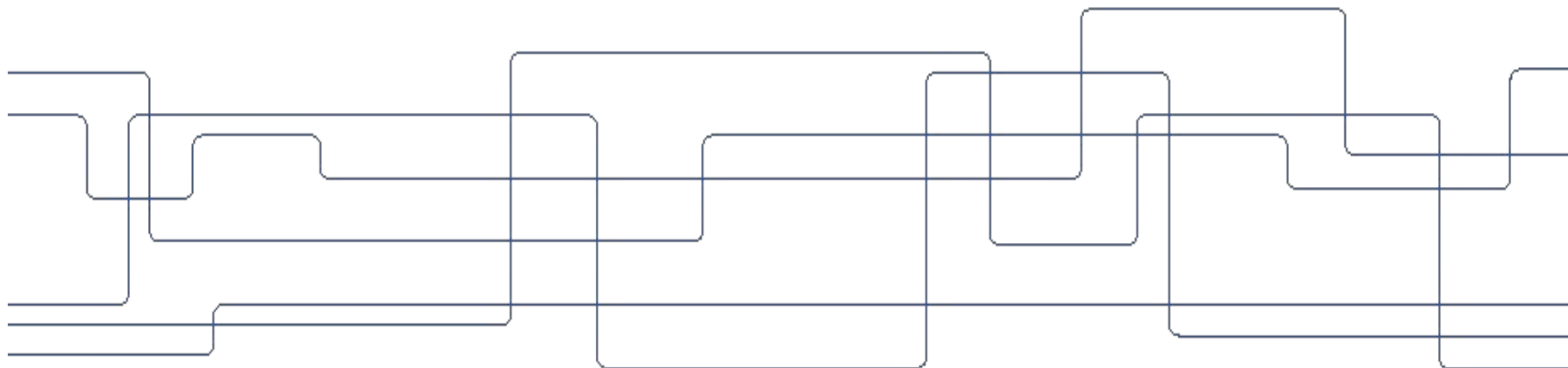


Assignment 1: Introduction to ROS

Introduction to Robotics
DD2410



What is Robot Operating System (ROS)?

- Middleware platform
- Simplifies integration of different components over the network
- Provides a set of common tools for debugging and visualization of data

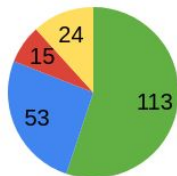


Why ROS?

- Very large user community
- Standard in many robotics labs around the world, even in some companies

ROS Users of the World

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



<http://metrorobots.com/rosmap.html>


Why ROS?

- Many commercially available robots use ROS nowadays
- <https://robots.ros.org/>



Recently Added

ROSbot 2.0 PRO




Category: [ground](#)
Resources: [Website](#) [Wiki](#)

ROSbot 2.0 is a successor of ROSbot - an autonomous, open source robot platform - now with a RGBD camera and improved design. It can be used as a learning platform for ROS as well as a base for a variety of robotic applications.

[autonomous](#) [education](#) [ground](#) [lidar](#) [mobile base](#)
[research](#) [wireless](#)

Doosan-Robot



Category: [manipulator](#)
Resources: [Website](#) [Wiki](#)

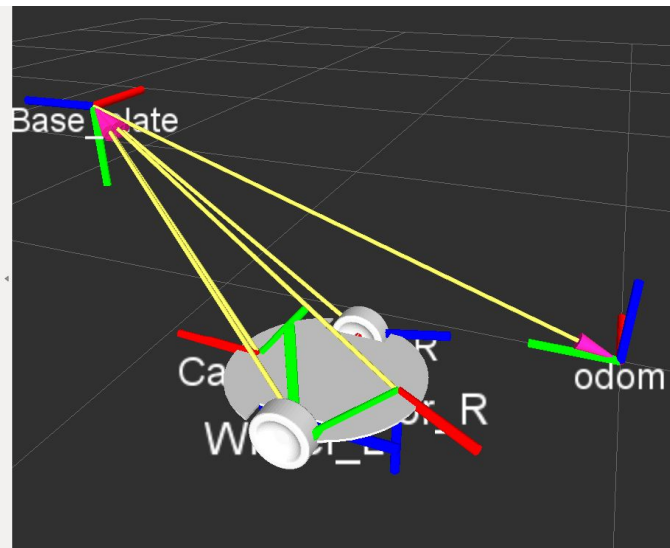
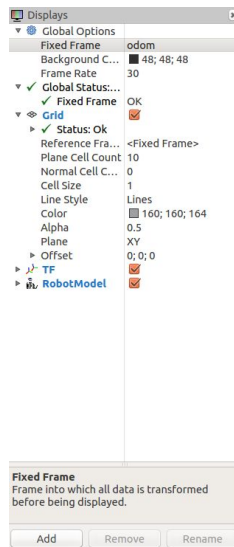
This repository is about 6-axis Doosan robots. It includes packages such as moveit, Gazebo, as well as Doosan controller interlocking packages for ROS.

[6dof](#) [arm](#) [manipulator](#) [co-bot](#) [doosan-robots](#)
[doosan-robotics](#)

Why ROS?

