

Plataformas de Bajo Coste para la Realización de Trabajos Prácticos de Mecatrónica y Robótica

A. Valera*, A. Soriano, M. Vallés

Instituto U. de Automática e Informática Industrial (ai2), Universitat Politècnica de València, Camino de Vera, nº 14, 46022, Valencia, España.

Resumen

Al igual que ocurrió con la introducción a la programación, el aprendizaje de la automatización, el control por computador, la robótica y los sistemas mecatrónicos en general son contenidos que se están impartiendo cada vez más a estudiantes universitarios y de educación secundaria. Por esta razón, la elección de plataformas adecuadas para el trabajo de laboratorio se convierte en una decisión crítica para promover la experimentación de los conceptos teóricos y la motivación de los estudiantes. Gracias a los avances en la tecnología, hoy en día hay muchas opciones disponibles, tanto a nivel de hardware como a nivel de lenguajes de programación. Este artículo presenta una plataforma de bajo coste multidisciplinar que permite cubrir diferentes cuestiones relacionadas con la realización de trabajos prácticos relacionados con el control automático. *Copyright © 2014 CEA. Publicado por Elsevier España, S.L. Todos los derechos reservados.*

Palabras Clave: Sistemas controlados por computador, control de robots, sistemas multi-agente, sistemas embebidos, robots móviles, educación en control.

1. Introducción

Para el aprendizaje de las disciplinas relacionadas con la ingeniería y, en concreto, para el aprendizaje de la teoría del área de la automatización y de ingeniería de sistemas, resulta sumamente importante poder experimentar en el laboratorio sobre los conceptos desarrollados a nivel teórico en las clases de aula. Esta necesidad se percibe de la misma forma por parte de las industrias. En este sentido, en la citada área de conocimiento, la robótica y el control de sistemas mecatrónicos en general, está teniendo un gran éxito en contextos educacionales en los últimos años [Weinberg and Yu (2003), Jung (2013)]. Uno de esos motivos reside en la posibilidad de que el alumno ponga en práctica el trabajo planteado de manera teórica en el aula.

El principal problema a la hora de poder conseguir esta experimentalidad reside en el precio de los equipos. Generalmente, el coste de estos equipos es muy alto, lo que obliga a tener que compartir un único equipo por todos los alumnos del curso. Algunos autores proponen salvar este inconveniente mediante el uso de procesos simulados (laboratorios virtuales) (Márquez and Sanguino (2010)) y/o el acceso a procesos físicos a través

de Internet (laboratorios remotos) (Valera et al. (2005)). En este último caso, la multiplicidad de uso se consigue, o bien, habilitando franjas temporales de uso para cada uno de los alumnos, o bien, mediante la compartición de equipos entre distintos departamentos o universidades (Guo et al. (2007), J. Henry (2005), Ionescu et al. (2013), Chaos et al. (2013)). Afortunadamente, en los últimos años se pueden adquirir varias plataformas de robots económicas (Adept Mobile Robots (ADEPT (2014)), Moway (2014), epuck (2014), LEGO Mindstorms (2014), etc.). Estas plataformas se componen habitualmente de controladores, sensores electrónicos, sistemas mecánicos de bajo coste y/o pequeños robots. Éstos no proporcionan la misma precisión que los robots industriales, pero son suficientes para los procesos educativos. Además, debido a su naturaleza multidisciplinar, estas plataformas son muy interesantes para la promoción del trabajo en grupos por equipos Danahy et al. (2014). Así, cada miembro del equipo se puede hacer responsable del desarrollo de una parte específica del trabajo, como el diseño mecánico y desarrollo físico del robot, la programación de los distintos subsistemas de éste, el desarrollo de estrategias y la implementación de comportamientos, validación del software programado, etc.

En los últimos años, el sistema Lego Mindstorms NXT se ha convertido en una de las plataformas más populares para la realización de trabajos prácticos relacionados con el control

*Autor en correspondencia.

Correos electrónicos: giuprog@ai2.upv.es (A. Valera),
ansovi@ai2.upv.es (A. Soriano), mvalles@ai2.upv.es (M. Vallés)
URL: <http://robotica.ai2.upv.es> (A. Valera)

discreto y la mecatrónica. Esto se debe a que el producto ofrece una destacable flexibilidad para desarrollar una gran variedad de proyectos y actividades a través de su unidad de control programable, los sensores, los actuadores y las piezas que lo componen. Otra ventaja que aporta es su precio, ya que esta plataforma, con las piezas de construcción y los dispositivos electrónicos tiene un coste aproximado de 250 euros, mucho más económico que el precio de otras plataformas robotizadas como el Pioneer 3-DX o el Khepera III. Por estos motivos, pueden encontrarse ejemplos de uso de esta plataforma con fines docentes (Kim et al. (2014)), (Danahy et al. (2014)), para la docencia de técnicas de inteligencia artificial (Cuellar and Pegalajar (2014)) o como prácticas de control de flujo, nivel o temperatura (Moor et al. (2013)). Es posible también encontrar trabajos en los que se usa la plataforma LEGO para docencia de disciplinas relacionadas con la programación de sistemas de tiempo real (Bradley Valdenebro et al. (2012)) y de sistemas empujados (Klassner and Continanza (2007)). En (Chin et al. (2009)) se utilizan los Lego para estudiar tareas de control remotas y en (Grega and Pilat (2008)) se usa para la docencia de tareas de control y procesos de tiempo real.

Por otro lado, en los últimos años se ha producido una verdadera explosión en la proliferación de tarjetas embebidas de bajo coste y hardware libre. Con estas tarjetas se pueden planear y desarrollar actividades relacionadas con el control por computador puesto que permiten abordar actividades con un uso de recursos de computación similares a las obtenidas con un PC completo pero con una diferencia sustancial de precio. Las tarjetas Arduino (2014), Raspberry (2014) y BeagleBoard (2014) son ejemplos de ello, y ya actualmente están muy presentes en aplicaciones de robótica e investigación. En Araújo et al. (2014) por ejemplo, se presenta el desarrollo de un robot móvil basado en este tipo de tarjetas.

Las capacidades de computación y comunicación de estas tarjetas unidas a su bajo coste, permiten el desarrollo de laboratorios completos con un mayor número de equipos por alumno, donde sea posible que los alumnos pongan en práctica los conocimientos adquiridos en el aula ((Zachariadou et al. (2012)), (Saleiro et al. (2013))). Además, proponer al alumno que adquiriera el material podría no quedar fuera de lugar dado el bajo coste que ello supone.

Además, como se sabe bien, las plataformas de hardware cerrado usualmente cambian aspectos entre una versión y otra (como por ejemplo los tipos de conexiones o el tipo de comunicaciones entre los dispositivos que venden), lo que dificulta la compatibilidad entre el software nuevo y el hardware antiguo. Esto suele provocar la necesidad de comprar nuevas plataformas para no estancarse tecnológicamente. En este sentido, el hardware libre permite la conexión de cualquier elemento hardware, viejo o nuevo (como por ejemplo motores o sensores provenientes de plataformas cerradas como LEGO), de modo que es posible reciclarlos o aprovecharlos para nuevos proyectos que incluyan este tipo de tarjetas, con el ahorro de coste en material que eso conlleva.

