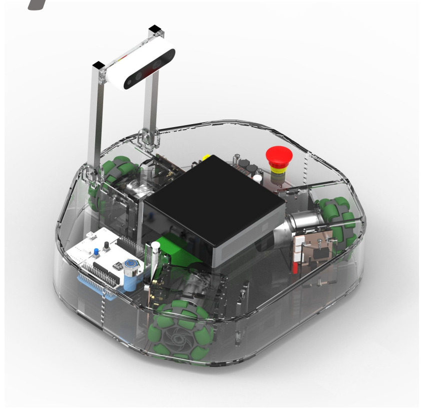
ROS Training for Industry



Veiko Vunder

September 16-20, 2019

Tartu, Estonia

Trainers:

Veiko Vunder	Organizer, Lecturer
Houman Masnavi	Masters student in Computer Engineering
Karl Kruusamäe	assoc prof in robotics (IMS robotics)
Robert Valner	PhD student in Science & Technology
Madis Kaspar Nigol	MSc in Computer Engineering & Robotics

Learning objectives

1) Introduce the fundamental concepts of ROS

2) Practical experience in setting up ROS and using its tools

3) Demonstrate how ROS interacts with real hardware

Learning outcomes:

1) knows ROS command line tools and syntax;
2) can implement publisher/subscriber structures for reading sensor data and controlling the robots;
3) can implement ROS-based solutions for most common robotics problems, e.g., coordinate transformation, path-planning, inverse kinematics, and collision-free motion planning;
4) able to use ROS packages for mapping and navigating using simulated and real robots.

Acknowledgements!

- The training is supported by ROSIN project.

- This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement no. 732287.

Agenda: Day 1 (16.09)

- 09:15 Welcome and System Setup
- 10:00 Linux Introduction and Shell Basics
- 10:30 Coffee Break
- 10:45 Workshop: Linux & Shell
- 12:00 Lunch Break
- 13:00 ROS Introduction, Basic Concepts, ROS Filesystem
- 14:30 Coffee Break
- 14:45 Workshop
  - ROS Environment
  - Navigating ROS filesystem: rospack find, roscd, ...
  - Running ROS nodes
  - Teleop with Clearbot robots
- 17:00 End of Day 1

Agenda: Day 2 (17.09)

- 09:15 ROS Build/Debug/Visualization Tools
- 10:15 Coffee Break
- 10:30 Workshop
  - Catkin workspace, ROS package, Creating a node
  - Publisher & Subscriber
  - Rqt & RViz Visualization
- 12:00 Lunch Break
- 13:00 ROS Programming: Messages, Services, Actions, Launch files
- 14:30 Coffee Break
- 14:45 Workshop:
  - Parameters & Launch files
  - Messages & Services
- 17:00 End of Day 2

Agenda: Day 3 (18.09)

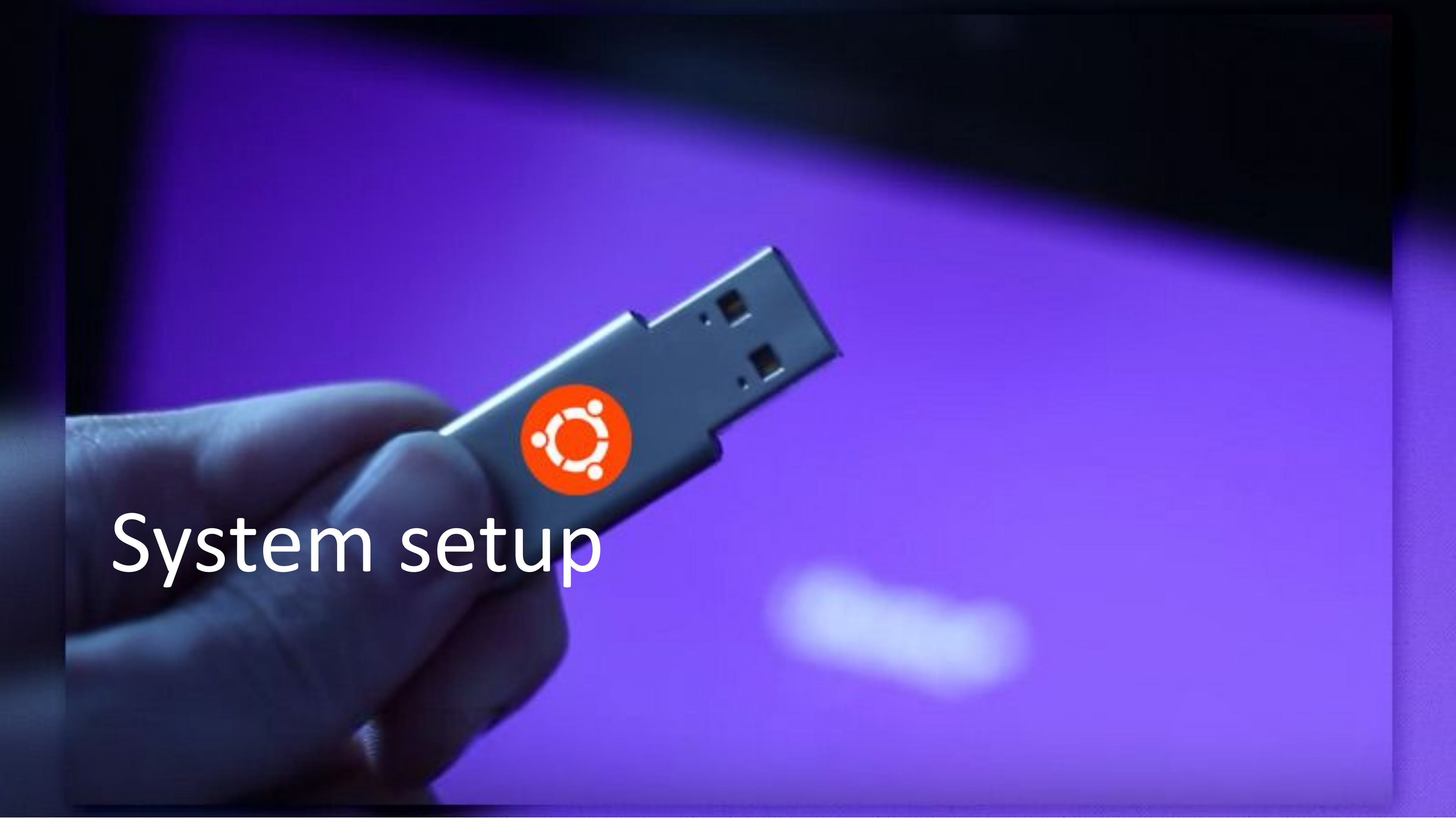
- 09:15 Hardware & drivers
- 10:15 Coffee Break
- 10:30 Workshop: Implementing ROS driver for Custom Hardware
  - Write driver for Arduino Sonar
  - Publish sonar range, IMU orientation, and visualize in RViz
- 12:00 Lunch Break
- 13:00 ROS Testing Tools & Continuous Integration
- 14:30 Coffee Break
- 14:45 Workshop
  - write tests and documentation for the ongoing package
  - 17:00 End of Day 3

Agenda: Day 4 (19.09)

- 09:15 Transforms in ROS, Gazebo
- 10:15 Coffee Break
- 10:30 Workshop: static TF, broadcaster programming
- 12:00 Lunch Break
- 13:00 Localization, Mapping, SLAM, Navigation with Path Planning
- 14:30 Coffee Break
- 14:45 Workshop
  - 2D mapping in Gazebo simulation
  - 2D mapping and navigation with Clearbot
  - 3D mapping on ClearBot
- 17:00 End of Day 4

Agenda: Day 5 (20.09)

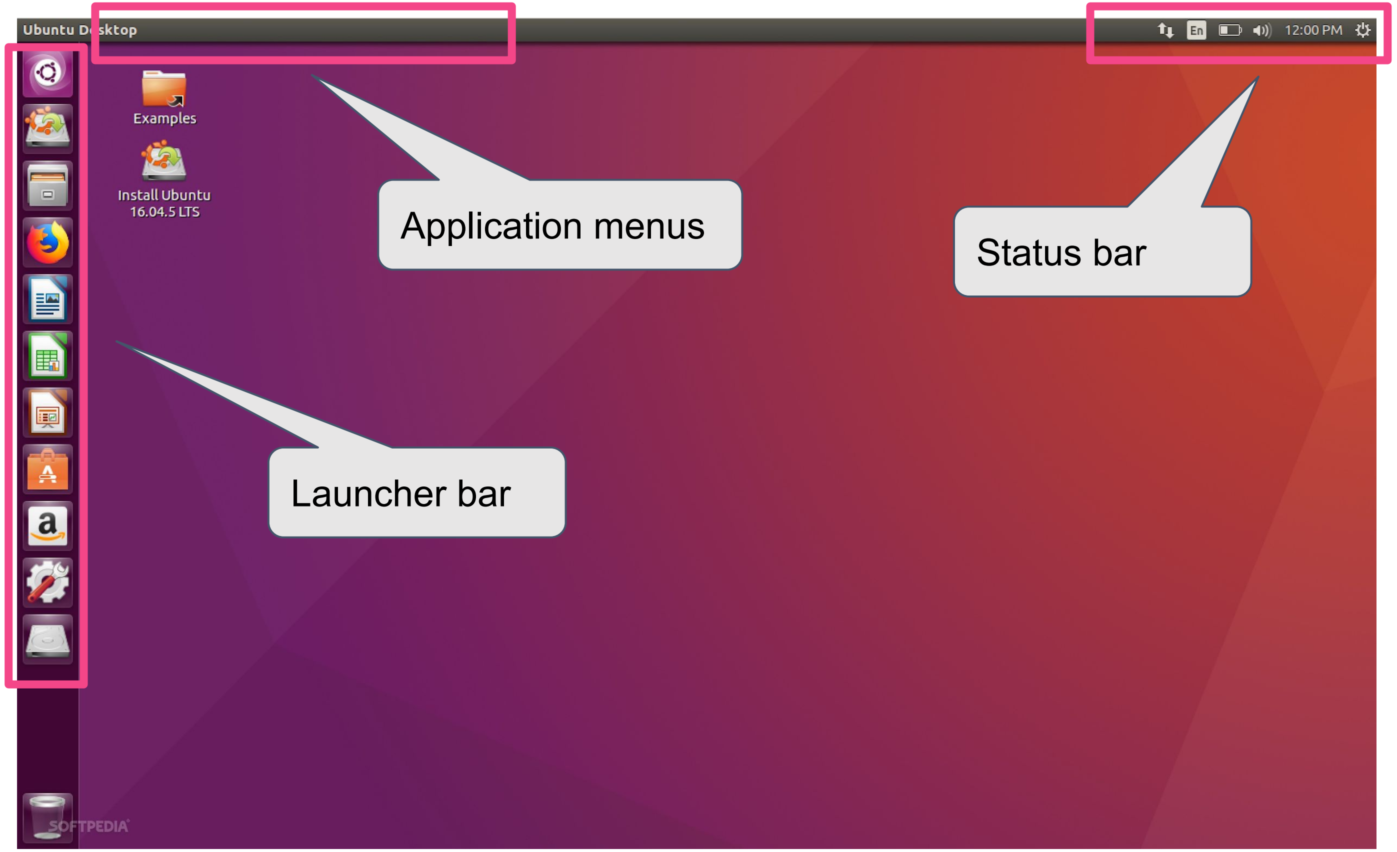
- 09:15 Robot Description (URDF), MoveIt!
- 10:00 Coffee Break
- 10:10 Workshop
  - MoveIt GUI
  - URDF
  - MoveIt Setup Assistant
- 12:00 Lunch Break
- 13:00 Workshop: MoveGroup C++ Interface
- 14:30 Coffee Break
- 14:45 Workshop: Motion planning with multiple robots
- 16:15 Conclusions, feedback, ROS2
- 17:00 End of Day 5



