ROS

Robot Operating System

Ignacio Escudero, Rodrigo Quinta, Andrés Tipoldi

Diciembre 2013

Agenda

- Introducción
- Conceptos
  - Nodos
  - ROS Master
  - Parameter Server
  - Mensajes
  - Tópicos
  - Services
- Integrar ROS con Butiá

Introducción

- ROS es un “meta-operating system”
  - Licencia de código abierto BSD
- Provee una forma de comunicación entre procesos por medio de mensajes en red.
- Brinda una abstracción de hardware, device drivers, librerías, visualizadores y herramientas.
  - Ayuda a los desarrolladores de software a crear aplicaciones para robots.

Introducción

- Runtime “graph” de ROS:
  - Red peer-to-peer de procesos poco acoplados, que usan la infraestructura de comunicación de ROS.

- ROS implementa diferentes formas de comunicación:
  - Comunicacion sincronica (RPC-style) sobre servicios
  - Comunicación asíncrona basada en tópicos
  - Almacenamiento de información en un “Parameter Server”.

Conceptos

- **Nodos**
  - Procesos encargados de un comportamiento.
  - Un sistema de control de un robot puede estar integrado por varios nodos.
  - Forman un grafo en el que se comunican entre ellos utilizando los topics y servicios.
  - Por lo general un nodo se encarga de una única función del robot (path planning, mover motores, etc).
  - Proporciona modularidad al sistema:
    - Mejor manejo de fallas
    - Individualidad de los nodos (exponen una API).

Conceptos

- **ROS Master**
  - Provee una forma de registro y búsqueda para el resto de la red.
  - Permite establecer conexiones entre los nodos (similar a DNS)
    - Los nodos no se comunican mediante el Master, la comunicación es entre nodos, el master simplemente establece esta comunicación.
  - Almacena la informacion para el registro a Tópicos y Servicios y actualiza a otros nodos sobre el status de los demás.
  - Provee el Parameter Server.

Conceptos

- **Parameter Server**
  - Servidor compartido que almacena parámetros que son consultados y almacenados por los nodos en runtime.
  - Utilizado principalmente para almacenar parámetros de configuración.
  - Implementado utilizando XMLRPC.
  - Por convención los parámetros son nombrados siguiendo una jerarquía para evitar colisiones de nombres.

Conceptos

- **Mensajes**
  - Los nodos se comunican a través de mensajes.
  - Estructura de datos con atributos tipados.
  - Existen varios tipos estándar predefinidos
  - Se pueden definir nuevos tipos.
  - Pueden tener una estructura anidada, un mensaje puede estar integrado por otros mensajes.
  - ROS genera una estructura de datos para los mensajes nuevos que contienen funciones estándar.

Conceptos

- **Tópicos**
  - Proporcionan un via de transporte con semántica publish / suscribe.
  - Los nodos envían mensajes a los tópicos y los nodos suscriptos a éste reciben una notificación.
  - Por lo general suscriptores y los que publican no se conocen.
  - Son un bus de mensajes y cualquiera se puede conectar para enviar o recibir mensajes

Conceptos

- **Services**
  - Proporcionan comunicacion one-to-one, Request/Reply.
  - Los nodos envían mensajes request a los nodos que provee services.
  - Los nodos que proveen servicios responden con un mensaje reply.
  - Llamado a servicio es bloqueante.

Integrar ROS con Butiá

- **Que hay que hacer?**
  - Crear un Robot Stack: Butia Stack
- **Componentes**:
  - Driver
  - Nodo
  - Modelo
  - Inicialización

Integrar ROS con Butiá

- **Driver:** butia_driver
  - Driver para el robot.
  - Usar PyBot Client?
- **Nodo:** butia_node
  - Wrapper para el driver.
  - API de ROS exponiendo los servicios (y tópicos) del robot.

Integrar ROS con Butiá

- **Modelo:** butia_description
  - URDF del robot
  - Descripción física: ubicación de sensores, actuadotes
  - Sensores móviles en Butia:
    - Modelo fijo con configuración tipica
    - Lego NXT – Lego Digital Designer

Integrar ROS con Butiá

- **Inicialización:** butia_bringup
  - Scripts de startup
  - Se invocan funciones de inicialización