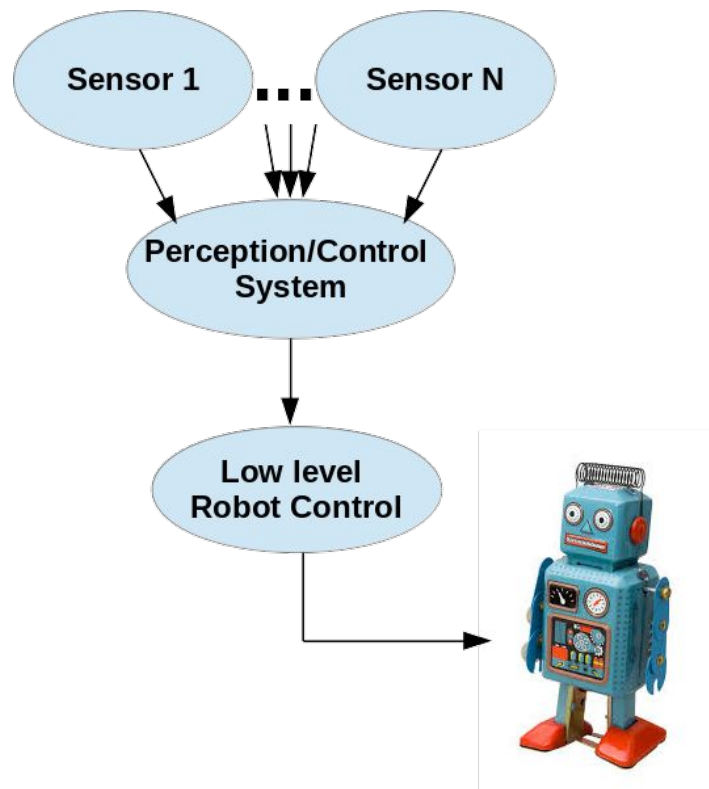
Open source + large community = **lots of packages, libraries, and tools available**

- Robot planning and control
- Navigation
- Visualization tools
- Hardware drivers and interfaces
- Etc ...



Why ROS?

- Modularization and abstraction
- Standardization/structure
- Easier to collaborate with others
- **Make roboticists' life easier!**





ROS: An overview



ROS - Overview

- A typical ROS system has the following basic elements
 - A master
 - Nodes
 - Topics / services / actions
 - TF





ROS - Nodes

- ROS nodes are executables
- Each running node is able to exchange information through topics/services
- Should perform well-defined task (motor control, localization, sensor processing, etc.)





ROS - Topics

- ROS topics implement communication channels
- Many-to-many relationship
- Each topic has a type of message





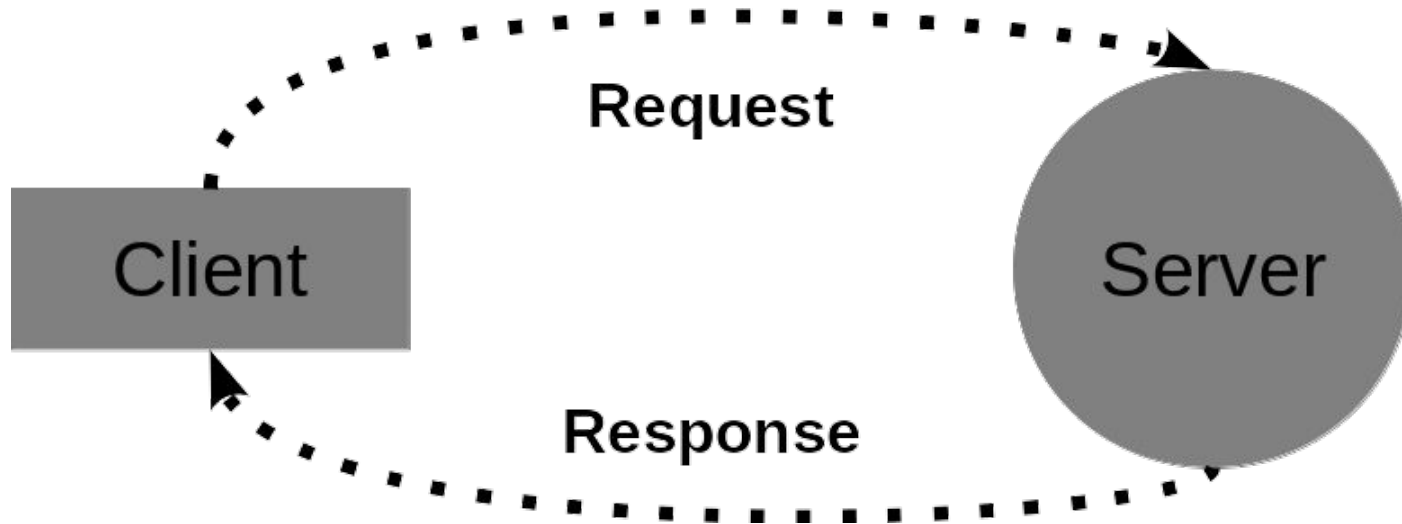
ROS - Topics

- Each topic has a type of message
- Messages define a communication interface
- They specify which type of information is sent over the communication channel



