PLATAFORMA DE ROBÓTICA MÓVIL PARA EXPERIMENTOS A TRAVÉS DEL SISTEMA OPERATIVO DE ROBOTS (ROS): DESARROLLO DE LA TARJETA DE TIEMPO REAL"

Edwin Lascano[1] , *Xavier Rosero*[2]

[1] Carrera de INGENIERÍA EN MECATRÓNICA, FICA, Universidad Técnica del Norte, Av. 17 de Julio, Ibarra, Ecuador
[2] Carrera, Facultad, Universidad, dirección, Ciudad, Provincia, País
Correo-e: edwinlb20@gmail.com

Resumen

El campo de la robótica comprende un sinnúmero de tecnologías y sistemas para su operación y control. La rama de la robótica que más ha evolucionado en los últimos años es la robótica móvil. Los robots móviles son máquinas capaces de trasladarse en cualquier ambiente sin fijarse a una sola ubicación física. Además, realizan movimientos complejos que son realizados en tiempo real, que ocurren con una planificación predecible y que deben ser realizados en un plazo de tiempo definido.
En la Universidad Técnica del Norte no se han desarrollado aún herramientas de robótica móvil que permitan visualizar, comprender y aplicar las teorías y fundamentos de las asignaturas de robótica y sistemas de control. En base a los aspectos antes mencionados nace la idea de realizar este proyecto mismo que se realizó con fines educativos para aportar al área de Ciencias Aplicadas de la Universidad. La conexión de sistemas integrados en tiempo real a ROS es una forma de aprovechar las capacidades de nivel superior de este meta-sistema operativo. Para la elaboración del presente proyecto se realizó la implementación del robot móvil a través de la plataforma robótica (ROS) que provee librerías, aplicaciones visuales de software y herramientas de comunicación. Éste fue ejecutado en un ordenador con aplicaciones de código abierto. Para la parte del sistema de tiempo real se empleó un microcontrolador y un sensor ultrasónico que están comunicados por nodos al ordenador.

Introducción

La robótica está experimentando un crecimiento explosivo propulsado por los avances en computación, sensores, electrónica y software[1]. Los robots están ya revolucionando los procedimientos que se emplean en la medicina, agricultura, minería y el transporte, al solventar la mayoría de sus necesidades de control y automatización. De esta manera, existen innumerables aplicaciones que día a día mejoran el diario vivir de la sociedad.
En la actualidad existen varios Framework de Desarrollo en Robótica tales como Player/Stage/Gazebo, Yarp, Orocos, OpenRave entre otros, todos ellos de código abierto, sin embargo, Sistema operativo de robots (ROS) ha logrado agrupar las mejores características de todos estos proyectos dando una solución integral y muy uniforme al problema de desarrollo de sistemas robóticos, se caracteriza por ser una plataforma multi-lenguaje (c++, python, java), peer2peer, orientado a herramientas, ligero y de código abierto (OpenSource)[2]. Un sistema construido utilizando (ROS) provee los servicios estándar de un sistema operativo tales como abstracción del hardware, control de dispositivos de bajo nivel, implementación de funcionalidad de uso común, paso de mensajes entre procesos y mantenimiento de paquetes[3]. Está basado en una arquitectura de punto a punto donde el procesamiento toma lugar en los nodos que pueden recibir, entregar y multiplexar mensajes de sensores, control, estados, planificaciones y actuadores, entre otros.
La realización de este proyecto se fundamenta en una necesidad la cual consiste, que en la actualidad la Universidad Técnica del Norte y particularmente a la Facultad de Ingeniería en Ciencias Aplicadas, no existen robots móviles que se utilicen como equipo para enseñanza e investigación, situación que repercute en las asignaturas relacionadas con robótica y control, convirtiéndolas netamente teóricas.

En base a la problemática descrita, creemos imperiosa la implementación de un robot móvil para experimentos de robótica que sea utilizada como equipo didáctico y de investigación en los laboratorios de la FICA.

Keywords

ROS (Plataforma de robótica móvil), tiempo real, sensor ultrasónico, distancia,

Metodología

Antes de la elaboración del proyecto se iniciara con una pequeña definición y con conceptos fundamentales para el entendimiento del tema como son: sistemas de tiempo real, ros y las herramientas que nos otorga cada uno de estos.

