DESARROLLO DE ESTRATEGIAS DE CONTROL DISTRIBUIDO EN ROBOTS MÓVILES SOBRE EL MIDDLEWARE DEL NÚCLEO DE CONTROL

Josep Tormo, Raúl Simarro, Eduardo Munera, José E. Simó, Juan-Luis Posadas-Yagüe
Instituto de Automática e Informática Industrial (ai2)
Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain.
jotorcos@etsii.upv.es, {rausifer, emunera, jsimo, jposadas}@ai2.upv.es

Resumen

En este trabajo se presenta la simulación de un robot móvil con control distribuido, cuyo objetivo es la validación de un middleware de control. Se definen las características y servicios que proporciona el middleware, así como las funciones más importantes de comunicación, modos de funcionamiento, calidad de control, etc. Se detallan las pruebas realizadas mediante un simulador de sistemas robóticos en el que se han implementado las funciones de middleware. Por último, se introduce el proceso de construcción del robot real, que será el siguiente paso en el proceso de validación del middleware.

Palabras Clave: control distribuido, middleware de control, simulación

1 INTRODUCCIÓN

Cuando se realiza la programación de cualquier tipo de aplicación software, una parte importante dentro de su desarrollo es la fase de testeo y pruebas para comprobar su correcto funcionamiento. En el caso de un middleware de control, en el que en su desarrollo intervienen varias personas, cada una especialista en un campo (comunicaciones, control, componentes hardware, etc.), y en el que en la implementación actúan diversos componentes físicos, así como diversas configuraciones disponibles, ésta fase de validación y testeo es muy importante y crítica. Por lo tanto, la posibilidad de disponer de demostradores, diversas configuraciones hardware, sensores, etc., es fundamental para probar y verificar el correcto funcionamiento.

Los simuladores permiten, en una primera fase, testear diversas configuraciones hardware, comunicaciones, condiciones de funcionamiento, etc., de una manera sencilla, provocando las situaciones más desfavorables para poder validar el correcto funcionamiento de la implementación realizada y la detección de posibles errores. Una vez superada esta primera fase de pruebas mediante simulación, el sistema está preparado para ser implementado sobre un sistema real [8].

A continuación se presenta el trabajo para realizar las pruebas de un middleware de control. Este middleware, denominado núcleo de control, permite la comunicación entre dispositivos, el tratamiento de la información de sensores inteligentes, distintos modos de funcionamiento, cálculo de la calidad del control, etc.

Las pruebas realizadas se centran en demostradores aplicados a la robótica móvil. La razón de utilizar este tipo de demostración es que el sistema de sensorización está distribuido, se dispone de varios nodos de cómputo y para la consecución del objetivo a conseguir se deben coordinar diversas tareas.

2 ESTRUCTURA DE CONTROL

La heterogeneidad de los sistemas actuales, formados por múltiples componentes de características diferentes conectados en red, sugiere el desarrollo de sistemas de control distribuido en los que las distintas funciones de control se implementen a diferentes niveles. Cuando en esta configuración se incluyen elementos dotados de movilidad (robots, vehículos,…), la capacidad de obtener y procesar información, así como la de generar acciones de control, es variable con el tiempo. Es más, los recursos disponibles en cada momento por cada elemento para realizar las actividades que se le asignen, también variables, son limitados. Por ello, es preciso definir una estructura de control jerarquizado de forma que se asegure que las acciones de máxima prioridad se ejecuten a nivel local y que el resto (opcionales, refinamientos, de coordinación) se lleven a cabo en niveles superiores, con menor prioridad.

2.1 MIDDLEWARE DE CONTROL

El núcleo de control, asimilable al núcleo de un sistema operativo, se define como el código mínimo que debe ejecutarse en una aplicación de control para que el funcionamiento sea seguro, y es el encargado

ISBN-13: 978-84-697-0589-6

de la conexión entre los nodos del sistema distribuido [1].