ROS - Services

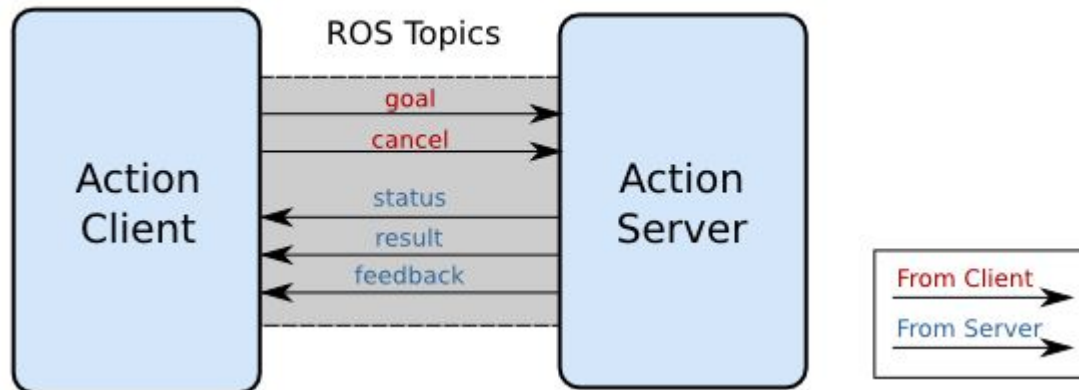
- Nodes may also communicate through services
- Client-server relationship
- Defined by a pair of messages: request and response



ROS - Actions

- Services are great, however not in all situations
- If the service takes a long time to execute
- If you want to cancel the request during execution
- If you want to get periodic feedback about the request
- This is where you should use actionlib instead
- Can be preempted
- Use simple action server/client

