Robot Operating System (ROS) como plataforma para extender las capacidades de Legos NXT

María Fernanda Utreras Abad[1], Diana Decimavilla Alarcon[2], Daniel Ochoa[3]
Facultad de Ingeniería Eléctrica y Computación (FIEC)
Escuela Superior Politécnica de Litoral (ESPOL)
Campus Gustavo Galindo, Km 30.5 Vía Perimetral
Apartado 09-01-5863 Guayaquil-Ecuador
mutreras@espol.edu.ec[1], ddecimav@espol.edu.ec[2], dochoa@fiec.espol.edu.ec[3]

Resumen

El presente trabajo muestra el estudio de Robot Operating System (ROS), explica que es ROS, para qué sirve, la arquitectura de ROS y por supuesto muestra a ROS como plataforma para extender las capacidades de Legos NXT. El objetivo de esta investigación nace de la necesidad de fomentar el uso de ROS ya que cabe recalcar que es de libre distribución y de código abierto. ROS es un framework de desarrollo de algoritmos de robótica, el cual permitió implementar la aplicación "LASER SCANNER". El objetivo principal de esta aplicación es implementar ROS en varios lenguajes de programación, como en este caso se utilizo Python y C++. Para esta aplicación se uso Legos NXT, una cámara web USB y un laser de línea. Lo que hace la aplicación es escanear un objeto el cual se encuentra en cierta posición para que un láser de línea lo ilumine de arriba hacia abajo y así poder tomar una foto del mismo, cada cierta posición tomara una foto, la cual se guardara en una ruta especificada y así poder obtener la forma del objeto.

Palabras claves: *ROS, Lego Nxt, Nodo, Python.*

Abstract

This paper presents the study of Robot Operating System (ROS). It explain what is ROS, what it is for, the architecture and of course shows ROS as a platform to extend the capabilities of Legos NXT. The objective of this research stems from the need to promote the use of ROS, as it should be stressed that is freely available and open source. ROS is a development framework for robotics algorithms, which allowed the implementation for the aplication "LASER SCANNER". The main objective of this application is to implement ROS in various programming languages like it was used for Python and C + +. For this application we use Legos NXT, a USB webcam and a line laser. What the application does is scan an object which is in a certain position for a line laser illuminates it from top to bottom in order to take a picture of it. It will take a picture of the object in certain position, which will be stored in a route specified and thereby obtain the object's shape.

Keys words: *ROS, Lego Nxt, Nodo, Python.*

1. Introducción

Los robots son complicados en lo que a software se refiere, debido a que en la práctica este software puede ser extenso y complejo.

ROS es un framework de Robótica que ayuda a los desarrolladores a crear aplicaciones que actualmente se usan en centros de estudio y en la Industria.

ROS es similar a otros frameworks de robótica como: Carmen, Player, Yarp, Orocos, etc. Si se compara Player con ROS, se puede destacar de ROS que es un framework mucho más robusto y soporta plataformas robóticas más complejas, a diferencia de Player que es más enfocado a plataformas robóticas simples.

2. ¿Qué es ROS?

ROS es un framework de Desarrollo de Algoritmos de Control en Robótica.

Contiene paquetes que incluyen librerías de control de dispositivos para actuadores, sensores, motores

Es de código abierto, esta soportado por Unix-Ubuntu actualmente y tiene un soporte experimental para Mac, Fedora, Windows, OpenSuse entre otros.

3. Arquitectura de ROS

ROS esta divido en tres niveles: nivel grafico, nivel sistema de archivos y nivel comunitario.

3.1. Nivel Gráfico

El Nivel Gráfico permite ver de manera gráfica cómo trabajan los procesos de ROS al momento de procesar datos en conjunto, tal como las redes peer-to-peer. Los conceptos básicos de este nivel grafico son: Nodos, Servicios, Topics, Mensajes, Master y Servidor de Parámetros.

3.1.1. Nodo. Nodo es un programa ejecutable que realiza el computador que debe realizar varias tareas para cumplir con una función u objetivo. Cada nodo que se ejecuta tiene un nombre único que los identifica del resto del sistema. Un sistema de control puede tener varios nodos.

El uso de nodos de ROS proporciona varios beneficios para el sistema. No hay tolerancia a errores, los nodos son individuales por lo tanto los accidentes o fallos que ocurran son aislados. El mecanismo de comunicaciones de los nodos se basa en que estos pueden publicar o suscribirse a ‘”Topics”, también pueden usar “Servicios”, enviar parámetros conceptos que explicaremos más adelante.

