REVISTA ELECTRÓNICA

Programación Matemática y Software

ISSN: 1947-2193

VOL. 6 NÚM. 1 JUN

AÑO: 2014

FAS: 2014

PÁGINA LEGAL

PROGRAMACIÓN MATEMÁTICA Y SOFTWARE, Volumen 10, Número 3, Octubre de 2018–Enero de 2019, es una publicación cuatrimestral, digital en línea, editada por la Universidad Autónoma del Estado de Morelos, a través de la Dirección General de Publicaciones de Investigación, Mezzanine de la Torre de Rectoría, Campus Norte, Av. Universidad 1001 Col. Chamilpa, C.P. 62209, Morelos, México, Teléfonos: (01-777) 329-7909, http://www.progmat.uaem.mx: progmat@uaem.mx. Editor responsable: Marco Antonio Cruz Chávez. Reservas de Derechos al uso Exclusivo No. 04-2014-070114141100-203, ISSN: 2007-3283. Ambos otorgados por el Instituto Nacional del Derecho de Autor. Responsable de la última actualización de este número, Dra. Beatriz Martínez Bahena, Centro de Investigación en Ingeniería y Ciencias Aplicadas, Av. Universidad 1001 Col. Chamilpa, C.P. 62209, Morelos, México. Fecha de última modificación: 12 de noviembre de 2018.

Programación Matemática y Software (2018) 10(3): 21-30. ISSN: 2007-3283

Plataformas de Software para el Desarrollo de Sistemas Robóticos

Software Frameworks to Develop Robotics Systems

Marco Antonio Aguilar Tadeo[1], José Martín Flores Albino, Víctor Manuel Landassuri Moreno, Saúl Lazcano Salas

Universidad Autónoma del Estado de México, Centro Universitario UAEM Valle de México
Km. 11.5 C.P. 54500 Carretera Atizapán de Zaragoza-Nicolás Romero S/N.
Boulevard Universitario S/N Predio San Javier Atizapán de Zaragoza, Estado de México.

maguilart045@alumno.uaemex.mx, {jmfloresa, vmlandassurim, slazcano}@uaemex.mx

PALABRAS CLAVE:

ROS, Open Source, Cloud Robotics, middleware.

RESUMEN

Los continuos avances tecnológicos ocurren a velocidades vertiginosas, impactando de una manera muy importante nuestro entorno inmediato. En este sentido, la robótica ha experimentado cambios significativos, los cuales han sido motivados en gran medida por el surgimiento de entornos y plataformas de desarrollo de software que facilitan la creación de sistemas robóticos. Estas plataformas poseen diferentes características y orientaciones de servicios, coincidiendo todas ellas en la necesidad de estandarizar el control de los robots. El presente trabajo analiza algunas de las plataformas de software para el desarrollo de sistemas robóticos más usadas y disponibles en Internet, como ROS y ORCA, entre otras. Así, la contribución del presente trabajo es mostrar las tendencias actuales de las plataformas de software para el diseño de robots.

KEYWORDS:

ROS, Open Source, Cloud Robotics, middleware.

ABSTRACT

Nowadays technological advances happen fast, impacting our immediate environment in a significant way. In this sense, robotics has undergone important changes motivated mainly by the emergence of development environments and software platforms that facilitate the creation of robotic systems. These platforms have different characteristics and services, all of them coinciding in need to standardize robots' control. The present work analyzes some of the software platforms for robotic system's development most used and available on the Internet, such as ROS and ORCA among others. Thus, the contribution of the present work is to show the current trends of software platforms for robot design.

Recibido: 4 de agosto del 2017 • Aceptado: 3 de marzo del 2018 • Publicado en línea: 28 de agosto del 2018

1. INTRODUCCIÓN

Robótica es un campo de investigación que se encuentra en crecimiento, de tal manera que la tecnología de los robots ha evolucionado y se ha vuelto más accesible. Hoy es posible comprar robots comerciales que tienen funciones específicas, por ejemplo: limpieza y entretenimiento. Así mismo se pueden adquirir robots a los cuales es posible agregarles características de hardware y software para adaptarse a nuevas tareas. Lo anterior era difícil hace unos años, sin embargo, con la aparición de tecnologías como son los sistemas de cómputo embebido, la miniaturización de sensores y actuadores (MEM's Micro-Electro-Mechanical Systems) y los sistemas de comunicación, hoy es posible ajustar y personalizar los robots.