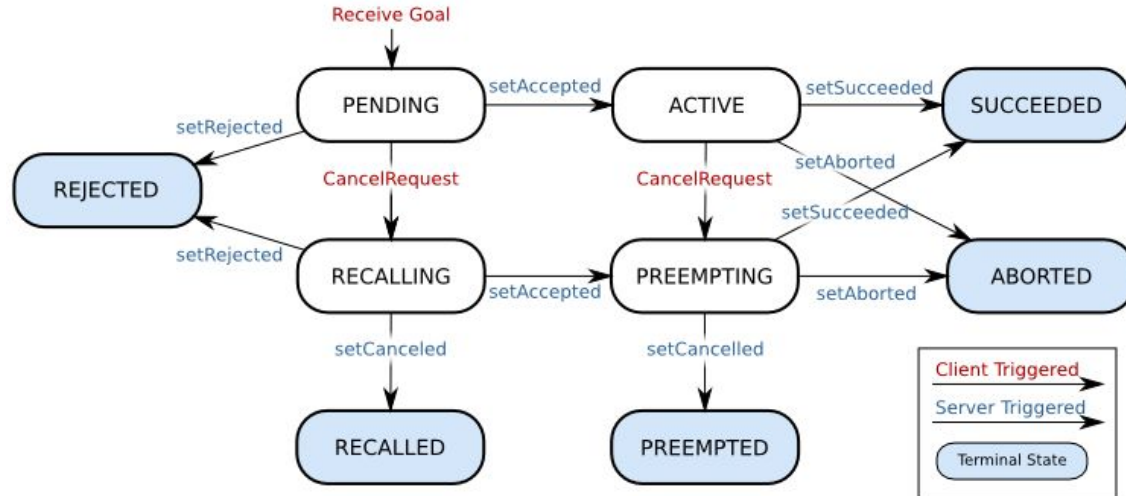
Action Interface



ROS - Action server

- The action server state transitions

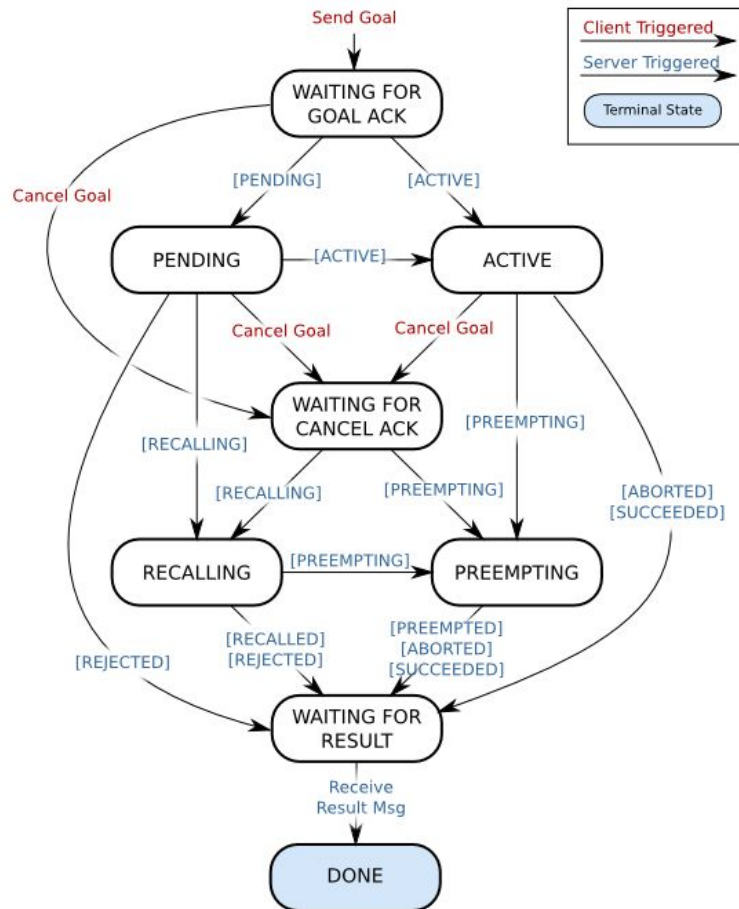
Server State Transitions



ROS - Action client

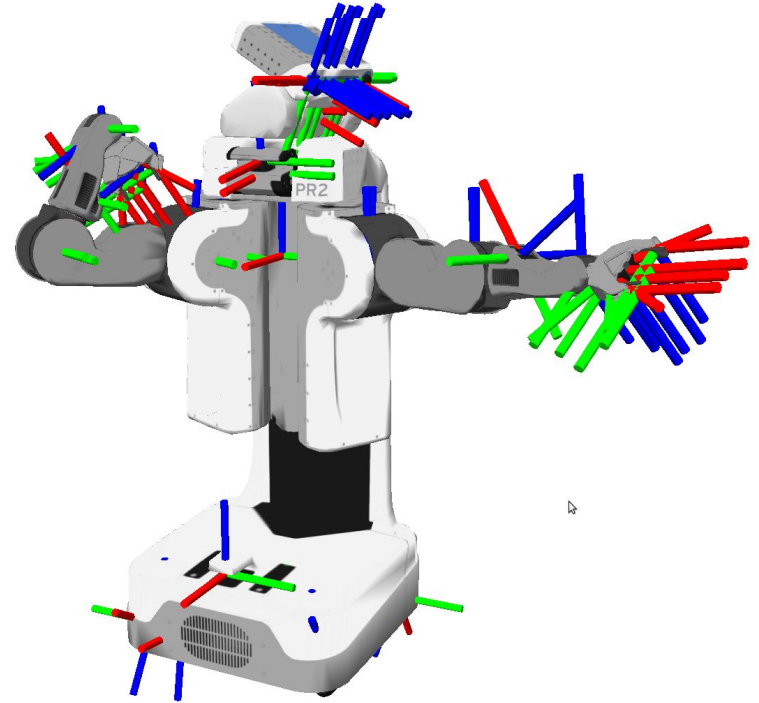
- The action client state transitions

Client State Transitions



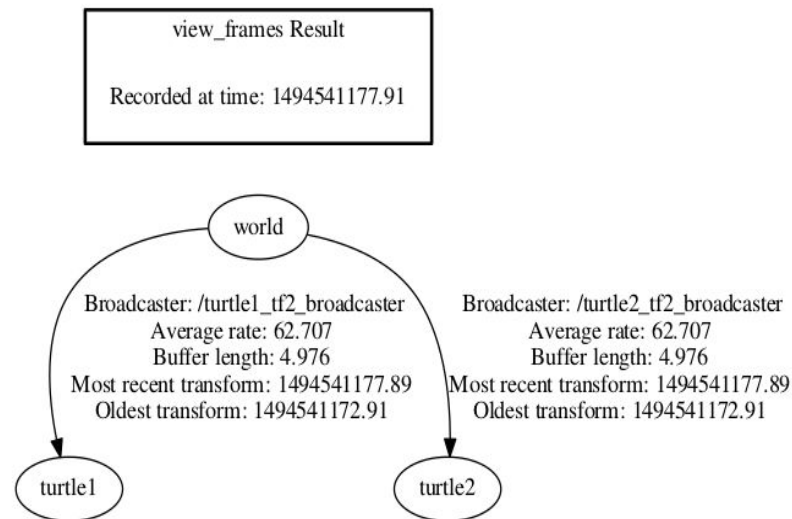
ROS - TF

- Keep track of multiple coordinate frames over time



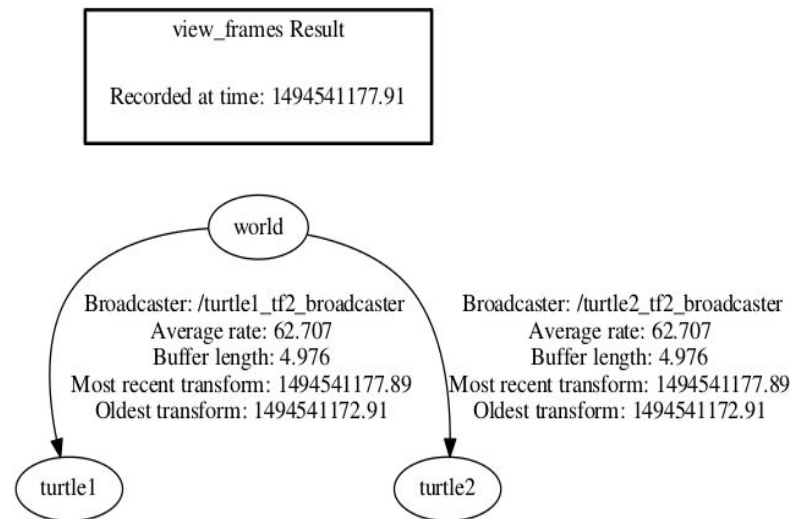
ROS - TF

- Keep track of multiple coordinate frames over time
- Maintains the relationship between coordinate frames in a tree structure buffered in time

