XXXVIII Jornadas de Automática
https://doi.org/10.17979/spudc.9788497497749.0766

PLATAFORMA BASADA EN LA INTEGRACIÓN DE MATLAB® Y ROS PARA LA DOCENCIA DE ROBÓTICA DE SERVICIO

Carlos G. Juan, Álvaro García, Jose M. Vicente, Jose M. Sabater-Navarro
Grupo de Neuroingeniería Biomédica de la Universidad Miguel Hernández de Elche, Av. Universidad, s/n, 03202, Elche, España
j.sabater@umh.es

Resumen

En este artículo se presenta una propuesta docente basada en una plataforma compuesta por Matlab® y ROS desarrollada en la asignatura "Robótica de Servicio" en la titulación de Máster Universitario en Ingeniería Industrial de la Universidad Miguel Hernández de Elche. Se ha preparado un proceso práctico en el que se introduce al alumno en el desarrollo de algoritmos de navegación, así como la interacción con los sistemas sensoriales del robot y sus actuadores. La componente innovadora se encuentra en el uso de una herramienta previamente conocida por el alumno (Matlab®) integrada con una plataforma más compleja y realista (ROS) para poder ofrecer la formación práctica adecuada con los recursos habituales.

Palabras Clave: Robótica, docencia, navegación, simulación

1 INTRODUCCIÓN

El entorno de desarrollo Matlab® [3] es una herramienta optimizada para la computación numérica ampliamente utilizada en múltiples ámbitos de la ingeniería, matemáticas, física, y diversas ciencias experimentales, especialmente en entornos académicos e investigadores. Ofrece una elegante y conveniente combinación de facilidad de programación, rapidez de desarrollo, modularidad e integración con una gran variedad de plataformas, circunstancia que justifica su extendido uso, sobre todo en universidades y centros de investigación.

Por otro lado, ROS (*Robot Operation System*) [5] es una plataforma de código abierto que presenta una adecuada recopilación de herramientas para el desarrollo de software para robótica. Está orientada al control de robots y la comunicación entre los diferentes módulos de un sistema robótico.

Una de las mayores ventajas de este sistema es que ROS define los diferentes procesos involucrados en un proyecto como nodos software, los cuales pueden enviar, recibir y multiplexar datos fácilmente. Esto se puede realizar de manera periódica y automática desde nodos servidor a nodos subscriptor (mediante *topics*) o de forma aislada bajo petición de nodos cliente a nodos servidor (mediante *services*).

La robótica es una asignatura que está ganando presencia en los ámbitos formativos a pasos agigantados, y su carácter técnico, actualizado e innovador abre el debate sobre la mejor manera de enseñarla [2]. En este sentido, la innovación docente se postula como imprescindible, y es conveniente revisar las propuestas más recientes, e.g. [1], [6]. Esta innovación no sólo se da en la forma de la docencia, también en los recursos y su optimización, puesto que no siempre se cuenta con el equipamiento necesario. En este ámbito, el uso de simuladores a menudo se presenta como la mejor solución [4].

Este trabajo presenta una estrategia didáctica innovadora que combina ROS con Matlab® para la docencia de la robótica en la asignatura Robótica de Servicio del Máster Universitario en Ingeniería Industrial [7]. La robótica es una disciplina joven en constante evolución y actualización, y el uso de ROS es cada vez más extendido debido a sus numerosas ventajas. Sin embargo, es una plataforma compleja basada en Linux, y dar una correcta formación a los alumnos en este sentido requiere un tiempo y unos recursos de los que no se dispone en los programas educativos habituales.

No obstante, desde la versión R2015a Matlab® cuenta con los comandos de ROS integrados si se instala la MathWorks Robotics System Toolbox. Esto permite que los alumnos puedan interactuar con plataformas ROS desde Matlab®, un sistema que conocen y con el que se sienten cómodos, eliminando así esta necesidad de formación previa y permitiendo centrar el curso en la docencia de la robótica.

En el presente artículo se expone el proceso docente propuesto y los resultados de su aplicación. En el Apartado 2 se detallan las herramientas utilizadas y se describe la plataforma empleada, el entorno de simulación y el modelo del robot considerado. En el apartado 3 se narra la implementación del proceso

docente, mientras que en el Apartado 4 se exponen los resultados obtenidos. Por último, el apartado 5 ofrece las conclusiones y consideraciones oportunas.