3.1.2. Servicios (SRV). Los servicios se definen mediante archivos SRV, que se compilan en el código fuente de una biblioteca de ROS. Los servicios están conformados por dos partes: un mensaje que es el que pregunta y otro mensaje que es el que responde. A diferencia de los topics, un nodo solo puede llamar a un servicio bajo un nombre en específico.

3.1.3. Topics. Mecanismo mediante el cual los nodos intercambian mensajes. El nodo envía un mensaje de publicación en un determinado “Topic”. Cuando un nodo está interesado en un tipo de dato se suscribirá al topic correspondiente.

Puede haber varios editores y suscriptores simultáneos para el mismo tópico y además un mismo nodo puede publicar o suscribirse en diferentes tópics. Nombre de Topic define el contenido del mensaje.

3.1.4. Mensajes (MSG). Los Nodos utilizan mensajes para comunicarse. Un mensaje se define como un archivo de texto en el que se describen los campos utilizados en el mensaje ROS.

Los mensajes son enviados a través de los topics que los enruta con el sistema Publicación/Suscripción. Los tipos de datos básicos que se usan son flotante, booleano, entero, etc y estructuras de C.

3.1.5. Máster. Es el Nodo principal de ROS, el cual proporciona un registro de nombres.

El máster es el encargado de hacer las conexiones correctas entre los nodos que se quieran comunicar.

3.2. Nivel Sistema de Archivos

Los conceptos de nivel de sistema de archivos son recursos ROS que se encuentran en el disco [1] tales como: paquetes y pilas.

3.2.1. Paquetes. Son la Unidad principal para organizar el software en ROS. Contiene: Nodos, Bibliotecas Independientes, Datos, Archivos de Configuración.

Los Paquetes son especificados por un archivo manifest.mxl, este archivo es similar al conocido archivo readme.txt de los programas comunes. En este archivo se especifica el autor, licencia, dependencias del lenguaje de programación, etc [2].

3.2.2. Pilas. Son la unidad de organización de los paquetes, el objetivo de las pilas es simplificar el proceso de compartir código. Las pilas son especificadas en un archivo stack.xml y cualquier paquete dentro de ella se considera parte de dicha pila.

Los archivos stack.xml suelen incluir etiquetas <depend> que declaran las pilas que se deben instalar como pre-requisitos. Además las pilas no pueden contener pilas [3].

3.3. Nivel Comunitario

Comprende los recursos disponibles a las comunidades que incluyen:

3.3.1. Distribución. Son colecciones de pilas con versiones que se pueden instalar. Es como una distribución de Linux que tratan de hacer fácil la instalación de programas informáticos y que logra mantener versiones consistentes de un software.

3.3.2. Repositorios. Gran cantidad de repositorios de código que forman parte de la red de donde las instituciones reutilizan código para luego lanzar su propio software.

3.3.3. Wiki de ROS. Aquí se concentra el más grande y principal foro donde se encuentra toda la información sobre ROS. Sólo creando una cuenta puedes contribuir con nueva documentación para ayudar a todos quienes están desarrollando alguna aplicación con ROS.

Además de facilitar correcciones y actualizaciones, entre otros beneficios.

4. Aplicación Láser Scanner

En la presente sección se explica el proyecto "Láser Scanner". Además se describe el programa utilizado con imágenes que resultaron de la realización y estudio del proyecto usando ROS.

4.1. Descripción General del Proyecto

Láser Scanner tiene como objetivo escanear un objeto el cual se encuentra en cierta posición para que un láser de línea lo ilumine de arriba hacia abajo para poder tomar una foto del mismo. Cuando el laser llega a su punto más bajo, volverá a subir para hacer girar al objeto y repetir la secuencia de arriba hacia abajo hasta que el objeto haya girado por completo y vuelva a su posición inicial. Las fotos se guardan dentro de una carpeta en la ruta especificada dentro del código.

4.2. Objetivos de la Aplicación:

- Investigar la plataforma robótica ROS para que pueda ser introducida en la materia de fundamentos de robótica que se dicta en la Facultad.
- Desarrollar el proyecto de una forma eficiente mediante la aplicación de diferentes lenguajes de programación según convenga.
- Implementar con Lego NXT.
- Fomentar la plataforma robótica ROS en centros de educación y área industrial.