1. **Sistema de tiempo real**

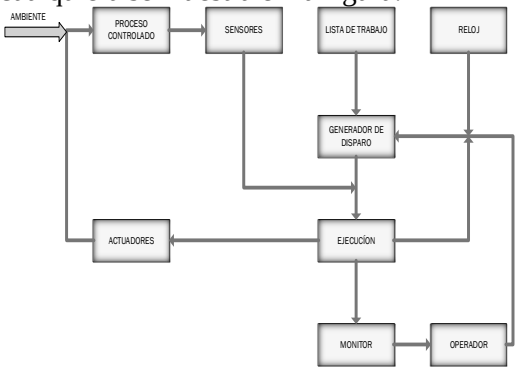
Un sistema es el conjunto de componentes que trabajan en conjunto para alcanzar un propósito común. Los sistemas de tiempo real son sistemas basados en un ordenador que debe resolver diferentes aspectos de forma simultánea, rápida respuesta, reacciona ante estímulos, fallo en los componentes o en sus conexiones y posibles necesidades de adaptarse a lo largo del tiempo (mientras está en funcionamiento) ante los cambios de requerimiento y circunstancias.

Un sistema en tiempo real debe cumplir con tres condiciones fundamentales:

- Interactúa con el mundo real.
- Emite respuestas correctas.
- Cumple restricciones temporales.

Estructura general de un sistema en tiempo real

La estructura general de un sistema en tiempo real que controla un proceso cualquiera se muestra en la figura.



Aplicaciones de sistemas en tiempo real

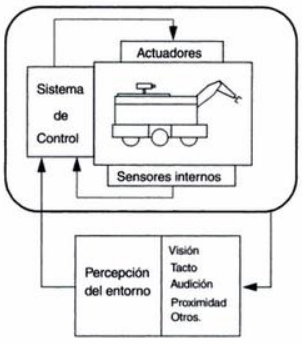
Son numerosas las aplicaciones donde se utilizan sistemas en tiempo real:

- Sistemas de defensa
- Sistemas de radar
- Control de procesos
- Aviación control de tráfico aéreo servidores multimedia
- Sistemas de comunicación
- Sistemas de satélites
- Sistemas de procesamientos de señales
- Sistemas de navegación autónoma
- Sistemas de control y adquisición de datos. Etc...

Un grupo importante de sistemas en tiempo real son los sistemas empotrados (Embedded systems). Son sistemas diseñados para una aplicación específica. Por ejemplo un teléfono móvil, el control del abs en un coche, etc. Sin embargo un ordenador personal no sería un sistema empotrado ya que no se ha concebido para que realice una aplicación concreta. Por tanto la afirmación de que todo sistema empotrado es un sistema en tiempo real es cierta, mientras que la afirmación contraria no lo es. [4]

Tiempo real en los robots

Un robot es una máquina que realiza trabajos productivos y de imitación de movimientos y comportamientos de seres vivos. En la actualidad los robots son obras de ingeniería que están compuestos por sistemas mecánicos, actuadores, sensores y sistemas de control. Este últimos es importante ya que es el encargado de obtener la información del exterior por medio de los sensores, interpretar los datos y enviar la información a los actuadores.[1] En la Figura se detalla al robot sus elementos y la interacción con el entorno.



En la industria, los robots cada vez son más indispensables en los procesos de

producción y con una población en crecimiento la demanda de productos exige que estos sean más rápidos en la recolección, interpretación y envió de datos para que sean más eficientes, además, puedan realizar acciones en respuesta a eventualidades que se pueden suscitar en su entorno en lapsos de tiempo muy cortos por ellos es necesario la implementación de sistemas en tiempo real para los robot. [5]

Las ventajas de utilizar sistemas en tiempo real en los robots son: su mayor exactitud, seguridad y adaptabilidad a contingencias variadas de su entorno y en cada uno de los procesos que esté realizando ya que continuamente está adquiriendo información del medio externo.