2 DESCRIPCIÓN DE LAS HERRAMIENTAS

Como se ha comentado, para poder implementar la integración de ROS y Matlab® es necesario contar con la versión 2015a (o posterior) del software de computación numérica. Este software fue instalado en ordenadores con sistema operativo Windows® por comodidad de cara al alumnado. Por otro lado, ROS se ejecuta sobre una distribución de Linux, en nuestro caso Ubuntu 14.04 LTS con ROS Indigo Igloo.

Para conectar ambas plataformas lo más cómodo es dotar a la unidad Linux de una dirección IP fija. Así, se cuenta con tantas unidades de Matlab® como requiera el proceso docente, basta con establecer la comunicación con el servidor ROS mediante UDP, asegurándose de que todas las unidades se encuentran en la misma área local.

Además, en el escenario ideal para ofrecer unas prácticas satisfactorias de un curso de robótica, los alumnos deberían manejar un robot real mediante nuestra plataforma. Sin embargo, por una simple cuestión de limitación de recursos, no resulta viable tener tantas unidades robóticas como alumnos. Así, en el sistema operativo ROS se ha integrado el simulador gratuito Gazebo, para simular el comportamiento de un determinado robot en un escenario real. En la Figura 1 se puede ver un esquema del sistema propuesto.

A continuación se presenta el entorno de simulación escogido para fines docentes así como el robot simulado con el fin de poner en práctica los conceptos vistos en el curso.

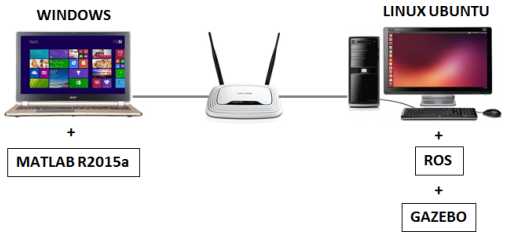

Figura 1: Esquema del sistema propuesto

2.1 ENTORNO DE SIMULACIÓN GAZEBO

Gazebo es un simulador gratuito que encuentra su mayor campo de aplicación en el diseño de robots. Concretamente, ofrece la posibilidad de modelar con precisión y efectividad poblaciones de robots con cualquier configuración en entornos complejos, tanto de interior como de exterior.

El simulador está basado en herramientas open source. La física está fundamentada en la dinámica del sólido-rígido mediante recursos como ODE o Bullet. El renderizado gráfico se realiza mediante OpenGL (OGRE) y las interfaces cuentan con plugins e IPCs basados en Google Protobuf y Boost ASIO. En cuanto a las interfaces de usuario, son de tipo GUI por comodidad, basadas en plataformas como QT o CEGUI. A continuación se puede observar un esquema de la arquitectura principal de Gazebo.

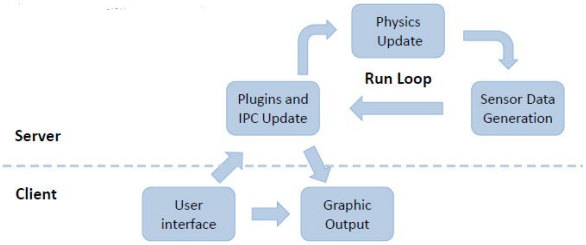

Figura 2: Arquitectura del simulador Gazebo

Las posibilidades que ofrece Gazebo para sus entornos son muy variadas. Se distinguen diversos tipos de elementos de manera jerárquica, como sensores que pueden recoger datos externos de otro dispositivo, objetos visuales que conforman los gráficos u objetos de colisión, que definen superficies que no pueden ser rebasadas. En un nivel superior se definen los enlaces, que combinan objetos visuales y de colisión, y articulaciones, que definen restricciones entre diferentes enlaces. Por encima se encuentran los modelos, que recopilan enlaces, articulaciones y sensores. Por último, los mundos reúnen modelos, configuraciones para los elementos de luz y propiedades globales.

La conveniencia de Gazebo es su integración con ROS. Para acceder a la interfaz de uso de ROS con el simulador Gazebo hay que instalar los paquetes correspondientes, los cuales se encuentran en la librería GAZEBO_ROS_PKGS. Con esto se puede enviar y recibir estructuras de datos para interactuar con Gazebo desde ROS o establecer sensores, motores y componentes dinámicos reconfigurables en Gazebo, entre otras funcionalidades.

2.2 ROBOT PIONEER DX-2

Como se ha comentado anteriormente, aunque lo deseable en un curso de robótica es enriquecer la formación ofrecida mediante su aplicación en robots y entornos reales, las limitaciones de recursos a menudo obligan a buscar soluciones alternativas, con frecuencia basadas en la simulación.