El presente tutorial pretende hacer una revisión de las herramientas u opciones que, bajo la experiencia en los últimos años, se consideran más adecuadas para la docencia y la inves-

tigación de materias como la robótica o el control de sistemas mecatrónicos. Así, se describe tanto el hardware como las posibles herramientas software disponibles a la hora de llevar a cabo su programación y se muestra un conjunto de actividades desarrolladas para este fin.

La organización del artículo es la siguiente: en el apartado 2 se describe la plataforma para el desarrollo y programación de robots de bajo precio LEGO Mindstorms. En el apartado 3 se describen distintos entornos de programación posibles a utilizar con la plataforma LEGO. El apartado 4 realiza una comparativa entre distintas opciones existentes actualmente en el mercado de hardware libre como son Arduino, Raspberry Pi y Beaglebone. En el apartado 5 se muestran distintos demostradores y ejemplos desarrollados para la docencia de distintas materias relacionadas con el control de sistemas mecatrónicos y la robótica. Finalmente, el apartado 6 presenta las conclusiones más interesantes de este trabajo.

2. Plataforma LEGO Mindstorms

En 1998 Lego proporciona el primer conjunto Mindstorms: Robotics Invention System (RIS 1.0). En él, además de las piezas típicas de Lego, el kit proporcionaba motores de corriente continua, sensores y, lo más importante, el RCX. El RCX es el *ladrillo* programable de LEGO que permitía no sólo el movimiento, sino sensorizar y responder al entorno. Está basado en el microprocesador H8 de Hitachi, y proporciona convertidores analógico/ digital, comunicación serie y temporizadores.

El RCX se desarrolló en colaboración entre LEGO y el MIT (Resnick et al. (1996)), (Papert (2000)). En la primera versión el RCX permitía 6 puertos de entrada y de salida, aunque posteriormente se limitaron a los 3 actuales por razones de consumo.

Además de contar con entusiastas que han contribuido a ampliar tanto el hardware como el software de este sistema de varios modos (Baum (2000)), (Baum and Zurcher (2003)), también se han publicado numerosos trabajos basados en esta plataforma en ediciones especiales de revistas como IEEE Robotics and Automation Magazine (Weinberg and Yu (2003)), (Klassner and Anderson (2003)), (Greenwald and Kopena (2003)) o IEEE Control Systems Magazine (Gawthrop and McGookin (2004)).

En enero de 2006 se presentó en el International Consumer Electronics Show la siguiente generación: el Lego Mindstorms NXT. La nueva versión además de otros cambios menores en los sensores electrónicos y las piezas de construcción, incorporaba una unidad de control nueva: el NXT. Esta nueva unidad de control, estaba basada en un potente microcontrolador de 32 bits: ARM7, con 256 Kbytes FLASH y 64 Kbytes de memoria RAM.

Para la programación y las comunicaciones, el NXT está equipado con un puerto USB 2.0 y con un dispositivo inalámbrico Bluetooth clase II, V2.0. Además, también incorpora una pantalla LCD gráfica matricial de 100x64 píxeles, un altavoz de sonido real y 4 botones que permiten una programación simple gracias a un entorno muy intuitivo basado en iconos. Así mismo, el NXT dispone de 4 entradas (una de ellas incluye una

expansión IEC 61158 Type 4/EN 50 170 para usos futuros) y 3 salidas analógicas, por lo que, además de disponer de 1 entrada más que en la versión anterior y dado que los sensores de rotación están completamente integrados en los actuadores eléctricos en esta versión, es posible tener conectados un número mayor de dispositivos sensores.

Al igual que pasaba con la versión anterior, combinando los bloques de construcción, la fácil programación del NXT y su interfaz de entrada y salida se puede obtener un sistema de prototipado rápido para el desarrollo de una gran variedad de actividades, lo que ha permitido que este sistema haya sido ampliamente aceptado como una herramienta para la investigación y la educación universitaria.

En septiembre de 2013 apareció la última generación de Lego: Lego Mindstorms EV3. Como en las versiones anteriores, el kit de Lego incorpora motores de corriente continua (2 grandes y 1 pequeño), sensores electrónicos (de distancia, contacto, color y giroscopo), más de 550 piezas de construcción y, lo que es más destacable, la nueva unidad de control EV3.

La unidad de control está basada en un procesador ARM9 más rápido (Texas Instruments Sitara AM1808, a 300MHz) y con más memoria (64MB RAM y 16MB Flash), lo que le permite por ejemplo, leer los sensores 3 veces más rápido que lo hacía la NXT. La EV3 incorpora 4 puertos de entrada para la lectura de los sensores, 4 puertos de salida para controlar los actuadores eléctricos, 1 puerto USB y una ranura Micro SD que permite leer tarjetas de memoria de hasta 32GB. Estas mejoras permiten por ejemplo instalar y utilizar el sistema operativo Linux y tener conexión wi-fi.

La Figura 1 muestra las unidades de control y los actuadores eléctricos de las 3 versiones de Lego Mindstorms desarrolladas.



Figura 1: Unidades de control RCX, NXT y EV3 y sus respectivos actuadores eléctricos de Lego Mindstorms desarrolladas.

Además de los sensores originales suministrados por Lego, en la actualidad hay una gran variedad de sensores completamente compatibles con las unidades de control. El precio de estos sensores suele ser bastante económico, y se pueden encontrar en empresas como HiTechnic (2014), Mindsensors (2014) o DexterIndustries (2014).

De esta forma se pueden adquirir los típicos sensores de navegación como GPS, brújulas magnéticas o sensores inerciales basados en giroscopos capaces de medir ± 2000 grados por segundo, acelerómetros que miden aceleraciones de hasta 8G en los ejes $X - Y - Z$ y sensores de distancia con una resolución de milímetros. Además, también se puede encontrar una gran variedad de diferentes sensores electrónicos, como cámaras de visión artificial que son capaces de seguir a 8 objetos diferentes a una velocidad de 30fps, sensores térmicos capaces de leer temperaturas desde -70°C hasta 380°C , sensores de presión con un rango máximo de 500kPa (70 PSI), sensores de flexión/deformación y de fuerza, barómetros para medir la presión atmosférica, sensores de presencia, etc.

Cabe destacar que además de los sensores electrónicos también se pueden encontrar accesorios muy diversos, como multiplexores de actuadores y sensores, sistemas de comunicaciones inalámbricas que soportan TCP, HTTP, UDP o módulos de radio de comunicaciones XBee.

3. Entornos de Programación del LEGO Mindstorms

Para el desarrollo de programas con el LEGO Mindstorms, la compañía LEGO proporciona, junto con el hardware, un software gratuito de programación visual mediante bloques basado en LabView. Dada su sencillez y su alto nivel de abstracción, dicho software resulta muy limitado tanto en el acceso a bajo nivel del hardware como en la versatilidad de la programación. Esto ha desembocado por un lado, en el desarrollo de una librería para Labview específica para LEGO Mindstorms (Labview (2014)) y por otro, en la aparición de una gran variedad de alternativas de desarrollo software. Dichas alternativas ofrecen la posibilidad de utilizar lenguajes de programación como *C*, *C#*, *Java*, *Python*, *Ada* o *Lua* para la implementación de aplicaciones en el Lego.