4.3. Recursos Físicos

- 1 Brick NXT.
- 1 Cámara web USB.
- 1 Láser lineal.
- 2 Servo motores del Kit Lego Mindstorm NXT.
- Varias piezas del Kit Lego Mindstorm NXT, como muestra la Figura 2.

4.4. Descripción Técnica del Proyecto

Láser Scanner de acuerdo a la teoría de ROS consiste en 3 Nodos los cuales se describen a continuación:

4.4.1. Nodo Motores. Este nodo está encargado de controlar los motores que utilizados en este proyecto. Usa Lenguaje de Programación Python y tiene la característica de ser un Nodo Publicador. Para este proyecto el nodo publicador es llamado "NodoMotores" el cual da una orden a otro nodo llamado "NodoCamara" para que tome una foto.

El NodoMotores está programado para que controlar dos motores que son:

- **Motor Base:** Motor Base está encargado de controlar el giro del objeto a escanear. Está programado para girar la base del objeto 90º cada vez que el Motor Láser haya concluido su ejecución de arriba hacia abajo. Dentro de la estructura del código se identifica al motor base bajo el nombre: m_base.
- **Motor Láser:** Motor Láser se encarga de controlar el láser. El cual está en una posición inicial y a partir de esta comienza a girar el motor para que el laser vaya iluminando de arriba hacia abajo el objeto. Para este proyecto se programó al motor del láser para que gire cada 15º. Dentro del código se puede identificar el motor del láser bajo el nombre: m_ laser.

Los Nodos que intervienen en la realización del proyecto tienen características bien definidas y funcionan de la siguiente manera:

El NodoMotores envía una publicación definida por un "Topic" al cual denominaremos "foto". Esto recibe el NodoCamara, el que ya tenemos programado para que solo se subscriba al Topic "foto" y ejecute el código escrito que toma una foto cada vez que reciba el "Topic". Logrando con esto una comunicación sincronizada entre ambos nodos. El proceso concluye cuando el Motor Base haya vuelto a su posición inicial. Ver figura 1.

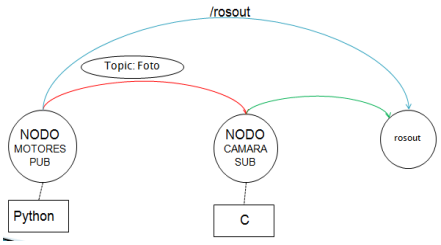


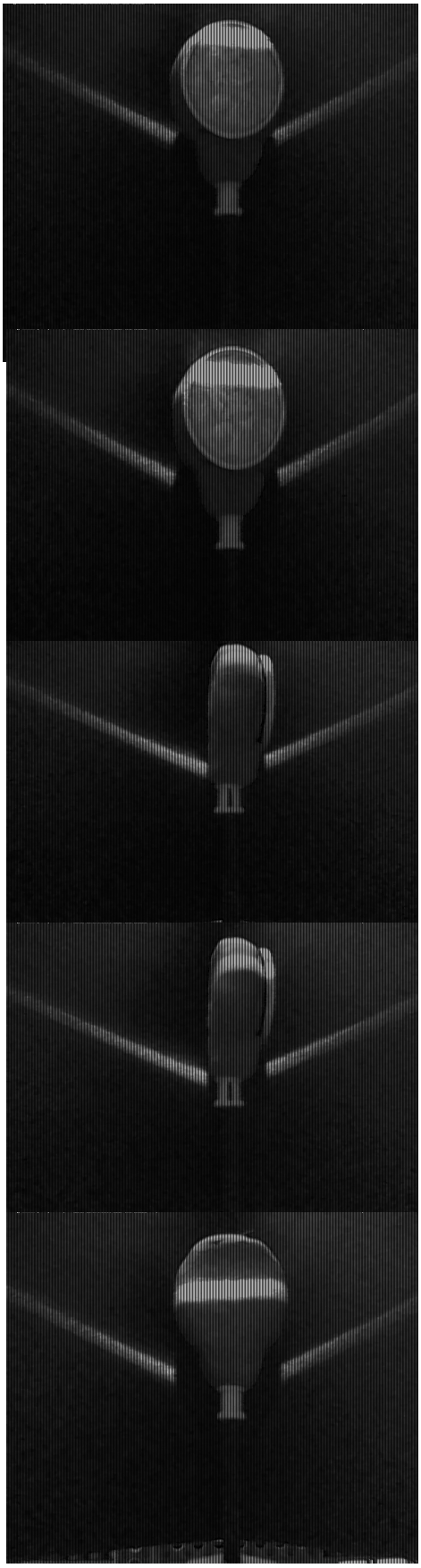
Figura 1. Descripción técnica Láser Scanner.