En el proceso docente presentado se proporciona a los discentes un determinado escenario en el que se ha posicionado un robot, mediante el entorno de simulación Gazebo. El objetivo fundamental es que los alumnos realicen la comunicación adecuada con el robot para implementar el control mediante Matlab®, y así poder aplicar los conceptos estudiados mediante un entorno sencillo y que les es familiar, sin la necesidad de invertir un tiempo del que no se dispone en enseñarles ROS. El robot escogido para tal fin es el Pioneer DX-2, el cual se puede ver en la Figura 3.

Figura 3. Robot Pioneer DX-2

Se trata de un robot comúnmente empleado en la educación y la formación en robótica, debido a su sencillez y facilidad de uso. Cuenta con dos ruedas motrices con motores independientes más una tercera rueda pasiva de apoyo. El control independiente de las ruedas motrices le permite girar y desplazarse en cualquier dirección. En la Figura 4 aparece un esquema de la parte inferior donde se aprecia esta configuración.

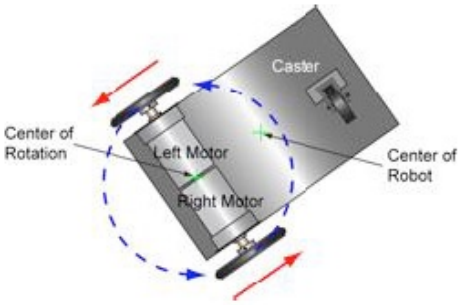

Figura 4. Esquema de la parte inferior del robot

El robot presenta una anchura máxima (de rueda a rueda) de 0.39 m (0.277 m sin las ruedas) y una longitud de 0.445 m. Las ruedas motrices tienen un diámetro de 0.22 m y están separadas 0.3 m de la rueda de apoyo.

En cuanto a los sensores, este robot habitualmente cuenta con anillos de sonares tanto en la parte delantera como en la trasera (como se puede apreciar en la Figura 3). No obstante, como en esta ocasión el robot se modela en un entorno de simulación, con fines docentes se han sustituido los sonares por un sensor láser Hokuyo 2D, el cual proporcionará la información necesaria sobre los obstáculos que se presentan en el entorno cercano del robot, y por tanto permitirá su control y navegación. En la Figura 5 se muestra un ejemplo del funcionamiento del sensor láser Hokuyo 2D simulado en Gazebo.

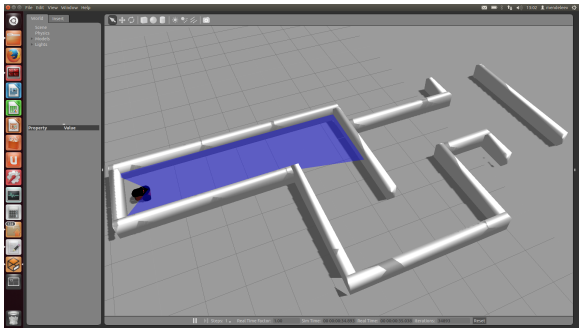

Figura 5. Sensor láser Hokuyo 2D simulado en Gazebo

La figura siguiente muestra el esquema rqt de las comunicaciones entre Matlab y Gazebo para la ejecución de la practica.

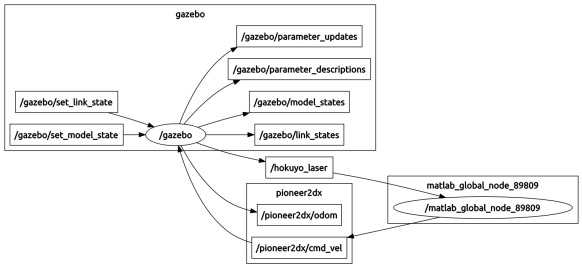

Figura 6. Esquema rqt

Como ejemplo, se muestra parte del código del fichero .sdf de definición del modelo del robot Pioneer DX-2.

```
<?xml version='1.0'?>
<sdf version='1.4'>
  <model name="mobile_base">
    <link name='chassis'>
      <pose>0 0 .25 0 0 0</pose>

      <inertial>
        <mass>20.0</mass>
        <pose>-0.1 0 -0.1 0 0 0</pose>
        <inertia>
          <ixx>0.5</ixx>
          <iyy>1.0</iyy>
          <izz>0.1</izz>
        </inertia>
      </inertial>

      <collision name='collision'>
        <geometry>
          <box>
            <size>2 1 0.3</size>
          </box>
        </geometry>
      </collision>

      <visual name='visual'>
        <geometry>
          <box>
            <size>2 1 0.3</size>
          </box>
        </geometry>
      </visual>


<…>
```

