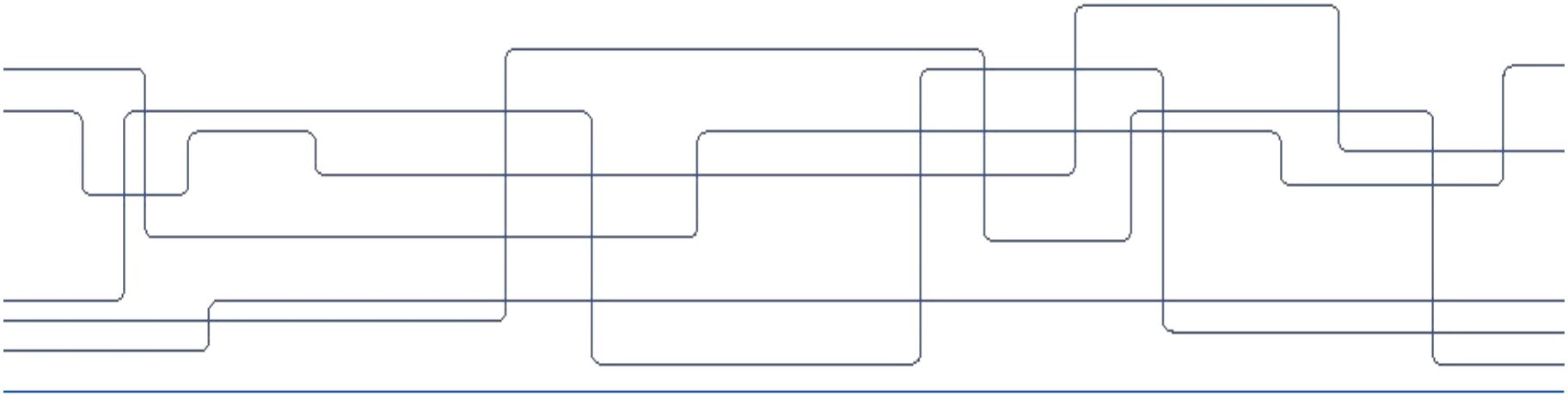
KTH ROYAL INSTITUTE OF TECHNOLOGY

Assignment 1: Introduction to ROS

Introduction to Robotics
DD2410

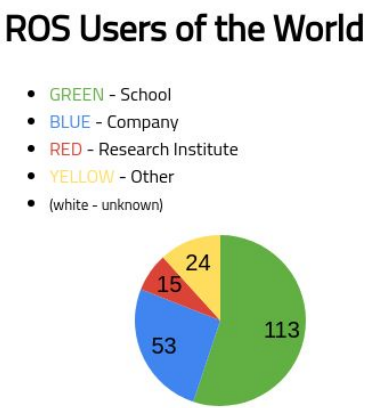


What is Robot Operating System (ROS)?

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

Why ROS?

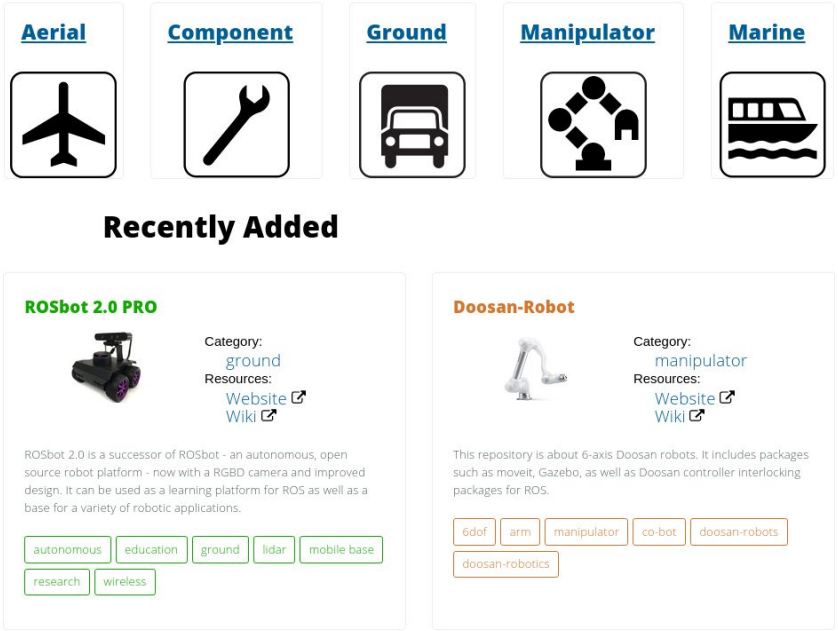
- Very large user community
- Standard in many robotics labs around the world, even in some companies



http://metrorobots.com/rosmap.html

Why ROS?