A la hora de escoger entre estas opciones de programación hay que tener en cuenta tres aspectos importantes: el sistema operativo bajo el cual se pretende desarrollar, el entorno de programación necesario para ello y si se requiere un cambio del firmware del robot. Por ejemplo, para la opción de programar en lenguaje *C* o *C#* únicamente es posible trabajar bajo el sistema operativo de pago Microsoft Windows y además para la programación en *C* en concreto se requiere el pago del entorno de desarrollo *RobotC* y un cambio del firmware original del robot al propio de *RobotC*.

Para la selección del lenguaje de programación también es importante tener en cuenta otras consideraciones como por ejemplo si éste permite el manejo de eventos, la programación multitarea, el manejo a bajo nivel de las comunicaciones Bluetooth y los dispositivos I2C, el manejo de ficheros, la programación en coma flotante, la compatibilidad con hardware no oficial de Lego, etc.

Gracias a la amplia experiencia con las distintas opciones de programación, este trabajo se va a centrar en las soluciones que a día de hoy ofrecen mayor versatilidad y aún mantienen una comunidad de usuarios activa por ser las que más utilizadas dentro del campo de la investigación y la mecatrónica, y

que además son pioneras en dar soporte al reciente modelo de LEGO Mindstorms EV3.

3.1. RobotC

Entre las soluciones más recomendables se encuentra RobotC (2014), ya mencionado anteriormente por funcionar solamente bajo Microsoft Windows, requerir una licencia para el entorno de desarrollo y por la necesidad del cambio del firmware del robot.

A pesar de no ser software libre, el precio de una licencia para laboratorio (12 puestos) es muy asequible. En la página web de RobotC se puede descargar una licencia de prueba para 60 días, existiendo además un repositorio de material multimedia muy interesante con múltiples actividades de programación de robots. Dicho material está enfocado a la docencia de la robótica y la programación, donde los profesores pueden encontrar documentos PDF con los capítulos y lecciones, instrucciones de montaje, propuesta de examen con preguntas, cuestiones y problemas para los estudiantes, etc.

El entorno de programación de RobotC ha sido desarrollado en la universidad Carnegie-Mellon, se trata de un entorno de desarrollo integrado (IDE) basado en el lenguaje de programación C, que incluye la verificación sintáctica en tiempo real, una ayuda contextual y el relleno automático de funciones y variables. Además, RobotC permite que después de compilar el programa y descargarlo en la unidad de control del Lego, se pueda utilizar un depurador de código muy completo, permitiendo la ejecución de programas paso-a-paso, el uso de puntos de ruptura, la visualización de los valores de los temporizadores y las variables del programa, el estado de las tareas y otras opciones en tiempo de ejecución.

Hay que recordar que para poder programar en C, el primer paso es realizar el cambio del firmware del robot. En la documentación del programa se encuentra detallado la metodología que hay que seguir según el modelo de Lego que se desee utilizar.

Para el caso concreto de la última versión de Lego, el Mindstorms EV3, hoy en día RobotC tan sólo admite una funcionalidad limitada de sus librerías. No obstante su desarrollo se encuentra en continua evolución para poder ofrecer pronto un acceso total a todas sus librerías.

3.2. LeJOS

Otra de las opciones de programación más destacables es LeJOS (2014), la cual se basa íntegramente en el lenguaje de programación Java. Como es bien conocido, Java provee de una máquina virtual Java que permite ejecutar código compilado Java sea cual sea la plataforma que exista por debajo. Es por eso que esta solución no requiere un sistema operativo concreto para funcionar como por contra ocurre con RobotC.

Además, leJOS es un software totalmente libre por lo que se puede descargar desde su página web en su versión más completa, ofreciendo incluso el código fuente de todas las librerías que implementa. Esto ha facilitado que haya disponible una gran cantidad de proyectos de libre distribución y código abierto por parte de la comunidad desarrolladora.

Junto con las librerías que componen el software de leJOS, se adjuntan herramientas de compilación, depuración de programas, manejo de ficheros dentro del robot, gestión de la conectividad inalámbrica, monitorización en tiempo real, etc. Entre dichas herramientas se encuentra el programa *nxjflash.exe*, el cual se utiliza para llevar a cabo el cambio del firmware del robot, ya que como ocurre con RobotC, es necesario sustituirlo para poder utilizar leJOS.

Las opciones para la programación en leJOS, son las mismas que para cualquier programa en Java. Los dos IDEs de distribución libre más utilizados para Java son Eclipse y NetBeans. No obstante, la comunidad de leJOS ha desarrollado un plugin para Eclipse el cual proporciona algunas funcionalidades directas sobre el robot, como cargar los programas directamente, configurar automáticamente proyectos para el Lego o cambiar el firmware del robot pulsando tan sólo en un icono, entre otras.

Por otro lado, al contrario de como ocurre en otros casos, las librerías de leJOS dan soporte al hardware original de Lego (sensores y motores) y al de compañías como las citadas en el apartado 2 anterior.

En este caso, al aprovecharse de la estructura orientada a objetos de Java, los programadores de los robots pueden trabajar con abstracciones de alto nivel, lo que evita tener que enfrentarse con detalles de bajo nivel como por ejemplo las direcciones hexadecimales de los componentes hardware. De esta forma leJOS no sólo incluye la implementación de controladores tipo PID y filtros de Kalman, sino que se tienen librerías con funciones más abstractas como la navegación, la cartografía, la programación basada en comportamientos, etc.

Para la nueva versión de Lego Mindstorms EV3, leJOS ha conseguido situarse como la opción de desarrollo más completa, ofreciendo hoy en día la total funcionalidad de sus librerías.

3.3. Matlab Simulink

Una última opción a considerar para su programación es la utilización del software Matlab-Simulink proporcionado por la empresa MathWorks (2014). Esta opción ofrece una programación mediante la conexión entre bloques gráficos, lo cual resulta muy intuitivo para alumnos que están iniciándose en la programación o bien no están aún familiarizados con lenguajes de programación.

A pesar de que Matlab no forma parte de la comunidad de software libre, es un software multiplataforma ampliamente implantado en los laboratorios de las universidades y que gracias a las licencias de estudiantes es comúnmente familiar y utilizado dentro del ámbito académico.

MathWorks ha desarrollado un paquete de distribución gratuita para Matlab-Simulink llamado Simulink Support Package for LEGO MINDSTORMS Hardware, el cual se ofrece en dos versiones, una con soporte para la versión NXT y otra para EV3. Estos paquetes incluyen una librería de bloques Simulink para configurar y acceder a los sensores, actuadores e interfaces de comunicación de Lego, lo que permite generar y ejecutar modelos Simulink directamente en el hardware de Lego sin necesidad de cambiar el firmware original al robot. Además, este

sistema permite la monitorización o la generación de gráficas de programas en ejecución o el ajuste de los parámetros de los controladores a partir del mismo esquema de Simulink.


