

**PRACTICA 3**

**Alumno: Zepeda Rosales Ana Yadira**

**Carrera: Mecatronica**

**Grado/Grupo: 8 °A**

**Profesor: Moran Garabito Carlos Enrique**

**Asignatura: Cinemática de Robots**

Periodo Cuatrimestral: enero-abril 2019

**¿Qué es ROS?**

Robot Operating System, o simplemente ROS, es un sistema operativo ideado en 2007 por ingenieros de la Stanford University (EEUU), para dar soporte a un robot con inteligencia artificial.

ROS porporciona al usuario y desarrollador un conjunto de herramientas y algunos atajos para iniciar su aplicación desde un nivel superior. No tiene que preocuparse de cómo se comunicarán las diferentes partes. Además, dispondrá de herramientas integradas gratuitas, como la planificación del movimiento, el reconocimiento de objetos, la navegación 2D, la asignación de salas en 3D, etc. De esta manera, no necesita volver a hacer lo que han hecho antes cientos de desarrolladores, y puede ceñirse al núcleo de su proyecto.

Los desarrolladores pueden armar un sistema complejo al conectar soluciones existentes para pequeños problemas. La forma Como se implementa el sistema nos permite:

* Reemplazar componentes con interfaces similares sobre la marcha, quitando la necesidad de detener el sistema para varios cambios.
* Multiplexar salida de múltiples componentes a una entrada para otro componente, permitiendo la solución paralela de varios problemas.
* Conectar componentes hechos en varios idiomas de programación con tan solo implementar los conectores apropiados al sistema de mensajería, haciendo más fácil el desarrollo de software al conectar módulos existentes de varios desarrolladores.
* Crear nodos sobre una red de dispositivos sin preocuparse sobre dónde se ejecuta un código e implementar los sistemas de comunicación entre proceso (IPC) y llamada a procedimiento remoto (RPC).
* Conectar directamente a transmisiones en demanda de hardware remoto sin escribir código extra, esto al emplear los dos puntos anteriores.

**Conceptos**

En general ROS funciona como una arquitectura cliente-servidor.A las aplicaciones de ROS se les llama nodos, y los canales de comunicación se conocen como tópicos. Los nodos se suscriben a tópicos publicados por otros nodos, y de esta forma los nodos que están publicando en el tópico le envían la información relevante a los nodos suscritos.

Nodo

Es un programa ejecutable en ROS, dedicado a una función específica, como interactuar con actuadores o sensores, procesar datos, entre otros. Para comunicarse con otros componentes del sistema hace uso del sistema de mensajes de ROS.

**Mensaje**

Es una estructura de datos definida estáticamente que sirve para enviar información entre nodos. Se declaran en archivos de texto simple con un formato determinado y las herramientas de compilación generan las clases para poder ser usados en los nodos.

**Tópico**

Permite que los nodos se comuniquen con mensajes, suscribiéndose o publicando en un tópico dado. Se tiene una relación n a n entre nodos y tópicos pudiendo un nodo suscribir y publicar simultáneamente en cuantos tópicos quiera.

**Servicio**

Es una forma de comunicación síncrona entre nodos que usa mensajes diferentes para la petición y la respuesta. La relación entre nodos y servicios es 1 a n, muchos nodos pueden llamar a un servicio pero éste solo responde al que hizo la petición.

**Paquete y Meta-paquete**

Un nodo o conjunto de nodos, sus servicios, mensajes definidos y otros archivos se conocen como paquete, y un conjunto de paquetes como un meta-paquete.

**Servidor de parámetros:**

ROS provee un servidor para guardar parámetros que permite que distintos nodos puedan compartir información, mantener un estado por defecto para el nodo o cambiar parámetros internos sin tener que recompilar. Para acceder a estos parámetros se puede usar la herramienta rosparam en línea de comandos, las librerías clientes en Python

Y C++ o la librería y herramienta dynamic\_reconfigure La herramienta rosparam tiene un costo asociado para aplicaciones en tiempo real, dado que las consultas y respuestas de este se entregan usando el sistema de mensajes de ROS por red, lo que podría producir una pausa en la ejecución esperando la respuesta por red. Este impacto se vería en condiciones de alta pérdida de paquetes y con llamadas a rosparam dentro de un ciclo pues para cada llamada habría que esperar la respuesta.

**Herramientas**

**Dynamic reconfigure**

Es una herramienta que usa una función de callback en el código de usuario para atender las modificaciones en los parámetros del nodo, permitiendo modificar variable global directamente. Es una forma eficiente de modificar parámetros, ya que solamente se llama al Servidor de parámetros cuando se modifica el parámetro y no en cada ciclo del programa. dynamic\_reconfigure además provee una interfaz gráfica, en la que se pueden modificar los parámetros. Para esto cuenta con un archivo de configuración donde se pueden separar los parámetros en grupos y se definen valores mínimos, máximos, por defecto y tipo de datos para cada uno. Los grupos de parámetros se puede definir como visibles, invisibles o colapsables, de forma de no mostrar parámetros cuando no son necesarios.