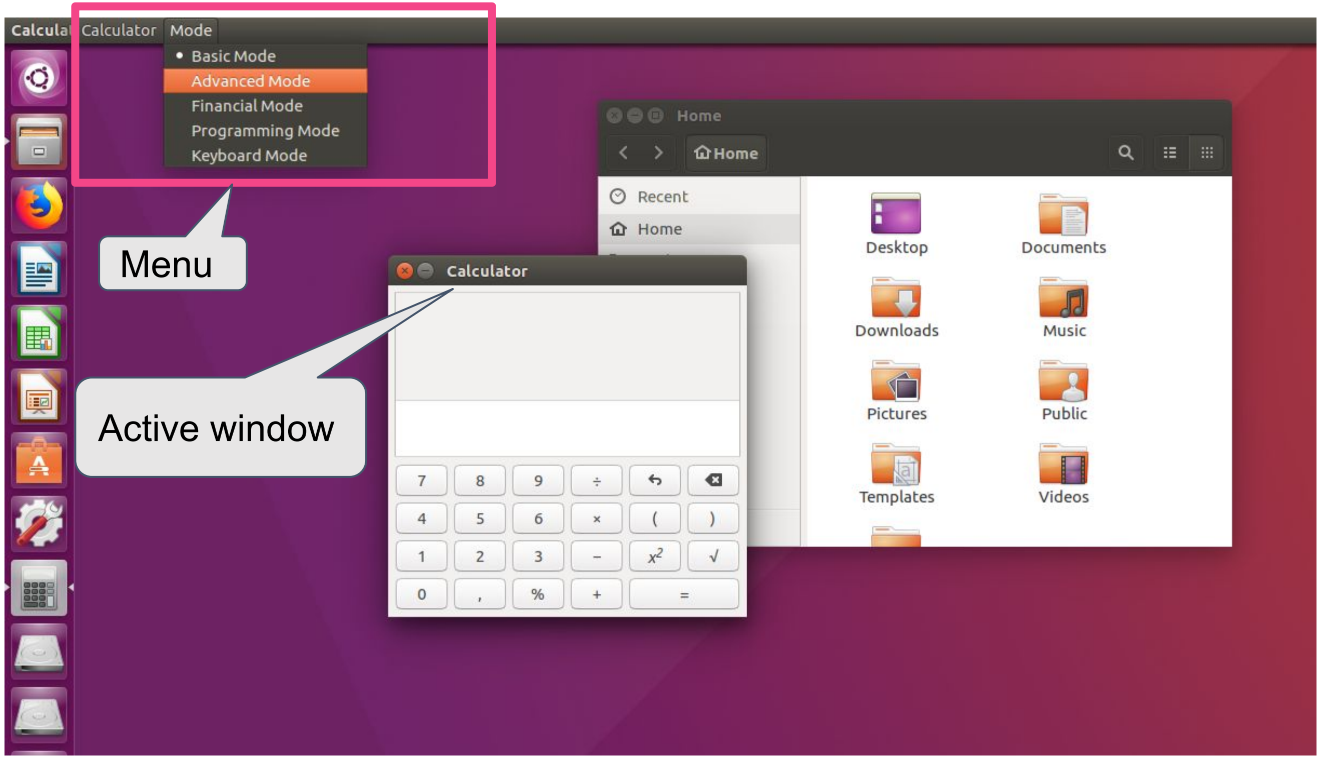
System setup

Ubuntu Linux & Shell Basics

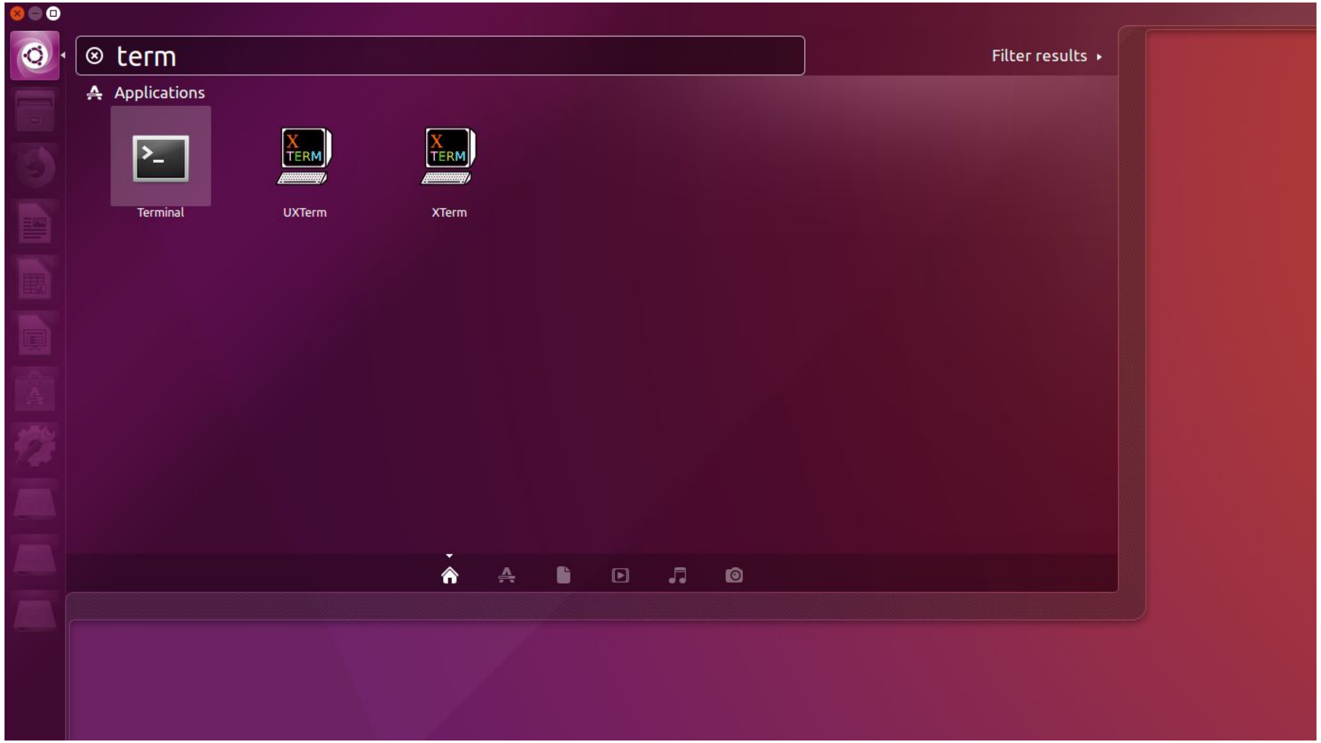
The Ubuntu GUI (16.04)



Application menus



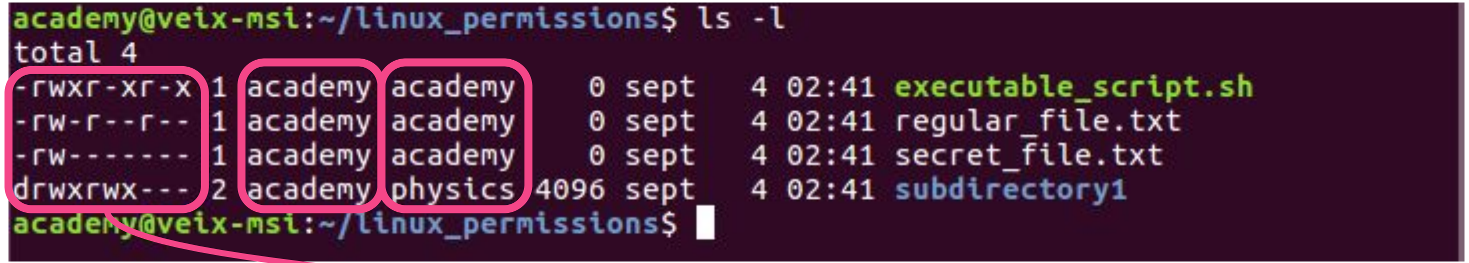
Ubuntu button & Dash



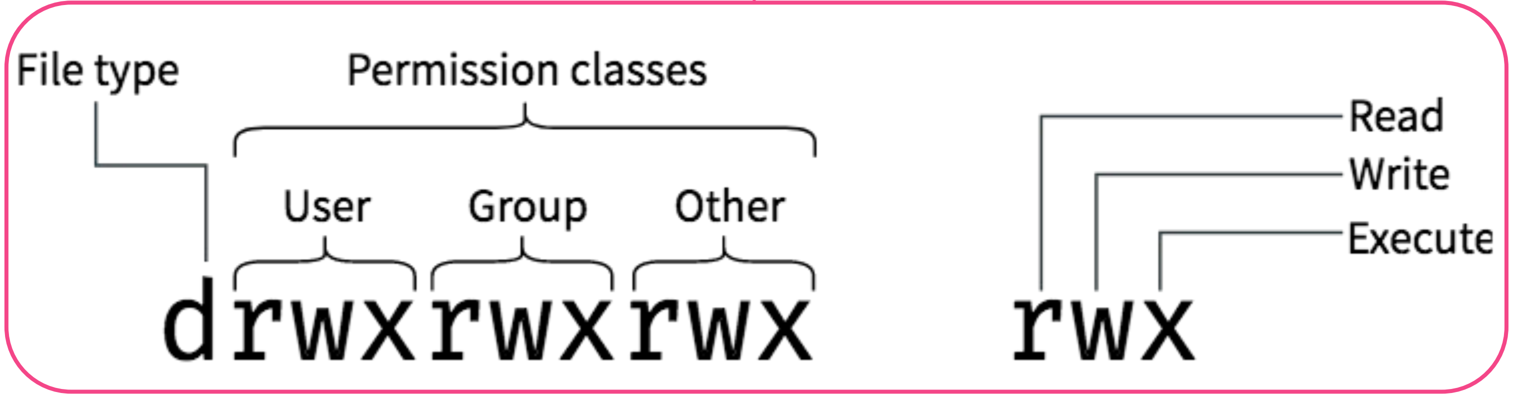
The Linux File System

- Hierarchical, similar to Windows/Mac
- Case sensitivity
- Linux uses / character for separating directories
- No Drive Letters – It's All Under root directory (/)
- Storage devices are mounted as subfolders of the root, e.g.:
    - /media/THUMBDRIVE
    - /cdrom
- Linux file system can contain more than files (disk drives, serial ports, etc.)
    - /dev/input/mouse0
    - /dev/ttyAMA0

The Linux File Permissions



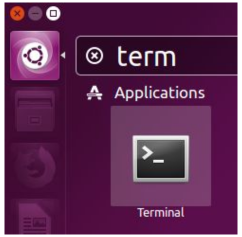
```
academy@veix-msi:~/linux_permissions$ ls -l
total 4
-rwxr-xr-x 1 academy academy    0 sept   4 02:41 executable_script.sh
-rw-r--r-- 1 academy academy    0 sept   4 02:41 regular_file.txt
-rw------- 1 academy academy    0 sept   4 02:41 secret_file.txt
drwxrwx--- 2 academy physics 4096 sept   4 02:41 subdirectory1
academy@veix-msi:~/linux_permissions$ 
```

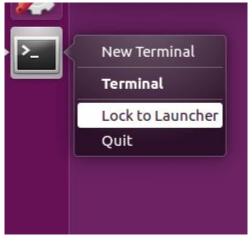


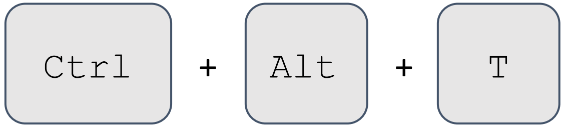
Linux terminal

Choose a convenient method to open!

Need to do this a lot when using ROS.