La forma de trabajar con este entorno es la usual de Matlab: primero hay que abrir la librería en el navegador de Simulink para poder acceder a los diferentes bloques que conforman el paquete, generando a continuación el esquema de control de Simulink. Una vez creado sólo queda conectar el Lego al puerto USB y configurar y ejecutar el modelo en el robot. La configuración es muy simple puesto que sólo hay que hacer click en el menú *Tools > Run to Target Hardware > Prepare to Run* y especificar la opción *LEGO MINDSTORMS NXT* o *LEGO MINDSTORMS EV3* en la ventana de configuración de parámetros que aparece. Por último, solo queda ejecutar el programa. Para ello hay que caer click en el botón *Deploy To Hardware* del modelo Simulink.

4. Tarjetas de Control Embebido

Como se ha comentado anteriormente, recientemente el mercado del hardware libre se ha convertido en un área en plena expansión. Existen hoy en día multitud de tarjetas microcontroladoras que debido a su bajo coste y su gran versatilidad han logrado hacerse un hueco dentro de la robótica educacional y de investigación.

Dentro del gran abanico de tarjetas diferentes de hardware libre que han surgido en los últimos años, este artículo se ha centrado en las tres que hoy en día lideran el mercado internacional: Arduino, Rapsberry Pi y BeagleBoard. Es cada vez más frecuente la elección de este tipo de tarjetas para el desarrollo de prototipos hardware o de desarrollo de prácticas de electrónica o programación en laboratorios.

Dado que el enfoque de este trabajo está dirigido al uso de plataformas para la enseñanza o desarrollo dentro de la robótica móvil y la mecatrónica, es estrictamente necesaria la capacidad de controlar motores y sensores desde las propias tarjetas. En la figura 2 se muestra una comparativa esquemática entre las tarjetas embebidas y a continuación se detallan sus características principales así como algunas tarjetas de expansión que facilitan su aplicación dentro del campo de la robótica.



	Arduino Uno	Raspberry Pi	BeagleBone Black
Tarjeta:	Arduino Uno	Raspberry Pi	BeagleBone Black
Precio:	20,99€	29,95€	47,95€
Procesador:	ATmega328	ARM1176JZF5	ARM Cortex-A8
Velocidad del procesador:	16 MHz	700 MHz	1 GHz
Pines analógicos:	6	-	7
Pines Digitales:	14 (6 PWM)	8 Digital GPIO	65 GPIO (8 PWM)
Memoria:	SRAM 2KB EEPROM 1KB	512MB RAM	512MB RAM DDR3L, eMMC 2GB
Lenguajes de Programación:	Arduino IDE, C Variant, Matlab	Cualquiera	Cualquiera
Tarjeta de expansión para motores:	Arduino Motor Shield	Ryanteck Motor Controller Brick Pi	Dual Motor Controller Cape Motor Cape with NXT Connectors

Figura 2: Comparativa entre tarjetas embebidas.

4.1. Tarjeta Arduino

El microcontrolador Arduino se inició en el año 2005 como un proyecto para estudiantes del instituto IVREA en Italia. Es un tipo de controlador de código abierto y software de programación libre, muy versátil y asequible. Cuenta con multitud de formatos (versión Uno, Mega, Mini, etc.), que se diferencian principalmente por la capacidad de la memoria, por el número de entradas y salidas disponibles, por la frecuencia del procesador o por las diferentes opciones de comunicación que ofrecen.

El microcontrolador de Arduino está basado en el chip Atmega (versiones Atmega168, Atmega328 y Atmega1280), y en la mayoría de modelos su voltaje de trabajo se comprende en un rango de 7 a 12V DC, aunque también se puede alimentar con una fuente de voltaje de 5V DC estabilizada. La tarjeta ofrece una serie de entradas analógicas en base a 5V digitalizadas mediante un conversor AD de 10 bits, lo que proporciona una resolución en la lectura de 4.9mV con una velocidad de lectura máxima aproximada de 10000 lecturas por segundo. En cuanto a las salidas, Arduino ofrece salidas digitales en base a 3.3V o 5V según el modelo de tarjeta y salidas de tipo PWM que generan una onda cuadrada estable con un determinado ciclo de trabajo especificado por el programador y con una frecuencia de aproximadamente 490 Hz.

Mediante el uso de estas salidas y su comunicación con un sencillo esquema eléctrico es posible llevar a cabo diversas tareas de control de motores. Existen algunas tarjetas de bajo coste que facilitan la integración de motores con cualquier dispositivo que pueda proporcionar pulsos de 5V. como *easy driver* o *big easy driver* (Schmalzhaus (2014)). No obstante la misma empresa Arduino ha desarrollado una serie de tarjetas de expansión que se acoplan directamente sobre las tarjetas originales y que ofrecen distintas funcionalidades como comunicaciones WiFi, Ethernet, control de servos, control de motores, etc. En concreto, para el control de motores podemos destacar entre otras *Arduino Motor Shield*, la cual implementa un puente en H para el control de como máximo dos motores de corriente continua. La instalación de la tarjeta es puramente física y a la hora de su programación no es necesario la importación de ninguna librería adicional para utilizarla.

Debido a su bajo coste y su política de hardware libre, en pocos años Arduino ha conseguido extenderse ampliamente dentro de la comunidad investigadora y científica. Formando parte como cerebro de distintos proyectos como desarrollos de impresoras 3D (Evans (2012)), creación de robots (Al-Busaidi (2012)), gestión de redes (Faludi (2010)), investigación en vehículos submarinos (Busquets et al. (2012)), e incluso en sus versiones más ligeras, en vehículos aéreos no tripulados (Lim et al. (2012)).

Arduino se puede programar de manera rápida y sencilla, a través de su propio Arduino IDE disponible para las plataformas Windows, Linux y Mac OS de manera totalmente gratuita desde su web. Tan sólo es necesario instalar los drivers para el puerto FTDI que incorpora la placa y conectarla por USB. El entorno del Arduino IDE está basado en Processing, es un editor de texto simple que permite escribir el código, contiene un área de mensajes, una consola de texto útil para depurar y verificar el código y una barra de herramientas con iconos para

el rápido acceso a las funciones más comunes como compilar el código, permite descargar el programa a la placa, la creación, apertura y guardado de programas, la selección del puerto de comunicaciones, etc. Arduino está basado en el lenguaje de programación *C* y soporta algunas funciones del estándar *C* y de *C++*. No obstante, debido a que la transmisión de datos entre el PC y la tarjeta es serie, es posible llevar a cabo la programación desde cualquier lenguaje que ofrezca este soporte, desde *Java*, como ocurre por ejemplo con la librería integrada para la programación de Arduino en el software Easy Java Simulations (EJS (2014)), hasta otras librerías comerciales como es el caso de Matlab. Para ello, MathWorks ofrece un paquete gratuito para su software de programación mediante bloques Simulink. Dicho paquete ofrece una serie de bloques diseñados específicamente para Arduino de modo que es posible implementar el esquema deseado en Matlab-Simulink y transferirlo a la placa Arduino de una forma totalmente transparente para el programador, despreocupándose de la conversión entre los bloques de Simulink y el código para Arduino.

