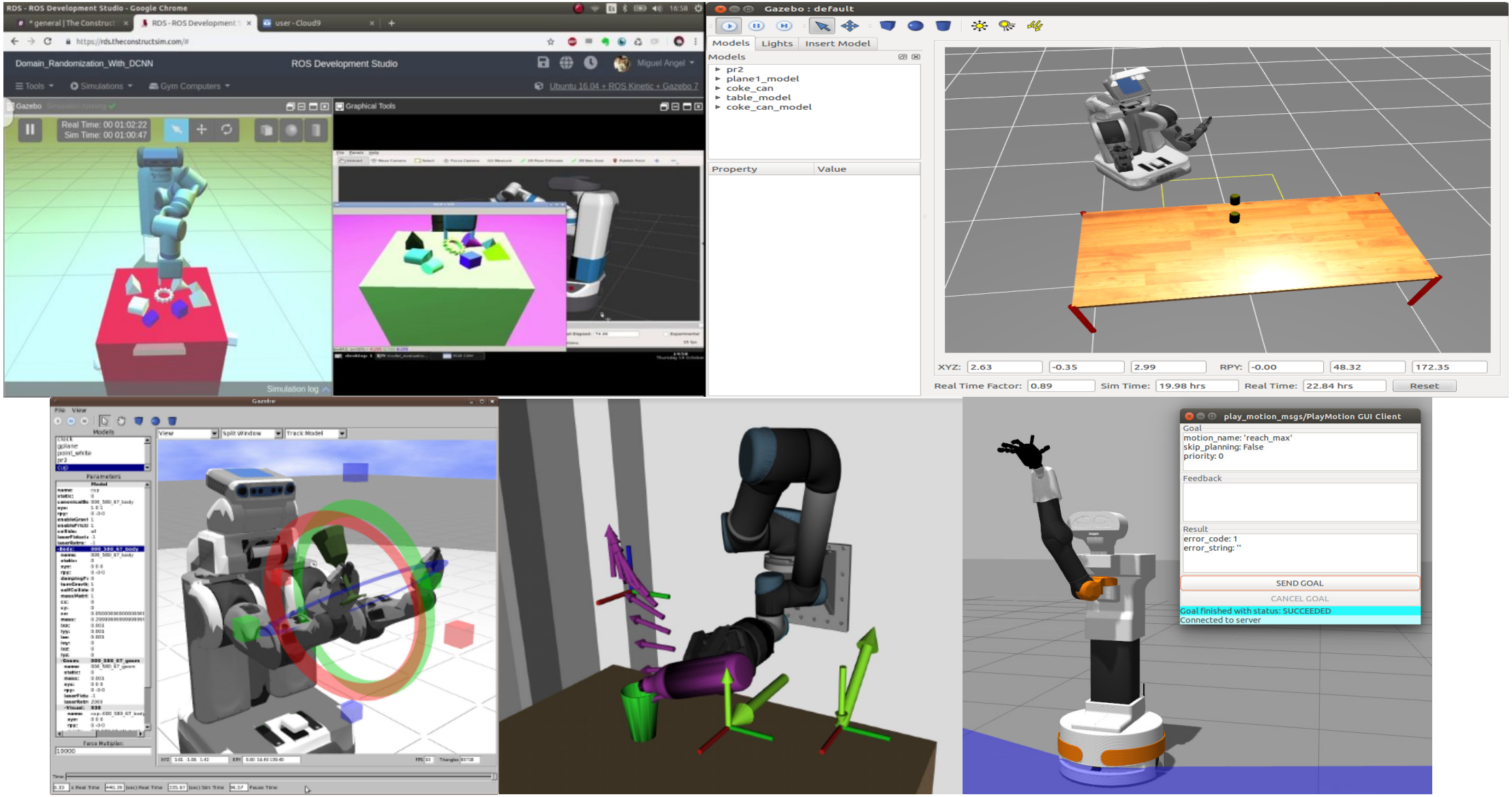


https://en.wikipedia.org/wiki/Robot_Operating_System

- To run RViz:

```
rosrun rviz rviz
```

ROS Simulation Environment GAZEBO



ROS Simulation Environment GAZEBO

ROS Computation Graph Node

- Executable part of ROS:
  - Scripts for Python
  - Compiled source code for C++
- Process that performs computation
- Meant to operate at fine-grained scale