Un sistema de control distribuido debe garantizar siempre la seguridad en el proceso que está regulando. La estrategia de control debe hacer frente a problemas como pérdida de datos, retardos o tiempo excesivo de cálculo, consiguiendo una degradación admisible de prestaciones. Los controladores desarrollados deben permitir la adaptación de los mismos a las condiciones de trabajo, utilizando los recursos adecuados en cada momento, para no tener pérdidas significativas de prestaciones en el logro de los objetivos a conseguir.

El núcleo de control se materializa en la forma de un middleware sobre el que se ejecuta la aplicación de control, que está compuesta por diferentes actividades computacionales. El middleware permite una adecuada separación entre el sistema operativo (SO), si lo hubiese, y la aplicación de control, brindando un diseño rápido de aplicaciones y un desarrollo transparente. El middleware gestiona la ejecución de las actividades, y ofrece componentes básicos necesarios para el diseño de aplicaciones de control (reguladores, conmutadores, muestreadores, actuadores, retenedores,…), además, incorpora servicios de evaluación de la "calidad del control" que permiten la detección de situaciones de excepción y la definición de acciones asociadas a estas situaciones.

El middleware ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas. Funciona como una capa de abstracción de software distribuida, que se sitúa entre las capas de aplicaciones y las capas inferiores (sistema operativo y red). El middleware nos abstrae de la complejidad y heterogeneidad de las redes de comunicaciones subyacentes, así como de los sistemas operativos y lenguajes de programación, proporcionando una API para la fácil programación y manejo de aplicaciones distribuidas. Dependiendo del problema a resolver y de las funciones necesarias, serán útiles diferentes tipos de servicios middleware.

2.1.1 Middleware reducido (Tiny Middleware)

El middleware de control en su versión más reducida, también conocida como Tiny Middleware, ha sido desarrollado para ofrecer una solución ligera y eficiente que pueda ser implementada en sistemas con recursos limitados. En un entorno de control, este tipo de sistemas serían encargados de realizar tareas sencillas de percepción y actuación al más bajo nivel. Para ello, intercambiará información con otros dispositivos del sistema, que usualmente implementan una versión completa del middleware. Al mismo tiempo, ofrecerá capacidades de monitorización del sistema y gestión de alarmas, a fin de detectar posibles fallos e informar de los mismos. Por lo tantos los principales servicios ofrecidos por el Tiny Middleware serán:

- Servicios de percepción y actuación: Deberán interactuar con los sensores y actuadores físicamente conectados al nodo, con el fin de garantizar un flujo constante de información preprocesada en el caso del sensor, e interpretando correctamente las acciones de control en el del actuador. En ambos casos será crítico poder asegurar el periodo de muestreo o actuación.
- Delegación de código: Son mecanismos para otorgar una mayor flexibilidad al sistema. Los nodos con una mayor carga computacional podrán delegar ciertas funciones a otros nodos más ociosos, asegurando un mejor aprovechamiento de los recursos.
- Monitorización del sistema y alarmas: Este servicio se encargará de monitorizar los recursos del sistema así como el correcto funcionamiento de los controladores. Al mismo tiempo, un sistema de alarmas avisará de posibles errores del sistema que gestionará estas excepciones a fin de proporcionar un sistema tolerante a fallos.
- Sistema de comunicaciones: Manejará las diferentes interfaces de comunicación ofreciendo un interfaz común a todas ellas. Dependiendo de los recursos de cada nodo se implementarán diversos tipos de protocolos, tales como RS-232 o RS-485 para los más limitados y TCP/IP para aquellos con mayores prestaciones.
- Gestión de datos: Ofrece una transparencia en el intercambio de información, permitiendo acceso tanto a datos locales como a variables compartidas en la red mediante un sistema "Data Centric Publisher Subscriber" (DCPS) [6].