3 PROPUESTA DOCENTE

Paralelamente al desarrollo teórico del curso, se ha desarrollado la componente práctica mediante la plataforma ROS-Matlab® propuesta y el simulador Gazebo con el modelo del robot Pioneer DX-2 con sensor láser Hokuyo 2D con fines ilustrativos. Tras unas primeras sesiones de aproximación a la plataforma consistentes en sencillos ejercicios, se ha ideado una práctica basada en proyecto por grupos de 2 o 3 alumnos para resolver un caso realista, la cual ha sido el fundamento de la evaluación del curso.
Concretamente, al alumnado se le ha presentado un entorno que modela un laberinto, en cuyo interior se ha posicionado el modelo del robot escogido. Las tareas a realizar consisten en establecer la correcta comunicación con el robot a través de Matlab® (mediante los comandos de MathWorks Robotics System Toolbox), para obtener la información del sensor láser y de la posición y orientación actuales del robot, identificar la configuración del entorno e implementar el control del robot de manera que pueda avanzar e ir navegando hasta que consiga salir del laberinto, y enviar los datos correspondientes a los actuadores del robot. En la Figura 5 se puede apreciar una captura del entorno propuesto con el robot en su interior, mientras que en la Figura 7 aparece el esquema del flujo de la información propuesto.

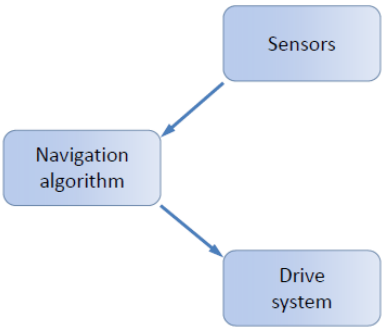

Figura 7. Flujo de la información

Para implementar el algoritmo de navegación, el software desarrollado por los alumnos en el entorno Matlab® debe subscribirse a los *topics* de ROS /hokuyo_laser y /pioneer2dx/odom. En el primero de ellos se publican los valores tomados por el sensor láser, mientras que en el segundo se publican la posición y orientación actuales del robot. Además, el software debe publicar un nuevo *topic*, pioneer2dx/cmd_vel, con los comandos para el movimiento del robot. A este *topic* debe estar subscrito el modelo del robot en Gazebo.
Los comandos para el movimiento del robot se componen de dos velocidades, una lineal (desplazamiento hacia delante) y una angular (giro hacia derecha o izquierda). Dado que solo hay dos ruedas directrices y cada una tiene un motor independiente, estableciendo los valores y los signos adecuados para cada una de estas dos velocidades se puede conseguir que el robot se desplace en cualquier dirección y sentido. Como medida de seguridad, se establece que las velocidades jamás podrán ser superiores a 1 m/s (lineal) o 1 rad/s (angular).

En cuanto al laberinto, se proporciona una primera versión a los discentes para que puedan elaborar su proyecto e ir comprobando el funcionamiento, corrigiendo errores y optimizando el software. El laberinto se simula en Gazebo y se garantiza que la anchura de los pasillos es de 2 m, desde una pared hasta la opuesta. Además, se asegura que para resolverlo tan solo será necesario implementar giros tanto a derecha como a izquierda de 90º, intersecciones en T o cambios de sentido (giros de 180º).
Un video con la demostración del ejemplo mostrado a los alumnos puede verse en:

https://youtu.be/SSGLPlawTm4

Tras las correspondientes sesiones para el desarrollo de los proyectos, se realiza la evaluación. Para ello, cada grupo debe realizar una presentación oral en la que exponga el trabajo realizado, prestando especial atención al algoritmo de navegación y las comunicaciones Matlab®-ROS implementadas. En estas presentaciones se valora positivamente el uso de gráficos ilustrativos, así como el diseño modular del software, desglosando el algoritmo en subfunciones.

Seguidamente, el sistema diseñado por cada grupo será puesto en práctica en un nuevo entorno con un laberinto diferente (hasta el momento desconocido por los alumnos). En esta etapa se valorará la efectividad de los sistemas desarrollados a la hora de salir del laberinto, el número de colisiones con las paredes y, con especial interés, el tiempo que tarda cada robot en salir del laberinto. Se premiará al grupo que consiga salir en menor tiempo.