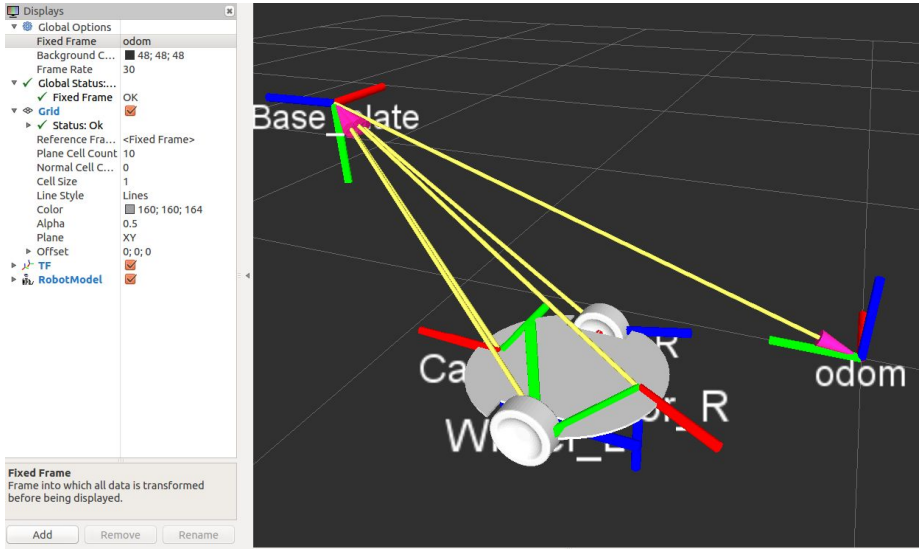
- Many commercially available robots use ROS nowadays
- https://robots.ros.org/



Why ROS?

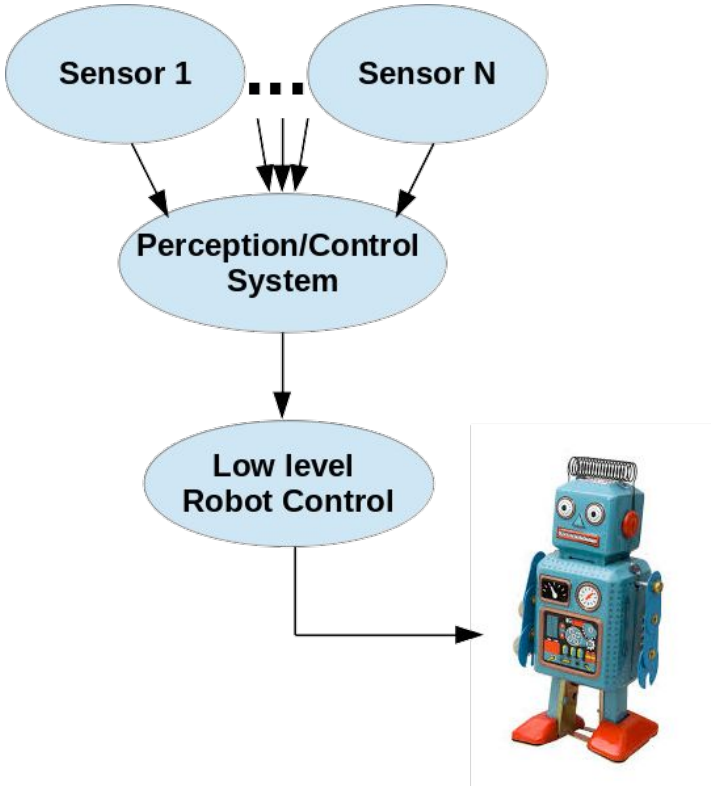
Open source + large community = **lots of packages, libraries, and tools available**

- 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

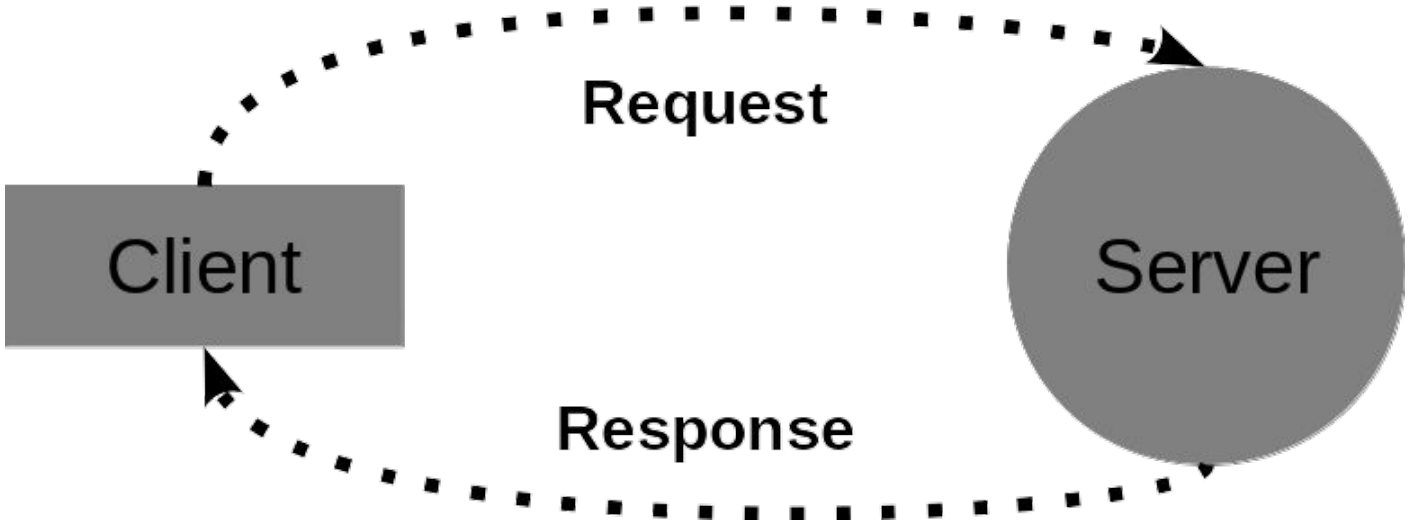
- ROS topics implement communication channels
- Many-to-many relationship
- Each topic has a type of message