Super key
Type 'term'
Hit Enter

Lock to Launcher
Open with a click

Use a keyboard shortcut

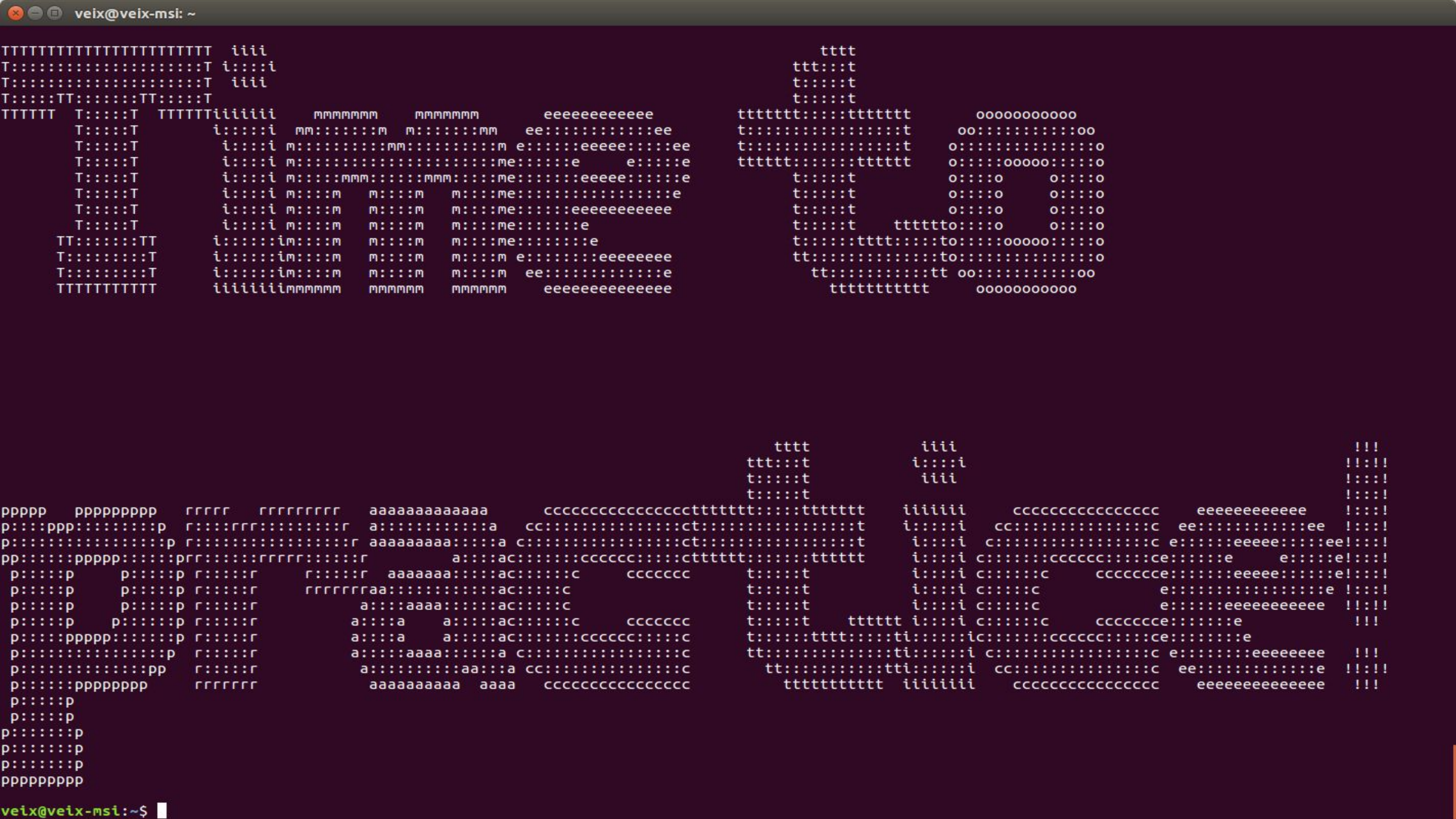
Linux terminal: Tips

- Use arrow keys to scroll previous commands.
- `Ctrl+C` to "kill" the command.
- `TAB` key is your friend! Press often to autocomplete commands.
- `Ctrl+Z` suspends a command.
  - `fg` to make it active again
  - `bg` to continue running it in background.

- `Ctrl+S` will freeze the terminal! Hit `Ctrl+Q` to restore.

Linux terminal: Standard commands

- `ls` – Lists files and folders. Specifying a file or wild card will show only the files listed
- `ls -a` – Lists hidden files as well
- `cd <folder>` - Changes the working folder to the given folder
- `pwd` – Prints the current working folder
- `cp <src> <dest>` - Copies <src> to <dest>
- `mv <src> <dest>` - Moves/renames <src> to <dest>
- `rm <file>` - Removes <file>
- `ps ax` – Shows all processes running on computer
- `kill <pid>` - Kills program with process <pid>



```
TTTTTTTTTTTTTTTTTTTTTTT  iiii                                                            tttt
T:::::::::::::::::::::T i::::i                                                        ttt:::t
T:::::::::::::::::::::T  iiii                                                         t:::::t
T:::::TT:::::::TT:::::T                                                               t:::::t
TTTTTT  T:::::T  TTTTTTiiiiiii    mmmmmmm    mmmmmmm       eeeeeeeeeeee         ttttttt:::::ttttttt       ooooooooooo
        T:::::T        i:::::i  mm:::::::m  m:::::::mm   ee::::::::::::ee       t:::::::::::::::::t     oo:::::::::::oo
        T:::::T         i::::i m::::::::::mm::::::::::m e::::::eeeee:::::ee     t:::::::::::::::::t    o:::::::::::::::o
        T:::::T         i::::i m::::::::::::::::::::::me::::::e     e:::::e     tttttt:::::::tttttt    o:::::ooooo:::::o
        T:::::T         i::::i m:::::mmm::::::mmm:::::me:::::::eeeee::::::e           t:::::t          o::::o     o::::o
        T:::::T         i::::i m::::m   m::::m   m::::me:::::::::::::::::e            t:::::t          o::::o     o::::o
        T:::::T         i::::i m::::m   m::::m   m::::me::::::eeeeeeeeeee             t:::::t          o::::o     o::::o
        T:::::T         i::::i m::::m   m::::m   m::::me:::::::e                      t:::::t    tttttto::::o     o::::o
      TT:::::::TT      i::::::im::::m   m::::m   m::::me::::::::e                     t::::::tttt:::::to:::::ooooo:::::o
      T:::::::::T      i::::::im::::m   m::::m   m::::m e::::::::eeeeeeee             tt::::::::::::::to:::::::::::::::o
      T:::::::::T      i::::::im::::m   m::::m   m::::m  ee:::::::::::::e               tt:::::::::::tt oo:::::::::::oo
      TTTTTTTTTTT      iiiiiiiimmmmmm   mmmmmm   mmmmmm    eeeeeeeeeeeeee                 ttttttttttt     ooooooooooo





                                                                                    tttt            iiii                                           !!!
                                                                                 ttt:::t           i::::i                                         !!:!!
                                                                                 t:::::t            iiii                                          !:::!
                                                                                 t:::::t                                                          !:::!
ppppp   ppppppppp   rrrrr   rrrrrrrrr   aaaaaaaaaaaaa      cccccccccccccccctttttttt:::::ttttttt    iiiiiii     cccccccccccccccc    eeeeeeeeeeee    !:::!
p::::ppp:::::::::p  r::::rrr:::::::::r  a::::::::::::a   cc:::::::::::::::ct:::::::::::::::::t    i:::::i   cc:::::::::::::::c  ee::::::::::::ee  !:::!
p:::::::::::::::::p r:::::::::::::::::r aaaaaaaaa:::::a c:::::::::::::::::ct:::::::::::::::::t     i::::i  c:::::::::::::::::c e::::::eeeee:::::ee!:::!
pp::::::ppppp::::::prr::::::rrrrr::::::r         a::::ac:::::::cccccc:::::ctttttt:::::::tttttt     i::::i c:::::::cccccc:::::ce::::::e     e:::::e!:::!
 p:::::p     p:::::p r:::::r     r:::::r  aaaaaaa:::::ac::::::c     ccccccc      t:::::t           i::::i c::::::c     ccccccce:::::::eeeee::::::e!:::!
 p:::::p     p:::::p r:::::r     rrrrrrraa::::::::::::ac:::::c                   t:::::t           i::::i c:::::c             e:::::::::::::::::e !:::!
 p:::::p     p:::::p r:::::r           a::::aaaa::::::ac:::::c                   t:::::t           i::::i c:::::c             e::::::eeeeeeeeeee  !!:!!
 p:::::p    p::::::p r:::::r          a::::a    a:::::ac::::::c     ccccccc      t:::::t    tttttt i::::i c::::::c     ccccccce:::::::e            !!!
 p:::::ppppp:::::::p r:::::r          a::::a    a:::::ac:::::::cccccc:::::c      t::::::tttt:::::ti::::::ic:::::::cccccc:::::ce::::::::e
 p::::::::::::::::p  r:::::r          a:::::aaaa::::::a c:::::::::::::::::c      tt::::::::::::::ti::::::i c:::::::::::::::::c e::::::::eeeeeeee   !!!
 p::::::::::::::pp   r:::::r           a::::::::::aa:::a cc:::::::::::::::c        tt:::::::::::tti::::::i  cc:::::::::::::::c  ee:::::::::::::e  !!:!!
 p::::::pppppppp     rrrrrrr             aaaaaaaaaa  aaaa   cccccccccccccccc          ttttttttttt  iiiiiiii    cccccccccccccccc    eeeeeeeeeeeeee   !!!
 p:::::p
 p:::::p
p:::::::p
p:::::::p
p:::::::p
ppppppppp

veix@veix-msi:~$
```

ROS Introduction

Fundamentals, Concepts, Filesystem

Session Outline

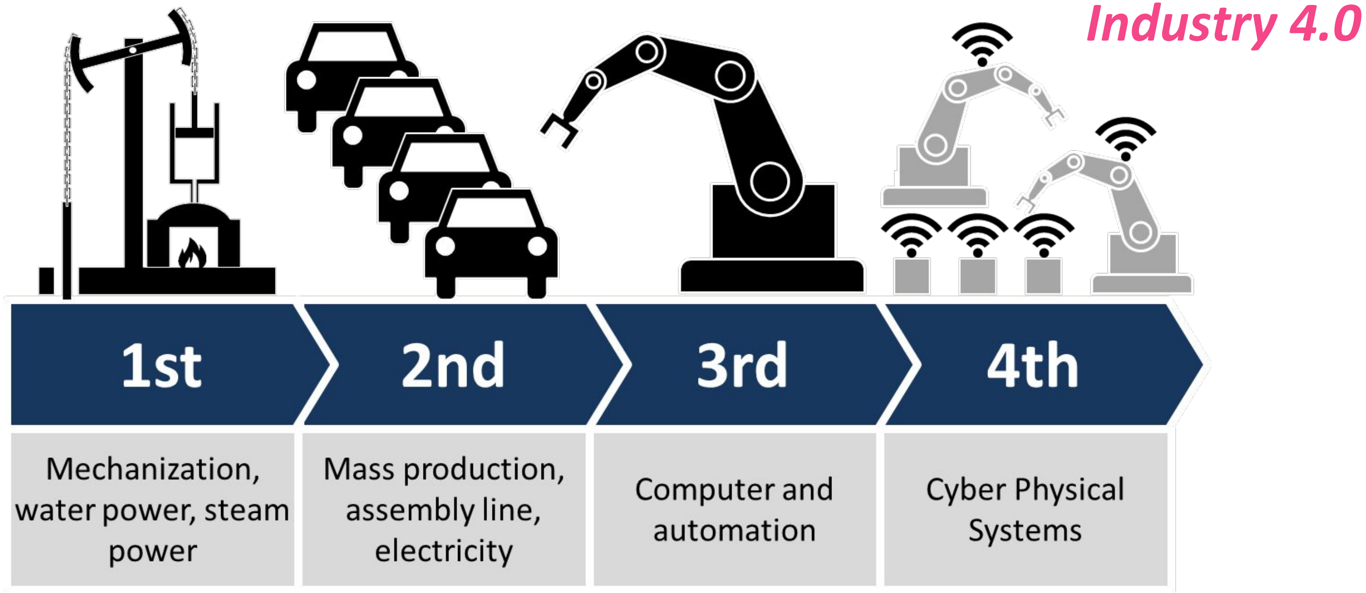
Robotics directions & motivation for ROS

The big picture of ROS?