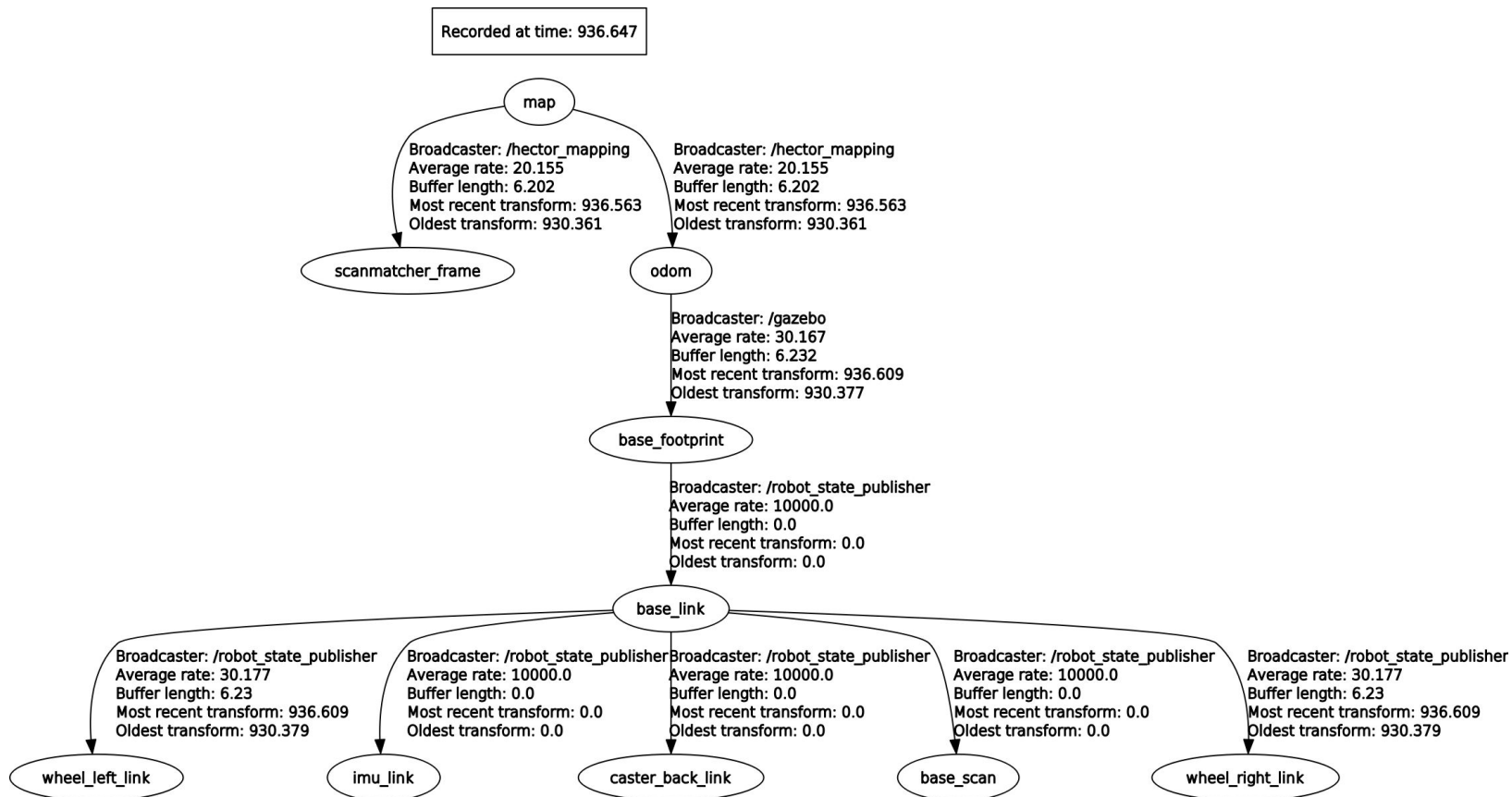


ROS - TF

- Keep track of multiple coordinate frames over time
- Maintains the relationship between coordinate frames in a tree structure buffered in time
- Lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time
- In a lot of messages you will see **header**
 - It contains information about the frame of the message



ROS - TF

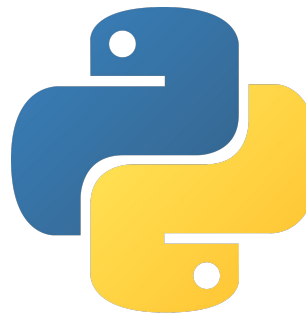


ROS - Programming

- Ubuntu **18.04**
- ROS **Melodic**
- Python
 - Version 2.7
 - Default with Ubuntu 18.04

If you are interested you can read more here: <http://www.ros.org/reps/rep-0003.html>

- Everything is installed for you in the computer labs
 - E Building: Röd, Orange, Gul, Grön, Brun, Grå, Karmosin, Vit, Magenta
 - D building: Spel, sport