2.1.2 Middleware completo (Full Middleware)

El middleware completo, o Full Middleware, ha sido diseñado para ser implementado en sistemas con unas limitaciones menos restrictivas que en el caso anterior, y por lo tanto la cantidad de servicios que pueden ofrecer ha sido ampliada. El Full Middleware ofrece un interfaz más completo, y será usualmente el encargado de la ejecución de los algoritmos de control más complejos, así como de aglutinar la información de rendimiento, mediante medidas de calidad de servicio (QoS) [7], y alarmas producidas por los nodos con el middleware reducido. Al mismo tiempo ofrecerá un interfaz de alto nivel para

ISBN-13: 978-84-697-0589-6

gestionar las configuraciones del sistema. Por lo tanto, los servicios exclusivos del Full Middleware son descritos a continuación:

- Gestión de Calidad de servicio: Este servicio está orientado a la supervisión de ciertas medidas de calidad en la ejecución del sistema, de forma que pueda efectuar una gestión más eficiente de las tareas y servicios.
- Gestión de recursos: Ofrece una interfaz para la correcta gestión de los recursos del sistema, tanto del propio nodo como de otros servicios de delegación de código entre nodos. Realizará las modificaciones pertinentes del sistema de control para adaptarse al funcionamiento
- Interfaz de configuración y monitorización: Ofrece un entorno "Human Machine Interface" (HMI) para la gestión de las configuraciones del sistema de forma gráfica. Al mismo tiempo este entorno servirá para la monitorización del sistema.

2.2 SISTEMA DISTRIBUIDO DE CONTROL

La figura 1 muestra la estructura propuesta del sistema distribuido de control, formada por nodos supervisores, que implementan un middleware completo, y nodos locales, con un middleware reducido [9].

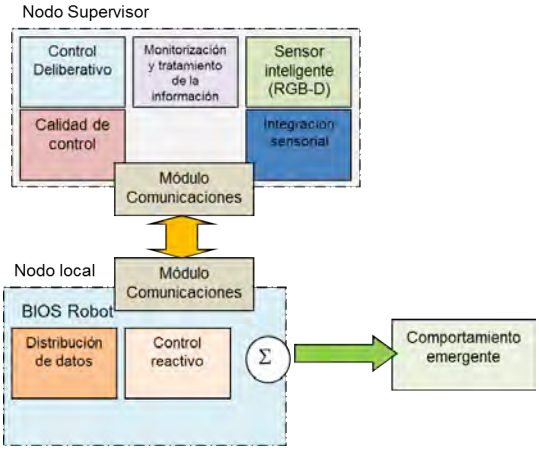

Figura 1: Estructura del sistema distribuido de control en el robot móvil

El supervisor se encarga de ejecutar las tareas de control de alto nivel que necesitan una gran capacidad de cómputo, o bien, las que necesitan la integración de varios sensores inteligentes que proporcionan una información adicional sobre el escenario, como por ejemplo el sensor Kinect, que es una cámara de tipo RGB-D (*Red, Green, Blue, Depth*). El supervisor es el que ejecuta la parte deliberativa del comportamiento del robot, como puede ser un seguimiento de la trayectoria, la realización de mapas del entorno, la integración de la información sensorial para la mejora de la navegación, etc.

El local es el que se encuentra integrado en el propio robot, y es el que permite realizar las tareas más inmediatas sobre el proceso. El objetivo fundamental de los nodos locales es asegurar que siempre exista una acción de control para mandar al proceso. En el caso de un sistema robótico es el que integra la parte reactiva de evitación de obstáculos o el alcanzar las zonas seguras del entorno.

3 SIMULACIÓN DEL SISTEMA ROBÓTICO

Para poder validar la estrategia de control propuesta, se propone la utilización de un simulador que permita implementar la estructura de control distribuida, la utilización del middleware de control y que tenga un interfaz de programación fácil. La plataforma de simulación elegida es V-REP [3].

V-REP es un potente simulador 3D de robots de software libre (en su versión académica) basado en una arquitectura modular y descentralizada. Ofrece la posibilidad de comunicar el simulador con C/C++, Java, Python, Matlab, Octave y Urbi mediante el uso de APIs de clientes remotos. De esta manera se puede controlar un mismo robot desde un script de Lua empotrado en el mismo simulador o desde fuera del simulador [4].

