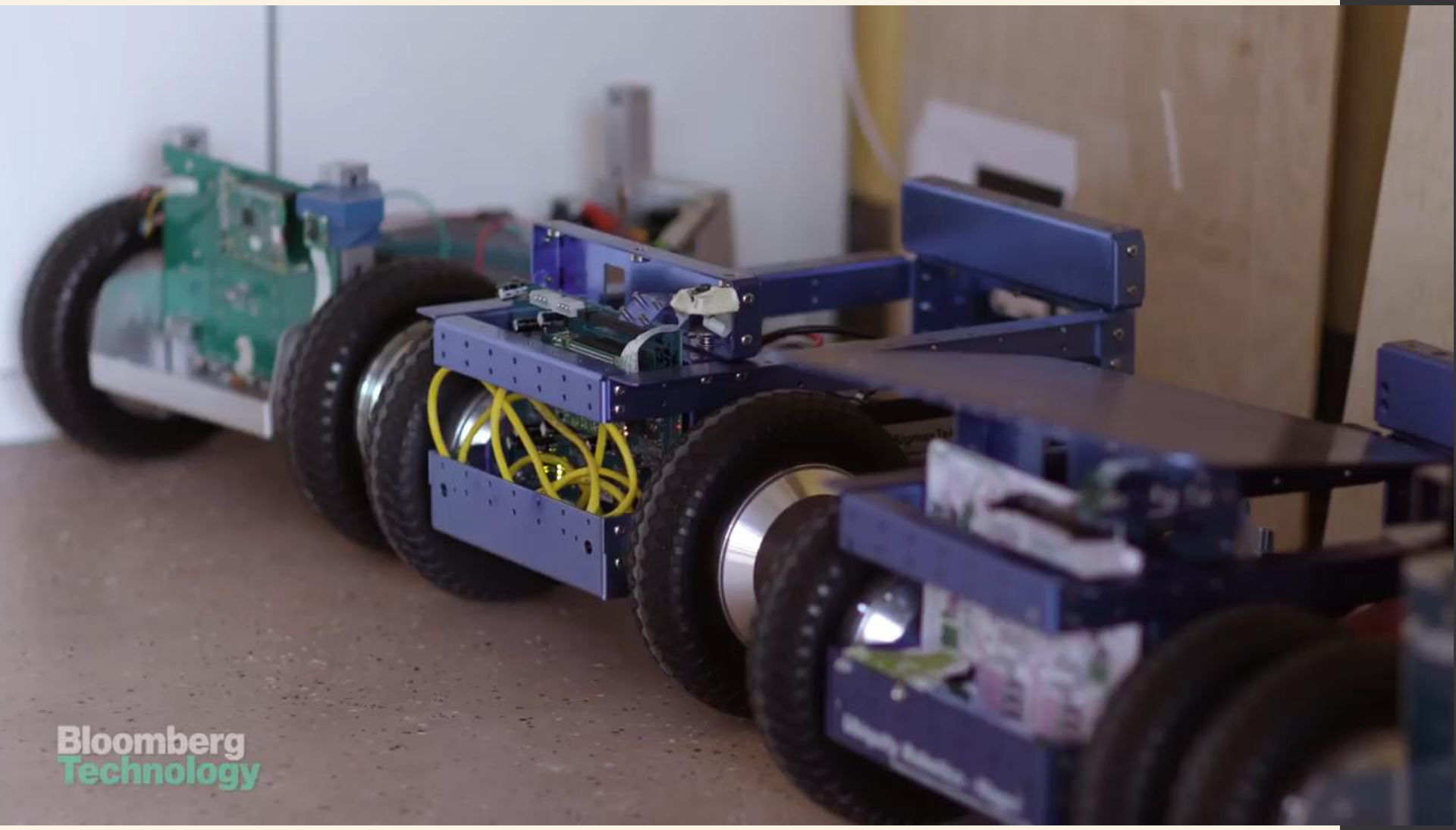
COMPSYS726: Robotics and Intelligent Systems

Topic: An introduction to ROS

ROS

- Robot Operating System
  - Open source - BSD-3-Clause licence
  - A flexible framework for writing robotic software
  - General-purpose robotic software is difficult
  - Encourage collaborative robotic software development
    - Modular and Distributed design
      - Scalable and easy to connect modules
- Reference materials:
  - **A Gentle Introduction to ROS:** https://www.cse.sc.edu/~jokane/agitr/
  - Main site - www.ros.org
  - Documentation - http://wiki.ros.org/
  - Cpp API - http://docs.ros.org/api/roscpp/html/
- We will be using:
  - Ubuntu 16.04
  - ROS Kinetic Kame – Long term support version