Fundamentals of ROS

ROS conventions (nodes, packages, and catkin workspace)

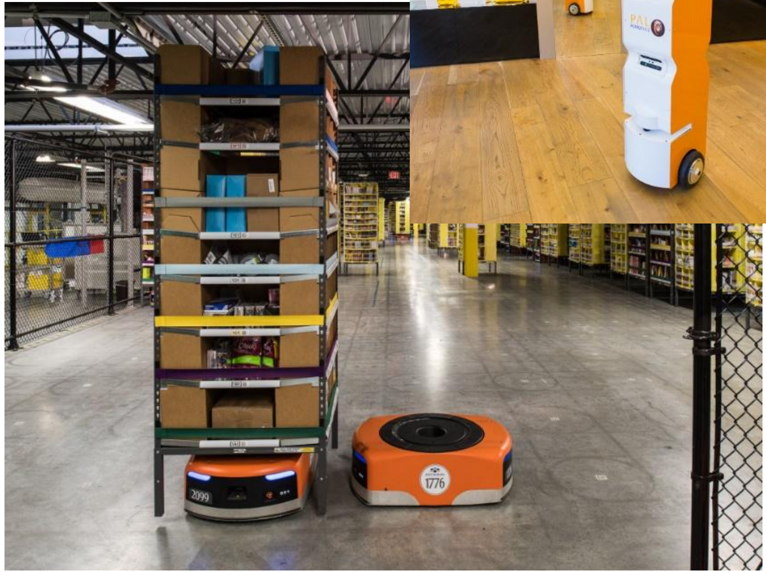
Robotics developments


Industry 4.0
1st
Mechanization, water power, steam power
2nd
Mass production, assembly line, electricity
3rd
Computer and automation
4th
Cyber Physical Systems

Robotics developments: Collaboration

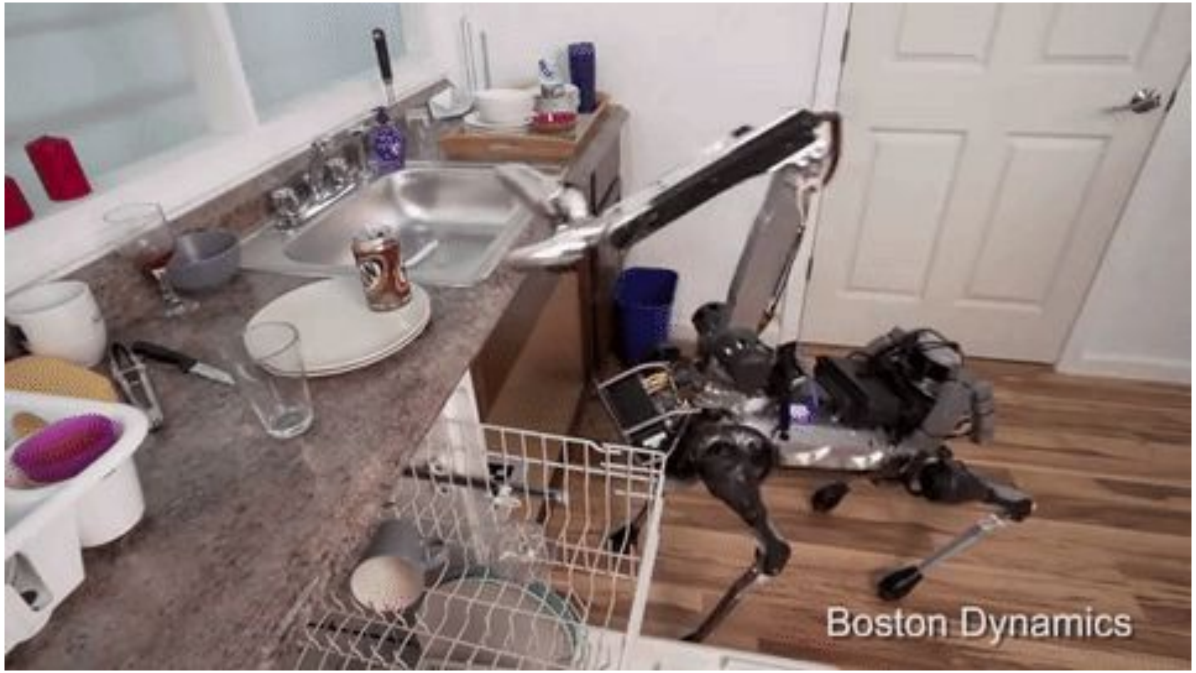
Images: ABB, Universal Robots, Sarcos Robotics,

Robotics developments: Logistics



Images: Amazon Robotics, PAL Robotics, MiR, StarShip Technologies, Cleveron

Implementing challenging tasks

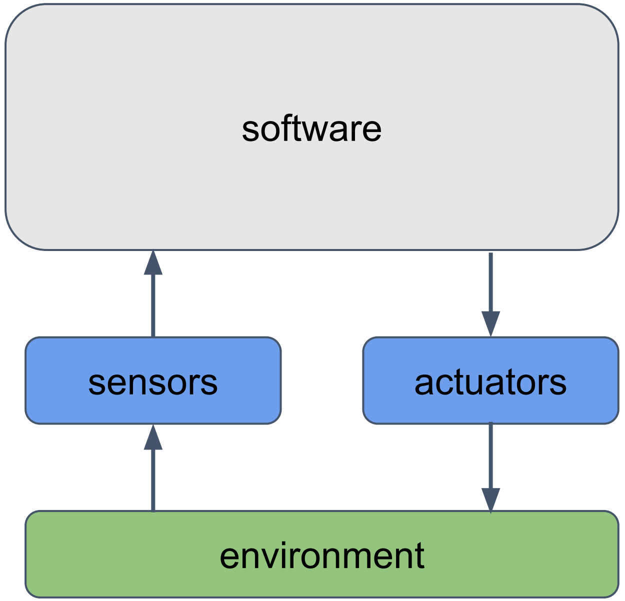


https://www.youtube.com/watch?v=c4z6RZXv5p8

The motivation for ROS

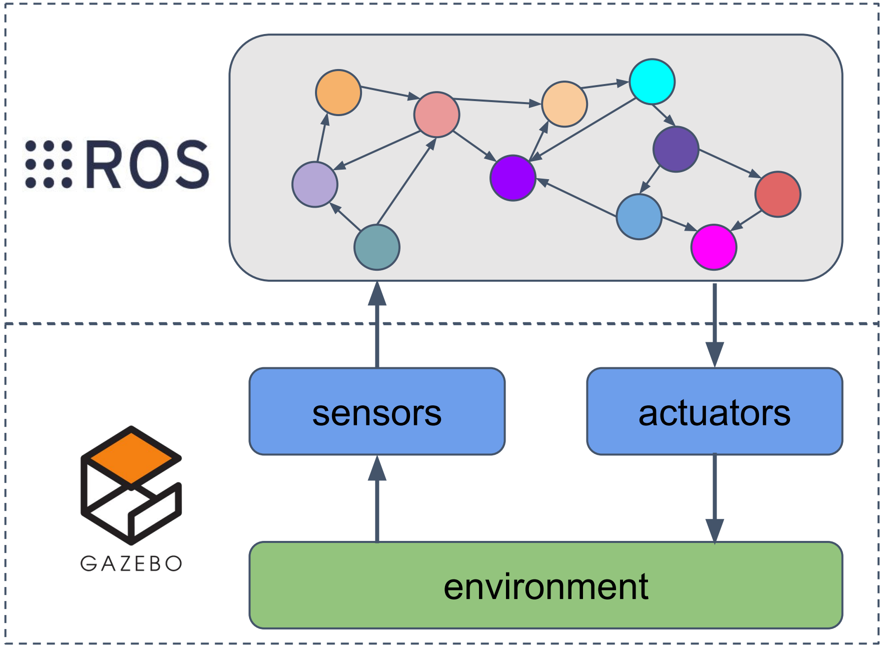
All robots are:

- Software connecting Sensors to Actuators to interact with the Environment

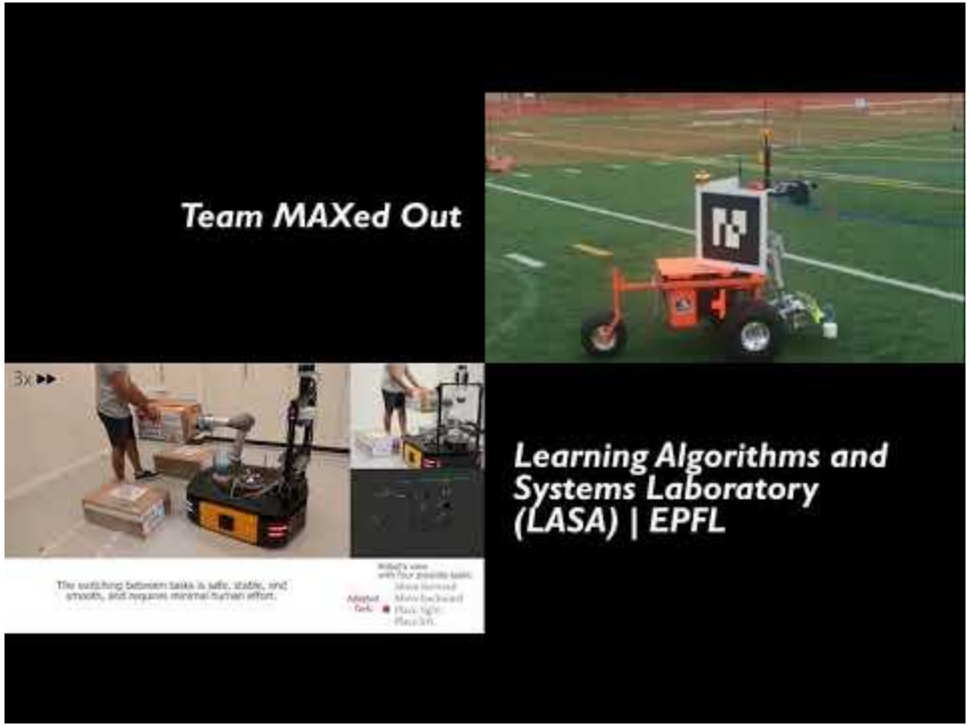


The motivation for ROS

- Break Complex Software into Smaller Pieces
- Provide a framework, tools, and interfaces for distributed development
- Encourage re-use of software pieces
- Easy transition between simulation and hardware



10+ years of ROS

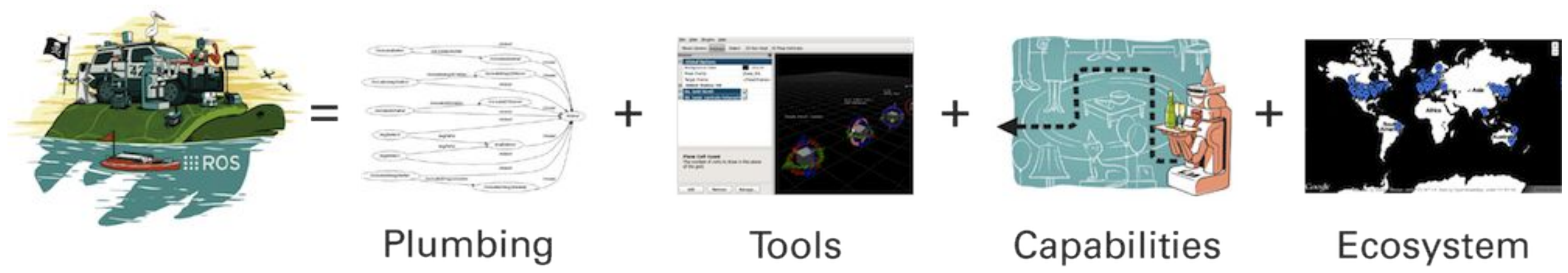


https://www.youtube.com/watch?v=mDwZ21Zia8s

What is ROS?

Sales pitch 😊

- **Open-source solution** for implementing cutting-edge robotics software
- Unified **framework for integrating hardware** from different manufacturers
- Easy-to-use existing functionality, i.e., **modular approach** for re-using previous code
- Huge selection of **amazing development tools** from robot builders to robot builders



What is ROS?