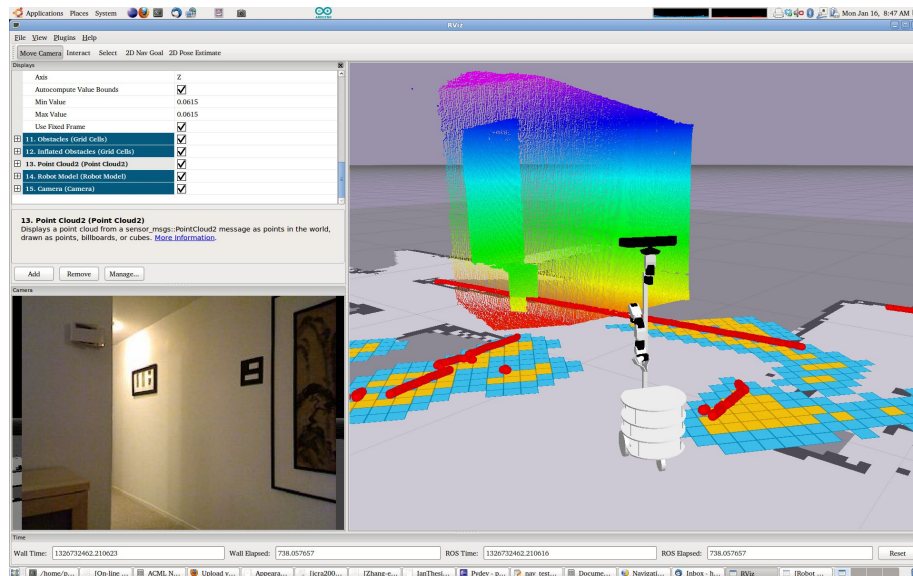
ROS - Packages

- Pieces of software in ROS are bundled together and distributed through **packages**
- Contains source code for compiling nodes
- Parameters, message/service/action files...
- Build and install instructions



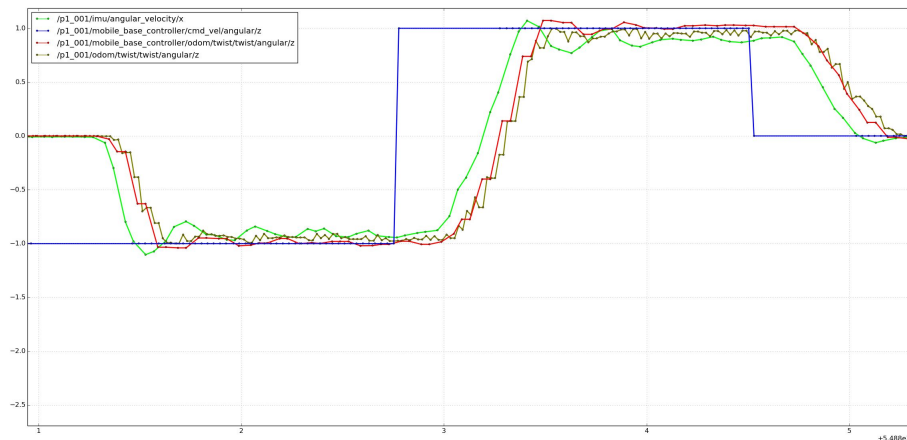
ROS - Visualization

- ROS provides tools to visualize your data
- They subscribe to your topics and display information
- **RViz** for general purpose visualization



ROS - Visualization

- ROS provides tools to visualize your data
- They subscribe to your topics and display information
- **RQT** for more specialized analysis





ROS - Code Editors

- Recommend using VS Code with plugins:
 - ROS
 - Python
- You can of course use whichever other editor you want, such as:
 - Sublime
 - Atom
 - Vim
 - Emacs
 - QtCreator
 - CLion
 - PyCharm
- Here you can find more information about IDEs for ROS: <https://wiki.ros.org/IDEs>



ROS - When you need help

- Lots of nice tutorials and information on the ROS webpage. Always look there first for information on ROS/ROS packages
 - ROS wiki: <https://wiki.ros.org/>
 - ROS Q/A: <https://answers.ros.org/>
 - ROS tutorials: <https://wiki.ros.org/ROS/Tutorials>



Kattis

You must register in Kattis - merely logging in is not enough!



Kattis

You must register in Kattis - merely logging in is not enough!

Kattis is the autograding system used by the EECS school. It is used for assignments 2, 3, and 4 in this course.

Use your personal KTH log-in. Kattis is equipped with a plagiarism checker, and if another student's solution is submitted with your account, this will count as attempted plagiarism.

Kattis is **not a debugging tool**. Ensure that all your code works in your own development environment, with all the supplied practice test cases, before submitting to Kattis.



Kattis

You must register in Kattis - merely logging in is not enough!

Introduction to Robotics – DD2410/irob18

This course offering ended 2018-12-31

I am a student taking this course and I want to register for it on Kattis.

- [Course website](#)
- [Problem list](#)
- [Students](#)
- [Export course data](#)

Teachers

- [Christian Smith](#)
- [Michael C. Welle](#)
- [Patric Jensfelt](#)
- [Petter Ögren](#)



Kattis

You must register in Kattis - merely logging in is not enough!

Introduction to Robotics – DD2410/irob18

This course offering ended 2018-12-31

You are registered for this course on Kattis.

- [Course website](#)
- [Problem list](#)
- [Students](#)
- [Export course data](#)

Teachers

- [Christian Smith](#)
- [Michael C. Welle](#)
- [Patric Jensfelt](#)
- [Petter Ögren](#)

You must register in Kattis - merely logging in is not enough!

Introduction to Robotics – DD2410/irob18

This course offering ended 2018-12-31

You are registered for this course on Kattis.

- [Course website](#)
- [Problem list](#)
- [Study hints](#)
- [Export course data](#)

Teachers

- [Christian S.](#)
- [Michael C. Weine](#)
- [Patric Jonsson](#)
- [Patrik Ögren](#)





Queue

- When you want to present or ask a question at the sessions
- Use the KTH queueing system
 - <http://queue.csc.kth.se/#/queue/lrob>



Canvas

- In Canvas we have created different *topics*
- When you want to ask something
 - Go to the topic that best matches what you want to ask
 - Look through the topic if someone else has already asked a similar question
 - *You can use the search field for this*
 - Ask your question if no one else has asked the question
- We are grateful if you answer each other's questions
 - As long as you do not solve the assignment for the other person

- Do you want a **live demo** of this?