2. Robot Operating System (ROS)

La plataforma (ROS) es un framework de desarrollo de algoritmos de control en robótica además cuenta con paquetes que incluyen librerías de control de dispositivos para actuadores, sensores, motores. Es de código abierto, esta licencia permite libertad para uso comercial y para la investigación, también esta soportado por Unix-Ubuntu actualmente y tiene un soporte experimental para Mac, Fedora, Windows, OpenSuse entre otros.

La principal ventaja de este tipo de comunicación es la creación de grandes bases de datos de manera gratuita ya que todos los ordenadores conectados en línea pueden descargar archivos de otros ordenadores también conectados.[6]

(ROS) cuenta con dos partes principales para su buen funcionamiento: el núcleo del sistema operativo y el ros-pkg. Este último un grupo de paquetes de licencia de código abierto, desarrollados por usuarios que implementan distintas funcionalidades como mapping, planning, etc.

Características Principales del Sistema Operativo de Robots (ROS)

Las principales características de ROS son:

Código Libre y Abierto.- (ROS) es de código libre bajo términos de licencia BSD, contiene libertad de usos tanto comercial como de investigación esto quiere decir que estas disponible al público en general. [6]

"Peer to Peer".- Es un sistema distribuido en el cual los diferentes proceso se comunican entre sí utilizando una topología "peer to peer". Con esta topología se utiliza un canal nuevo para la comunicación entre dos procesos distintos, evitando utilizar un servidor central para comunicar todos los procesos. [6]

Multi-Lenguaje.- Se puede trabajar en diferentes lenguajes de programación como lo son: C++, Phyton y Lisp. Ademas contiene bibliotecas en Java y Lua, en fase experimental.

Multi-Herramientas.- Contiene una gran cantidad de herramientas, que nos permite realizar diferentes tareas, desde navegar en los ficheros, también termite modificar los parámetros de configuración de un robot, proceso o driver, observar la topología de los procesos en ejecución y realizar la comunicación entre procesos.[6]

Conceptos Básicos del Sistema Operativo de Robots (ROS)

Para obtener un adecuado funcionamiento de (ROS) se debe tener conocimiento de los elementos fundamentales de esta plataforma como son: nodos, ROS Master, Mensajes y topics.

Nodos.- Son ejecutables que se comunican con otros procesos usando topics o servicios. El uso de nodos en (ROS) proporciona tolerancia a fallos y separa el código del sistema haciéndolo más simple. Un paquete puede contener varios nodos (cada nodo dispone de un nombre único), cada uno de ellos lleva a cabo una determinada acción.[2]

ROS Master.- Proporciona un registro de los nodos que se están ejecutando y permite la comunicación entre nodos. Sin el maestro los nodos no serían capaces de encontrar al resto de nodos, intercambiar mensajes o invocar servicios.[2]

Mensajes.- Los nodos se comunican entre sí mediante el paso de mensajes. Los tipos primitivos de mensajes ("integer", "floating point", "boolean", etc.) están soportados y se pueden crear tipos personalizados. [2]

Topics.- Son canales de comunicación para transmitir datos. Los topics pueden ser transmitidos sin una comunicación directa entre nodos, significa que la producción y el consumo de datos esta desacoplada. Los nodos pueden comunicarse entre sí mediante topics, pudiendo actuar como publicadores (publisher) y como suscriptores (subscriber).

Publicador.- El nodo que actúa como publicador es el encargado de crear el topic por el que va a difundir ciertos mensajes. Estos mensajes podrán ser visibles por los nodos que estén suscritos a este topic.[2]

Suscriptor. Este nodo se deberá suscribir a los topics correspondientes para poder acceder a los mensajes que hay publicados en ellos.
Un nodo puede publicar o suscribirse a un mensaje a través de un topic. En la Figura 6 se puede ver un ejemplo de comunicación entre dos nodos por medio de un topic. En este ejemplo, el nodo "publisher odometry" publica en el topic "Odom" un mensaje del tipo "nav_msgs/odometry" y el nodo "Base_movil" accede a este mensaje suscribiéndose a este topic. En la siguiente Figura se detalla la comunicación entre tópicos.