A slightly more technical pitch

=
Plumbing
+
Tools
+
Capabilities
+
Ecosystem

- Open-source solution for creating robot software
- Collection of **software libraries**, **tools**, and **conventions**
  - C++ and Python
- Hardware-agnostics and robust
- ROS is not operating system *per se*
  - Works *mostly* on **Linux** (typically **Ubuntu**)

Programming in ROS

- Language independence, easy to implement.
- Implemented in Python, C++, and Lisp
- Experimental libraries in Java and Lua.
-
- Builtin unit/integration test framework called rostest
- Scaling: ROS is appropriate for large runtime systems and for large development processes.

One-slide history of ROS

- Started during the 00's at **Stanford University**
- Official start in 2007 at **Willow Garage**
- **Open Source Robotics Foundation** (OSRF)

Image source: Willow Garage

List of ROS distributions

Distro	Release date	Poster	*Tuturtle*, turtle in tutorial	EOL date
ROS Melodic Morenia **(Recommended)**	May 23rd, 2018			May, 2023 (Bionic EOL)
ROS Lunar Loggerhead	May 23rd, 2017			May, 2019
ROS Kinetic Kame	May 23rd, 2016			April, 2021 (Xenial EOL)
ROS Jade Turtle	May 23rd, 2015			May, 2017
ROS Indigo Igloo	July 22nd, 2014			April, 2019 (Trusty EOL)

http://wiki.ros.org/Distributions

ROS Resources

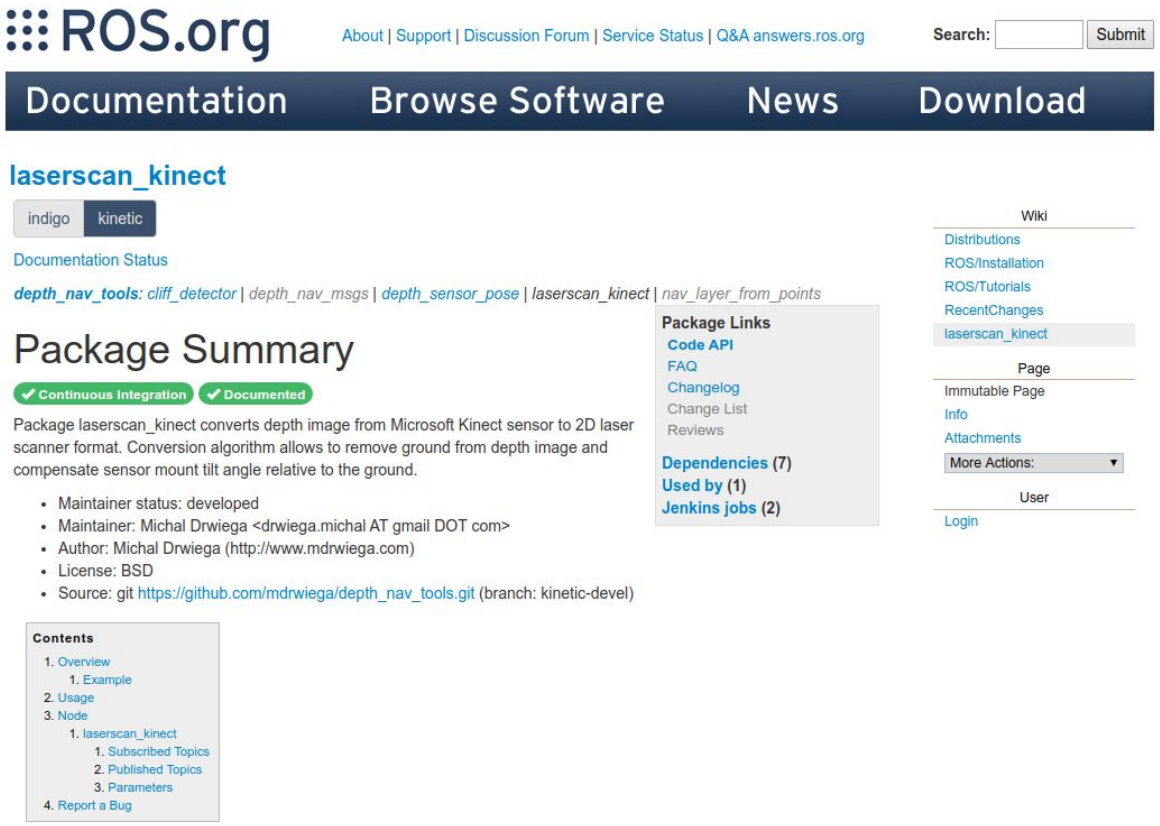
Package wiki

ROS wiki/github

ROS website

ROS Answers

http://wiki.ros.org/<package_name>



ROS.org

About | Support | Discussion Forum | Service Status | Q&A answers.ros.org

Search: Submit

Documentation Browse Software News Download

laserscan_kinect

indigo kinetic

Documentation Status

depth_nav_tools: cliff_detector | depth_nav_msgs | depth_sensor_pose | laserscan_kinect | nav_layer_from_points

Package Summary

✓ Continuous Integration ✓ Documented

Package laserscan_kinect converts depth image from Microsoft Kinect sensor to 2D laser scanner format. Conversion algorithm allows to remove ground from depth image and compensate sensor mount tilt angle relative to the ground.

- Maintainer status: developed
- Maintainer: Michal Drwiega <drwiega.michal AT gmail DOT com>
- Author: Michal Drwiega (http://www.mdrwiega.com)
- License: BSD
- Source: git https://github.com/mdrwiega/depth_nav_tools.git (branch: kinetic-devel)

Contents

Package Links

- Code API
- FAQ
- Changelog
- Change List
- Reviews

Dependencies (7)
Used by (1)
Jenkins jobs (2)

Wiki

- Distributions
- ROS/Installation
- ROS/Tutorials
- RecentChanges
- laserscan_kinect

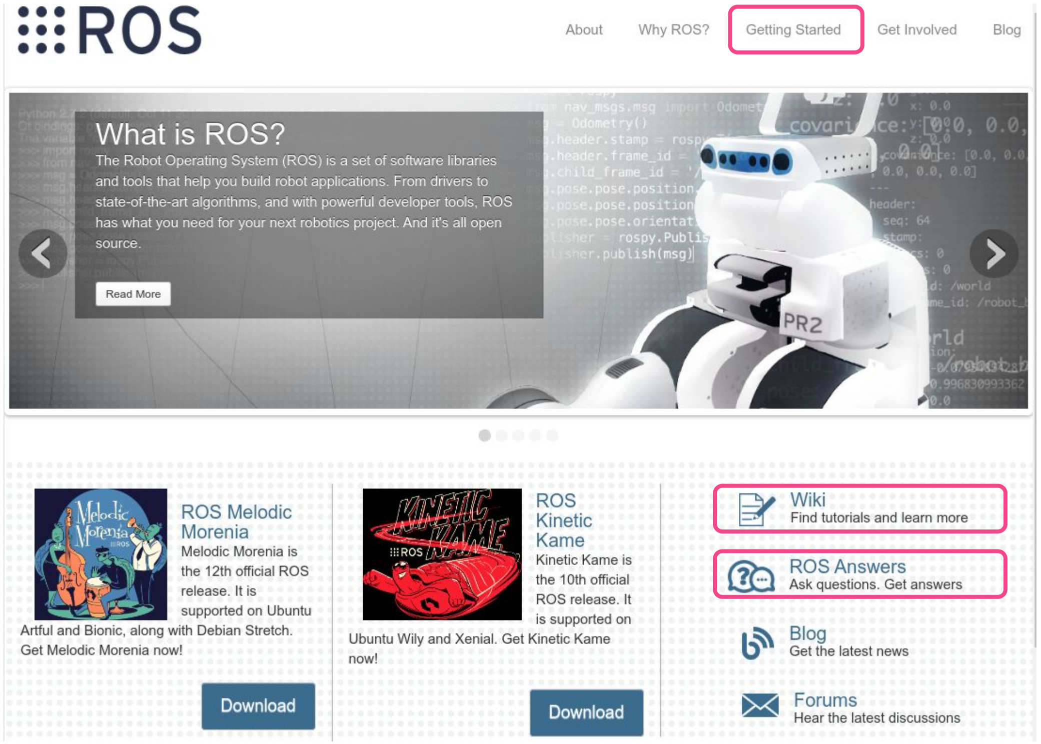
Page

- Immutable Page
- Info
- Attachments
- More Actions:

User

- Login

http://ros.org



What is ROS?

The Robot Operating System (ROS) is a set of software libraries and tools that help you build robot applications. From drivers to state-of-the-art algorithms, and with powerful developer tools, ROS has what you need for your next robotics project. And it's all open source.

ROS Melodic Morenia

Melodic Morenia is the 12th official ROS release. It is supported on Ubuntu Artful and Bionic, along with Debian Stretch. Get Melodic Morenia now!

ROS Kinetic Kame

Kinetic Kame is the 10th official ROS release. It is supported on Ubuntu Wily and Xenial. Get Kinetic Kame now!

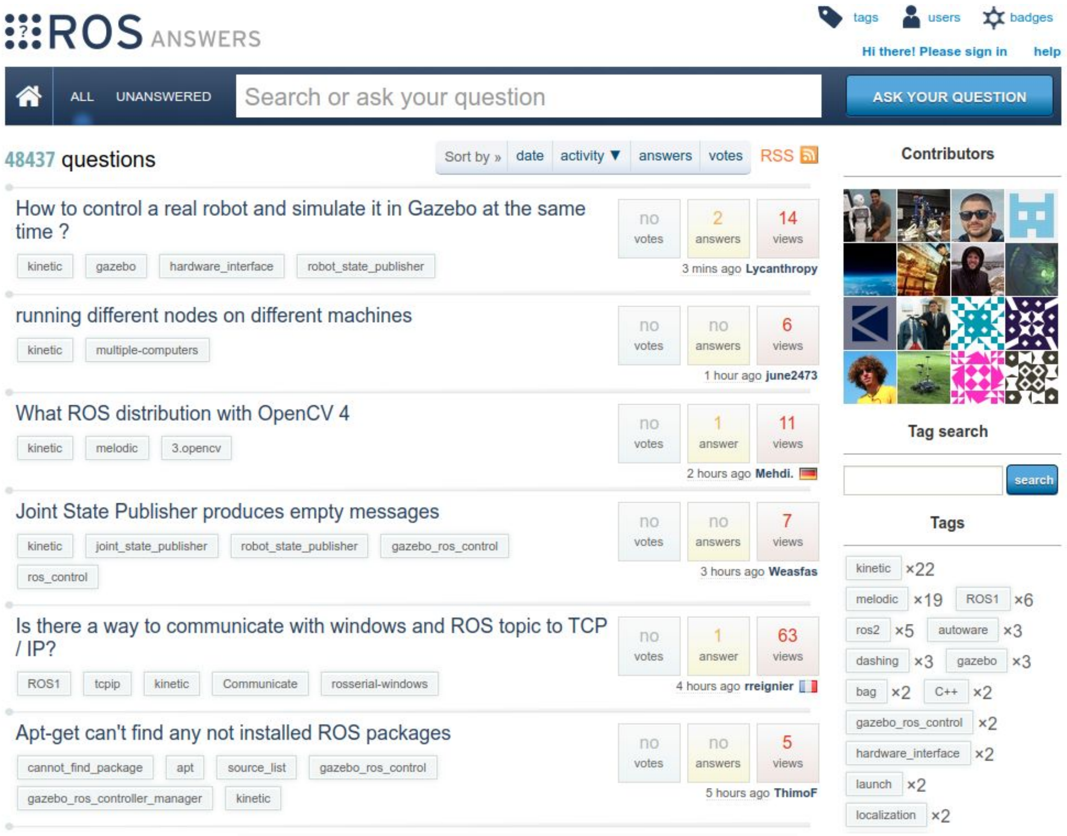
Wiki
Find tutorials and learn more

ROS Answers
Ask questions. Get answers

Blog
Get the latest news

Forums
Hear the latest discussions

http://answers.ros.org



48437 questions

Sort by » date activity ▼ answers votes RSS

How to control a real robot and simulate it in Gazebo at the same time ?
kinetic gazebo hardware_interface robot_state_publisher
no votes, 2 answers, 14 views — 3 mins ago Lycanthropy

running different nodes on different machines
kinetic multiple-computers
no votes, no answers, 6 views — 1 hour ago june2473

What ROS distribution with OpenCV 4
kinetic melodic 3.opencv
no votes, 1 answer, 11 views — 2 hours ago Mehdi.