Actualmente se desarrollan diferentes sistemas para control de robots, los cuales utilizan la filosofía del "Desarrollo de Software Basado en Componente" (CBSD por sus siglas en inglés). En CBSD se favorece la reutilización de código para facilitar la creación de nuevos programas [1]. Dichos sistemas van desde desarrollos sencillos y básicos, hasta los complejos. En muchas ocasiones al iniciar un proyecto de robótica se puede comenzar sin apoyarse en trabajos previos, no obstante, hay que considerar que actualmente existe trabajo generado por otras investigaciones. Este es el caso de ROS (Robot Operating System), el cual es una plataforma de desarrollo de software que provee una serie de ejemplos y librerías que simplifican la creación de aplicaciones para robots con características de hardware diferentes [2]. De esta forma se pueden reutilizar códigos que están probados y que se encuentran disponibles para su consulta y descarga, alcanzando más rápidamente los objetivos del trabajo, ganando tiempo para enfocarse en mejorar las aplicaciones de robótica.

Una de las ventajas que ofrecen las plataformas de software para el desarrollo de sistemas robóticos, es la de poder almacenar y compartir las investigaciones (proyectos de robótica) llevados a cabo por instituciones públicas y privadas, así como por investigadores, estudiantes y aficionados. Esto permite que muchos trabajos puedan ser retomados por alguien más y no se pierdan los avances generados, sin importar si el propósito del proyecto es de investigación, comercial o lúdico. En las plataformas de uso libre, incluso es posible contribuir a mejorar el proyecto mismo. Por ejemplo, si se encuentra un error en el código se hace una copia o clonación del sistema (fork en inglés), y una vez mejorado se envía a los administradores para su evaluación (pull request en inglés), en caso de hacer una mejora importante y significativa al sistema, se agregan los cambios generados a la versión más reciente de la plataforma de software.

El objetivo que tiene este artículo es presentar los sistemas o plataformas de software que actualmente se desarrollan para compartir y generar herramientas para el diseño de robots.

El resto de este trabajo está organizado de la siguiente manera: en la sección II se describen algunas plataformas de software para el desarrollo de sistemas robóticos. En la sección III se resumen mediante una tabla los proyectos, destacando las principales características de cada uno de ellos. En la sección IV se ejemplifican proyectos de actualidad que se desarrollan mediante algunas de las plataformas mencionadas en este trabajo, por último, la sección V da las conclusiones.

2. PLATAFORMAS DE SOFTWARE PARA EL DESARROLLO DE SISTEMAS ROBÓTICOS

La construcción de robots requiere de una gran inversión económica y de tiempo, razón por la cual no han alcanzado una mayor popularidad. No obstante, existe una gran demanda comercial de ellos donde el costo de materiales y componentes son un elemento de inversión importante. Además, los algoritmos para darles capacidades de operación y de respuesta desde bajo nivel (manejo de actuadores y sensores) hasta alto nivel (inteligencia: toma de decisiones) son obra de cada diseñador, lo que dificulta aún más su implementación. Por tal motivo, no contar con un estándar de desarrollo de software para robots es parte de este problema. Afortunadamente, han surgido distintos proyectos de plataformas de software para el desarrollo de sistemas robóticos, los cuales buscan convertirse en un estándar para el diseño de robots, reduciendo la inversión de tiempo, dinero e investigación.

Distintos proyectos alrededor del mundo se están desarrollando, todos ellos con diferentes características como son lenguajes de programación, propósitos (reconocimiento de voz, robots móviles, agarre de objetos, mapeo, etc.) o compatibilidad (hardware, sistemas operativos, aplicaciones), a continuación, se describen varios de ellos.

ARTOO. Es un micro-framework para robótica. Proporciona un Lenguaje de Dominio Específico (DSL por sus siglas en inglés) simple pero potente para robots.

ARTOO está basado en SINATRA [3], el cual es un DSL para la rápida creación de aplicaciones web con un mínimo esfuerzo, mediante el lenguaje de programación Ruby e incluso toma código prestado de SINATRA. Artoo se encuentra disponible en inglés, fue lanzado en "Los Ángeles Ruby Conference 2013", es una infraestructura de software para robots de código abierto que trabaja con el lenguaje de programación RUBY. Trabaja con los sistemas operativos Linux, Windows y Mac OS X. Ofrece la posibilidad de conectar varios dispositivos de hardware de manera sencilla (por ejemplo: ARDrone, Raspberry Pi, Joystick, Teclado, Arduino, Motores, Servomotores, Ledes, Sensores análogos, entre otros) [4].