Para implementar el nodo supervisor se utilizará Matlab, ya que permite la realización rápida y fácil de algoritmos de control, además de estar disponibles las funciones del middleware. V-REP dispone de la comunicación con Matlab mediante una API remota.

3.1 SIMULACIÓN LOCAL

Cuando no se necesitan utilizar funciones que requieran un gran procesamiento de la información, ni tareas de alto nivel, se puede implementar el comportamiento del robot sobre el nodo local directamente. Para realizar esta implementación se pueden utilizar dos alternativas: mediante un Child Script o con un Plugin en C.

3.1.1 Child Script

Se aplica utilizando scripts asociados a los objetos del escenario, es decir, un Child Script estará asociado al robot y el código se escribirá en éste, en lenguaje Lua [5]. En la figura 2 se muestra una prueba, utilizando las funcionalidades

ISBN-13: 978-84-697-0589-6

proporcionadas por el middleware mediante el Child Script, en que el robot implementa dos comportamientos: seguimiento de una trayectoria definida y evitación de obstáculos.

3.1.2 Plugin en C

Se implementa haciendo uso de la funcionalidad de plugins en C proporcionada por el simulador. En este caso, el simulador cargará automáticamente al inicio la librería compartida, con esto, en el child script asociado al robot se pone una función que pasará el control del robot a manos del código escrito en C. De este modo, se puede probar el funcionamiento del middleware cada vez que se desee con solo compilar el código C y generar una librería compartida.

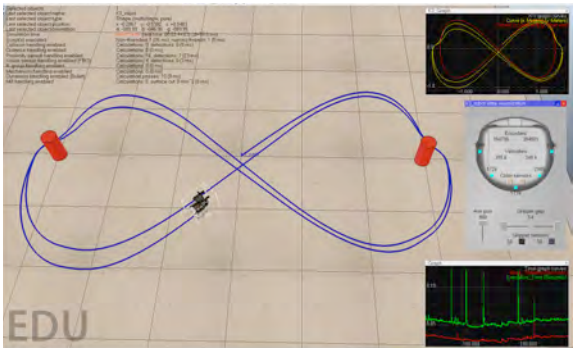

Figura 2: Prueba del seguimiento de trayectoria y evitación de obstáculos

3.2 SIMULACIÓN DISTRIBUIDA

Para simular la estructura de control de nodos mencionada, se ubica el código de control del nodo supervisor en Matlab, mientras que el código de control del nodo local se aloja en el script del propio robot en V-REP. En la figura 3 se muestran los dos modos de comunicación utilizados.

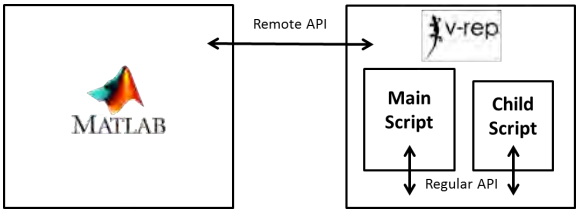

Figura 3: Esquema del sistema distribuido implementado en simulación

Por una parte, para comunicar el propio robot (Child Script) con el entorno de simulación (Main Script) se hace uso de la API regular, que está compuesta por varios cientos de funciones que se pueden llamar desde cualquier script de Lua empotrado.
Por otra parte, para comunicar V-REP con Matlab se utiliza la API remota, formada por un conjunto de funciones que se pueden llamar desde Matlab y permiten la comunicación con V-REP vía socket. Se pueden establecer comunicaciones tanto asíncronas como síncronas. En el lado de Matlab se llama a las funciones necesarias para la comunicación y se asegura la conexión. En el lado del simulador se introduce el puerto seleccionado y la estructura necesaria para una comunicación síncrona.