Joint State Publisher produces empty messages
kinetic joint_state_publisher robot_state_publisher gazebo_ros_control ros_control
no votes, no answers, 7 views — 3 hours ago Weasfas

Is there a way to communicate with windows and ROS topic to TCP / IP?
ROS1 tcpip kinetic Communicate rosserial-windows
no votes, 1 answer, 63 views — 4 hours ago rreignier

Apt-get can't find any not installed ROS packages
cannot_find_package apt source_list gazebo_ros_control gazebo_ros_controller_manager kinetic
no votes, no answers, 5 views — 5 hours ago ThimoF

Contributors

Tag search

search

Tags

kinetic ×22
melodic ×19 ROS1 ×6
ros2 ×5 autoware ×3
dashing ×3 gazebo ×3
bag ×2 C++ ×2
gazebo_ros_control ×2
hardware_interface ×2
launch ×2
localization ×2

ROS is a growing community

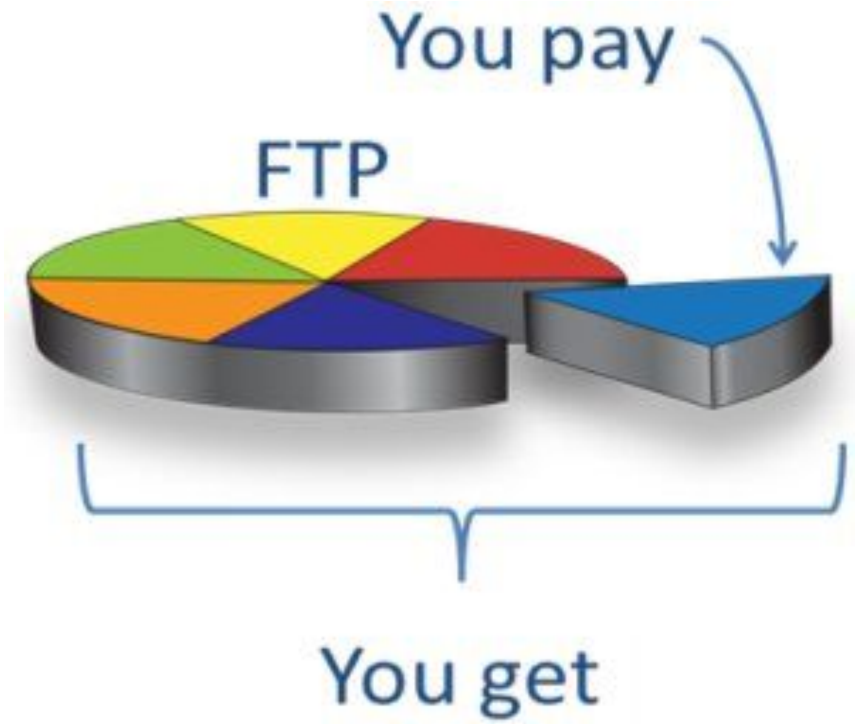
ROS is active:

- ROS Wiki has 2M pageviews/month
- ROS Answers has 650k pageviews/month
- Both have been increasing 20% / year

Data source: http://download.ros.org/downloads/metrics/metrics-report-2018-07.pdf

ROS Industrial

- Started in 2012
    - Yaskawa
    - SWRI
    - Willow Garage
- Focused Technical Projects
- Up to 2 years members only
- Present in 3 regions:
    - ROS Industrial Americas
    - ROS Industrial Europe
    - ROS Industrial Asia - Pacific



https://rosindustrial.org/ric/about-ftps/

ROS Industrial

3M · ABB Robotics · Acutronic Robotics · ADLINK · AF ResTech United States Air Force · AIRBUS · A*STAR Advanced Remanufacturing and Technology Centre · BA Systemes

Bastian Solutions · Blue Ocean Robotics · BlueWorkforce · Boeing · Bosch · BMW · CAT · Erle Robotics

EWI · FH Aachen University of Applied Sciences · Fraunhofer IPA · Fraunhofer IPK · Fraunhofer IML · FZI · GKN Aerospace

GL research · Hanwha Techwin · Harting · ifm · INESCTEC · Intelligrated · IIT · Joanneum Research · John Deere

KEBA Automation by innovation · Lely · M8M · Magna · Microsoft · Nanyang Technological University Singapore · NIST National Institute of Standards and Technology U.S. Department of Commerce · NOV · NRC CNRC

The University of Texas at Austin Nuclear Robotics Group · open robotics · Pepperl+Fuchs · PickNik · Pilz · Plus One Robotics · PPM · PushCorp, Inc.

Rensselaer CATS · Republic Polytechnic · Saxion Hogeschool · SEC · Siemens · SwRI · synapticon · TARDEC · Tony Robotics · Tokyo Opensource Robotics Kyokai

Tormach · traclabs · TU Delft · Unitech · Universal Robots · Vetex · Volvo · Wolf Robotics · Woodside · Yaskawa

https://rosindustrial.org/ric/current-members/

ROS FUNDAMENTAL
S

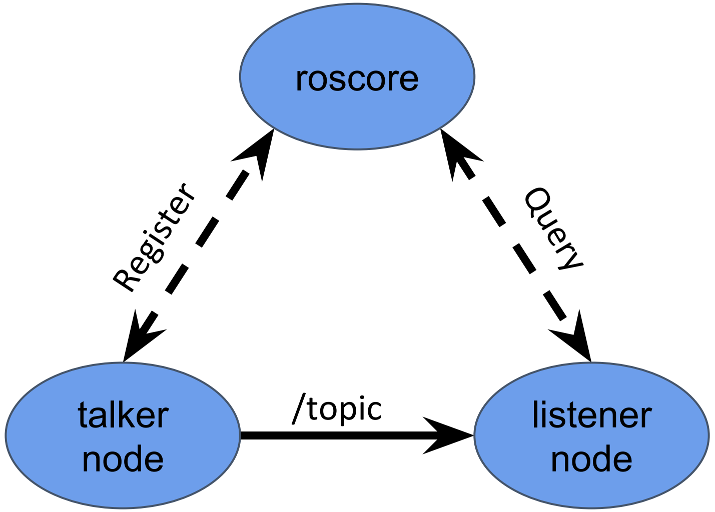
ROS terminology

- **ROSCORE/ROSMASTER** – always on the background, **roscore** is a service that provides connection information to **node**s so that they can transmit **message**s to one another
- **NODE** – software module that is sending or receiving **message**s
- **MESSAGE** – programming-language-independent „data type“
- **TOPIC** – name for a stream of **message**s of defined type
- **PUBLISHER** – sends out **message**s on a specific **topic**
- **SUBSCRIBER** – receives **message**s on a specific **topic**

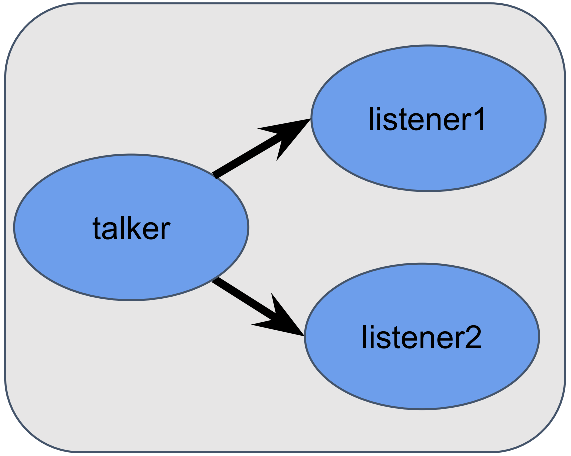
ROS Architecture: roscore and nodes

- - - **Temporary** connection

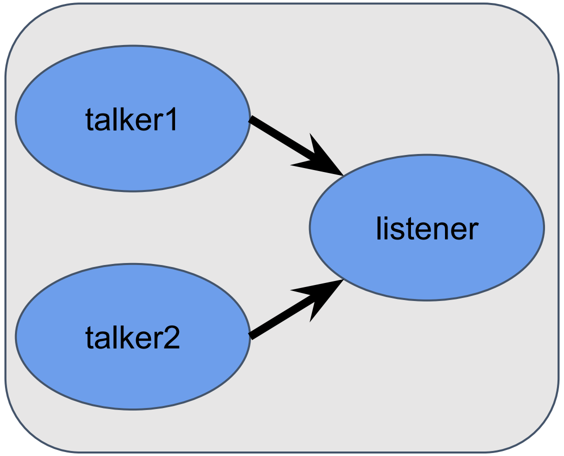
— **Permanent** connection



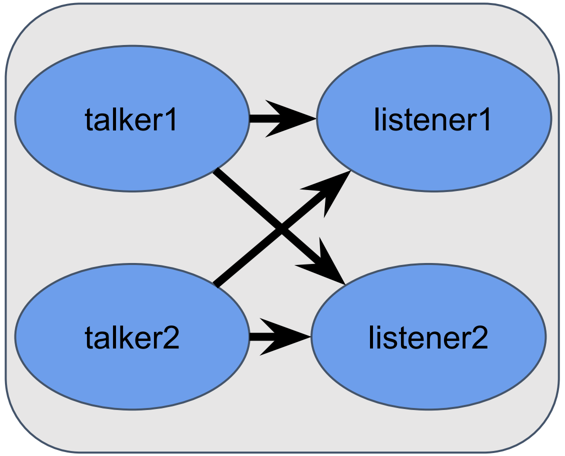
Relationship models



one-to-many



many-to-one



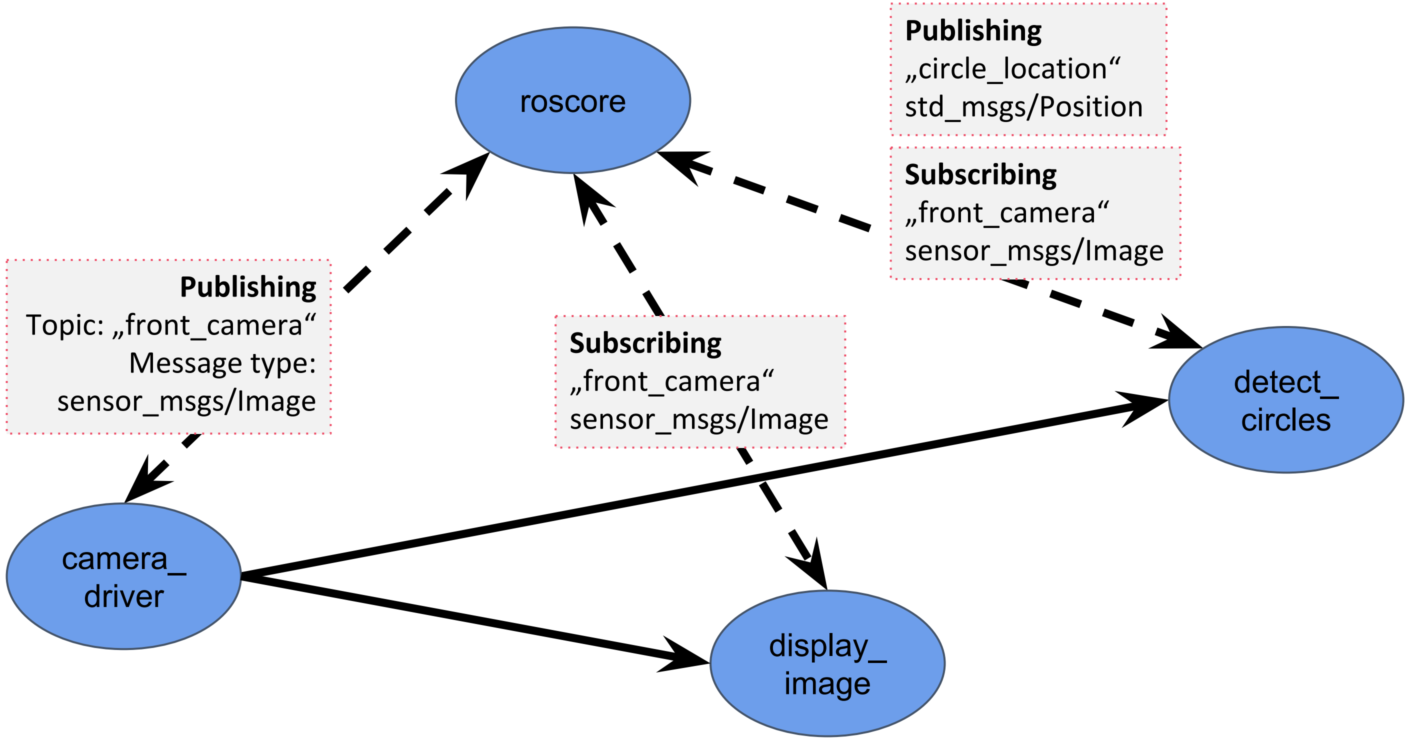
many-to-many

Example: Robot with a camera

Say we have a **robot** with a front-facing **camera** and we would like to **pinpoint** all **circular objects** in its field of view.