Tiempo real en ROS
A pesar de la importancia de la reactividad y la baja latencia en el control de robots, (ROS), en sí, no es un sistema operativo en tiempo real (RTOS), aunque es posible integrar (ROS) con código en tiempo real. Para ello debe cumplir con unos requisitos para su buen funcionamiento.
Los requisitos de un sistema en tiempo real varían dependiendo del caso de uso. En su esencia, el requisito en tiempo real de un sistema tiene dos componentes:

a) Estado latente
- Período de actualización (también conocida como fecha límite)
- Previsibilidad

b) Modo de fallo
- Cómo reaccionar ante un vencimiento del plazo [6]

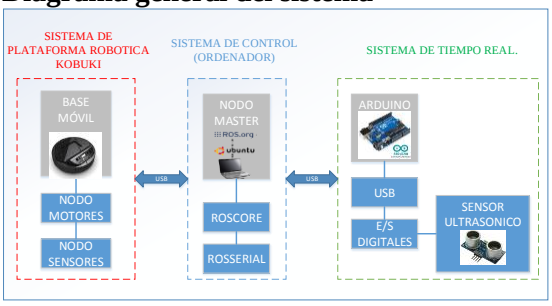
Cuando se hace referencia a los sistemas **duro / blando / firmes** en tiempo real generalmente se refiere al modo de fallo. Un sistema de tiempo real duro trata un plazo incumplido como un fallo del sistema. Un sistema de tiempo real blando trata de cumplir con los plazos, pero no falla cuando falta una fecha límite. Un sistema en tiempo real firme descarta cálculos realizados por el incumplimiento de plazos y puede degradar su requisito de rendimiento con el fin de adaptarse a un plazo incumplido.

Un modo de fallo de tiempo real se asocia a menudo con alta predictibilidad. Los sistemas de seguridad críticos a menudo requieren un sistema de tiempo real con alta predictibilidad.

Solución al problema planteado
La plataforma robótica Kobuki cuenta con muchos sensores como lo son: sensores de impacto son tres que le permite en el ínstate de llegar a tener contacto con un objeto la plataforma robótica se detiene inmediatamente, sensores de desnivel son tres estos sensores están situados debajo del parachoques que son infrarrojos que sirven para detectar si puede seguir por la dirección prevista caso contrario tomará otra ruta, sensor de caída de rueda son dos son conmutadores de dos posiciones con muelle de retorno a su poción inicial y son accionados por medio de una palanca. Se ubican en la parte interna de las ruedas.
Sin embargo ninguno de estos sensores envía una señal de los objetos que lo rodea sin colisionar por esta razón se ha visto conveniente implementar un sensor que de un aviso temprano y así el operador tome decisiones de evasión de obstáculos evitando daños ya sea en la plataforma robótica como en su entorno sin la necesidad que el operador se encentre cerca de la plataforma robótica.
Se seleccionó un sensor ultrasónico ya que proporciona un método sencillo de medición de distancia. Este sensor es perfecto para cualquier número de aplicaciones que requieren que se realice mediciones entre objetos en movimiento o estacionarios.
Consta de transmisor ultrasónico, receptor y circuitos de control, cuando se dispara, envía una serie de pulsos de ultrasonidos de 40KHz y recibe eco de un objeto. La distancia entre la unidad y el objeto se calcula mediante la medición del tiempo de desplazamiento del sonido y su salida como la anchura de un pulso TTL. [7] En la siguiente Figura se muestra un sensor ultrasónico.

Este sensor se puede acoplar a diferentes aplicaciones como por ejemplo: sistemas de seguridad, exposiciones animadas interactivas, sistemas de asistencia de parqueo y navegación robótica.

Diagrama general del sistema



Sistema de Plataforma Robótica (Kobuki)

IClebo Kobuki es una base de investigación móvil de bajo costo diseñada para la educación y la investigación sobre el estado de la robótica de arte. Con la operación continua en mente, Kobuki proporciona fuentes de alimentación para un ordenador externo, así como sensores y actuadores adicionales. Su odometría altamente precisa, enmendada por nuestro giroscopio calibrado en fábrica, permite una navegación precisa.