Carmen Robot Navigation Toolkit. Es un proyecto perteneciente a Christopher Fedor y Reid Simmons de la Universidad Carneige Mellon, comenzó en 1991 y trabaja bajo el sistema operativo Linux, está escrito en lenguaje de programación C, pero ofrece compatibilidad con lenguaje JAVA. Es una colección de software para control de robots móviles de código abierto. Disponible en inglés, Carmen está diseñado para proveer servicios de navegación como: control de base y sensores, registro, esquivar obstáculos, localización, planear rutas y mapeo. El proyecto ha sido patrocinado por el programa MARS (Mobile Autonomous Robot Software) de DARPA (Defense Advance Research Project Agency) [5].

EEROS. (Easy, Elegant, Reliable, Open and Safe) Es una infraestructura de código abierto para desarrollo de software para robots, opera bajo el sistema operativo Linux y con el lenguaje de programación C++. Se desarrolla por NTB Universidad de Tecnología en Suiza desde el año 2012. EEROS consta de tres sistemas conectados, el Sistema de Control, el Secuenciador y el Sistema de Seguridad, los cuales trabajan juntos para desarrollar rápidamente el nuevo software de robótica y minimizar errores potencialmente peligrosos. EEROS se encuentra aún en desarrollo, sin embargo, ha tenido buenos resultados en sus versiones prototipo, está disponible en inglés [6].

Microsoft Robotics Developer Studio. (MRDS). Es un Proyecto de Microsoft que inició en 2006, ofrece un entorno de simulación 3D basado en el motor de simulación física AGEIA PhysX, un lenguaje de programación visual y soporte en tiempo de ejecución. Opera bajo el sistema operativo Windows 7 y su uso es libre y de código abierto. Está desarrollado sobre el entorno .NET y soporta lenguajes de programación como VB.NET, Python, Visual Basic, VPL o C#. Disponible en inglés [7] [8].

miniBloq. Es un proyecto perteneciente a Julian U. Da Silva Gillic, desarrollado en el Instituto Wyss de la Universidad de Harvard desde 1993, está compilado con C++. Es un entorno de programación gráfica de código abierto que tiene como principal objetivo la robótica educativa a través de facilitar la programación y su aprendizaje en personas con pocos conocimientos en informática, además de contar con su robot llamado "Root". Opera bajo el sistema operativo Windows y Linux, este último con algunas limitaciones, cuenta con una interfaz de usuario avanzada donde se puede programar con los lenguajes de programación Python, JavaScript y Swift [9]. Disponible en inglés y con tutoriales en español.

MOOS. Por sus siglas en inglés Mission Oriented Operating Suite, es una plataforma construida en C++ para investigación en robótica, se comenzó en 2001 y su creador es Paul Newman del Departamento de Ingeniería Oceanográfica del Instituto de Tecnología de Massachusetts. MOOS se define como un conjunto de capas que se comunican entre sí y que en conjunto crean aplicaciones robustas. La última versión es llamada MOOS v10, la cual opera en los sistemas operativos Linux y Mac OS X [10] [11]. Es compatible con los lenguajes de programación Matlab y Java, está disponible en idioma inglés.

MyRobotLab. Es un proyecto de código abierto para robótica y control de máquinas, funciona con Java y con los sistemas operativos Windows, Linux y Mac. Ofrece servicios para visión artificial, reconocimiento de voz, control de motores y servomotores y comunicación con microcontroladores. Disponible en inglés [12].

OpenCV. (Open Source Computer Vision Library). Es una librería de código abierto creada por Intel en el año 1999, está orientada a la visión artificial y aprendizaje de máquinas. Opera bajo los sistemas operativos Linux, Mac, Windows y Android. Soporta los lenguajes de programación Python, Java, C/C++ y MATLAB. Cuenta con tutoriales en inglés, español y japonés [13].

OPRoS. (Open Platform for Robotics Services) Es una plataforma de código abierto para desarrollo de aplicaciones para robots. OPRoS ofrece un IDE (por sus siglas en inglés Integrated Development Environment) para desarrollo de componentes, contenido e integración de monitoreo, depuración y simulación. Además, un servidor para repositorio de servicios y componentes de robot. El proyecto comenzó en diciembre de 2007 y se encuentra disponible en inglés y coreano, ofrece también licencias comerciales. Opera bajo el sistema operativo Windows y con los lenguajes de programación Java y C/C++, cuenta con un ambiente virtual para

simulación de robots [14].