4 RESULTADOS

En el curso han participado un total de 17 alumnos, por lo que para la evaluación se han configurado 7 grupos: 4 grupos de 2 alumnos y 3 grupos de 3 alumnos. En términos generales, los trabajos presentados han sido satisfactorios, todos los grupos han desarrollado su algoritmo de navegación y han intentado darle el carácter modular buscado, aunque no todos han tenido éxito en la ejecución. Por otro lado, las comunicaciones Matlab®-ROS se han llevado a cabo correctamente en la totalidad de los casos, tal y como perseguía uno de los objetivos principales de esta propuesta.

En la figura 8 se muestra parte del código Matlab generado por un equipo de estudiantes para la

navegación del robot ejemplo en un laberinto con las características del laberinto planteado:

```
function Robot

    % Creation of the node with a custom name
    TestNode = robotics.ros.Node('Robot');

    % Creation of the subscribers with the name of the
topics (first parameter)
    % and the message format (second parameter).  A
different subscriber is
    % needed for each topic
    Laser = rossubscriber('hokuyo_laser',
'sensor_msgs/LaserScan');
    %Posicion = rossubscriber('pioneer2dx/odom',
'nav_msgs/Odometry');

    % Creation of the publisher with the name of the new
topic (first parameter)
    % and the message format (second parameter).  As for
the subscribers, a
    % different publisher is needed for each new topic
    Movimiento = rospublisher('pioneer2dx/cmd_vel',
'geometry_msgs/Twist');

    % Infinite loop to maintain active the data exchange
    while(1)

        % Creation of the input messages and reception
of the entry data
        InputMessage1 = receive(Laser);
        %InputMessage2 = receive(Posicion);

        % Algoritmo de Navegacion
        Rango = InputMessage1.Ranges;
%         Pose = InputMessage2

        FRONT = 0;
        for(i=283:358)
            if Rango(i)<1.25%0.7775
                FRONT = 1;
            end
        end

<…>

if(FRONT==0)
            % Creation of the output message and filling
of the variables
                OutputMessage = rosmessage(Movimiento);
                OutputMessage.Linear.X = 0.5;
                OutputMessage.Angular.Z = 0;
                if(RIGHT3==0)

<…>

% Sending the message
        send(Movimiento, OutputMessage);


    end
end
```

Figura 8. Código Matlab propuesto por un grupo de alumnos

Por ejemplo, uno de los algoritmos de navegación presentados consistía en la obtención de las distancias del robot a las paredes laterales y frontal. Así, la distancia a la pared frontal se evaluaba y se comprobaba si se encontraba por debajo de un valor umbral (0.7775 m, para intentar mantener el robot centrado con respecto al resto de paredes). En caso negativo se permitía el avance frontal, mientras que en caso positivo se detenía el robot y se evaluaba si la distancia a la pared derecha era superior a 2 m, lo que suponía que no había obstáculos a la derecha. En este caso, se iniciaba la rotación hacia la derecha. En caso negativo, se realizaba el mismo proceso respecto de la pared izquierda. En el caso de no encontrar libre ninguna dirección, se permitía el giro a la derecha para buscar otra posición. Todas estas rotaciones se mantenían hasta que la distancia a la pared frontal superaba el valor umbral establecido. En la Figura 9 se puede apreciar el diagrama de flujo propuesto por los alumnos., y en la Figura 10 se muestra una captura de pantalla de su robot resolviendo el laberinto inicial.