El Kobuki es una base móvil. Tiene sensores, motores y fuentes de energía, sin embargo por sí mismo, no puede hacer nada. Para ser funcional, se requiere construir una plataforma encima de la hoja de comandos. En el lado del hardware, esto suele implicar la adición de un notebook o una placa incorporada para ser el núcleo computacional para su sistema y por lo general algunos sensores extra para que sea realmente funcional. En el lado del software, esto implica construir cualquier software propio o integrar software de otros grupos con sus propias fuentes de desarrollo.

Sistema de Control (Ordenador)

Este sistema está compuesto del software como de hardware. El software comprende de un ordenador de marca Acer que cumple con los requerimientos necesarios para el funcionamiento óptimo del robot.

El hardware que se empleara para realizar el proyecto, es el sistema operativo Ubuntu 14.04 LTE. Para el buen funcionamiento y desarrollo del proyecto la elección del sistema operativo es fundamental. Teniendo en conocimiento que (ROS) también puede ser instalado en Windows, las aplicaciones desarrolladas (Paquetes) deben cumplir ciertas limitaciones computacionales. Además, cabe mencionar (ROS) solo es totalmente funcional en la plataforma Linux fundamentalmente para la distribución Ubuntu; con otras distribuciones no garantizan el correcto funcionamiento de (ROS).

Para concluir, se ha escogido Linux, concretamente la distribución Ubuntu 14.04 LTS, cuyo origen se basa en Debian, además se eligió esta versión por ser LTS sus siglas están en inglés (Long Term Support), esto significa que es una versión de Ubuntu que tendrá soporte y será actualizada más tiempo que una versión normal.

Del mismo modo, las versiones LTS suelen ser versiones más estables y probadas. Además, las versiones LTS de Ubuntu tienen soporte durante 5 años y las versiones normales tienen un soporte de 9 meses, luego de ese tiempo tendrá que actualizarse en un periodo relativamente corto de tiempo.

En el interior del sistema operativo robótico (ROS) se está ejecutando dos nodos sumamente importantes para la comunicación entre el ordenador y el sistema en tiempo real estos son: roscore y rosserial.[8]

Roscore

Es un conjunto de nodos y programas que es condición previa para la utilización de un sistema basado en (ROS). Se debe ejecutar un roscore con la finalidad de que (ROS) se pueda comunicar con todos los nodos. Se lo inicia con el comando roscore.

Rosserial

ROS Serial es una versión punto a punto de las comunicaciones (ROS) sobre serie, principalmente para la integración de microcontroladores de bajo coste (Arduino) en (ROS). ROS serie consta de bibliotecas para uso con Arduino, y nodos para el lado PC / Tablet (actualmente en Python y Java). [9]

Sistema de Tiempo Real

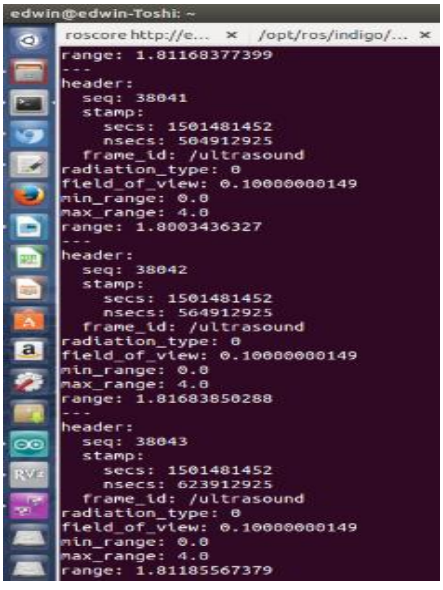
Para la parte del sistema en tiempo real se empleó el microcontrolador Arduino y el sensor ultrasónico que se conectan por medio de cables, y los dos elementos se comunica al ordenador por medio de UBS.

Frecuencia de muestreo del sensor ultrasónico es de 1 muestras por segundo.