Orca. Es una plataforma para desarrollo de sistemas robóticos basados en componentes, su uso es libre y opera de manera completa en los sistemas operativos Linux, Windows y el sistema operativo Mac aún se encuentra en fase experimental. En un principio Orca era parte de otra plataforma llamada OROCOS, sin embargo, nunca se integró al proyecto y en 2005 inició Orca con el objetivo de estimular la reutilización código para el continuo progreso en la robótica. El proyecto está construido con lenguaje de programación C++ y para compilar se pueden utilizar los lenguajes C ++, Java, Python, PHP, C #, Visual Basic, Ruby y Objective C, pero solo para escribir partes del código [15] [16]. Disponible en inglés.

Orocos. (Open Robot Control Software u Open Realtime Control Services) Es un proyecto de código abierto enfocado principalmente al control de robots y máquinas en tiempo real, fue una idea de Herman Bruyninckx en el año 2000 y patrocinado por la Unión Europea. Trabaja con el sistema operativo Linux y utiliza el lenguaje de programación C++, se encuentra disponible en inglés. Orocos provee a C++ cuatro librerías con las que trabaja RTT (Real-Time Toolkit), OCL (Orocos Components Library), KDL (Kinematics and Dynamics Library) y BFL (Bayesian Filtering Library) [17].

Player. Es desarrollado por un equipo internacional de investigadores en robótica, su propósito es crear software libre para hacer investigación en sistemas de robots y sensores. Fue creado por Brian Gerkey, Richard Vaughan y Andrew Howard en la Universidad del sur de California en el año 1999. Opera bajo los sistemas operativos Linux, Solaris, BSD y Mac OSX (Darwin). El proyecto Player es ampliamente utilizado e incluye los paquetes de software "Stage" y "Gazebo" los cuales son simuladores en 2D y 3D respectivamente. Tiene compatibilidad con una gran variedad de robots comerciales y hardware (sensores y actuadores). Es compatible con los lenguajes de programación C++, Tcl, Java, Python y cualquier lenguaje que admita sockets TCP. Disponible en inglés [18].

ROCK. (The Robot Construction Kit) Es una plataforma para desarrollo de sistemas robóticos de código abierto, basado en RTT de Orocos. Proporciona todas las herramientas necesarias para configuración y ejecución de sistemas robóticos. Inicialmente se desarrolló en el DFKI Centro de Innovación Robótica y su principal colaborador es la Universidad Católica Leuven. Trabaja con sistema operativo Linux (Ubuntu y Debian) y está disponible en inglés. Es compatible con los lenguajes de programación C++ y Ruby [19].

RT Middleware OpenRTM-aist. (Robot Technology) "es una plataforma de software para construir el sistema de robot combinando los módulos de software de los elementos funcionales del robot" [20]. Es un proyecto que comenzó en el año 2002, perteneciente al Instituto Nacional de Ciencia Industrial Avanzada y Tecnología de Japón (AIST), su uso es gratuito y se encuentra disponible en los idiomas inglés y japonés. En esta plataforma se busca separar mediante módulos los componentes del robot, tanto de software (como pueden ser los algoritmos de control) y de hardware (como pueden ser los sensores o un conjunto de ellos) organizándolos de manera jerárquica. Soporta lenguajes de programación como C++, Python y Java. Se encuentra disponible para los sistemas operativos Linux/Unix, Windows y Mac OS X.

ROS (Robot Operating System) es una infraestructura para el desarrollo de software para robots, provee una serie de ejemplos, librerías, herramientas y convenciones que simplifican la creación de aplicaciones para robots con características de hardware diferentes. La filosofía de ROS es hacer software que pueda ser reutilizado en otros robots, es decir, lograr que el código que se crea para un robot, pueda ser compartido y utilizado en otros robots para así ahorrar tiempo y esfuerzo en el desarrollo de aplicaciones. ROS está bajo la licencia open source. Se encuentra disponible en inglés, alemán, francés, italiano, japonés, coreano, portugués, chino simplificado y español. ROS surgió originalmente en el año 2007 bajo el nombre de "switchyard", fue desarrollado por el Laboratorio de Inteligencia Artificial de Stanford (SAIL). A partir del año 2008 el desarrollo del proyecto se lleva a cabo en el Instituto de Investigación de Robótica Willow Garage, en California, Estados Unidos [21]. ROS Opera con el sistema operativo Linux y soporta los lenguajes de programación C, C++, Python y Java, este último aún en fase experimental.