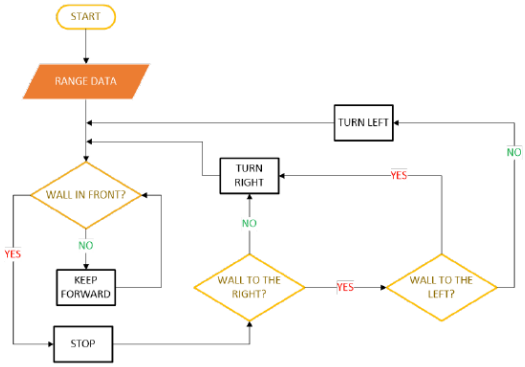

Figura 9. Diagrama de flujo propuesto por un grupo de alumnos

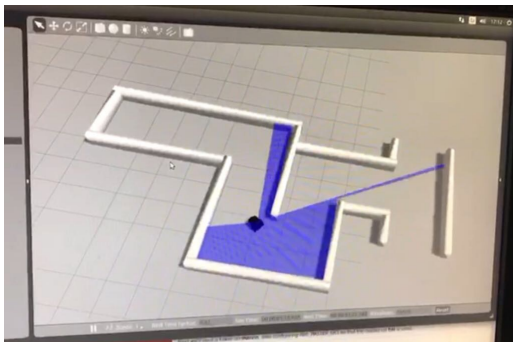

Figura 10. Captura de pantalla del robot resolviendo el laberinto inicial

En cuanto a la evaluación del funcionamiento de los sistemas desarrollados en un laberinto desconocido, los resultados obtenidos han sido variados. Cinco grupos consiguieron salir del laberinto con diferentes tiempos y números de colisiones, mientras que los algoritmos de dos de los grupos produjeron fallos de ejecución a lo largo de la prueba que les impidieron lograr el objetivo, en ambos casos en un punto del recorrido en el que se requería un giro de 180º, en el que el robot entró en un bucle de giro continuo. En la Tabla 1 se muestra un resumen de los resultados del proceso de evaluación.

Tabla 1: Resultados de la evaluación

Grupo	Conseguido	Colisiones	Tiempo (s)
1	SÍ	1	52
2	SÍ	0	45
3	NO	-	-
4	SÍ	0	43
5	SÍ	5	73
6	SÍ	1	58
7	NO	-	-

5 CONCLUSIONES

A la vista de los resultados expuestos, la valoración de la propuesta docente es positiva. La robótica es una disciplina que aúna una considerable carga teórica con una fuerte componente práctica, siendo la segunda imprescindible para la correcta comprensión de la primera. En este artículo se ha presentado una propuesta para solventar los problemas de recursos materiales y temporales a los que el ámbito docente tiene que hacer frente con más frecuencia de la deseable, y los resultados han mostrado un alto grado de consecución de los objetivos que lleva implícito un resultado satisfactorio del proceso docente.

Es de destacar la motivación extra que generan en el alumnado este tipo de propuestas, que consiguen acercarlo al entorno más realista posible sin requerirle un esfuerzo excesivo, y le permite centrarse en las enseñanzas propias del curso. Además, la componente competitiva siempre es un aliciente para todos los participantes, quienes se implican más y se interesan en mayor grado por todos los aspectos del curso. El hecho de poder ofrecer una formación práctica ajustada a los conceptos del curso que el alumno lleva a cabo mediante una herramienta que conoce y domina como es Matlab® y que le incita mayor participación y motivación es muy recomendable.

Agradecimientos

Los autores desean agradecer la financiación recibida por parte del Ministerio de Economía y Competitividad a través del proyecto DPI2013-47196-C3-2. Carlos G. Juan quiere expresar su gratitud al Ministerio de Educación, Cultura y Deporte por la financiación recibida a través del programa de Formación de Profesorado Universitario (FPU) en su convocatoria de 2014, mediante la solicitud FPU14/00401.

Referencias

[1] Agostini, A., Torras, C. y Wörgötter, F., (2017) "Efficient interactive decision-making framework for robotic applications", *Artificial Intelligence,* vol. 247, pp. 187-212.

[2] Kucuk, S. y Sisman, B., (2017) "Behavioral patterns of elementary students and teachers in one-to-one robotics instruction", *Computer and Education,* vol. 111, pp. 31-43.

[3] MathWorks, (2017) Matlab. En línea: https://es.mathworks.com/products/matlab.html. [Último acceso: 25 mayo 2017].

[4] Plaza, J., Quevedo, M. y Matellán, V., (2009) "Use of simulators in the teaching of mobile robotics [Uso de simuladores en docencia de robótica móvil]", *Revista Iberoamericana de Tecnologias del Aprendizaje,* vol. 4, nº 4, pp. 268-277.

[5] ROS, (2017) ROS. En línea: http://www.ros.org/. [Último acceso: 25 mayo 2017].

[6] Sooraksa, P., Sakorntanant, S., Jansri, A. y Klomkarn, K., (2016) "Tree robot: An innovation for STEAM education", *IEEE International Conference on Real-Time Computing and Robotics, RCAR*, Angkor Wat, Camboya, pp. 338-341.

[7] Universidad Miguel Hernández, (2017) Robótica de Servicio. En línea: http://www.umh.es/contenido/pdi/:asi_m_2987/datos_es.html. [Último acceso: 25 mayo 2017].