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

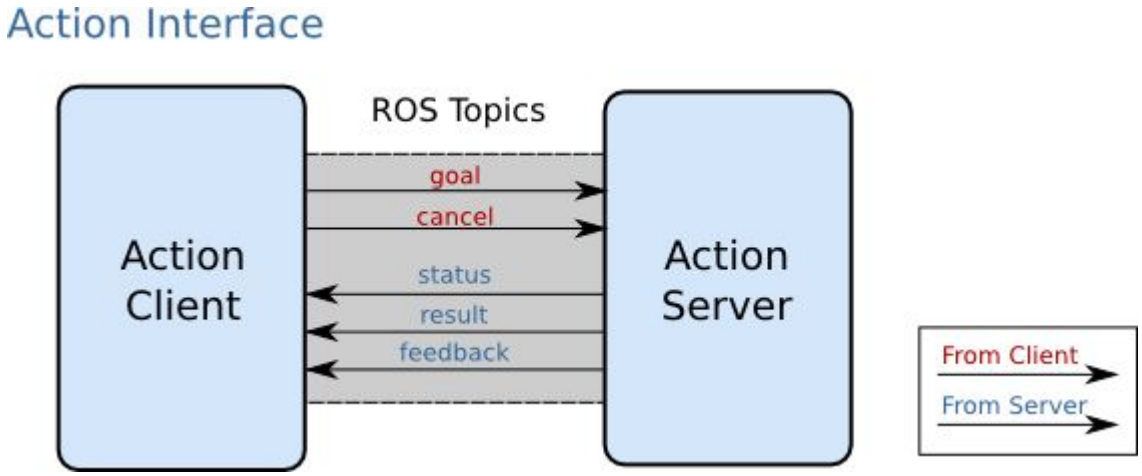
ROS - Services

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



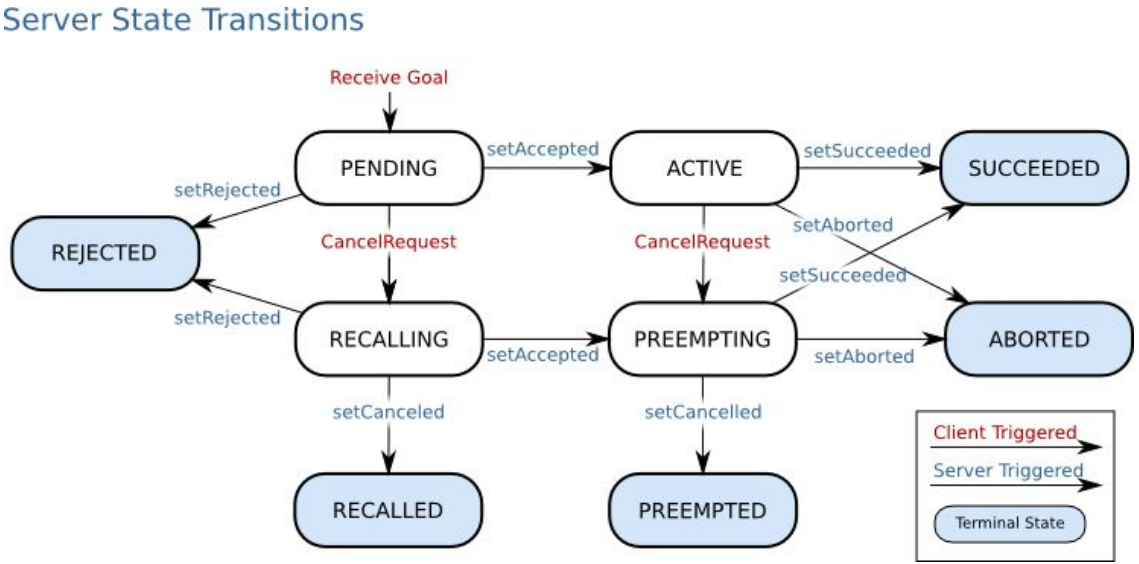
ROS - Actions

- 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



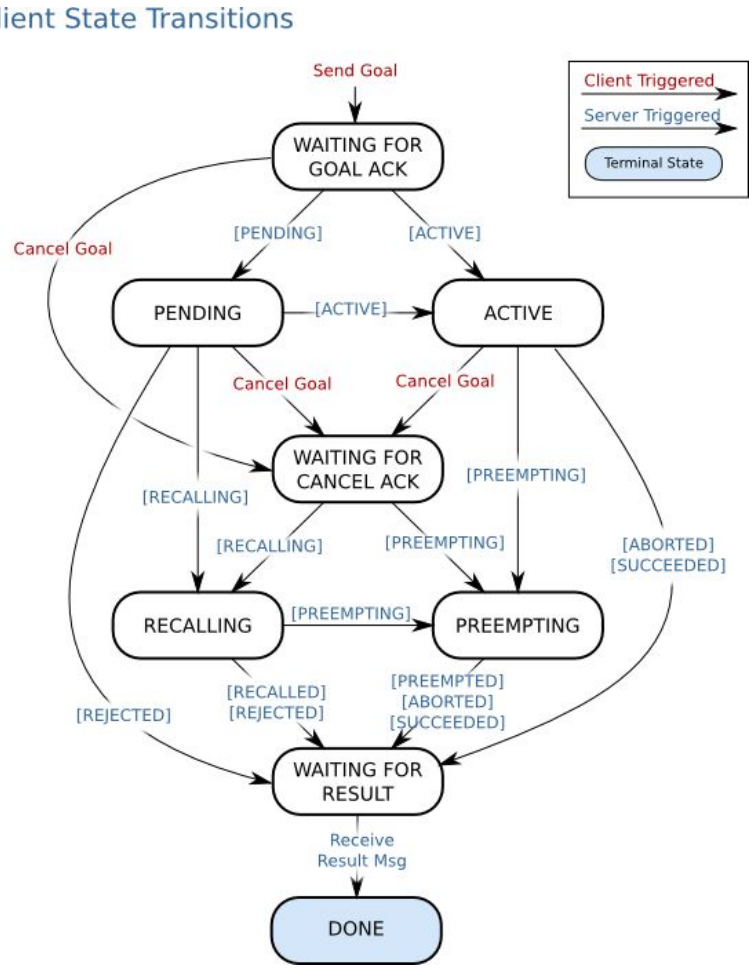
ROS - Action server

- The action server state transitions



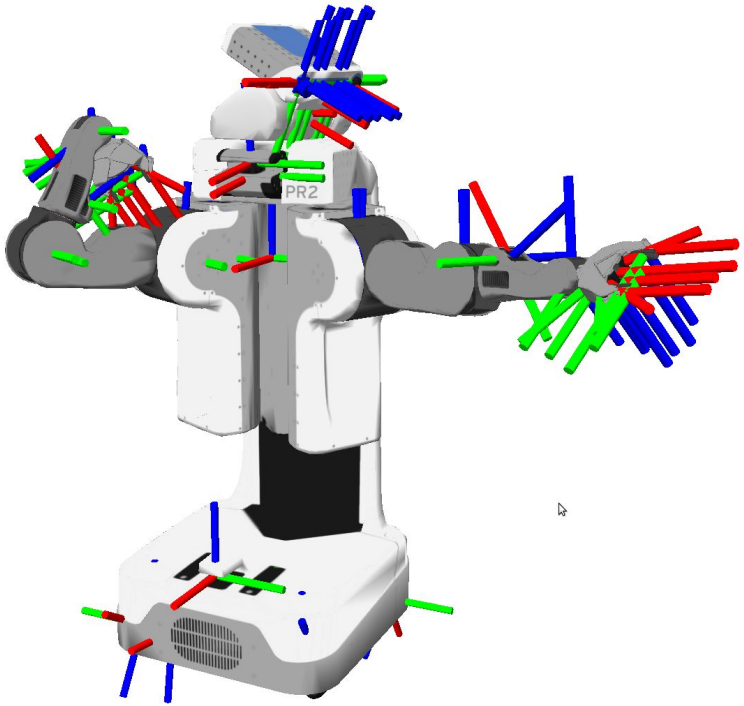
ROS - Action client

- The action client state transitions



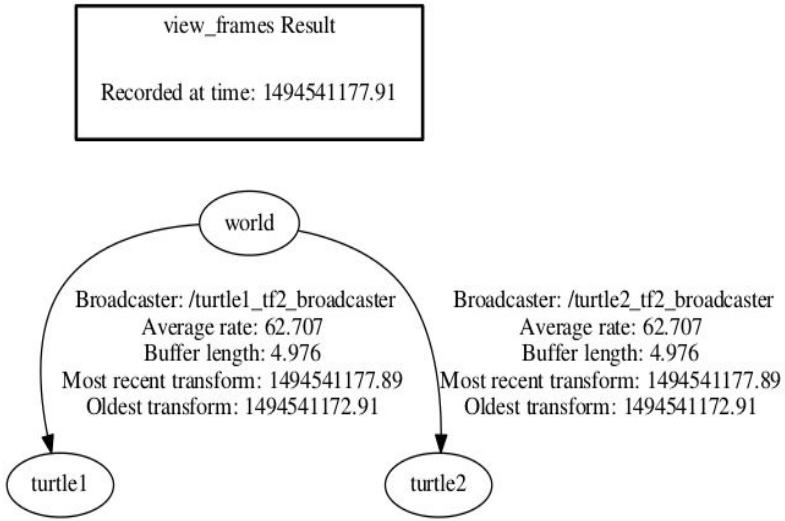
ROS - TF