Building a Robot Operating System for the Future

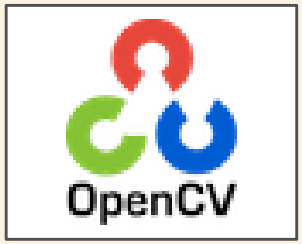


- https://www.youtube.com/watch?v=Dm7HnQb8n9Y

ROS

- Meta-operating system
  - Provides services
    - 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
- ROS is not a real-time framework
  - Possible to integrate ROS with real-time code.
- Language independent
  - Implementations in Python, C++, and Lisp,
    - Experimental libraries in Java and Lua.

ROS Integration



ROS Tools - http://wiki.ros.org/Tools

- Running ROS
  - roscore - Bringup core system
  - rosrun - Run a single program
  - roslaunch - Launching/configuring multiple programs
- Interacting and debugging
  - rostopic – reading and sending messages
  - rviz – sensor data visualisation
  - … (many more) ..
- Workspace creation and compiling
  - catkin
    - catkin_init_workspace
    - catkin_make
    - catkin_create_pkg

catkin - http://wiki.ros.org/catkin/conceptual_overview

- The official build system of ROS
  - Successor to the original ROS build (rosbuild)
    - http://wiki.ros.org/catkin_or_rosbuild
    - https://www.clearpathrobotics.com/2013/09/introducing-catkin/
  - Allows better distribution, cross-compiling support and portability
- Combines CMake and Python scripts
  - CMake - is an open-source, cross-platform family of tools designed to build, test and package software.
  - Custom CMake macros along with some Python code.
- catkin add onto Cmake
  - support for automatic 'find package' infrastructure and building multiple, dependent projects at the same time.
- Decoupled from ROS
  - Can be used even for non-ROS projects

catkin Workspaces -
http://wiki.ros.org/catkin/workspaces

- Workspace_folder/ -- workspace folder
  - src/ -- source space
    - CMakeLists.txt – The top-level CMake file
    - Package_1/
      - CMakeLists.txt
      - package.xml
      - …
    - Package_n/
      - …
  - build/ -- build space
  - devel/ -- development space (CATKIN_DEVEL_PREFIX)
    - Bin/
    - …
    - ….
    - setup.bash
  - install/ -- install space (CMAKE_INSTALL_PREFIX)
    - ….

Tools

- rospack:
  - find and retrieve information about packages
- **catkin_init_workspace**
  - Initialise a workspace
- **catkin_create_pkg**:
  - create a new package
- **catkin_make**:
  - build a workspace of packages
- rosdep:
  - install system dependencies of a package
- rqt:
  - In rqt there is a plugin called "Introspection/Package Graph", which visualizes package dependencies as a graph

rosout

- A mechanism for specifying different level of human-readable log messages:
  - DEBUG
    - Information only used during development and debugging. Should be turned off after deployment.
      - i.e., received message from node N with values x,y,z
  - INFO
    - Useful information to the user
      - i.e., Node initialized
  - WARN
    - Information that the user may find alarming and can affect the output of the application but is part of expected workings of the system.
      - i.e., "Could not load configuration file from <path>. Using defaults."
  - ERROR
    - Something serious has gone wrong but is recoverable.
      - i.e., “Unexpected input, treated as zero”
  - FATAL
    - Something unrecoverable has occurred
      - i.e., “Robot is on fire!”
- Better than printf since you can turn on/off certain levels at runtime.
  - Configuration file
  - Programming
  - Tool - rqt_logger_level
  - http://wiki.ros.org/roscpp/Overview/Logging