3.2.1 Pruebas realizadas

Para testear la estructura de control distribuida, se ha realizado la siguiente prueba: Se parte de un robot con tracción diferencial y un brazo manipulador Khepera III al que se le añade una cámara RGB-D (Kinect), tal y como se muestra en la figura 4. Se crea un entorno con obstáculos al que se añaden varios cilindros rojos. La misión del robot será encontrar y recoger un cilindro rojo para dejarlo en algún punto del escenario previamente indicado, evitando los obstáculos que se le presenten.

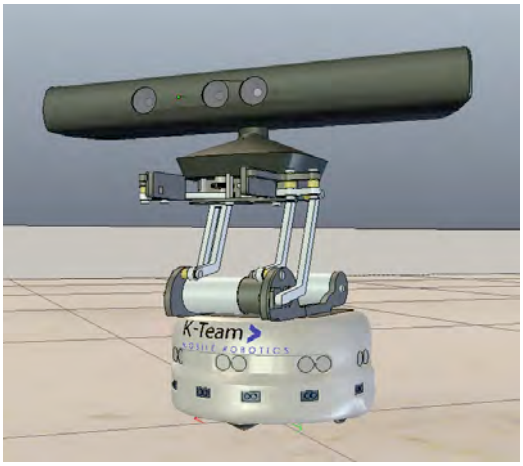

Figura 4: Robot con el brazo manipulador y sensor RGB-D

Siguiendo el esquema de comunicación distribuida, en el script del robot se implementan los comportamientos más básicos, que permitirán que ante cualquier fallo de comunicación con el supervisor se pueda continuar proporcionando una acción de control. Estos comportamientos son: la cinemática inversa y la evitación de obstáculos, basada en un vehículo de Braitenberg [2]. De esta forma, en Matlab es donde se implementan los comportamientos de más alto nivel y por tanto con mayores requerimientos de computación. Estos comportamientos son: el procesamiento de la imagen enviada por la Kinect, el cálculo de la trayectoria para llegar al punto objetivo y el mapeo del escenario combinando los sensores infrarrojos y la Kinect, además de la adquisición del punto de llegada pedido y la conmutación de misiones.

ISBN-13: 978-84-697-0589-6

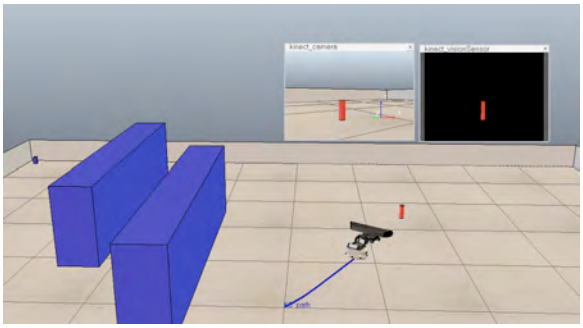

Figura 5: Prueba de reconocimiento y manipulación de objetos

3.3 UTILIZACIÓN DEL MIDDLEWARE DE CONTROL CON V-REP

Con el propósito de disponer de un interfaz común, tanto para robots físicos, como para robots simulados mediante el uso de V-REP, se ha adaptado el uso del Middleware de control en ambas arquitecturas. Gracias a ello, la migración de código entre dichas plataformas será instantánea. De esta forma, el simulador podrá ser utilizado para la realización de pruebas repetitivas de forma acelerada, relegando el uso de la plataforma física para la validación final, ahorrando así tiempo en el desarrollo de experimentos y evitando el deterioro del robot real como resultado de pruebas erróneas.

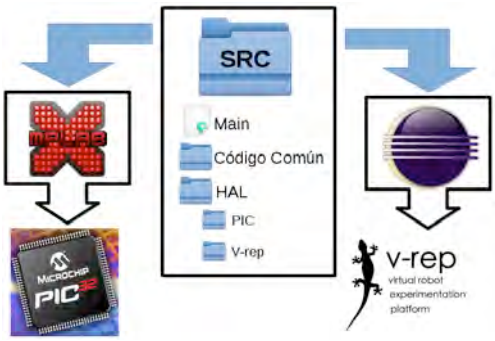

Figura 6: Migración de código entre plataformas.