- What would be the ROS structure?
- What would the C++ code look like?

Example: Robot with a camera



Coding example: publisher

```
#include "ros/ros.h"
#include "sensor_msgs/Image.h"
#include "camera.h"

int main(int argc, char* argv[]){
  ros::init(argc, argv, „camera_driver");    // ROS node initialisation
  ros::NodeHandle nh;                        // ROS node handle
  ros::Rate frequency(10);                   // Rate 10 Hz

  // Let's create a ROS publisher on topic called „front_camera"
  ros::Publisher pub_cam = nh.advertise<sensor_msgs::Image>(„front_camera", 10);

  while( ros::ok() )
  {
    pub_cam.publish( getCameraImage() );     // Publish single image
    ros::spinOnce();                         // Let other nodes work ;)
    frequency.sleep();                       // Sleep to meet the frequency
  }
  return 0;
}
```

Coding example: subscriber

```
#include "ros/ros.h"
#include "sensor_msgs/Image.h"
#include "std_msgs/Point.h"

ros::Publisher pub_position;

void findCircle(sensor_msgs::Image input_image) {
  std_msgs::Point circle_position;
  ...                                         // here be algorithm
  pub_position.publish( circle_position );    // publish circle position
}

int main(int argc, char *argv[]) {
  ros::init(argc, argv, „detect_circles");    // ROS node initialisation
  ros::NodeHandle nh;                         // ROS node handle
  // Let’s create a ROS subscriber to „front_camera‟
  ros::Subscriber subscriber_cam = nh.subscribe(„front_camera", 1, findCircle);
  // Let’s create a ROS publisher on „circle_location‟
  pub_position = nh.advertise<std_msgs::Point>(„circle_location", 1);
  ros::spin();
  return 0;
}
```

What else is there in ROS?

Query-based messaging

- **Service** – Query and response messages
- **Action** – Query, state, and response messages

Parameter server – maintaining runtime variables

Configuration files

URDF – Unified Robot Description Format

roslaunch – starting multiple nodes simultaneously, loading configuration values to parameter server, etc.

Packages – organization for ROS nodes, launch-files, etc

ROS

CONCEPTS & CONVENTIONS

ROS conventions: units & coordinates

- SI units (meter, kilogram, second, ampere)
- SI-derived units (radian, hertz, newton, watt, volt, celsius, tesla)
- Right handed coordinates:
    - x forward
    - y left
    - z up
- Preferred representation for rotations: Quaternions

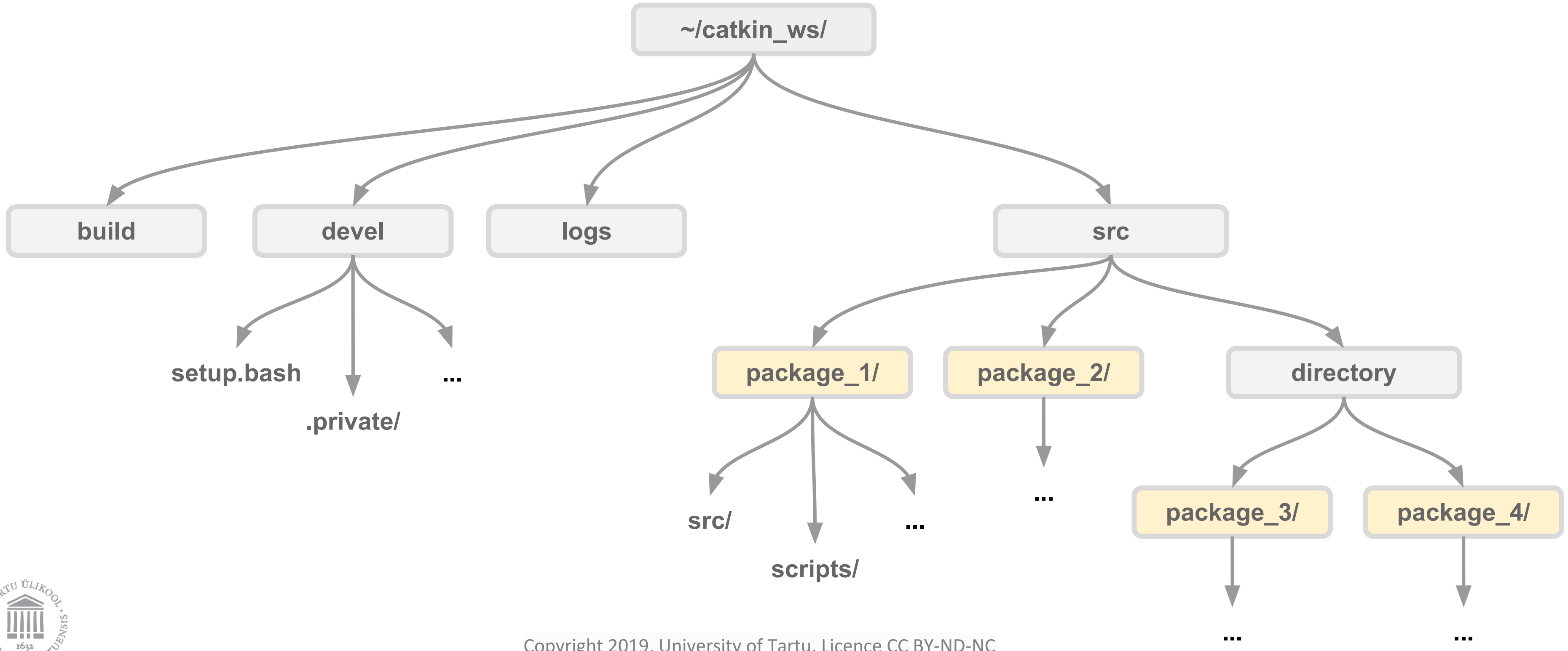
- https://www.ros.org/reps/rep-0103.html

ROS conventions: naming

Package names: lower case, underscore separators, e.g. **laser_scan**

REP 144: https://www.ros.org/reps/rep-0144.html

Structure of Catkin workspace



ROS packages

http://wiki.ros.org/Packages

A package might contain

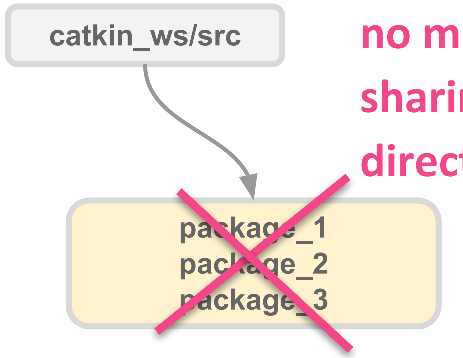
- ROS nodes,
- a ROS-independent library,
- a dataset,
- configuration files,
- a third-party piece of software, or
- anything else that logically constitutes a useful module.

ROS packages follow a "Goldilocks" principle:
enough functionality to be useful, but not too much that the package is heavyweight and difficult to use from other software.

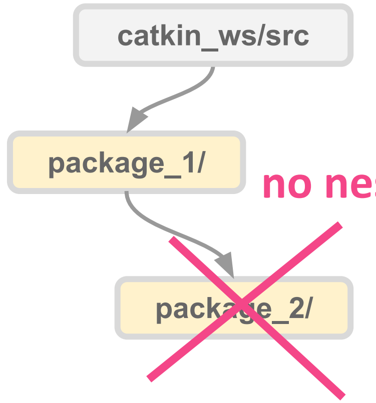
ROS packages
http://wiki.ros.org/Packages

By definition, the package must contain
- a catkin compliant **package.xml** file
- a **CMakeLists.txt** which uses catkin

Each package must have its own folder!



no multiple packages sharing the same directory!



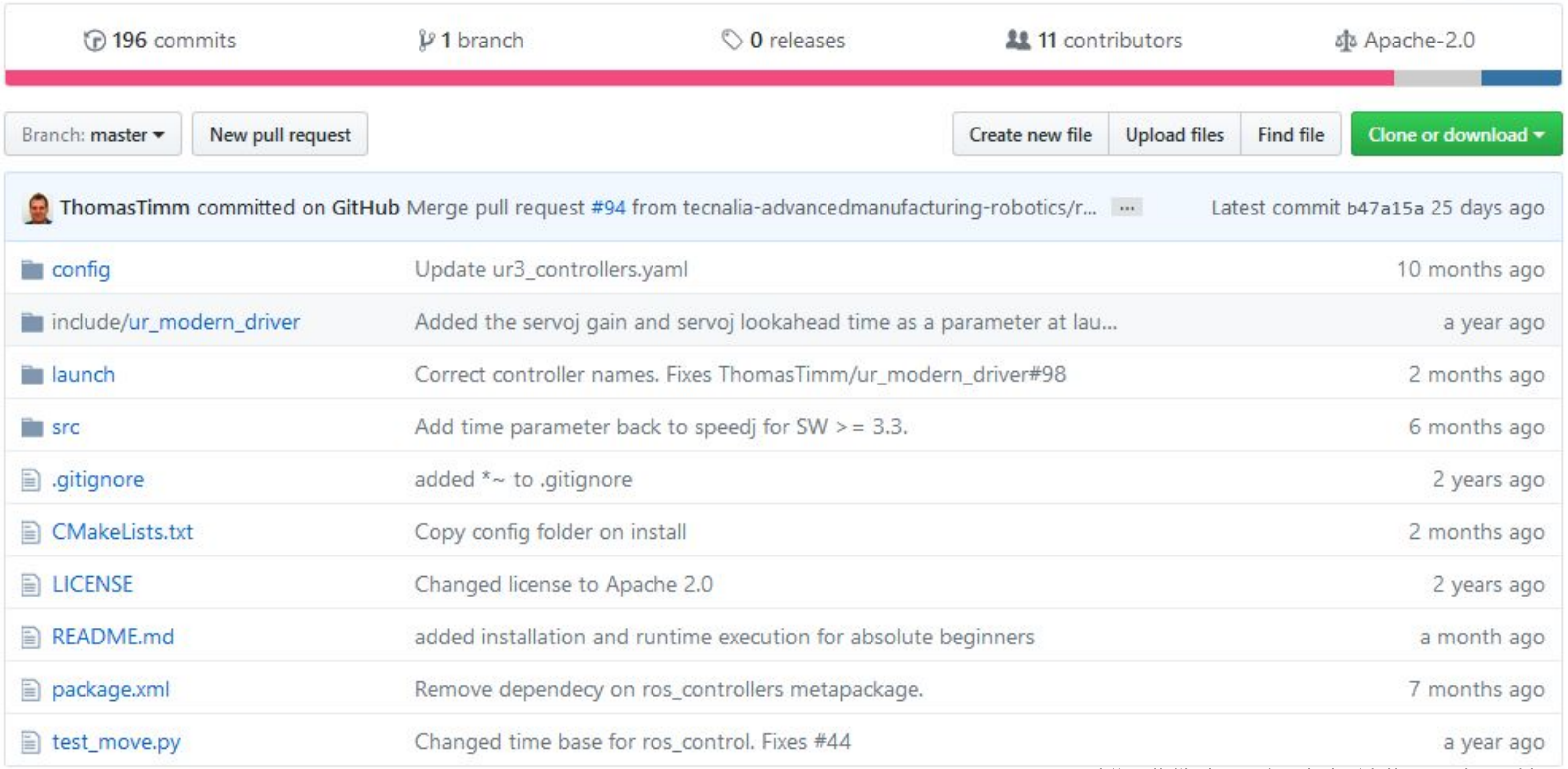
no nested packages!