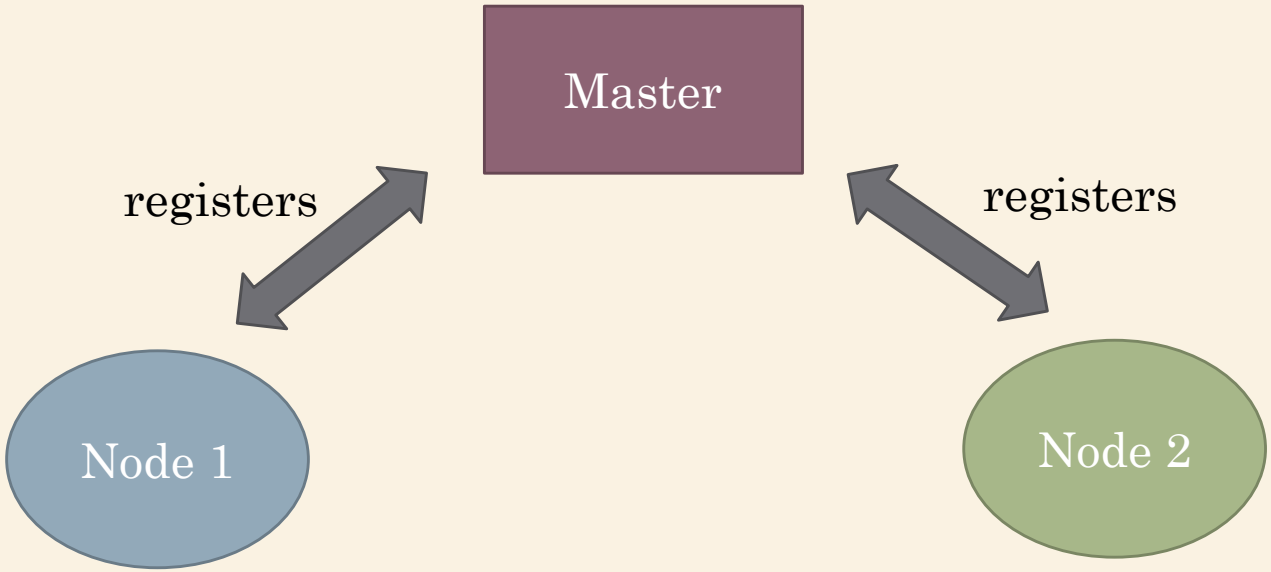
ROS Computation Graph

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

  - Master – provides name registration and lookup to the rest of the Computation Graph. Without the Master, nodes would not be able to find each other, exchange messages, or invoke services.
    - roscore:
      - ROS Master
      - ROS Parameter Server – shared node to store parameters
        - i.e., /camera/left/name : leftcamera
      - rosout – console log reporting mechanism in ROS
  - Envrionmental variables
    - ROS_MASTER_URI
      - export ROS_MASTER_URI=http://localhost:11311/

Nodes

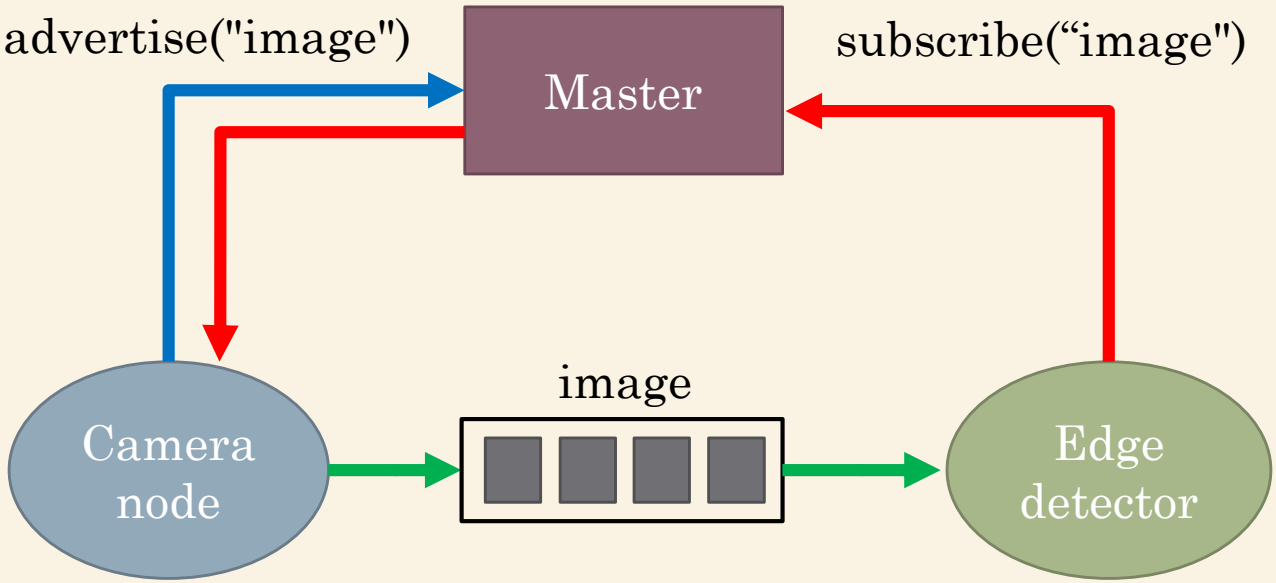
- Nodes are processes that perform computation. A robot control system usually comprises many nodes.
  - For example, laser range-finder node, wheel motors node, localization node … etc
  - An ***instance*** of a ROS program.
  - Environmental variable
    - ROS_IP and ROS_HOSTNAME – only required if the computer have multiple IP addresses/interfaces.