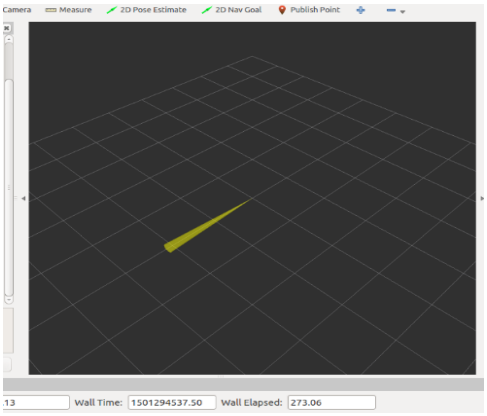
Pruebas, Resultados y Análisis

Para las pruebas se realizó los siguientes pasos:

a. Se carga el programa que se realizó en el software Arduino a la placa Arduino uno.
b. Se abre un nuevo terminal en Ubuntu y se ejecuta roscore.
c. Se ejecuta el código en un nuevo terminal
   roslaunch kobuki_node minimal.launch --screen para encender el robot kobuki.
d. Se ingresa el código en un nuevo terminal
   roslaunch kobuki_node minimal.launch –screen el que permite tele-operar al robot.
e. Utilizando el comando ls /dev/tty y pulsando dos veces la tecla tab nos imprime todos los puertos que se están utilizando. Se puede verificar si está conectado el Arduino con ordenador.
f. A continuación se ejecuta el comando:
   rosrun rosserial_python serial_node.py / dev / ttyUSB0
   Este comando ejecuta la aplicación cliente rosserial que reenvía sus mensajes desde el Arduino al resto de (ROS).
g. A continuación se ejecuta el comando:
   rostopic list
   Este comando nos permite imprime una lista de los temas a los que se puede suscribir.
h. Ejecutando el siguiente codigo
   rostopic echo /ultrasound
   Imprime la informacion del tema ultrasonico que para el caso es el sensor Ultrasonico hacia el ordenador por el puerto serial, que se visualiza en el terminal.



i. Para visualizar de una manera gráfica se lo puede hacer ejecutando el siguiente cogido.
   rosrun rviz rviz
   Lo que abre el simulador y mediante un haz de color amarillo representa las distancias que en ese momento está enviando el sensor.



Análisis y Resultados

Con esta prueba se pudo verificar, que el sensor está enviando datos hacia el microcontrolador, y que éste a su vez los está enviando al ordenador.

Conclusiones y Discusión

La plataforma robótica móvil Kobuki tiene la ventaja de ser un robot omnidireccional que permite realizar movimientos de desplazamiento y de rotación facilitando la implementación de algoritmos para la teleoperación.

La configuración del sistema se basó en tres segmentos fundamentales que son: sistema de plataforma robótica Kobuki, Sistema de control, Sistema de tiempo real. Los cuales trabajando a la par se pudo resolver el problema planteado.

La implementación del software con el hardware se la realizó rápidamente gracias a la facilidades que otorgo la plataforma robótica Kobuki.

Bibliografía.

[1] A. O. Baturone, Robótica: manipuladores y robots móviles. Marcombo, 2005.

[2] A. Koubaa, Robot Operating System (ROS): The Complete Reference. Springer, 2017.

[3] «ROS.org | Acerca de ROS». .

[4] Manuel Ortiz, «SISTEMAS INFORMÁTICOS EN TIEMPO REAL».

[5] A. Araújo, D. Portugal, M. S. Couceiro, J. Sales, y R. P. Rocha, «Desarrollo de un robot móvil compacto integrado en el middleware ROS», Rev. Iberoam. Automática E Informática Ind. RIAI, vol. 11, n.º 3, pp. 315-326, jul. 2014.
[6] «ROS/Introduction - ROS Wiki». [En línea]. Disponible en: http://wiki.ros.org/ROS/Introduction. [Accedido: 16-sep-2017].
[7] «Sensor de Distancia de Ultrasonido HC-SR04», Electronilab. .
[8] «The leading operating system for PCs, IoT devices, servers and the cloud | Ubuntu». [En línea]. Disponible en: https://www.ubuntu.com/. [Accedido: 16-sep-2017].
[9] «rosserial». [En línea]. Disponible en: http://library.isr.ist.utl.pt/docs/roswiki/rosserial.html. [Accedido: 13-sep-2017].