  - To run a node:

rosrun package_name node_name

→ a robot control system will usually comprise many nodes

ROS Computation Graph Master

- One Master per system
- Registry for:
  - Nodes
  - Topics
  - Services
  - Parameters
- Part of the *roscore*

→ essential for all kind of processing and communication

- To start the roscore:

roscore

http://starwars.wikia.com/wiki

ROS Computation Graph Parameter Server

> Allows data to be stored by key in a central location
> Globally viewable
> Not designed for high-performance

→ static, non binary data (configuration data)

> Examples

- ➢ adjustable hardware drivers:
  - webcams,
  - joysticks, …
- ➢ adjustable algorithms:
  - path planning,
  - sensor fusion, …

More flexibility

ROS Computation Graph Launch Files

- XML based
- Starts the **roscore**
- Tool to manage a robotic system:
  - starting of multiple nodes
  - setting of parameter values
  - including other launch files
  - defining of namespaces
  - respawning of died nodes

http://starwars.wikia.com/wiki

- To run a launch file:

```
roslaunch package_name launch_file_name
```

ROS2 Robotics de facto Standard

- The purpose of ROS 2 is to offer a standard software platform to developers across industries that will carry them from research and prototyping through to deployment and production. ROS 2 builds on the success of ROS 1, which is used today in myriad robotics applications around the world.
- Do you really want to create your own object recognition, speech recognition, mapping, localization, arm/gripper control, or collision detection code from scratch? Or would you rather reuse, and/or expand existing code/packages to better fit your application?

ROS2 Advantages over ROS1

- **Shorten Time to Market**

ROS 2 provides the robotics tools, libraries, and capabilities that you need to develop your applications, allowing you to spend your time on the work that is important for your business.

- **Designed for Production**

Drawing on a decade of experience in establishing ROS 1 as the de facto global standard for robotics R&D, ROS 2 was built from the ground up to be industry-grade and used in production, including high reliability and safety critical systems.

- **Multi-Platform**

ROS 2 is supported and tested on Linux, Windows, and macOS, allowing seamless development and deployment of on-robot autonomy, back-end management, and user interfaces.

- **Built on Industry Standards**

The default communications method in ROS 2 uses industry standards like IDL, DDS, and DDS-I RTPS, which are already widely deployed in a variety of industrial applications, from factories to aerospace.

Should you use ROS?

- USE CASE – DEVELOP A ROVER
- NEEDS : Have multiple sensors like camera, LIDAR's, force/torque sensor. Multiple actuators etc.

ROS	NON ROS
Almost all state of art sensors are plug and play having ROS Packages which can directly connect to the PC without need of writing drivers or need of data pre-processing.	Develop driver compatibility for the sensors used. Do your own sensor data processing. Yes it could make it more efficient but at the higher development cost?
SLAM and MAPPING Application Packages available. So you can actually concentrate on modifying and using them to develop your own Application.	Reinventing the wheel for SLAM and MAPPING algorithm could make it more efficient but is that your aim of the project?