- Keep track of multiple coordinate frames over time



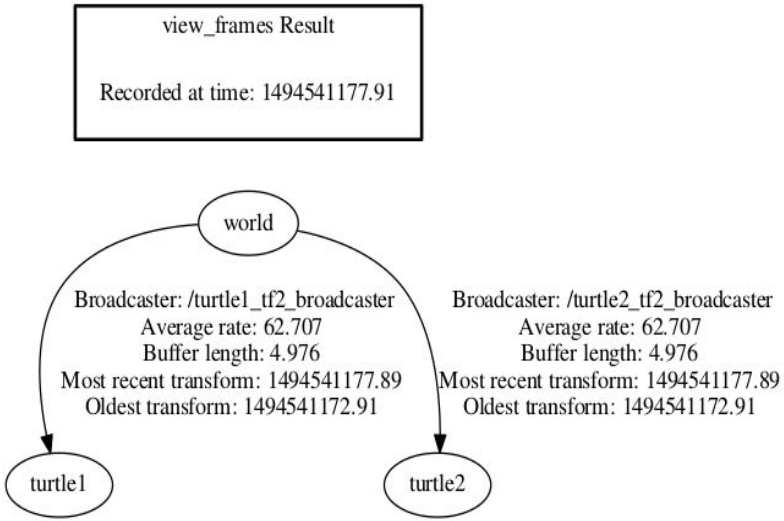
ROS - TF

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

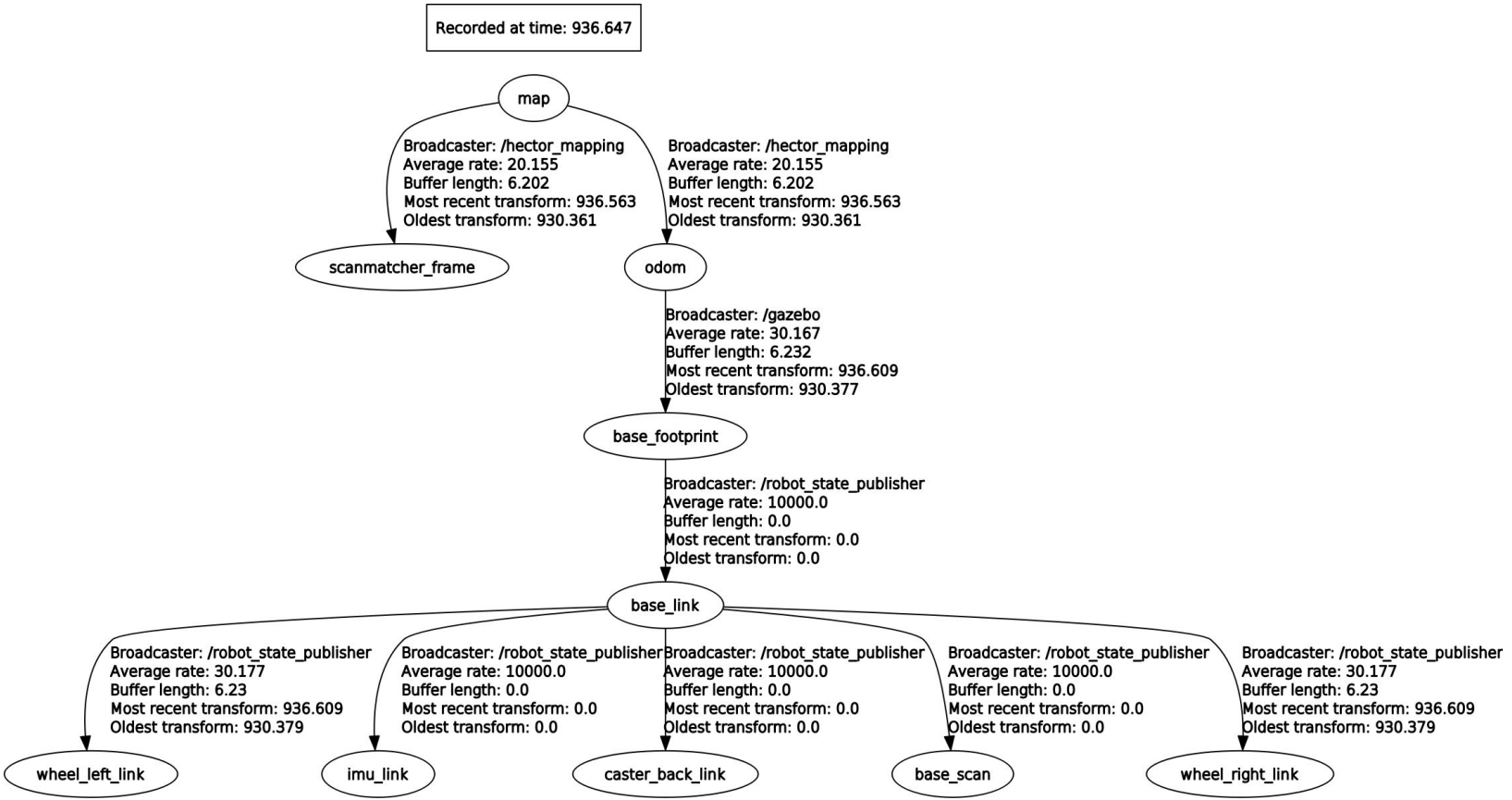


ROS - TF

- 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


Recorded at time: 936.647
map
Broadcaster: /hector_mapping
Average rate: 20.155
Buffer length: 6.202
Most recent transform: 936.563
Oldest transform: 930.361
Broadcaster: /hector_mapping
Average rate: 20.155
Buffer length: 6.202
Most recent transform: 936.563
Oldest transform: 930.361
scanmatcher_frame
odom