Assignment 1



Install - Own computer

```
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" >
/etc/apt/sources.list.d/ros-latest.list'
```

```
sudo apt-key adv --keyserver 'hkp://keyserver.ubuntu.com:80' --recv-key
C1CF6E31E6BADE8868B172B4F42ED6FBAB17C654
```

```
sudo apt update
```

```
sudo apt install ros-melodic-desktop-full
```

```
sudo rosdep init
```

```
rosdep update
```

```
sudo apt install python-rosinstall python-rosinstall-generator python-wstool
build-essential python-pip python-catkin-tools
```

```
sudo apt install ros-melodic-ros-tutorials ros-melodic-turtlebot3
ros-melodic-turtlebot3-simulations ros-melodic-navigation libspatialindex-dev
libqt4-dev
```

```
sudo apt install ros-melodic-rqt ros-melodic-rqt-common-plugins ros-melodic-turtlesim
```

```
sudo apt install ros-melodic-turtle-tf2 ros-melodic-tf2-tools ros-melodic-tf
pip install rtree sklearn
```



Install - School computer

```
pip install rtree sklearn
```

```
cd ~
```

```
mkdir not
```

```
mv .nv .nvidia-settings-rc not
```



Install - Source and create ROS workspace

```
echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc  
source ~/.bashrc
```

```
mkdir -p ~/catkin_ws/src
```

```
cd ~/catkin_ws/
```

```
catkin_make
```

```
echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc  
source ~/.bashrc
```



Tutorials

- You should do some of the basic ROS tutorials
 - Create a ROS package
 - Building a ROS package
 - Writing a publisher and subscriber
 - Writing a service and client
- TF2 tutorials
 - Introduction to TF2
 - Static broadcaster
 - Broadcaster
 - Listener
- Actionlib
 - Writing a action server
 - Writing a action client

All of these are mostly copy-paste and see what happens

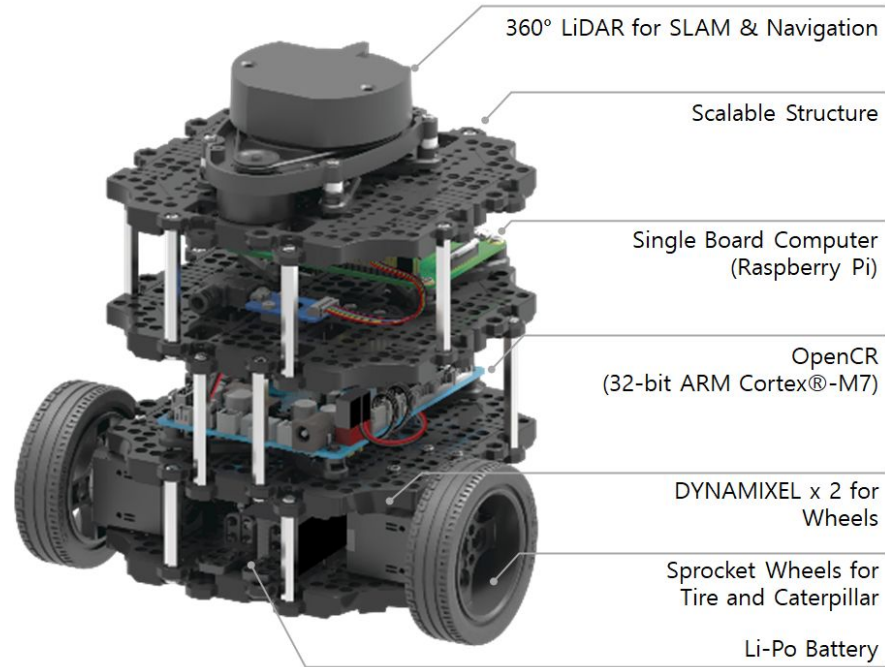


Mini-project

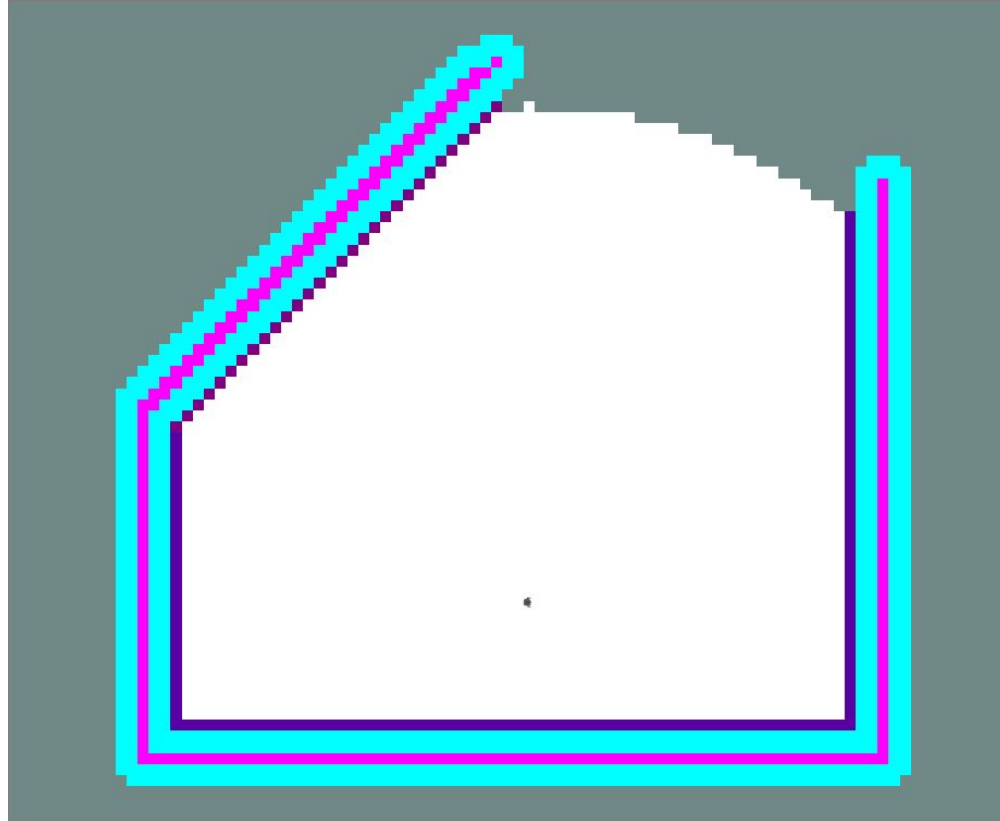
- At the end you should use what you learned from the tutorials on a real problem
- The problem in question is **exploration**