4.2. Tarjeta Raspberry Pi

Raspberry PI es una iniciativa de una compañía británica para crear un micro ordenador muy económico (alrededor de 30€) y lo suficientemente flexible para ser utilizado para distintos usos. Desde su aparición en 2012, el objetivo de su fundación se ha centrado en promover su uso en el campo académico, tanto a nivel de colegios, como a nivel universitario para proyectos de investigación. Su diseño incluye un Broadcom BCM2835 con un procesador ARM1176JZFS a 700 MHz con unidad de coma flotante, 512 MB de memoria RAM y un procesador gráfico (GPU) Videocore IV capaz de mover contenidos con calidad Blu-ray y con soporte para las librerías OpenGL y OpenVG. La tarjeta arranca desde una tarjeta de memoria SD y posee dos conectores USB 2.0, un conector de Ethernet RJ-45, Salida de vídeo Digital HDMI y salida de vídeo analógico S-Video, un conector GPIO, un conector de alimentación y una salida de audio analógica.

Lo más destacable en relación a su uso en el campo de la robótica además de su tamaño y su precio, es sin duda el conector GPIO formado por 26 pines, todos reconfigurables excepto los de alimentación y masa. Por defecto hay una serie de pines configurados para la conexión de interfaces UART, I2C y SPI y el resto están configurados para entradas y salidas digitales. Los pines son compatibles con niveles de 3.3V y como máximo la corriente que puede circular por cada pin es de 50mA, por tanto si se desea conectar dispositivos que funcionen a 5V o que consuman una corriente mayor, se debe hacer a través de un Buffer o cualquier otro circuito que adapte los niveles de tensión y de corriente. Hay que tener en cuenta que el puerto GPIO no está protegido ante cortocircuitos y sobretensiones.

Del mismo modo que ocurre con la tarjeta Arduino, el amplio uso a nivel internacional y multifuncional de la Raspberry Pi ha impulsado el desarrollo de múltiples tarjetas de expansión que añaden distintas funcionalidades como visualización en LCD, integración de cámaras de visión, compatibilidad con productos de Arduino, etc. Dos tarjetas de expansión destacables para su uso en el campo de la robótica son los casos de la

Ryanteck Motor Controller y la Brick Pi. La tarjeta Ryanteck Motor Controller distribuida por la empresa Adafruit, es un kit auto-ensamblable que permite controlar hasta dos motores de corriente continua con facilidad a través de los puertos GPIO. Por otro lado, la tarjeta de expansión Brick Pi desarrollada por Dexter Industries, aporta la integración directa a los sensores y motores del Lego Mindstorm a la Raspberry Pi, ofreciendo 9 puertos I2C para un máximo de 4 motores y 5 de sensores. La misma empresa ofrece una librería de código libre para desarrollar con la Brick Pi.

La Raspberry Pi soporta distintos sistemas operativos basados en arquitecturas ARM con núcleo Linux como Pidora (versión optimizada de Fedora) o Raspbian (basado en distribución Debian) e incluso hay una versión compatible para Android en fase de desarrollo.

La Fundación de Raspberry Pi recomienda Python como lenguaje de programación para el desarrollo de aplicaciones, no obstante se puede utilizar cualquier lenguaje compilable para ARMv6 como *C*, *C++*, *Java*, *Ruby* e incluso *Scratch*, los cuales están soportados por defecto en la tarjeta. Recientemente MathWorks también ha desarrollado una librería específica de distribución libre para la programación de la Raspberry Pi, la configuración y el acceso a los periféricos de entrada/salida mediante el software Matlab-Simulink. Desde su página web es posible acceder a sencillos tutoriales de iniciación. Una vez desarrollado el esquema en Simulink, el programa se transmite a la tarjeta mediante TCP/IP especificando la dirección IP de la tarjeta y el puerto, que por defecto es 17725. Por tanto no es necesario ningún software adicional o programar nada desde el interior de la tarjeta, tan sólo se requiere que el ordenador desde el que se programa esté conectado a la misma red que la Raspberry Pi.

4.3. Tarjeta BeagleBoard

La tarjeta BeagleBoard producida por la empresa Texas Instruments en asociación con Digi-Key y Newark element14, surge a mediados de 2008 como el primer micro ordenador de bajo coste del mercado. Desde su aparición ha ido evolucionando mediante modelos nuevos que incorporan ciertas mejoras en rendimiento y conectividad. Hoy en día el modelo más reciente es el BeagleBone Black, cuyo lanzamiento fue en abril de 2013. Este modelo ofrece un rendimiento y una capacidad de cómputo que pocas tarjetas de precios similares son capaces de superar. La BeagleBone Black tiene un precio de menos de 50€, y posee un procesador ARM Cortex-A8 a 1000MHz, una GPU PowerVR SGX530 a 200MHz, viene con 512MB de RAM de tipo DDR3, una conexión Ethernet, salida de vídeo y audio micro-HDMI, un puerto USB, una ranura para tarjeta microSD y dos filas de 46 pines de entrada o salida. Hay que tener en cuenta que el voltaje de entrada de la tarjeta es de 5V y que sin conectar nada, la tarjeta sola consume entre 210 y 460mA.

Otra de las ventajas más destacables de la BeagleBone Black es la conectividad que ofrece. Dispone de un total de 92 pines reconfigurables que incluyen hasta cuatro puertos serie diferentes, la generación de ocho PWM independientes, cuatro temporizadores, un bus CAN, siete entradas analógicas en base a 1.8V con una resolución del convertidor AD de 12 bits, varias salidas

de alimentación a 5 y 3.3V, dos conexiones SPI, dos puertos I2C para su interconexión con cualquier tipo de hardware compatible, además de 65 entradas y salidas digitales reconfigurables.

Las extensiones para las tarjetas Beagleboard se denominan capes y hay más de 40 modelos diferentes. Del mismo modo que con el resto de tarjetas, su funcionalidad es otorgar mayor facilidad de interconexión y/o dotar de nuevas características a la tarjeta. Para el control de motores en concreto, una de las capes más utilizadas para el control de motores es la *Dual Motor Controller Cape (DMCC) Mk.6*, la cual permite el manejo de dos motores DC (de 5V a 28V y de hasta 7A por motor) y ofrece una librería completa en el lenguaje de programación C para su desarrollo. Otra cape interesante es la *BeagleBone Motor Cape with NXT Connectors*, que aunque tan sólo es compatible con el modelo BeagleBone (no Black) como su propio nombre indica, añade la capacidad de conexión directa de los motores con conexiones I2C como son los del Lego NXT y el EV3.

Como ocurre con la Raspberry Pi, la BeagleBone Black soporta diversos sistemas operativos basados en la arquitectura ARM Linux como Fedora, Ubuntu u openSUSE, pero también tiene soporte para Android e incluso para Windows Embedded. Estos sistemas se ejecutan desde la microSD aunque cabe mencionar que la tarjeta dispone de un kernel de auto arranque Linux en su memoria eMMC de 2GB que se ejecuta cuando no se inserta ninguna microSD.

Las opciones de programación de la BeagleBoard son prácticamente las mismas que las de Raspberry Pi. Se pueden utilizar lenguajes como C, C++, Java, Python, JavaScript, Android e incluso también Matlab-Simulink. En este caso, MathWorks tan sólo da soporte al modelo de tarjeta BeagleBoard xM mediante la librería BeagleBoard Support from Simulink. En este caso, tampoco es necesaria la instalación o desarrollo de software dentro de la tarjeta BeagleBoard, no obstante Simulink requiere una conexión serie y una conexión de Ethernet desde el ordenador donde se desarrolla a la tarjeta. En la web de Mathworks se exponen diversos tutoriales y ejemplos para aprender a configurar y utilizar este software.

