An Embedded Computing Platform for Robots

Steven Erdmanczyk
Why A Robot Specific Platform?

- Abstract tasks specific to robotics
- Modularize tasks
- Separate device level control from higher level abstract control
Already Existing Platforms

- Microsoft Robotics Developer Studio
- iRobot Create
- ROS
Microsoft Robotics Developer Studio

- Provides Visual Programming Language
- Supports C#, Visual Basic .NET, Jscript, and IronPython
- 3D Simulating Environment

Components

- CCR (Concurrency and Coordination Runtime)
- DSS (Decentralized Software Services)
- VPL (Visual Programming Language)
- VSE (Visual Simulation Environment)
iRobot Create

- Specifically for robotics development
- Centers around the iRoomba robot
- Interfaces through a serial port on the robot
- Often an external computing platform is used
• **Application Layer**
  • High level layer where logic is written
  • API and drivers should be well encapsulated to allow easy to understand flow at this level

• **OS Layer**
  • Handles management of data flow
  • coordinates tasks

• **Processor**
  • Handles processing of tasks
  • Provides hardware mechanisms for software
  • e.g. i2c, uart, pwm, procesing

• **Device**
  • Sensors, motors, communications interfaces
ROS (R.obot O.perating S.ystem)
- Open-Source meta-operating system
- Provides
  - hardware abstraction
  - low-level device control
  - implementation of commonly-used functionality
  - message-passing between processes
  - package management
- Also provides code libraries for writing code across multiple computers
- Not real-time, but can be integrated into real-time code
Goals of ROS

- Thin: Designed so that your main code can easily be ported to other projects
- Agnostic Libraries
- Language Independence
- Easy Testing: built-in unit/integration test framework ‘rostest’
Three Levels of Concepts

- File-system Level
- Computational Graph Level
- Community Level
  - Distributions
  - Repositories
  - Wiki
Computational Graph Level

- **Nodes**
  - Processes that perform computation
- **Master**
  - Provides name registration and lookup
- **Parameter Server**
  - Allows data to be stored at a central location
- **Messages**
  - Allows nodes to communicate
- **Topics**
  - Messages are sent via a transport system with publish/subscribe semantics
Community Level

- Distributions
  - Similar to Linux distributions
- Repositories
  - Different institutions can develop their own software components
- Wiki
  - Community wide documentation
Higher Level Concepts

- Coordinate Frames/Transforms
- Actions/Tasks
  - Topic based interface for preemptive tasks in ROS
- Message Ontology
- Plugins
  - Dynamically loading libraries in C++ code
- Filters
  - C++ library for processing data using filters
- Robot Model
  - Provides an xml format for representing a robot model
Currently Available Stacks

- **3d pipeline**
  - Algorithms and tools for dealing with 3D point cloud data from a variety of sensing devices

- **arm navigation**
  - Contains action/behavior used to execute collision free motion planning and control for a robot arm manipulator
Example Libraries

- Roscpp
  - C++ library
- Rospy
  - Python Library
- Rosoct
  - Library for Octave, an open source Matlab alternative
- OpenCV2
  - Open source computer vision library 2