Broadcaster: /gazebo
Average rate: 30.167
Buffer length: 6.232
Most recent transform: 936.609
Oldest transform: 930.377
base_footprint
Broadcaster: /robot_state_publisher
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
base_link
Broadcaster: /robot_state_publisher
Average rate: 30.177
Buffer length: 6.23
Most recent transform: 936.609
Oldest transform: 930.379
Broadcaster: /robot_state_publisher
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
Broadcaster: /robot_state_publisher
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
Broadcaster: /robot_state_publisher
Average rate: 10000.0
Buffer length: 0.0
Most recent transform: 0.0
Oldest transform: 0.0
Broadcaster: /robot_state_publisher
Average rate: 30.177
Buffer length: 6.23
Most recent transform: 936.609
Oldest transform: 930.379
wheel_left_link
imu_link
caster_back_link
base_scan
wheel_right_link

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

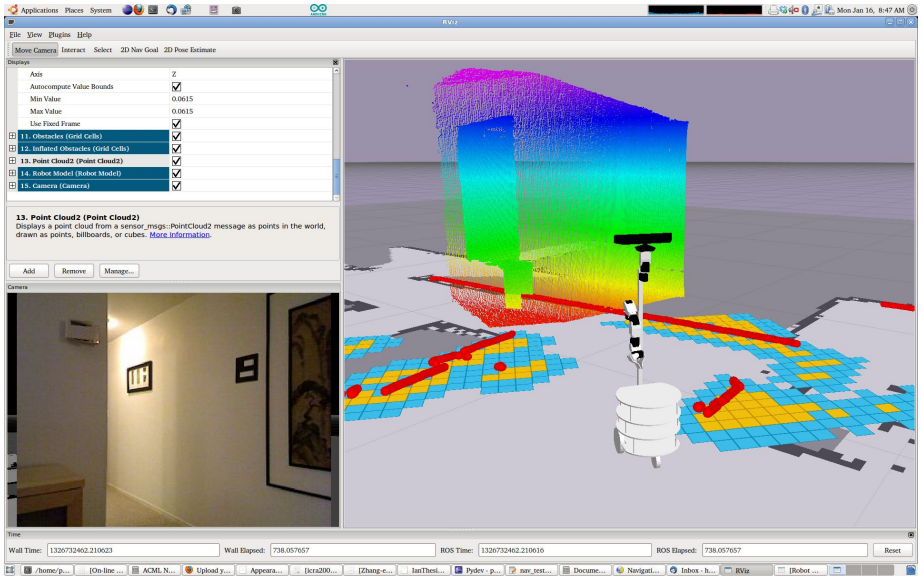
- Everything is installed for you in the computer labs
  - E Building: Röd, Orange, Gul, Grön, Brun, Grå, Karmosin, Vit, Magenta
  - D building: Spel, sport

