Intro to ROS

Pitt's Robotics and Automation Society

What is ROS?

- ROS stands for Robot Operating System
  - NOT an operating system in the traditional sense
- ROS is:
  - A build and installation system
  - A development environment manager
  - A single or multi-machine launch system
  - A single or multi-machine process “manager”
  - A communication network
  - A parameter managing service
  - A community of open-source packages
  - And so much more.
- http://ros.org

Today's Plan

- For today, we are going to focus on:
  - Development environment manager
  - Single machine process manager
  - Fundamentals of the ROS communication network
  - Launching on a single machine
- Language of choice today will be **Python**
  - If you don't know Python, don't worry- it has a very simple syntax you will be able to pick up on very quickly
- For more tutorials ranging from beginner to expert, always go to the ROS Tutorials and Wiki
  - http://wiki.ros.org/ROS/Tutorials
  - http://wiki.ros.org/

Developer Environment

- ROS has what's called the ‘catkin workspace’
- Directory with all of the packages in your system in its ‘src’
- Inside src are your ‘catkin packages’ (tutorial)
    - From tutorial:
        - For a package to be considered a catkin package it must meet a few requirements:
            - The package must contain a catkin compliant package.xml file.
                - That package.xml file provides meta information about the package.
            - The package must contain a CMakeLists.txt which uses catkin.
                - If it is a catkin metapackage it must have the relevant boilerplate CMakeLists.txt file.
            - Each package must have its own folder
                - This means no nested packages nor multiple packages sharing the same directory.

Making a Package

- Navigate to the part of the tutorial (here) and make a package called 'ras_workshop'
    - If you need help, please let me know

Process Manager

- Processes (aka programs or apps) are called ‘Nodes’
- A Node is a single running process that (typically) does one particular thing
  - Ex. roomba or cone detector node
- Each Node should have a defined function and should not be codependent on any other
  - Rather, it should only be dependent on its inputs (more on this later)
  - There are exceptions to this, but 80% of the time it holds
- Tutorial: http://wiki.ros.org/ROS/Tutorials/UnderstandingNodes

Making our first Node

- Inside the 'src' folder of your new package, create a new folder called 'ras_workshop'
  - NOTE: this is the typical layout of a Python package
- Create a file called `__init__.py` and save it (yes keep it empty)
- Create a file called my_first_node.py (or whatever else you like)
  - Move on to example code here-> https://github.com/Pitt-RAS/ras_ros_workshop_example/blob/master/src/ras_workshop/node_example.py

ROS Communication

- Communication happens via messages
  - Ex. see std_msgs String here
- Topics define an exchange for messages, all of same type
- Multiple nodes can publish and listen to the same topics
- Works across all nodes, no matter what language was used to write it
- Typically have a concept of “publishers and subscribers”

Publishing and Subscribing: Examples

- A node can “publish” to a topic
  - Create an example node
    - See here
- Message is broadcast to all subscribers listening on that topic
  - Create an example subscriber
    - See here

Launching

- ROS makes it easy to define launch files that start all your nodes, load your parameters into ROS, map topics, and so much more
- Make a launch file for our publisher and subscriber
  - See here

Further Resources

- Only covered rospy (not roscpp)
  - But the concepts learned here apply in C++
  - Introduce complexities of building C++ nodes
- The ROS tutorial and ROS Wiki are well laid out and provides a ton of useful information
  - http://wiki.ros.org/ROS/Tutorials
  - http://wiki.ros.org/
- Feel free to always ask questions

Questions?