4.4. Otras tarjetas embebidas

Como se ha comentado anteriormente, la continua evolución de este tipo de tarjetas embebidas de bajo coste requiere estar atento a las nuevas posibilidades de productos que van surgiendo. Es por ejemplo el caso de la reciente versión Intel Galileo (Fig. 3 izquierda) desarrollada a partir de una colaboración entre Arduino y la empresa Intel, o la anunciada tarjeta Arduino TRE (Fig. 3 derecha), otra colaboración de Arduino con Texas Instruments y la Fundación BeagleBoard. Esta última tarjeta se estima que saldrá al mercado a finales de este año.

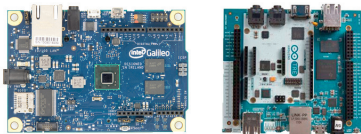


Figura 3: Tarjeta Intel Galileo (izquierda) y tarjeta Arduino TREE (derecha).

5. Demostradores y Ejemplos de Aplicación

Una vez se han descrito las posibles soluciones hardware/software de bajo coste, a continuación se van a mostrar algunas de las aplicaciones que se han desarrollado y utilizado. Estas actividades avalan la utilidad de este tipo de plataformas para la impartición de diferentes contenidos relacionados con la robótica y el control por computador.

5.1. Identificación de Motores de Corriente Continua

Antes de abordar el diseño o ajuste de controladores simples de velocidad o posición que permitirán la obtención de controladores más avanzados (basados en la cinemática del robot) es necesario obtener la función de transferencia de los motores que se van a utilizar. Por ello, una de las primeras actividades que se pueden desarrollar es obtener la identificación de los motores de corriente continua del Lego.

A continuación se muestra la parte más destacable de un programa de RobotC para la unidad de control del Lego utilizado para la identificación en posición de uno de los motores del Lego.

```
1  const string  sFileName = "iden-pos.txt";
   task main()
2  {
3      long i = 0;
4      long niteraciones;
5      float tfinal=5.0;
6      float ts=0.05; //p. muestreo: 50ms
7      float pulsos2rad=PI/180.0;
8      float pos_rad=0.0;
9      float acontrol;
10     createTextFile(sFileName, 30000);
11     nMotorEncoder[motorA] = 0;
12     niteraciones=(int)((tfinal)/ts);
13     i=0;
14     while(i < niteraciones)
15     {
16         // lectura posición motor A
17         pos_rad=nMotorEncoder[motorA]*pulsos2rad;
18         acontrol=60;
19         // proporcionar acción de control al motor
20         motor[motorA] = acontrol;
21         writeFloatNumber(pos_rad);
22         writeFloatNumber(ontrol);
23         wait1Msec(ts*1000);
24         i++;
25     }
26 }
```

El motor que se va a identificar se conecta en el puerto de entrada A de la unidad de control. Además, puesto que cada motor está equipado con un encoder incremental, es necesario inicializar su valor al principio de la ejecución de cada programa. Eso se consigue con la instrucción *nMotorEncoder[]*, escribiendo un determinado valor (0 en este caso).

A cada iteración del bucle de control, se obtiene mediante el comando *nMotorEncoder[]* la posición del eje del motor (expresada en radianes), y se suministra la acción de control al motor mediante el comando *motor[]*. La posición y la acción de control se almacenan en un fichero de texto (*iden-pos.txt*). Hay que tener en cuenta que el encoder de los motores de los Lego tiene una resolución de 1 pulso por grado, por lo que se utiliza una constante (*pulsos2rad*) para transformar el valor del encoder (en pulsos) a un ángulo expresado en radianes.

Para poder verificar del periodo de muestreo de cada iteración del bucle se tienen varias opciones. La más simple es

utilizar las típicas instrucciones de retardo (`wait1Msec()`) o, si lo permite el firmware de la unidad de control, mediante el uso de temporizadores.

Después de la ejecución del programa (5 segundos, en este caso), lo único que hay que hacer es transferir el fichero de datos desde la unidad de control al PC. En el caso de RobotC lo único que hay que hacer es conectar un cable USB entre el ladrillo inteligente y el computador, y utilizar las facilidades del manejo de ficheros de dicho entorno.

Una vez el fichero en el PC sólo resta analizar la respuesta obtenida. En nuestro caso esto se hace con la aplicación de Matlab, de forma que después de cargar el fichero y tener la variable en el entorno de trabajo, se puede utilizar cualquier comando y herramienta de Matlab para realizar la identificación.

La función de transferencia de posición de los motores obtenida es:

$$G_{pos}(s) = \frac{\theta(s)}{U(s)} = \frac{0,1362}{s(1 + 0,073s)}$$

Con cualquier otro lenguaje y unidad de control, la identificación se realiza de un modo completamente análogo. La única diferencia reside en el nombre de los comandos y la forma de transferir el fichero de datos. Por ejemplo, si se utiliza como unidad de control la tarjeta Raspberry Pi con la tarjeta BrickPi, y aunque se puede utilizar 2 lenguajes diferentes (C y Python), los comandos son iguales: `BrickPi.MotorSpeed[]` para proporcionar las acciones de control y `BrickPi.Encoder[]` para obtener la posición del eje del motor. Si por otro lado, se utiliza el firmware leJOS, el comando para acciones del control es `setPower(int power)` y para obtener la lectura del encoder se utiliza la función `getTachoCount()`.

En el caso de la identificación en velocidad del motor de corriente continua hay que tener en cuenta que los motores de Lego no disponen de un sensor que mida directamente la velocidad. Por ello, la forma más simple de obtener dicha magnitud es la de utilizar una aproximación matemática a partir de los datos de posición. En este caso es necesario precisar que si se realiza una aproximación simple de primer orden los resultados obtenidos no serán demasiado buenos puesto que la señal suele estar afectada de un ruido muy alto.

La función de transferencia en velocidad obtenida de los motores es:

$$G_{vel}(s) = \frac{\dot{\theta}(s)}{U(s)} = \frac{0,1361}{1 + 0,097s}$$

Las figuras 4 y 5 muestran los datos experimentales de posición y velocidad, y las salidas simuladas a partir de los modelos identificados.

5.2. Control de Motores de Corriente Continua

A partir de los datos de posición y velocidad del motor del Lego y/o de las funciones de transferencia obtenidas se puede proceder al diseño y la implementación de diversos controladores. Para la obtención de los controladores en este trabajo se ha utilizado el método basado en Ziegler-Nichols. Para ello se deben obtener primero los parámetros significativos del proceso,

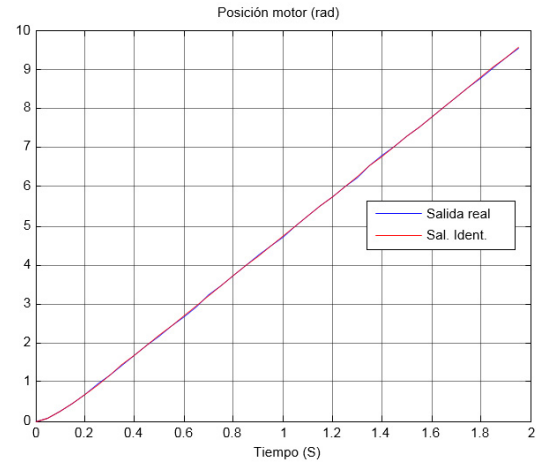


Figura 4: Respuesta a un escalón en posición del sistema real y del identificado.