Communication

- Topics
  - Name – package name / message name
    - i.e., geometry_msgs/Twist
  - Type – constants and fields
    - .msg file
    - “rosmsg list” – display all type
      - http://wiki.ros.org/common_msgs?distro=kinetic
  - TCP/IP (default) and UDP
  - Multiple *publisher* and *subscribers*
  - Asynchronous “stream-like” communication
- Services
  - Like topics have name and a type (.srv file)
  - For request – reply interactions
    - A message for request and a message for reply
    - A single service can have multiple clients
  - TCP/IP and UDP
  - Like a remote function call
  - Synchronous

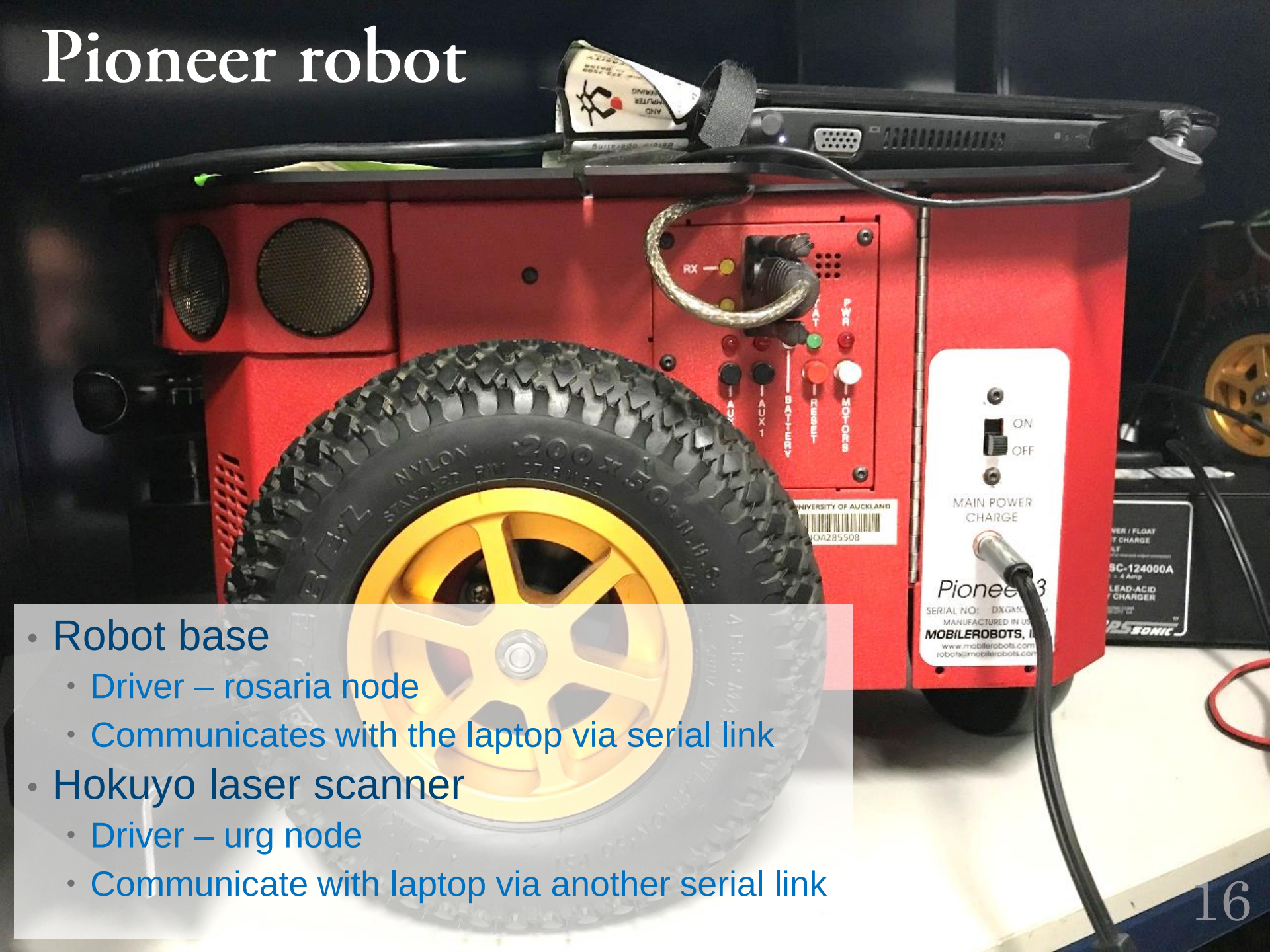
Communication



Demo

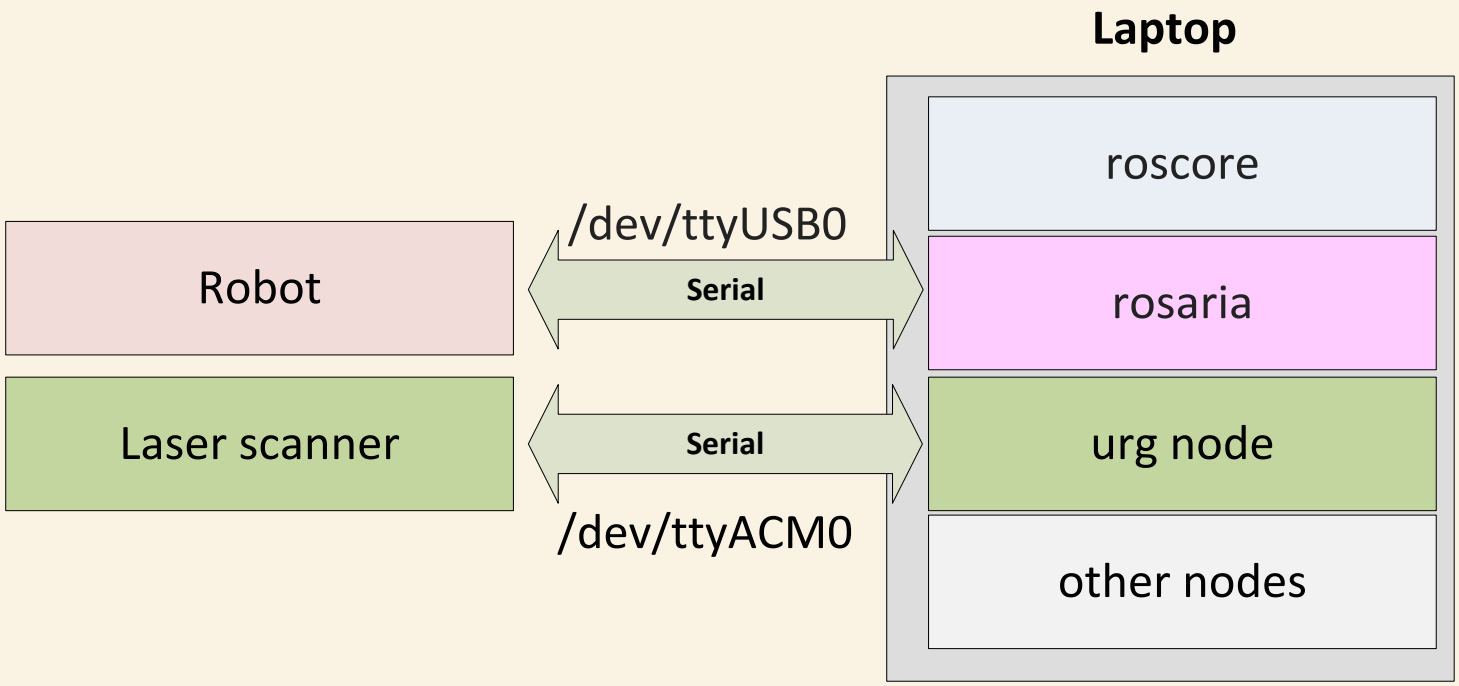
1. Publisher creation
2. rostopic – echo
3. Subscriber creation
4. rostopic - pub
5. Tool - rqt_graph
   - rqt_graph provides a GUI plugin for visualizing the ROS computation graph.
   - http://wiki.ros.org/rqt_graph

Pioneer robot



- Robot base
  - Driver – rosaria node
  - Communicates with the laptop via serial link
- Hokuyo laser scanner
  - Driver – urg node
  - Communicate with laptop via another serial link