BREAK

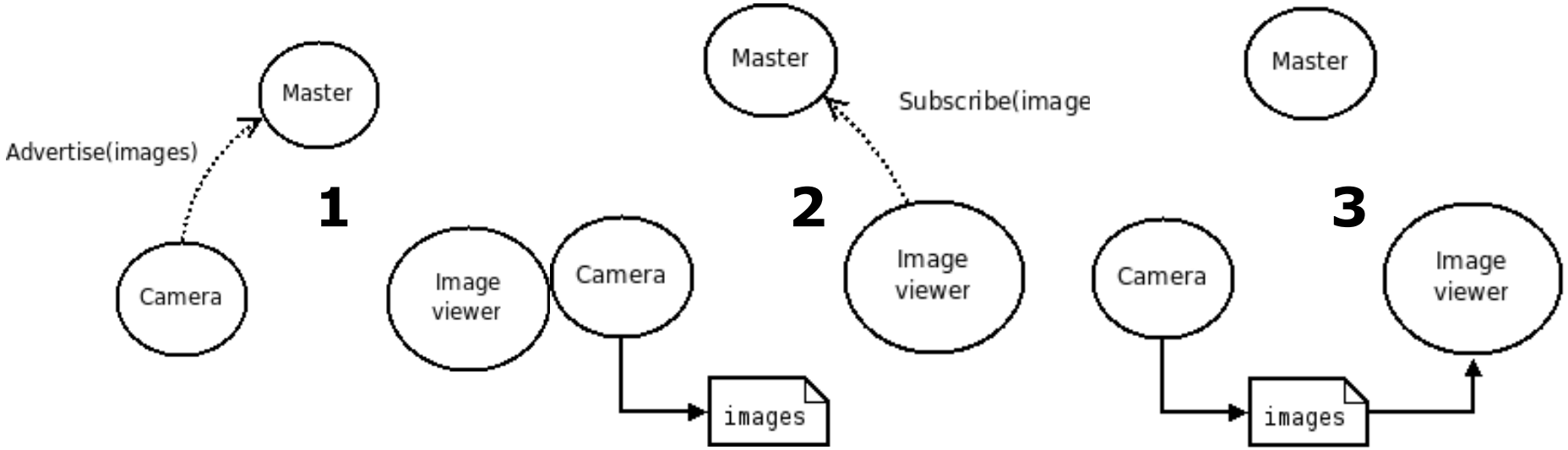
Practicals and Demo to Follow

Tutorial 1

ROS Topics

ROS Computation Graph Master

> ROS Master provides naming and registration services:
  - **Nodes** Topics Services Parameters



Camera likes to publish an image

Viewer likes to display an image

Master notifies Camera and viewer

After the nodes have located each other they communicate "peer-to-peer"