ROS packages

http://wiki.ros.org/Packages

```
my_ros_package/
|---CMakeLists.txt: CMake build file
|---package.xml: Manifest containing meta information
|---include/package_name: C++ include headers
|---launch/: Folder containing launch-files
|---msg/: Folder containing Message (msg) types
|---src/package_name/: Source files
|---srv/: Folder containing Service (srv) types
|---scripts/: executable scripts
```

ur_modern_driver
ROS package example in GitHub



196 commits | 1 branch | 0 releases | 11 contributors | Apache-2.0

Branch: master ▾ | New pull request | Create new file | Upload files | Find file | Clone or download ▾

ThomasTimm committed on GitHub Merge pull request #94 from tecnalia-advancedmanufacturing-robotics/r... ··· Latest commit b47a15a 25 days ago

config	Update ur3_controllers.yaml	10 months ago
include/ur_modern_driver	Added the servoj gain and servoj lookahead time as a parameter at lau...	a year ago
launch	Correct controller names. Fixes ThomasTimm/ur_modern_driver#98	2 months ago
src	Add time parameter back to speedj for SW >= 3.3.	6 months ago
.gitignore	added *~ to .gitignore	2 years ago
CMakeLists.txt	Copy config folder on install	2 months ago
LICENSE	Changed license to Apache 2.0	2 years ago
README.md	added installation and runtime execution for absolute beginners	a month ago
package.xml	Remove dependecy on ros_controllers metapackage.	7 months ago
test_move.py	Changed time base for ros_control. Fixes #44	a year ago

https://github.com/ros-industrial/ur_modern_driver

ROS packages and nodes

- Packages can be created with tools like `catkin_create_pkg`
- Every ROS node belongs to a ROS package
- A package can contain multiple nodes (name is set with `ros::init`)

```
ros::init(argc, argv, "camera_driver");
```

- Nodes are executables

```
$ rosrun <package_name> <node_name>
$ rosrun camera_package camera_driver
$ rosrun camera_package camera_driver.py
```

ROS nodes

http://wiki.ros.org/Nodes

- A node is a process that performs computation
- A robot control system will usually comprise many nodes

- Benefits of using ROS nodes:
  - Additional ***fault tolerance*** as crashes are isolated to individual nodes
  - ***Code complexity* is reduced** in comparison to monolithic systems
  - Implementation details are also well hidden as the nodes expose a minimal API to the rest of the graph and **alternate implementations**, even in other programming languages, **can easily be substituted**.

roslaunch

- Launch-files enable:
  - Running multiple nodes with a single command
  - Specifying arguments for nodes
  - Remapping
  - Loading parameters to ROS parameter server
- Uses XML

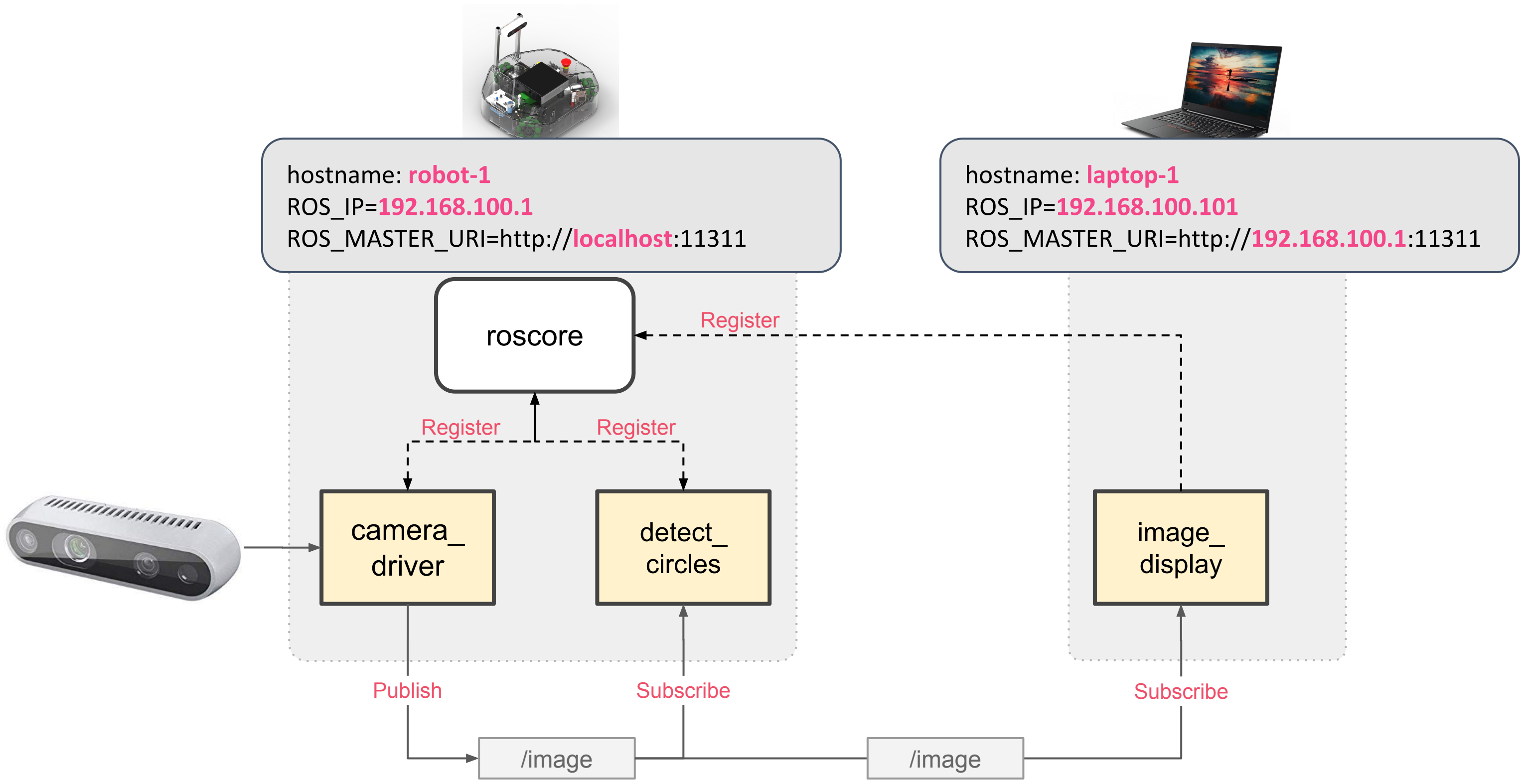
```
$ roslaunch <package> <launch-file>
$ roslaunch ur_modern_driver ur5_bringup.launch
```

ROS as a distributed system

Configured with environmental variables

```
export ROS_MASTER_URI=http://<master_ip>:11311
export ROSIP=<interface_address>
```

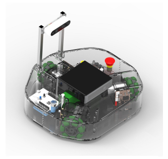
ROS as a distributed system



hostname: robot-1
ROS_IP=192.168.100.1
ROS_MASTER_URI=http://localhost:11311

hostname: laptop-1
ROS_IP=192.168.100.101
ROS_MASTER_URI=http://192.168.100.1:11311

Example



roscore

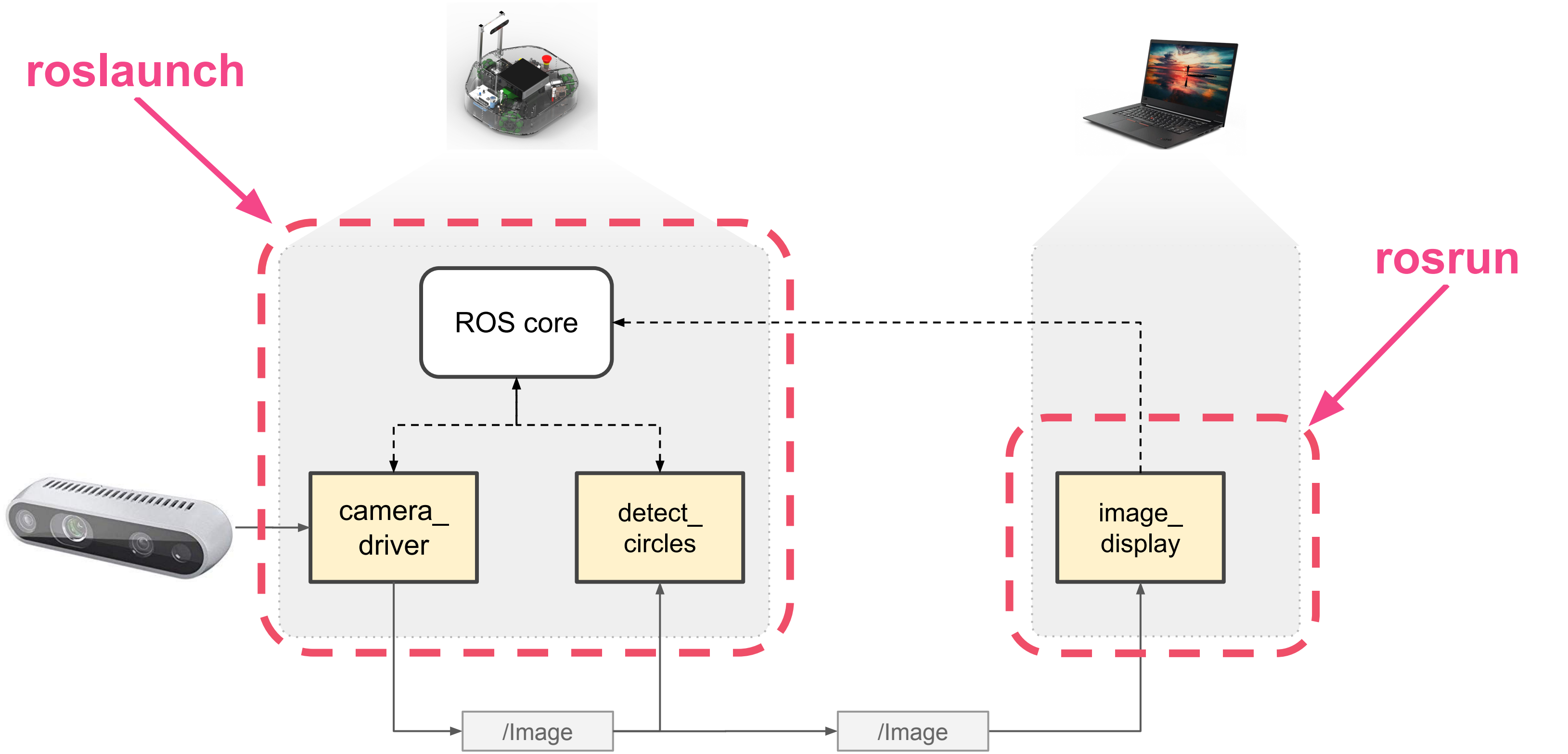
hostname: robot
ROS_IP=192.168.1.1
ROS_MASTER_URI=http://localhost:11311

```
127.0.0.1         localhost
127.0.1.1         laptop-1
192.168.100.1     robot-1
```

hostname: laptop
ROS_IP=192.168.1.101
ROS_MASTER_URI=http://192.168.100.1:11311

```
127.0.0.1         localhost
127.0.1.1         laptop-1
192.168.100.1     robot-1
```

rosrun vs roslaunch



Workshop

ROS Environment
Navigate through packages
Run ROS programs
Teleoperate Clearbot robot