System architecture



Workflow

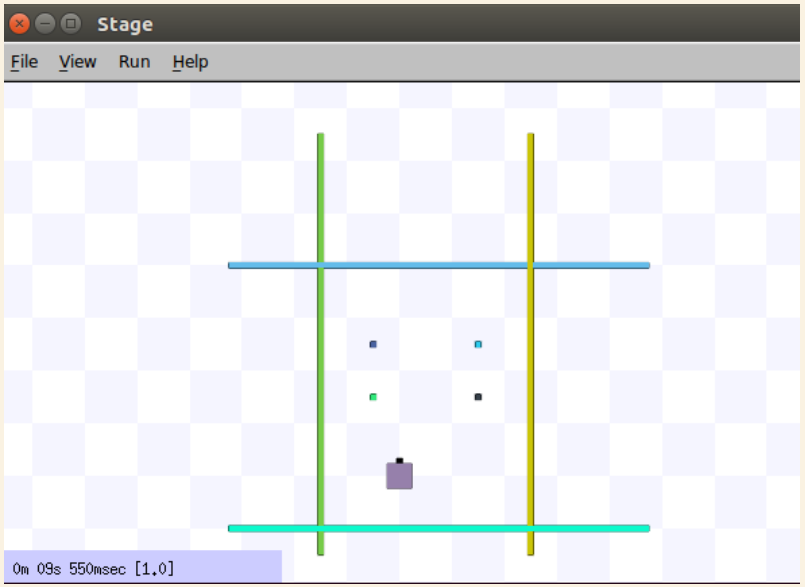
- Desktop
  - Simulation
  - desktop_workspace
  - H drive / GIT
- Micro SD card
  - workspace.tar.gz
  - source files
- Robot laptop
  - Deployment
  - workspace

uSD card to USB

uSD card to SD card

Stage

- Stage is a robot simulator. It provides a virtual world populated by mobile robots and sensors, along with various objects for the robots to sense and manipulate.
- http://playerstage.sourceforge.net/doc/stage-svn/group__model.html
- http://player-stage-manual.readthedocs.io/en/stable/



Stage

- World file (.world)
  - The .world file tells Player/Stage what things are available to put in the world. In this file you describe your robot, any items which populate the world and the layout of the world.
- Configuration file (.cfg)
  - The .cfg file is what Player reads to get all the information about the robot that you are going to use.This file tells Player which drivers it needs to use in order to interact with the robot,
- Include file (.inc)
  - The .inc file follows the same syntax and format of a .world file but it can be *included*. So if there is an object in your world that you might want to use in other worlds, such as a model of a robot, putting the robot description in a .inc file just makes it easier to copy over, it also means that if you ever want to change your robot description then you only need to do it in one place and your multiple simulations are changed too.

Stage – common

```
define model_name model
(
        # parameters
)
```

- bitmap: A bitmap file for the object
  - “path/file.png”
- size: This is the size in metres of the simulation.
  - [x y z]
- color: Color of the object
  - “random”, “orange”
- pose:
  - [x y z heading]
- origin:
  - [x y z heading]
- interval_sim:
  - Milliseconds between simulation update

Laser

```
define myLaser ranger(
sensor( range [ 0.02 5.6 ] fov 180.0 samples 512 )
size [0.1 0.1 0.1]
color "black"
block(
        points 4
        point[0] [0 0]
        point[1] [0.1 0]
        point[2] [0.1 0.1]
        point[3] [0 0.1]
        z [0 0.1]
   )
)
```