Mini-project - Meet Burger

TurtleBot3 Burger



Mini-project - Meet the environment





Mini-project

Given:

- Exploration node
- Collision avoidance node
- SLAM node
- ...
-



Mini-project

Given:

- Exploration node
- Collision avoidance node
- SLAM node
- ...

What you should do:

- Controller



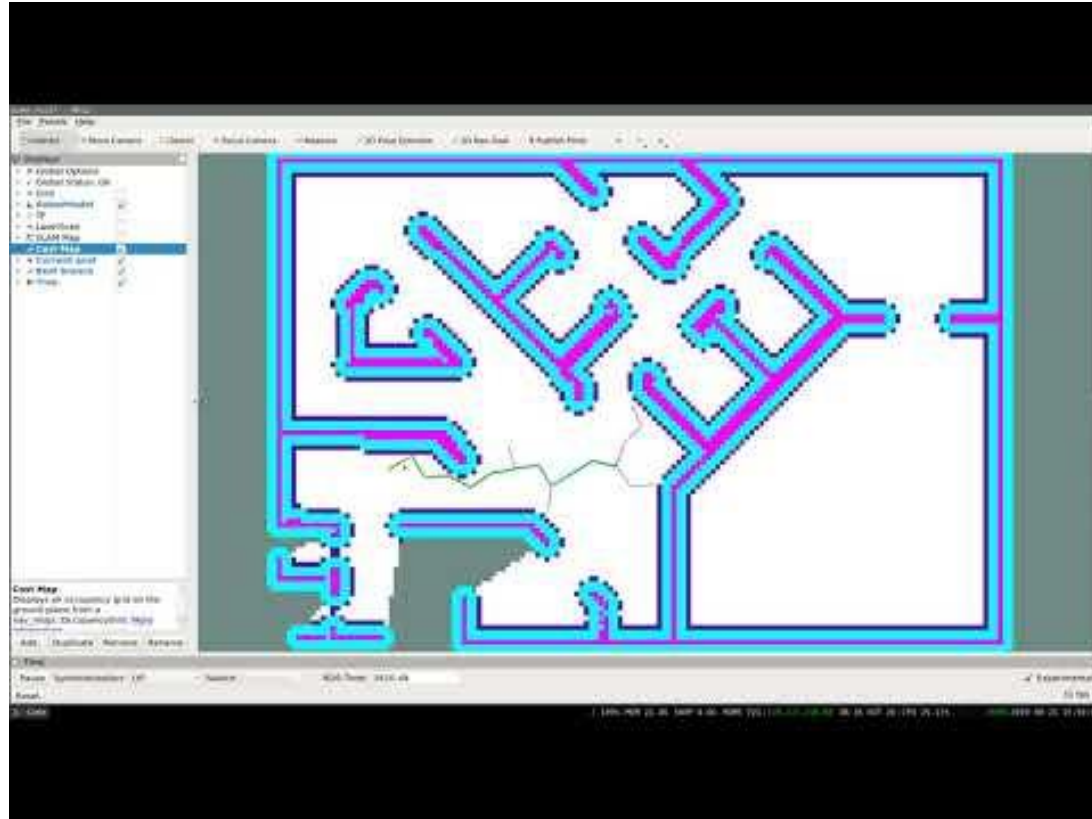
Mini-project - Controller

The controller should do:

1. Call the exploration *action server* to get a **path**
2. Call the collision avoidance *service* to get **new_path** and a **setpoint**
3. Transform the **setpoint** from frame *X* to frame *base_link* using TF2
4. Publish the transformed setpoint to the topic */cmd_vel*
5. If **new_path** is not empty then set **path** to **new_path** and go back to 2
6. When **path** is empty go to 1
7. When the **path** from the action server is empty right away -> exit



Mini-project - Video 2





The presentation

- The presentation will focus on ROS
- You do **not** have to write superb code
- Focus on understanding the different ROS concepts
- You should be able to explain what every line of your code does
- You should be able to answer questions such as
 - What is the difference between topics and services?
 - What is TF used for?
 - Why would you use actionlib instead of a service?
 - What does `init_node(...)` do?
 - ...



Questions?



Live demo