3. TABLA COMPARATIVA

Con base en la información previamente descrita para algunas plataformas de software para el desarrollo de sistemas robóticos, se presenta la siguiente tabla que contiene las principales características de cada una.

Tabla 1: Comparación entre distintas plataformas de software para el desarrollo de sistemas robóticos, destacando sus principales características.

Plataforma de desarrollo	*Tipo de licencia.*	*SO con que trabaja.*	*Lenguajes de programación soportados.*	*Idiomas soportados.*	*Servicios que ofrece.*
ARTOO.	*Open Source.*	*Linux, Windows y Mac OS X.*	*Ruby.*	*Inglés.*	*Proporciona un DSL sencillo pero potente para robots. Compatible con diferentes dispositivos de hardware.*
Carmen Robot Navigation Toolkit	*Open Source*	*Linux.*	*C y Java.*	*Inglés.*	*Brinda servicios de navegación para robots móviles.*
EEROS	*Open Source*	*Linux.*	*C++.*	*Inglés.*	*Proporciona un framework y una arquitectura viables para robots en educación e industria.*
Microsoft Robotics Developer Studio	*Open Source*	*Windows.*	*VB.NET, Python, Visual Basic, VPL o C#.*	*Inglés.*	*Entorno de simulación 3D, un lenguaje de programación visual y soporte en tiempo de ejecución.*
miniBloq	*Open Source*	*Windows y Linux (este último con algunas limitaciones)*	*Python, JavaScript y Swift.*	*Inglés y español.*	*Robótica educativa y un lenguaje de programación sencillo para niños y adultos, además de un robot llamado "Root".*
MOOS	*Open Source*	*Linux y Mac OS X.*	*C++, Matlab y Java.*	*Inglés.*	*Brinda servicios de navegación para robots móviles marinos.*
MyRobotLab	*Open Source*	*Linux, Windows y Mac OS.*	*Java.*	*Inglés.*	*Visión artificial, reconocimiento de voz, control de motores y comunicación con microcontroladores.*
OpenCV	*Open Source*	*Linux, Windows, Mac OS y Android.*	*Python, Java, C/C++ y MATLAB.*	*Inglés, español y japonés.*	*Visión artificial y aprendizaje de máquinas.*
OPRoS	*Open Source y comercial.*	*Windows.*	*Java y C/C++.*	*Inglés y coreano.*	*IDE, Servidor para repositorios y simulador 3D.*
Orca	*Open Source*	*Linux, Windows y el SO Mac OS aún en fase de prueba.*	*C++, Java, Python, PHP, C #, Visual Basic, Ruby y Objective C.*	*Inglés.*	*Repositorio para estimular la reutilización de software y compartir librerías de alto nivel.*
Orocos	*Open Source*	*Linux.*	*C++.*	*Inglés.*	*Control de robots y máquinas en tiempo real. Además cuatro librerías para C++.*

Player	*Open Source*	*Linux, Solaris, BSD y Mac OSX (Darwin).*	*C++, Tcl, Java, Python y cualquier lenguaje que admita sockets TCP.*	*Inglés.*	*Paquetes de software "Stage" y "Gazebo" los cuales son simuladores 2D y 3D respectivamente. Compatibilidad con robots y hardware comercial.*
ROCK	*Open Source*	*Linux.*	*C++ y Ruby.*	*Inglés.*	*Plataforma para desarrollo de sistemas robóticos con todas la herramientas necesarias para configuración y ejecución de los mismos.*
MT Middleware OpenRTM-aist	*Open Source*	*Linux/Unix, Windows y Mac OS X.*	*C++, Python y Java.*	*Inglés y japonés.*	*Plataforma de software para construir sistemas robóticos a través de módulos de hardware y software del robot. Organización jerárquica de los componentes del robot.*
ROS	*Open Source*	*Linux y Mac OS X. Windows con algunos errores.*	*C, C++, Python. Java aún en modo experimental.*	*Inglés, alemán, francés, italiano, japonés, coreano, portugués, chino simplificado y español.*	*Infraestructura para desarrollo de software para robots, repositorios, localización y mapeo simultaneo, simulación 2D y 3D, identificación de objetos, robots móviles, control, planificación de rutas, agarre de objetos, visión artificial y reconocimiento facial y de gestos. Gran compatibilidad con hardware.*