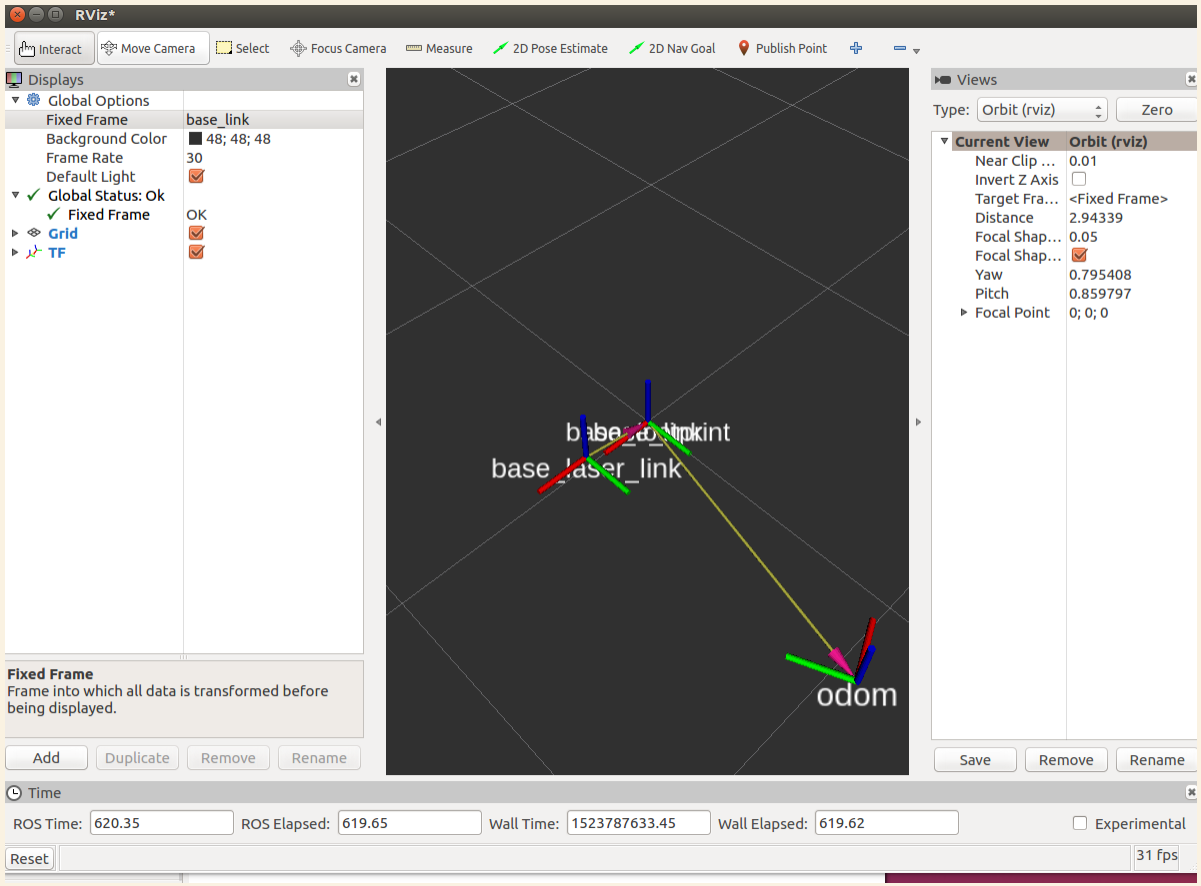
Robot

The position model simulates a mobile robot base. It can drive in one of two modes; either differential, i.e. able to control its speed and turn rate by driving left and right wheels like a Pioneer robot, or omnidirectional, i.e. able to control each of its three axes independently.

```
define myRobot position(
    size [0.46 0.46 0.25]
    drive "diff"
    myLaser(pose [ 0.28 0 -0.2 0 ])
)
```

rviz

- 3D visualization tool for ROS
- Allow you to see though the “robot’s eyes”
- http://wiki.ros.org/rviz/UserGuide
- Axis:
  - red - x
  - green - y
  - blue -z



Common Topics

- /RosAria/cmd_vel - geometry_msgs/Twist Message
  - geometry_msgs/Vector3 linear (m/s)
    - float64 x
    - float64 y
    - float64 z
  - geometry_msgs/Vector3 angular (rad/s)
    - float64 x
    - float64 y
    - float64 z
- http://docs.ros.org/api/geometry_msgs/html/msg/Twist.html
- Note: stage uses the /cmd_vel topic instead

Demo:

- Keyboard input to publish twist to cmd_vel
  - rosrun teleop_twist_keyboard teleop_twist_keyboard.py