4.4.2. Nodo Cámara. Este nodo está encargado de controlar una cámara web. Usa C++ como Lenguaje de Programación y tiene como característica principal ser un Nodo Subscriptor, dentro del código denominado "NodoCamara". El cual ejecuta su código solo cuando recibe la orden del NodoMotores, para que tome una foto por cada vez que se recibe esta orden. Las imágenes se guardan en una carpeta dentro de nuestro espacio, la cual está definida y puede ser modificada dentro del código del "NodoCamara".

4.4.3. Nodo rosout. El Nodo rosout siempre se encuentra activo ya que este es el encargado de suscribir, registrar y publicar los mensajes. Este nodo se crea automáticamente al ejecutar el nodo principal roscore, que permite la comunicación entre los nodos de ROS, el cual se ejecuta al inicio de trabajar con ROS.

5. Resultados:

Figura 2. Arquitectura Láser Scanner.



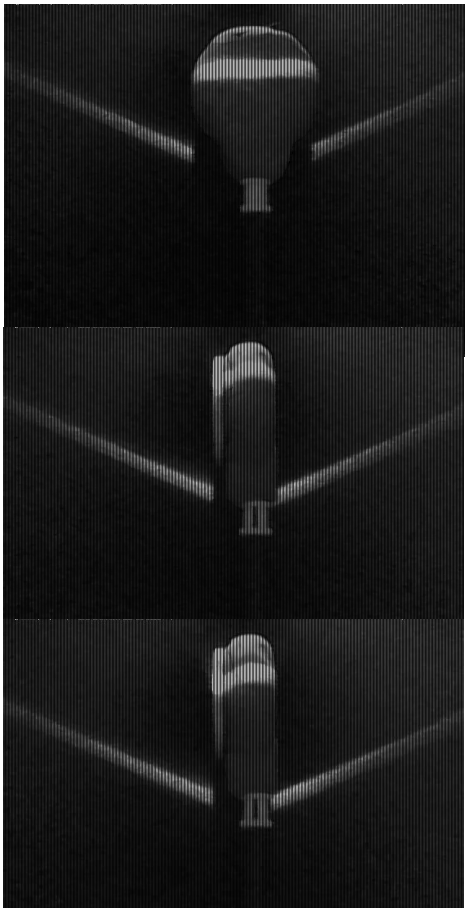


Figura 3. Imágenes resultantes de la Aplicación Láser Scanner.

6. Conclusiones

- ROS proporciona una gran cantidad de recursos, que tienen la característica de ser de código abierto, lo que permite reutilizarlos, extenderlos y tener resultados más eficientes sin tener que utilizar grandes y complejos códigos.

- Se verifico que en ROS es posible solucionar errores de una manera más sencilla, debido a que las aplicaciones pueden ser divididas en nodos, los cuales cumplen diferentes y fundamentales funciones para las aplicaciones.

- En el proyecto Laser Scanner, se comprobó que en ROS se pueden usar varios lenguajes de programación, dentro de una misma aplicación sin tener problemas de ejecución, ya que no se tuvo inconveniente al trabajar con Python y C++.

- Laser Scanner, es un proyecto ambicioso, que a pesar de ser creado con elementos de poca complejidad como NXT Lego, un laser lineal y una cámara web que no implicaron altos costos, puede ser aplicado en diversos y complicados campos como en Botánica, Robótica y demás fines académicos e investigativos debido a que su propósito aunque sencillo puede llegar a ser bien remunerado.

7. Referencias

[1] ROS.org, Conceptos, http://www.ros.org/wiki/ROS/Concepts, 17 de septiembre 2012.

[2] Massachusetts Institute of Technology, Nivel de Sistema de Archivos de ROS, http://courses.csail.mit.edu/6.141/spring2012/pub/labs/lab4/docs/Visual-Servo-Lab-Procedure.pdf, 08 de octubre 2012.

[3] ROS.org, Pilas, http://www.ros.org/wiki/Stack%20Manifest, 22 de octubre 2012.