Introduction and ROS Overview
Administration

- Mrs. Mor Vered
- http://u.cs.biu.ac.il/~veredm/89-689/
- E-mail: mvered89230@gmail.com

Please state full name, course, ID when applying.

- There will be computing exercises and a final project (percentage not yet determined).
- For updates check website daily.
What is ROS – Robot Operating System?

• ROS is an open-source platform, meta-operating system for your robot by WillowGarage. It provides libraries and tools to help software developers create robot applications.
• Primary goal of ROS is to support code reuse in robotics research and development.
• ROS framework is easy to implement in any modern programming language such as Python, C++, and Lisp (experimental libraries in Java and Lua).
• ROS currently only runs on Unix-based platforms.
What is STAGE – Robot Simulator?

- 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.
- Stage provides several sensor and actuator models, including sonar or infrared rangers, scanning laser rangefinder, color-blob tracking, bumpers, grippers and mobile robot bases with odometric or global localization.
- There are three ways to use Stage:
  1) The "stage" program: a standalone robot simulation program that loads your robot control program from a library that you provide.
  2) The Stage plugin for Player/ROS (libstageplugin) - provides a population of virtual robots for the popular Player networked robot interface system.
  3) Write your own simulator: the "libstage" C++ library makes it easy to create, run and customize a Stage simulation from inside your own programs.
Helpful Links

- [http://www.willowgarage.com/pages/software/ros-platform](http://www.willowgarage.com/pages/software/ros-platform) - Robotics research lab in California, devoted to developing hardware and open source software for personal robotics applications.
- [http://playerstage.sourceforge.net/doc/ps_getting_started-1.6.txt](http://playerstage.sourceforge.net/doc/ps_getting_started-1.6.txt) - How to install player/stage
- [http://www.ros.org/wiki/stage](http://www.ros.org/wiki/stage) - Running ROS with Stage simulator.
ROS BASIC COMPONENTS

• **Nodes** - Nodes are processes that perform computation. A robot control system will usually comprise many nodes. For example, one node controls a laser range-finder, one node controls the wheel motors, one node performs localization, one node performs path planning, one node provides a graphical view of the system, and so on. A ROS node is written with the use of a ROS client library, such as roscpp or rospy.

• **Messages** - Nodes communicate with each other by passing messages. A message is simply a data structure, comprising typed fields. Standard primitive types (integer, floating point, boolean, etc.) are supported, as are arrays of primitive types. Messages can include arbitrarily nested structures and arrays (much like C structs).

• **Topics** - Messages are routed via a transport system with publish / subscribe semantics. A node sends out a message by *publishing* it to a given topic. The topic is a name that is used to identify the content of the message. A node that is interested in a certain kind of data will *subscribe* to the appropriate topic. There may be multiple concurrent publishers and subscribers for a single topic, and a single node may publish and/or subscribe to multiple topics. In general, publishers and subscribers are not aware of each others' existence. The idea is to decouple the production of information from its consumption.
- **Master** - The ROS 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.

- **Parameter Server** - The Parameter Server allows data to be stored by key in a central location. It is currently part of the Master.

- **Services** - The publish / subscribe model is a very flexible communication paradigm, but its many-to-many, one-way transport is not appropriate for request / reply interactions, which are often required in a distributed system. Request / reply is done via services, which are defined by a pair of message structures: one for the request and one for the reply. A providing node offers a service under a name and a client uses the service by sending the request message and awaiting the reply. ROS client libraries generally present this interaction to the programmer as if it were a remote procedure call.
The ROS Master acts as a nameservice in the ROS Computation Graph. It stores topics and services registration information for ROS nodes. Nodes communicate with the Master to report their registration information. As these nodes communicate with the Master, they can receive information about other registered nodes and make connections as appropriate.

Nodes connect to other nodes directly; the Master only provides lookup information, much like a DNS server. Nodes that subscribe to a topic will request connections from nodes that publish that topic. The most common protocol used in ROS is called TCPROS, which uses standard TCP/IP sockets.
Install ROS (helpful doc on TA website)
Read Tutorial