Common Topics

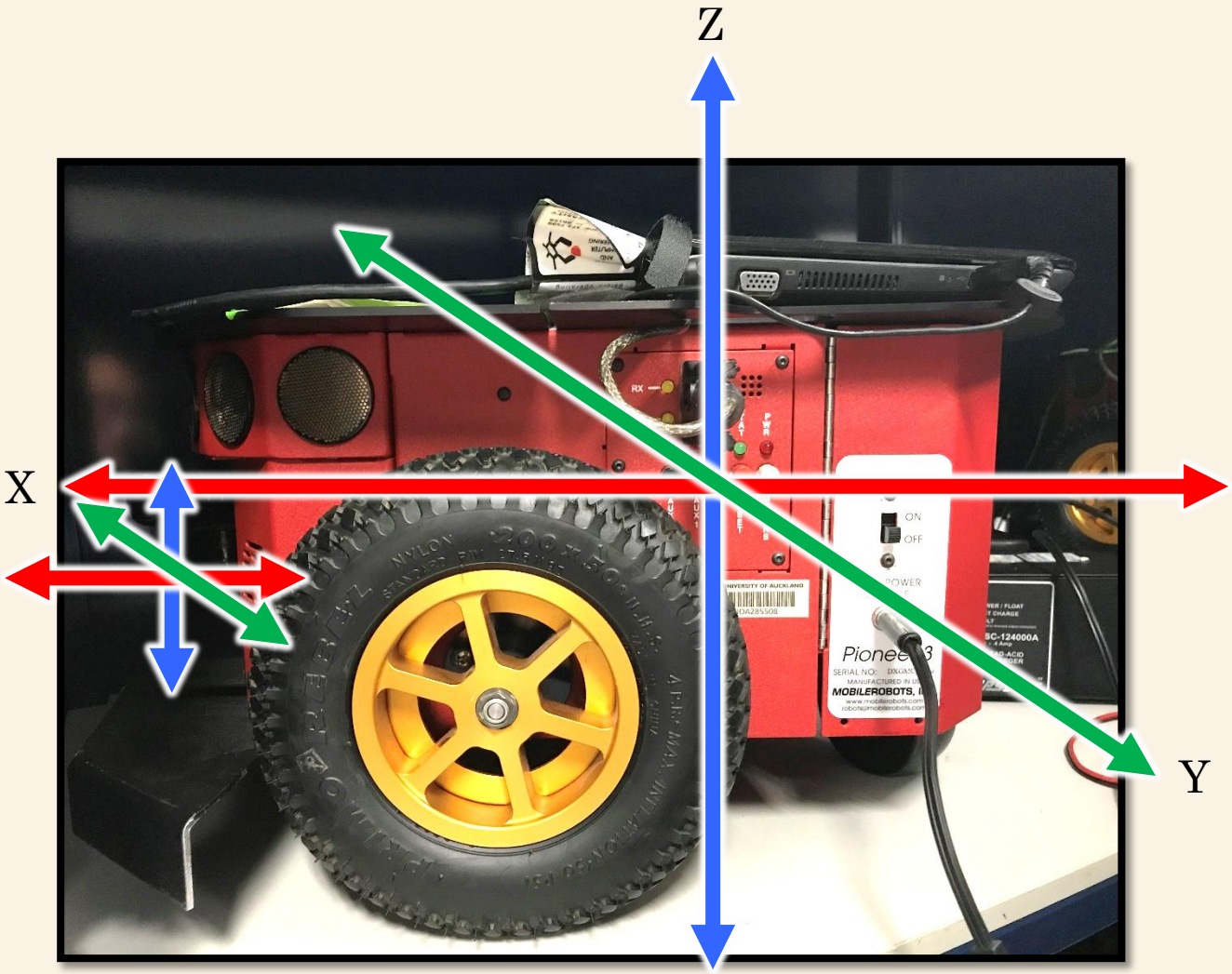
- /scan - **sensor_msgs/LaserScan.msg**
  - Header header
    - SequenceID Timestamp, frame_id,
  - float32 angle_min (rad)
  - float32 angle_max (rad)
  - float32 angle_increment (rad)
  - float32 time_increment (seconds)
  - float32 scan_time (seconds)
  - float32 range_min (m)
  - float32 range_max (m)
  - float32[] ranges (m)
  - float32[] intensities (device-specific unit – maybe empty)
- http://docs.ros.org/api/std_msgs/html/msg/Header.html

Common Topics

- /odom - **nav_msgs/Odometry Message**
  - Header header
  - string child_frame_id
  - geometry_msgs/PoseWithCovariance pose
    - geometry_msgs/Pose pose
      - geometry_msgs/Point position
        - float64 x, y, z
      - geometry_msgs/Quaternion orientation
        - float64 x,y,z,w
    - float64[36] covariance
  - geometry_msgs/TwistWithCovariance twist
    - geometry_msgs/Twist twist
      - geometry_msgs/Vector3 linear
    - float64[36] covariance
- http://docs.ros.org/api/nav_msgs/html/msg/Odometry.html

Coordinate frames

- Move in X and Rotate around Z



TF Transformation Systems

- rosrun tf tf_monitor
  - Prints information about the current transform tree
- rosrun tf view_frames
  - Prints information about the transformation between two frames
- rosrun tf tf_echo *source_frame target_frame*
  - *Creates a visual graph (PDF) of the transform tree*

- *More in http://wiki.ros.org/tf*

Launch files

- roslaunch is a tool for easily launching multiple ROS nodes
  - http://wiki.ros.org/roslaunch
  - http://wiki.ros.org/roslaunch/XML/launch
- Launch files (.launch) are xml
- Root element - <launch> </launch>
  - A ROS Node - <node></node> -
    - pkg – package
    - type – node type. Must contain an executable of the same name
    - output – log ($ROS_HOME/log) or screen (terminal)
  - Parameters - <param></param>
  - Remapping (i.e., topics)- <remap></remap>