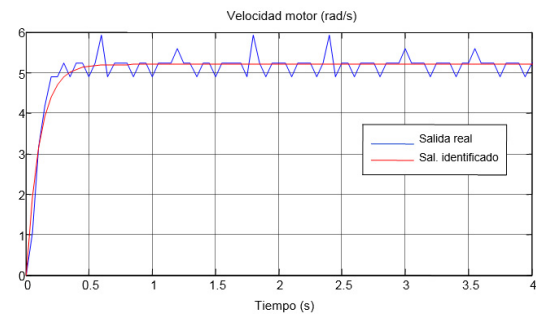


Figura 5: Respuesta a un escalón en velocidad del sistema real y del identificado.

de forma que a partir de ellos se realiza el ajuste de los controladores.

Para el ajuste de los controladores de posición se ha utilizado el método de la respuesta sostenida. Mediante este método se cierra el bucle con un regulador proporcional y se va incrementando el valor de la ganancia hasta que la respuesta del sistema se hace críticamente estable. Los parámetros significativos que se obtienen son 2: K_c (constante de repetición) y T_c (periodo de repetición de la respuesta del sistema).

Para el ajuste de los controladores de la velocidad del motor se ha utilizado el método de la respuesta a un escalón. Se trata de introducir un escalón a la entrada del sistema y analizar la respuesta del sistema. En este caso se obtienen 3 parámetros: K (ganancia del sistema), T_p y T_0 (tiempos obtenidos a partir de la constante de tiempo del sistema). La Tabla 1 muestra los parámetros obtenidos para los motores de Lego:

Tabla 1: Parámetros significativos motores Lego.

Respuesta sostenida		Respuesta ante escalón		
K_c	T_c	K	T_p	T_0
319.00	0.3000	0.1361	0.0713	0.0362

A partir de estos parámetros se pueden ajustar directamente los diferentes valores de los controladores de posición y velocidad de los motores de corriente continua. A continuación se muestra la parte más destacable de un algoritmo de RobotC para establecer el control proporcional de la velocidad de un motor conectado en el puerto A de la unidad de control.:

```
1 const string sFileName = "esc_velP.txt";
2 task main()
3 {
4     long i = 0;
5     long niteraciones;
6     float tfinal=10.0; //tiempo ejecución: 10seg.
7     float refVel=2*PI*2.0; //ref: 2 vueltas por seg.
8     float ts=0.05; //p. muestreo: 50mS
9     float pulsos2rad=PI/180.0;
10    float pos_rad_actual=0.0;
11    float pos_rad_anterior=0.0;
12    float vel_rueda=0.0;
13    float acontrol;
14    float error_vel=0.0;
15    float q0=9.43; //parámetro controlador P
16    createTextFile(sFileName, 30000);
17    nMotorEncoder[motorA] = 0;
18    niteraciones=(int)((tfinal)/ts);
19    i=0;
20    while(i < niteraciones)
21    {
22        // lectura posición motor A
23        pos_rad_actual=nMotorEncoder[motorA]*pulsos2rad;
24        // estimación velocidad motor
25        vel_rueda=(pos_rad_actual-pos_rad_anterior)/ts;
26        // calculo error velocidad y acción de control
27        error_vel=refVel-vel_rueda;
28        acontrol=error_vel*q0;
29        // proporcionar acción de control
30        motor[motorA] = acontrol;
31        writeFloatNumber(refVel);
32        writeFloatNumber(vel_rueda);
33        writeFloatNumber(ontrol);
34        pos_rad_anterior= pos_rad_actual;
35        wait1Msec(ts*1000);
36        i++;
37    }
38 }
```

La estimación de velocidad del motor en el instante k se obtiene a partir de la posición actual (instante k) y la posición de la iteración anterior $k - 1$. El error de velocidad se calcula restando a la referencia ($refVel$) dicha estimación. A continuación se calcula la acción de control multiplicando la constante proporcional (q_0) por dicho error. Por último se suministra la acción de control mediante el comando `motor[]` y se almacenan los datos en un fichero de texto para poder realizar, una vez transferido desde la unidad de control a un PC, su análisis mediante (por ejemplo) Matlab.

Por supuesto, adaptar este algoritmo a otras tarjetas de control y/o lenguajes de programación es una tarea muy simple y directa.

La figura 6 muestra la respuesta real y la referencia de la posición del sistema con un regulador proporcional. La figura 7 muestra la velocidad real y la referencia del motor con un regulador proporcional-integral.

En la figura 6 se puede apreciar como las respuestas de los controladores P y PD tienen un error constante de seguimiento ante la referencia tipo rampa introducida al motor. Para eliminar dicho error se necesita un controlador que incorpore una acción integral. También se puede apreciar como la incorporación de un término derivativo mejora el comportamiento dinámico puesto que proporciona una sobreoscilación y un tiempo de posicionado menor.

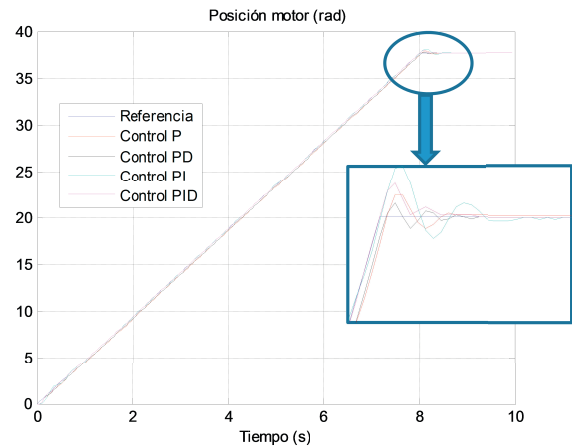


Figura 6: Respuesta control de posición motores Lego.

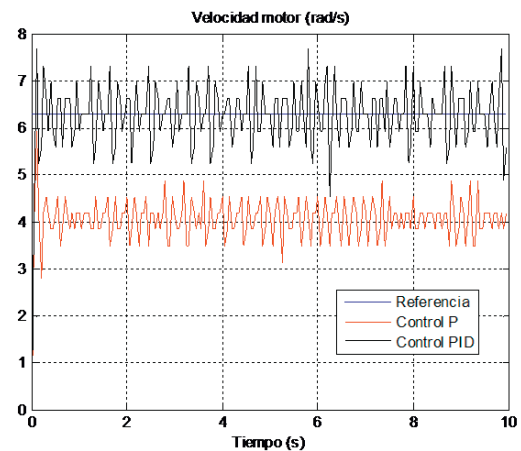


Figura 7: Respuesta control de velocidad motores Lego.

La figura 7 muestra la referencia de velocidad (valor constante 2π) y la respuesta de los controles P y PID. Se puede apreciar que el control proporcional tiene un error constante de seguimiento ante el escalón. Si se desea eliminar dicho error es necesario incorporar una acción integral.