ROS Computation Graph Topics

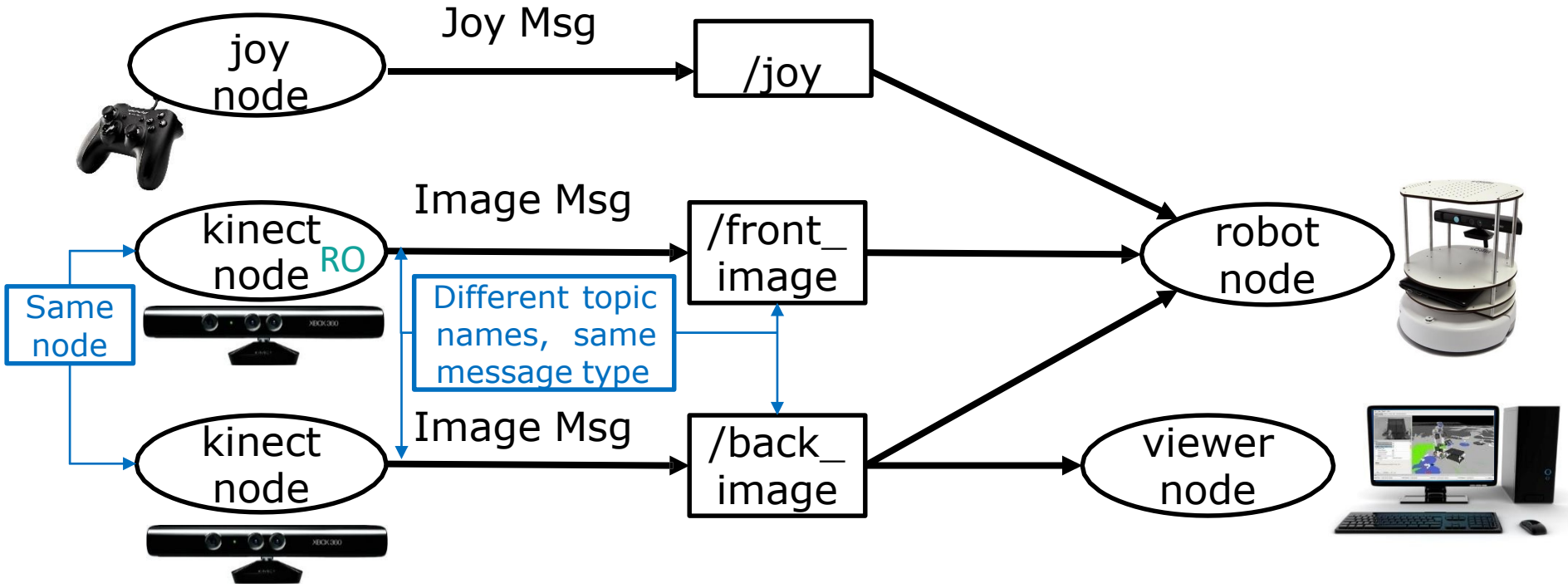
Node A

Go there

Node B

I can see

ROS Computation Graph Topics



- **Topics** are named software buses over which nodes exchange messages
- A node sends out a message by publishing it to a given topic
- A node that is interested in a certain kind of data will *subscribe* to the appropriate topic
- A single node may publish and/or subscribe to multiple topics

Tutorial 2

ROS
Publisher
Subscriber

ROS Computation Graph Publisher

Publisher

Definition

```
pub = rospy.Publisher('topic_name', std_msgs.msg.String, queue_size=1)
```

Fill message with data

```
ros_string      = std_msgs.msg.String()
ros_string.data = "Hello ROS!"
```

Hint: Queue size for asynchronous publishing. Too large queue will lead to large latency. One lagging subscriber can block all publishing!

Publication

```
pub.publish(ros_string)
```

ROS Computation Graph Subscriber

Subscriber

Definition

```
rospy.Subscriber('topic_name', std_msgs.msg.String, callback)
```

Topic name

Message type

Invoked function

Accessing to incoming message data

```
def callback(msg):
    value = msg.data
```

Always called if new data appear

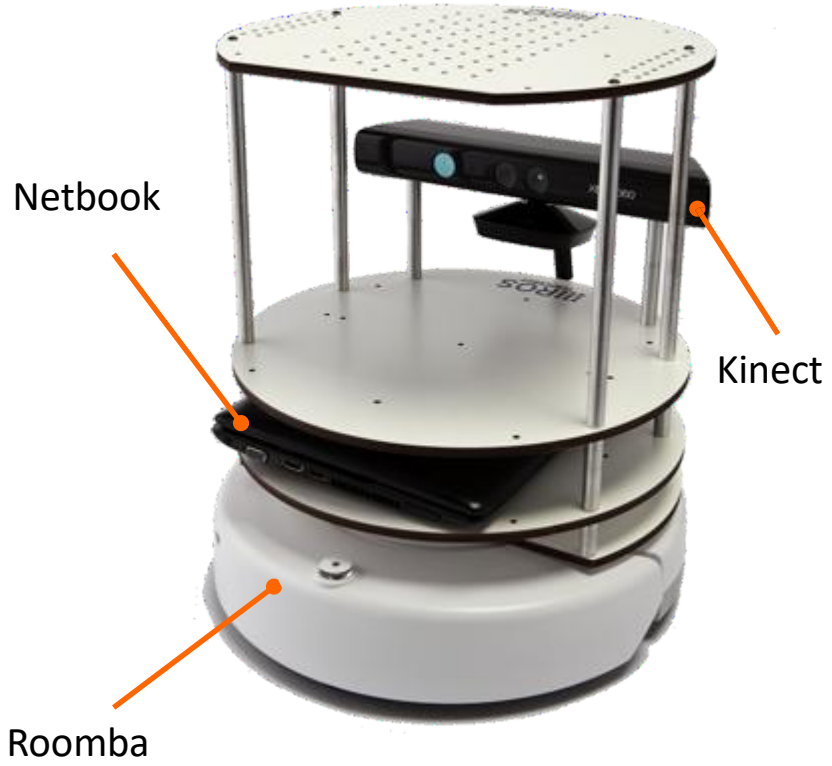
First argument is the incoming message

Tutorial 3

ROS Mapping and Navigation

TurtleBot3

Introduction and Demo



Much more than a vaccum cleaner…

> A Complete system
> Runs out of the box
> Many tutorials like: teleoperation, pointcloud,
> Perception, navigation and much more.
> Controllable via Android based systems

If you are not interested in hardware design, just software, this is the way to go!