Además de esto, el poder ejecutar el Middleware en ambas plataformas implica también heredar su jerarquía de nodos así como sus capacidades de comunicación. Teniendo esto en cuenta, surge la posibilidad de combinar nodos reales con nodos simulados que interactuarán de forma totalmente transparente.

Una de las pruebas realizadas es la implementación del nodo supervisor sobre un PIC32, que se comunica con el simulador. En el simulador se ejecuta el nodo local sobre el robot simulado, que enviará la información de los sensores del robot hacia el nodo supervisor, y recibirá las acciones de control calculadas en el PIC32.

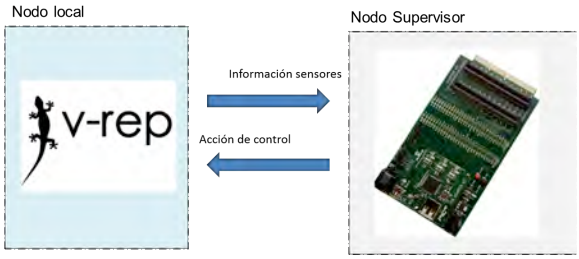

Figura 7: Implementación del nodo supervisor sobre un PIC32

4 CALIDAD DE CONTROL

Una de las tareas del nodo supervisor es la de monitorizar los datos del sistema. Estos servicios permiten comprobar el cumplimiento de las especificaciones de control, así como valorar su correcto funcionamiento. Estos parámetros son una medida de la calidad de control (QoC) obtenida con la estrategia implementada. En caso de que la calidad no sea la esperada se pueden realizar acciones correctoras.

Uno de los parámetros para estimar la QoC son los incrementos en la acción de control, así como el mantenimiento de la acción de control en valores cerca del límite de actuación. En caso de que se detecten acciones fuera de rango se podría cambiar la estrategia de control relajando las especificaciones.

Desde luego un parámetro fundamental para estimar la QoC es el error, es decir, la diferencia entre la referencia y las variables controladas. En caso de que ese error supere un cierto valor definido, el supervisor podría enviar una acción de control correctora o cambiar el tipo de control.

4.1 PRUEBAS REALIZADAS

A continuación se muestran las pruebas realizas en un robot móvil en el seguimiento de una trayectoria (figuras 8 y 9). El control local realiza la estimación de la posición en función de la odometría. El supervisor dispone de un sistema de localización global del robot, pero que sólo puede ser enviado en ciertos instantes. En caso de que el supervisor detecte un error muy grande envía la posición global para que el robot recalcule su posición con datos más precisos.

La figura 8 muestra el error entre la posición del robot y la trayectoria a seguir. El nodo supervisor cuando envía la posición global permite reducir el error del robot y así mejorar el seguimiento en la trayectoria (figura 9).