Dada la información anterior se puede resumir que todas las plataformas de software para desarrollo de sistemas robóticos mencionadas ofrecen open source, sin embargo, OPRoS cuenta con licencias comerciales. El sistema operativo que predomina es Linux, con excepción de OPRoS y MRDS los cuales trabajan con Windows, los sistemas operativos Mac y Android también figuran como una opción para algunas plataformas. La diversidad en lenguajes de programación es amplia, pero los más utilizados son C++, Python y Java. El idioma que aparece disponible para todas las plataformas es el inglés y en muchos casos es el único idioma soportado. Es importante analizar los objetivos que se persiguen al desarrollar un proyecto para poder hacer una correcta elección de la plataforma de desarrollo de software, además de evaluar los conocimientos del desarrollador en cuanto a lenguajes de programación, sistema operativo e idioma.

4. PROYECTOS ANUALES CON PLATAFORMAS DE SOFTWARE PARA EL DESARROLLO DE SISTEMAS ROBÓTICOS

Diversos proyectos y trabajos de investigación se están llevando a cabo alrededor del mundo con base en las plataformas de software para el desarrollo de sistemas robóticos descritos, por ejemplo, con ROS se están realizando trabajos de navegación con vehículos autónomos sumergibles (AUV del inglés Autonomous Underwater Vehicles) [22], con vehículos aéreos no tripulados (UAV del inglés Unmanned Aerial Vehicles) [23] [30], con vehículos móviles teledirigidos [24], para estimación de posición [25], con robots agrícolas [29] y para control de robots manipuladores [27] [28], incluso la NASA cuenta con un proyecto llamado robonaut [31] el cual se desarrolló en conjunto con General Motors, el proyecto se trata de un robot humanoide y ha estado a prueba en la Estación Espacial Internacional. Con

la plataforma Orocos se está investigando acerca del control de robots bípedos [26] y con control de robots manipuladores industriales [27]. Con la plataforma Player/Gazebo/Stage se está trabajando sobre diseño de algoritmos mediante su simulador Stage [32]. Con MRDS se desarrollan escenarios donde se evalúan algoritmos [33]. En el caso de Orca se está desarrollando un robot para hospital [34], además se está trabajando con evitación de obstáculos en ambientes dinámicos para evitar colisiones en robots móviles [35]. Esto demuestra la vigencia y la demanda que existe hacia las plataformas de software para el desarrollo de sistemas robóticos, los proyectos son muy variados, así como los objetivos que se persiguen.

Aunque aún no se define una plataforma como el estándar para el diseño de aplicaciones de robótica, ROS goza de una mayor aceptación sobre los demás sistemas gracias a la compatibilidad que tiene con robots comerciales y hardware, e incluso con otras plataformas, por ejemplo, los simuladores Gazebo y Stage, pertenecientes al proyecto Player.

5. CONCLUSIONES

Por lo tratado anteriormente, se concluye: 1.- Existen proyectos para el desarrollo de plataformas de software para la programación de robots en Internet. Esto permite que se puedan retomar investigaciones y trabajos sobre robótica y así dar continuidad a los proyectos. Las plataformas existentes para el desarrollo de sistemas robóticos se basan en la filosofía de aprovechar el conocimiento colectivo y ponerlo para su uso en la industria, investigación y para todo aquel interesado. 2.- Se promueve la estandarización de las plataformas de software para sistemas robóticos. Lo anterior es fundamental para lograr avances en la robótica que permitan facilitar el diseño y la investigación en el campo. 3.-Otro punto importante es que las plataformas de software sean de código abierto. Debido a que estimulan que se contribuya a la mejora continua, al incorporar los últimos avances en robótica y haciéndolo disponible a la comunidad.