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

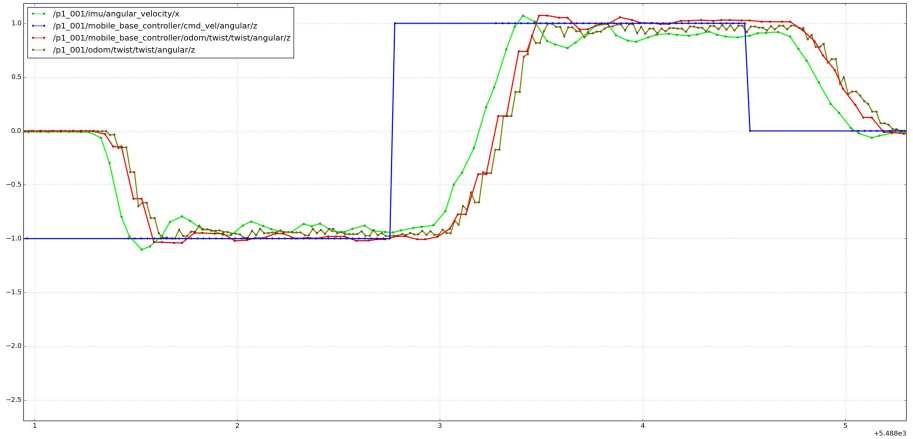
ROS - Visualization

- 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

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

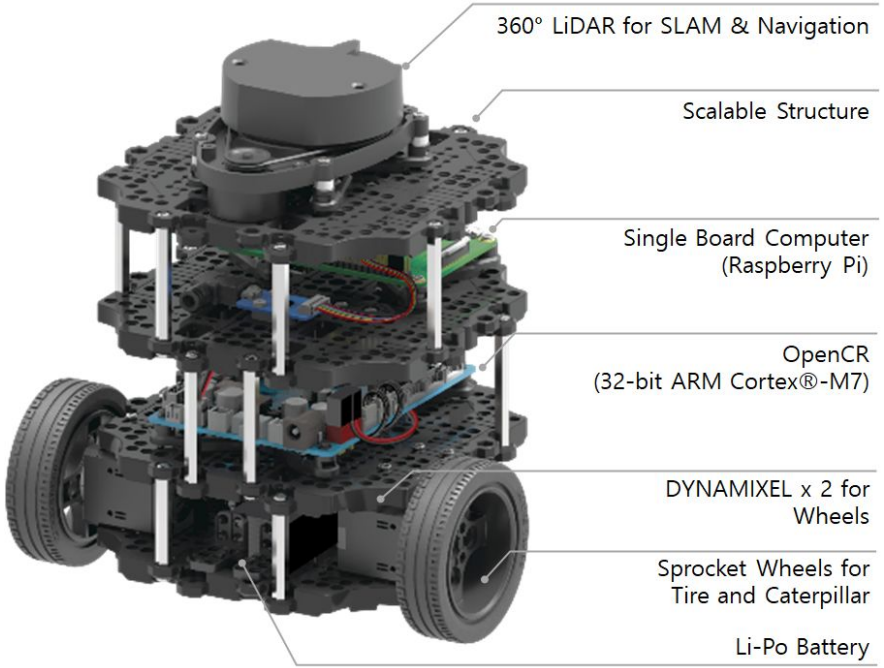
All of these are mostly copy-paste and see what happens

Mini-project

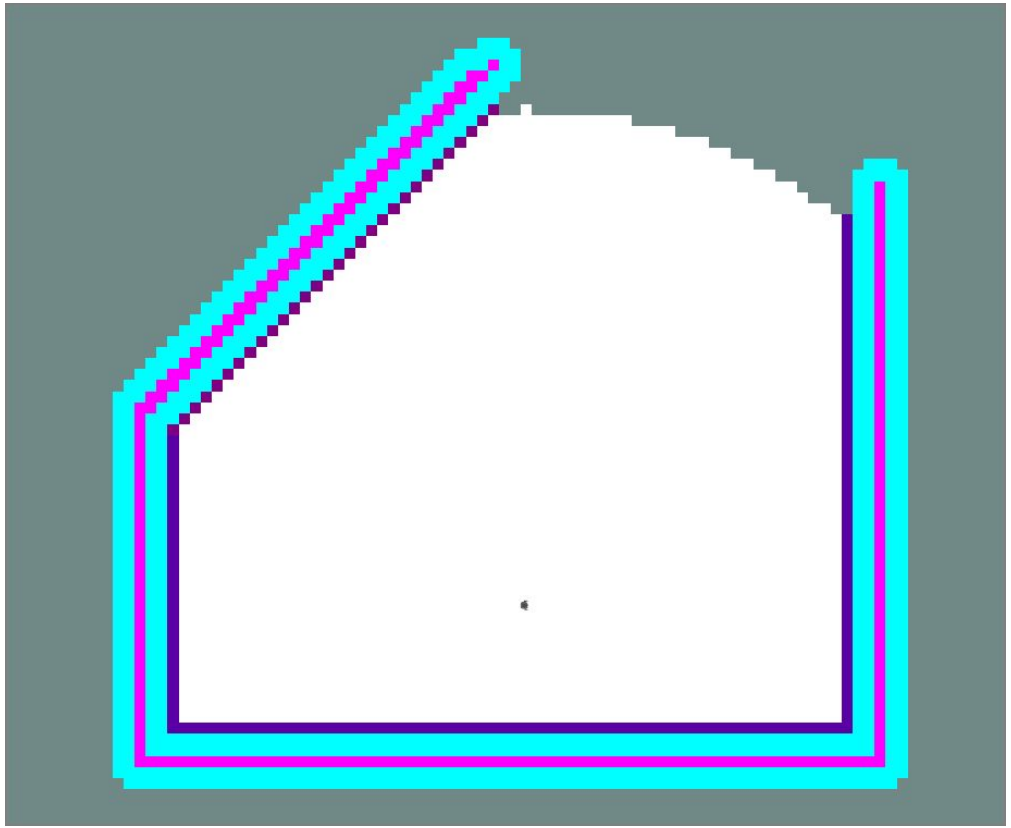
- At the end you should use what you learned from the tutorials on a real problem
- The problem in question is **exploration**

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:

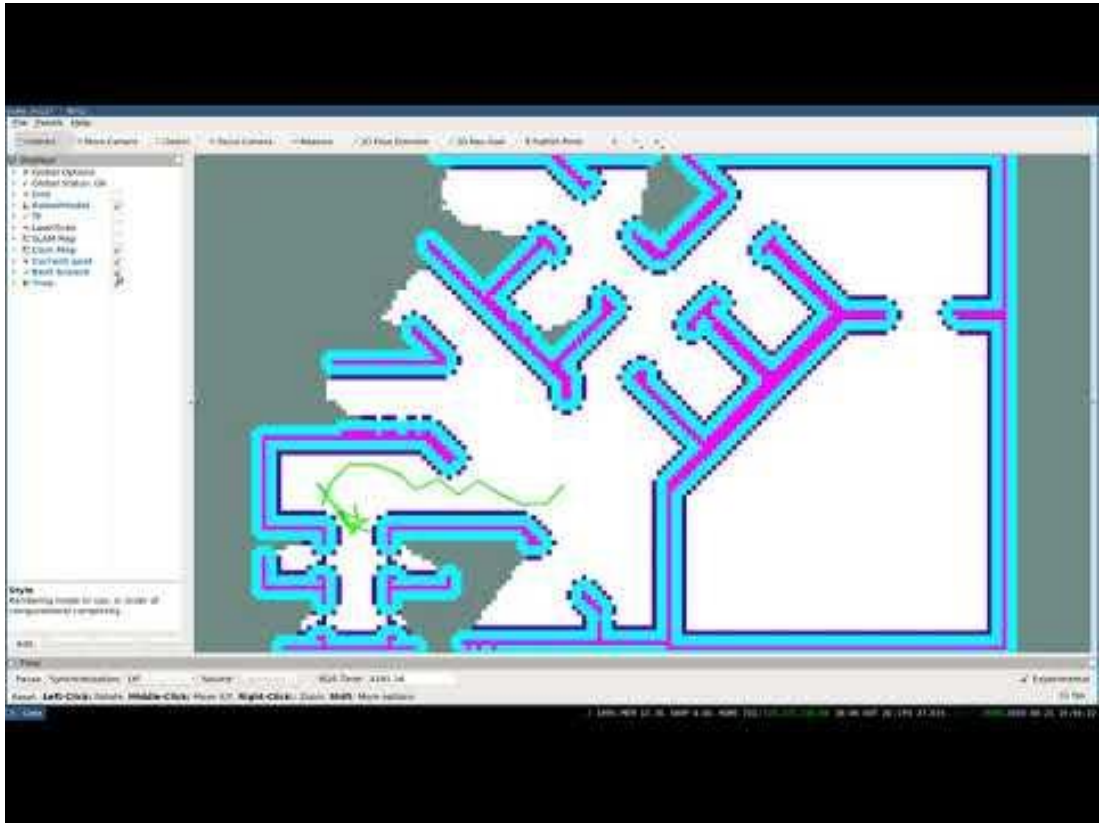
- Controller

Mini-project - Controller

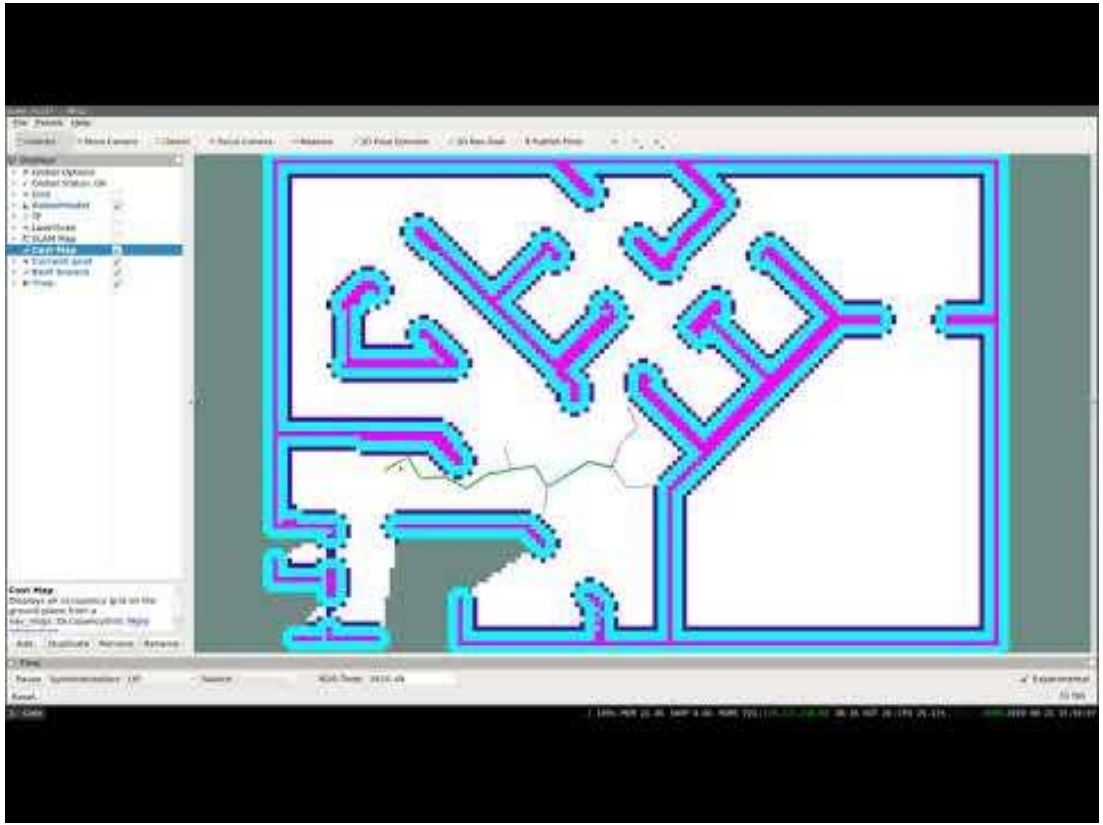
The controller should do:

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 the exit

Mini-project - Video 1



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?