**Roslaunch**

Es una herramienta que permite la ejecución de múltiples nodos, definiendo parámetros y opciones con una sola llamada. Hace uso de archivos XML o YAML (lanzadores) de extensión ‘.launch’ para definir nodos a correr o incluso otros archivos launch. Esto permite modularizar la ejecución de un paquete.

**Visualización con rviz**

Permite la visualización de datos de sensores, modelos de movimiento de robots, y otros datos mediante la suscripción a los tópicos correspondientes y la selección de plugins adecuados al dato seleccionado. Además permite interactuar con los robots dándoles instrucciones sobre planificación de trayectorias, movimiento, entre otros; usando los plugins para cada caso. Existe un plugin que funciona con MoveIt! para el PR2.

**Rosbag y rqt\_bag**

Son herramientas que permiten guardar la información publicada en diferentes tópicos en archivos, permitiendo la reproducción de éstos a futuro. Esto permite en un sistema separar el proceso de adquisición de datos del de procesamiento. Rosbag5 es la versión de línea de comandos de que tiene interfaz gráfica.

En el caso particular de la Kinect, no se puede grabar la nube de puntos en el formato de trabajo porque es un formato descomprimido, y la alta cantidad de datos recibidos saturaría la red y el disco. En cambio se guarda su versión en bruto comprimida y luego se procesa usando el nodo que la descomprime. Para ello es necesario iniciar openni con un id de dispositivo inválido, de forma de que tome la información publicada por rosbag en vez de esperar por una Kinect. openni publica en los mismos tópicos que la Kinect haciendo transparente, para el resto del sistema la existencia de una Kinect real o datos grabados.

**Gazebo**

Es un simulador de robots que se integra perfectamente con ROS, aunque funciona independiente de este también. Permite simular entornos con objetos genéricos o diseñados, para ver el comportamiento de algoritmos antes de probarlos en robots reales. Existen paquetes de integración con ROS para el robot PR2 que simulan los controladores del robot siguiendo las interfaces de ROS.

**Actionlib**

Es un paquete de ROS que presenta una forma estándar de comunicarse con tareas de largo plazo que sean cancelables. Trabaja definiendo un cliente y un servidor que se comunican en función de metas a cumplir solicitadas por el cliente, y funciones de callback y resultado entregado por el servidor. El servidor mantiene una máquina de estados para cada acción solicitada y el cliente intenta copiar el estado del servidor para informar al usuario el estado de su solicitud. Internamente estas entidades realizan la comunicación estándar de ROS de tópicos. La especificación de su comunicación se hace a través de una estructura particular de mensajes de ROS llamadas Acciones, donde se define la estructura de datos de la meta, los callbacks y el resultado.

**Sistemas de coordenadas: tf**

La librería de transformaciones entre sistemas de coordenadas es llamada tf. Esta librería almacena un árbol de transformaciones geométricas entre los sistemas de coordenadas de cada frame del robot. Este árbol permite transformar entre los sistemas de coordenadas de cualquier frame y solicitar transformaciones en el pasado o futuro (extrapolación) de configurarse.

Para el PR2 el árbol parte con el frame /odom\_combined que representa el mundo y luego base\_footprint marca el centro del robot entre las ruedas (a la altura del piso).

Para obtener transformaciones, éstas deben buscarse en el árbol en el tiempo específico en que fue solicitada, o la más cercana de estar disponible, por lo que hay un costo asociado

Importante. Deben evitarse llamadas masivas a tf, como por ejemplo pedir transformaciones para todos los puntos de una nube por separado, en vez de transformar la nube completa con su transformación asociada. Los sistemas de coordenadas usados por ROS son Z arriba, X adelante e Y a la izquierda8

, excepto para aquellos frames con el sufijo optical\_frame, los que corresponden a cámaras, puesto que el estándar en estos es Z en la dirección de la cámara, X abajo e Y izquierda. Representación de transformaciones Se usan un tipo de dato tf: Vector3 para la traslación y tf: Quaternion para la rotación. Los Quaternion es son un sistema de vectores que extienden los números complejos y tienen la particularidad de que pueden representar rotaciones de forma compacta y eficiente. Esta representación difiere de la usada mayormente en PCL, donde se usa la librería eligen con las estructuras Eigen: Vector3f y Quaternionf.

**Point Cloud Library (PCL)**

PCL es una librería que facilita el trabajo con nubes de puntos, pues contiene varios métodos para manipularlas, y algoritmos del estado del arte para tareas comunes como segmentación, filtrado, visualización, reconocimiento, descriptores, etc. La librería tiene licencia BSD de 3 cláusulas, es código abierto y funciona independiente de otros frameworks o hardware.

**Move It**

Es una librería de manipulación móvil que hace uso de algoritmos del estado del arte en planificación de movimiento, manipulación, percepción 3D, cinemática, control y navegación. Su diseño modular y extensible permite integrar fácilmente nuevos algoritmos y expone una interfaz de fácil uso para el desarrollo de soluciones robóticas. Como es de código abierto se puede consultar, extender y desarrollar con facilidad. Esta librería funciona en conjunto con el framework ROS desde su versión Hydro con Ubuntu 12.04. Move It! Realiza la planificación de movimiento, cálculo de colisiones y ejecución final del grasping.15