Hay que destacar que el sistema ROS actualmente ha ganado popularidad como plataforma de software para el desarrollo de robots, gracias a la compatibilidad que tiene con hardware (robots comerciales, sensores, actuadores, microcontroladores, cámaras, etc.) y software (sistemas operativos Linux, Mac y Android), además de hacer uso de los lenguajes de programación como C, C++, Python y Java. Entre las ventajas de ROS esta que cuenta con librerías para resolver la geometría, navegación, mapeo, localización, estimación de posición y diagnóstico de robots, entre otras, lo que permite que se pueda utilizar para el desarrollo de robots de tipo terrestres, aéreos, submarinos y espaciales. Así como incorporarles capacidad reconocimiento de objetos, planeación de rutas, evasión de obstáculos, reconocimiento de voz, visión artificial, por mencionar algunas. ROS cuenta con amplia documentación en distintos idiomas y es además compatible con otras plataformas.

AGRADECIMIENTOS

Para la Comisión Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico brindado al estudiante de maestría Marco Antonio Aguilar Tadeo a través de la beca número 611432.

REFERENCIAS

1. Alonso, D., Pastor, J. Á., Sánchez, P., Álvarez, B., & Vicente-Chicote, C. (2012). Generación automática de software para sistemas de tiempo real: Un enfoque basado en componentes, modelos y frameworks. Revista Iberoamericana de Automática e Informática Industrial RIAI, 9(2), 170-181.

2. Sánchez, F. Á. B., & Guzmán, A. F. (2012). La robótica como un recurso para facilitar el aprendizaje y desarrollo de competencias generales. Teoría de la Educación. Educación y Cultura en la Sociedad de la Información, 13(2), 120-136.

3. SINATRA [en línea], junio 2017, disponible en: http://www.sinatrarb.com/

4. artoo Ruby on Robots [en línea], junio 2017, disponible en: http://artoo.io/

5. Carmen robot Navigation Toolkit [en línea], junio 2017, disponible en: http://carmen.sourceforge.net/

6. EEROS [en línea], junio 2017, disponible en http://eeros.org/wordpress/

7. Microsoft [en línea], junio 2017, disponible en: https://www.microsoft.com/en-us/download/details.aspx?id=29081

8. Tomás, F. A., Aponte, A. C., & Álvarez, C. M. (2013). Plataforma de simulación reconfigurable basada en Microsoft Robotics Developer Studio.

9. miniBloq [en línea], junio 2017, disponible en: http://blog.minibloq.org/

10. MOOS [en línea], junio 2017, disponible en: http://www.robots.ox.ac.uk/~mobile/MOOS/wiki/pmwiki.php/Main/HomePage

11. Newman, P. M. (2008). MOOS-mission orientated operating suite.

12. Myrobotlab [en línea], junio 2017, disponible en: http://myrobotlab.org/

13. OpenCV [en línea], junio 2017, disponible en: http://opencv.org/

14. OPROS [en línea], junio 2017, disponible en: http://ropros.org/display/opros/OPRoS+Wiki

15. Orca [en línea], junio 2017, disponible en: http://orca-robotics.sourceforge.net/index.html

16. Makarenko, A., Brooks, A., & Kaupp, T. (2006, October). Orca: Components for robotics. In International Conference on Intelligent Robots and Systems (IROS) (pp. 163-168).

17. The Orocos Project Samarter control in robotics & automation! [en línea], junio 2017, disponible en: http://www.orocos.org/

18. Player [en línea], junio 2017, disponible en: http://playerstage.sourceforge.net/index.php?src=index

19. ROCK the Robot Construction Kit [en línea], junio 2017, disponible en: http://rock-robotics.org/stable/index.html

20. RT Middleware OpenRTM-aist [en línea], junio 2017, disponible en: http://openrtm.org/openrtm/en

21. ROS.org [en línea], septiembre 2016, disponible en: http://www.ros.org/

22. Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., ... & Carreras, M. (2015, April). ROSPlan: Planning in the Robot Operating System. In ICAPS (pp. 333-341).

23. Abeywardena, D., Pounds, P., Hunt, D., & Dissanayake, G. (2015). Design and Development of ReCOPTER: An Open source ROS-based Multi-rotor Platform for Research. In Australasian Conference on Robotics and Automation. The Australian National University.

24. Cepeda Castellanos, E. F. (2015). Desarrollo de móvil teledirigido basado en ROS (Bachelor's thesis, Universidad Militar Nueva Granada).

25. Díaz García, M. L. (2016). Odometría visual aplicada a la localización de un robot con Kinect en interiores.

26. Peekema, A. T. (2015). Template-based control of the bipedal robot ATRIAS (Doctoral dissertation).

27. Estévez, E., García, A. S., García, J. G., & Ortega, J. G. (2017). Aproximación Basada en UML para el Diseño y Codificación Automática de Plataformas Robóticas Manipuladoras. Revista Iberoamericana de Automática e Informática Industrial RIAI, 14(1),

82-93.