ISBN-13: 978-84-697-0589-6

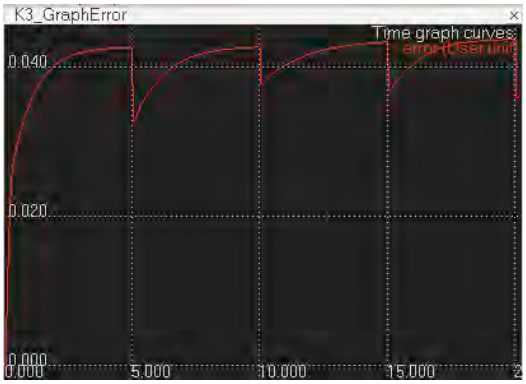

Figura 8: Error cometido en función del tiempo

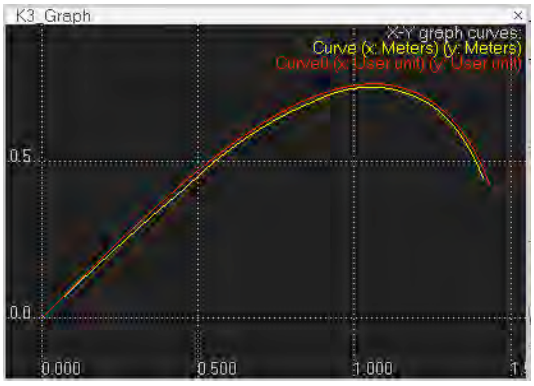

Figura 9: Seguimiento de la trayectoria con la corrección del error cometido

5 SISTEMA ROBÓTICO REAL

El robot real, que va a servir de demostrador de las funcionalidades del middleware de control, ha sido diseñado dentro del proyecto de investigación CICYT denominado COBAMI con referencia DPI 2011-28507-C02-01/02. Se trata de un robot diferencial con sensores de infrarrojos y ultrasonidos, cuya principal novedad es la de montar el sensor inteligente de visión RGB-D, basado en Kinect, y toda la electrónica asociada.

Las piezas de este robot se han diseñado mediante un programa de diseño asistido por computador, para posteriormente generarlas mediante una impresora 3D. Este mismo diseño se ha aprovechado para importar el modelo al simulador, con lo que se puede ver tanto su aspecto físico como su comportamiento con la estrategia de control implementada. Además, de esta forma, se puede seleccionar la mejor ubicación para los sensores (anillo de infrarrojos, encoders, Kinect,..) y los actuadores (motores). En la figura 10 se muestra el robot real y su simulación mediante V-REP.

Figura 10: Robot real y simulado

6 CONCLUSIONES

En este trabajo se presenta una estructura de control distribuida aplicada a sistemas robóticos móviles. La estructura de control está formada por nodos supervisores y locales, que hacen uso de un middleware de control realizado en el entorno del proyecto en el que se inscribe el presente trabajo.

Para probar la estrategia de control se ha utilizado un entorno de simulación, en el que se han implementado los distintos nodos de control haciendo uso de la API proporcionada por el middleware de control.

En el simulador se ha realizado varias pruebas que permiten desde validar el middleware como las distintas estrategias de control propuestas.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el proyecto CICYT COBAMI con referencia DPI 2011-28507-C02-01/02, del Ministerio de Educación y Ciencia.

Referencias

[1] Albertos, P., Crespo, A., y Simó, J.E. (2006) Control Kernel: A key concept in embedded control systems. 4th IFAC Symposium on Mecatronic Systems.

ISBN-13: 978-84-697-0589-6

[2] Braitenberg, V. (1984) Vehicles: Experiments in synthetic psychology. Cambridge, MA. MIT Press.

[3] Coppelia Robotics (2014) V-REP: Virtual Robot Experimentation Platform. http://www.coppeliarobotics.com/

[4] Freese, M., Singh, S., Ozaki, F., & Matsuhira, N. (2010). Virtual robot experimentation platform V-REP: a versatile 3D robot simulator. In *Simulation, Modeling, and Programming for Autonomous Robots* (pp. 51-62). Springer Berlin Heidelberg.

[5] Lua The Programming Language (2014) Lua 5.2 Reference Manual. http://www.lua.org/manual/5.2/

[6] Pardo-Castellote, G. (2003, May). Omg data-distribution service: Architectural overview. In Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on (pp. 200-206). IEEE..

[7] Poza, J. L., Posadas, J. L., & Simó, J. E. (2009, January). QoS-based middleware architecture for distributed control systems. In International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008) (pp. 587-595). Springer Berlin Heidelberg.

[8] Pressman, R. S. (2005) "Ingeniería de Software. Un enfoque práctico". Editorial McGraw-Hill

[9] Simarro, R., Coronel, J., Simó, J., & Blanes, J. F. (2008, July). Hierarchical and distributed embedded control kernel. In *17th IFAC World Congress*.

ISBN-13: 978-84-697-0589-6