28. Macanás Valera, J. (2015). Interacción entre webcam y brazo robot para el posicionamiento del efector final.

29. Rengifo, H. F. C., & Preciado, J. A. C. CONTROL AUTONOMO DE ROBOTS DE APLICACION AGRÍCOLA CON PLEXIL.

30. Rodríguez Martín, E. (2015). Sistema de posicionamiento para un drone.

31. Badger, J., Gooding, D., Ensley, K., Hambuchen, K., & Thackston, A. (2016). ROS in Space: A Case Study on Robonaut 2. In Robot Operating System (ROS) (pp. 343-373). Springer International Publishing.

32. Saavedra Alcoba, M., & Enríquez Paz, L. P. (2015). Diseño de un algoritmo de búsqueda informada mediante el simulador robótico stage. Fides et Ratio-Revista de Difusión cultural y científica de la Universidad La Salle en Bolivia, 10(10), 39-59.

33. Rosado, J. F. Sistema de Simulación de Laberintos 3D y Robots Legos para la asignatura de Inteligencia Artificial. CIIE, 516.

34. Mamun, K. A., Sharma, A., Islam, F. R., Hoque, A. S. M., & Szecsi, T. (2016). Patient Condition Monitoring Modular Hospital Robot. JSW, 11(8), 768-786.

35. Khan, S. A., Yasar Ayaz, M. J., Gillani, S. O., Naveed, M., Qureshi, A. H., & Iqbal, K. F. (2015). Collaborative Optimal Reciprocal Collision Avoidance for Mobile Robots. International Journal of Control and Automation, 8(8), 203-212.

SEMBLANZA

Marco Antonio Aguilar Tadeo recibió el título de Técnico en Mantenimiento en Equipo de Computo en el año 2006 por el Colegio de Estudios Científicos y Tecnológicos del Estado de México (CECyTEM), el grado de Ingeniero en Computación en el año 2013 por parte de la Universidad Autónoma del Estado de México (CU UAEM VM). Actualmente es estudiante de la Maestría en Ciencias de la Computación en el Centro Universitario Valle de México de la Universidad Autónoma del Estado de México, es además becario de CONACyT con el número de beca 611432. Sus áreas de interés son: Robótica e Inteligencia Artificial.

Dr. en C. José Martín Flores Albino. Actualmente profesor en la carrera de Ingeniería en Sistemas y Comunicaciones y de la Maestría en Ciencias de la Computación en el Centro Universitario Valle de México de la Universidad Autónoma del Estado de México. Es ingeniero en Comunicaciones y Electrónica por parte del Instituto Politécnico Nacional, Maestro y Doctor en Ciencias (CINVESTAV-IPN). Trabaja en áreas de investigación sobre los temas de Control Automático, Electrónica e Inteligencia Artificial.

Dr. en Ing. Saúl Lazcano Salas es egresado de la carrera de Ingeniería en Telecomunicaciones de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México (UNAM). Realizó estudios de especialización en gestión de las Telecomunicaciones en la Escuela de Organización Industrial, Madrid, España y finalmente, el Doctorado en Ingeniería Eléctrica, área de Telecomunicaciones. Actualmente, se desempeña como profesor de tiempo completo en el Centro Universitario UAEM Valle de México. Sus áreas de interés son: codificación de canal, procesamiento de señales e inteligencia artificial.

Víctor Manuel Landassuri-Moreno recibió el título de Ingeniero en Computación en el año 2003 por parte de la Universidad Autónoma del Estado de México (UAP–VM, UAEM), el grado de Maestro en Ciencias en el Centro de Investigación en Computación del Instituto Politécnico Nacional (CIC-IPN) en el 2006 y el grado de Doctor en Ciencias de la Computación en la Universidad de Birmingham en el Reino Unido en el 2012. Actualmente es profesor de tiempo completo en el Centro Universitario UAEM Valle de México y ha publicado más de 35 artículos científicos, y más de 50 ponencias en congresos nacionales e internacionales, en el área de cómputo evolutivo y redes neuronales artificiales. Sus áreas de interés son: Evolución de Redes Neuronales Artificiales, Algoritmos Evolutivos y Análisis, Predicción y Clasificación de series de tiempo.