5.3. Filtrado de Señales

Como se ha comprobado en la figura 7, cuando se aborda el control de velocidad de los motores de Lego aparece un ruido muy evidente que provoca que la respuesta del sistema no sea buena. Dicho ruido está motivado básicamente porque al carecer los motores de sensores de velocidad, se ha obtenido una estimación de la velocidad mediante una aproximación numérica de primer orden a partir de la posición.

Para tratar de evitar este fenómeno se puede abordar diferentes soluciones como por ejemplo el diseño de filtros. Para el diseño del filtro se pueden proponer algunas actividades muy interesantes. La primera podría ser diseñar e implementar en el

bucle de control un filtro digital que realiza un procesamiento previo de la información antes del cálculo del error de velocidad. En este caso se han planteado diferentes filtros pasa-bajo IIR, tanto de diseño indirecto (a partir de la discretización de filtros prototipo analógicos) como de diseño directo a partir de la aproximación de mínimos cuadrados.

Como alternativa a estos filtros pasa-bajo se puede diseñar un Filtro de Kalman (KF). Como es bien sabido, el KF proporciona una herramienta computacional para estimar el estado de un proceso, de forma que minimiza la media cuadrática del error estimado del proceso. Su objetivo es estimar el estado x_k del sistema basado en el conocimiento de las dinámicas de éste, las características de la perturbación (matrices de covarianza) y la disponibilidad de medidas ruidosas z_k , de manera que el filtro minimizará los efectos de w_k y v_k en la estimación del estado.

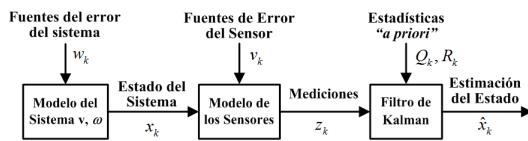


Figura 8: Esquema modular del funcionamiento del Filtro de Kalman.

El algoritmo del KF requiere de dos etapas o estados:

- *Actualización del tiempo (o Predictor)*: calcula la siguiente estimación del estado a priori utilizando la estimación previa del estado y del valor actual de las entradas.

- *Actualización de medidas (o Corrector)*: utiliza la medida actual para refinar el resultado dado por la etapa anterior para obtener una estimación a posteriori mejorada.

La Figura 9 muestra uno de los KF implementados y ejecutado en la unidad de control de Lego. En este caso se trata de un filtro que mejora la estimación de la velocidad lineal de los motores a partir de la medida de los encoders. Se puede apreciar fácilmente como mediante la implementación del KF se ha podido eliminar el ruido de la estimación de la velocidad, siendo esto una cuestión crítica puesto que estas velocidades se utilizarán para obtener la estimación de la posición global y la orientación del robot.

5.4. Control Cinemático de Robots Móviles

Una vez desarrollado el control de posición y velocidad de los motores de corriente continua del sistema se puede abordar el control de movimiento de los robots móviles. Este control cinemático consiste en determinar las acciones necesarias para llevar al robot desde su posición actual a una posición final deseada, considerando para ello las velocidades y la orientación del robot.

En el control de robots móviles se pueden abordar dos problemas: el control de la trayectoria y el control del camino. En el primer caso se dispone de la trayectoria deseada, que es la curva temporal para cada una de las coordenadas sobre las que

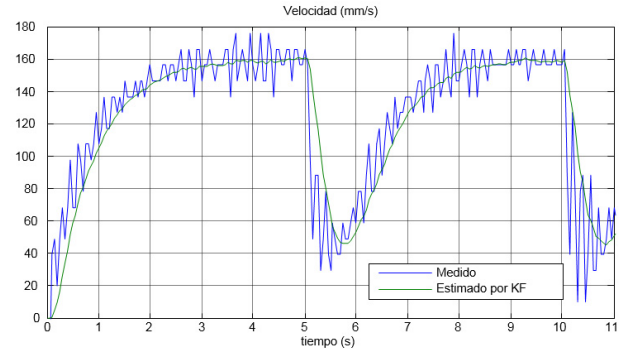


Figura 9: Velocidad y estimación mediante KF.

se debe realizar un control del robot. En el caso del control del camino la diferencia radica en que no se tiene en cuenta el factor tiempo, sino que únicamente se debe proporcionar la información geométrica en el espacio cartesiano.

Para el control de movimiento del robot es necesario tener en cuenta la configuración cinemática de éste. En el caso de Lego se recomienda o bien la configuración diferencial o (en menor medida) la configuración Ackerman.

La configuración diferencial es muy fácil de implementar con las piezas de construcción de Lego. Se trata de una configuración donde el movimiento del robot se establece a partir de la diferencia de velocidades de la rueda derecha e izquierda. Por ello, en esta configuración se deben establecer dos controles de velocidad.

En la configuración Ackerman el control del robot se realiza a partir de la orientación de las ruedas directrices y de la velocidad de avance de las ruedas motrices. Por ello en esta configuración se necesitan dos clases de controles: un control de posición y un control de velocidad.

De esta forma, el control de movimiento de los robots móviles tendrá 2 niveles. En el nivel más interno se tendrá un control de posición y/o velocidad que calcule las acciones de control necesarias a partir de las referencias y de la posición y velocidades actuales de las ruedas del robot.

El nivel más externo se encargará de comparar la trayectoria o camino deseado con la posición y orientación actual del robot, calculando las referencias para el control interno del robot.

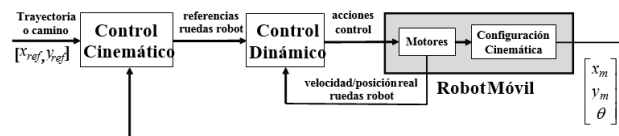


Figura 10: Arquitectura control cinemático de robots móviles.

A continuación se muestra la parte más importante de un programa de RobotC que establece el control de trayectoria por punto descentralizado para un robot móvil con configuración diferencial.

```

//definición del fichero de texto
2 const string sFileName = "ctrl_cine_P.txt";
task main() //tarea principal
4 {
    long i=0;
    float tfinal=30.0;
    float vel_ang_d, vel_ang_i;
    float vel_lin_d, vel_lin_i;
    float vref_d, vref_i;
    float wref_d, wref_i;
    float q0=9.4276
    float theta=6.2832; // orientación inicial del robot
    float x=-0.05; // coordenada inicial X del robot
    float y=-0.05; // coordenada inicial Y del robot
    float b=0.02; // separación entre las ruedas)/2: 20mm
    float e=0.056; // distancia punto descentrado: 56 mm
    float pul2rad=0.0174533; // pul2rad=2*pi/360
    float radiorueda=0.028; // radio de rueda es 28 mm
    float Ts=0.05; // periodo muestreo: 50mS
    nMotorEncoder[motorA]= 0; // reset encoder rueda dcha.
    nMotorEncoder[motorB]=0; //reset encoder rueda izqda.
    createTextFile(sFileName, 30000);
    //inicialización iteraciones
    i=0;
    niteraciones=(int)((tfinal)/Ts);
    //bucle de control
    while(i < niteraciones)
    { // cálculo referencias pos. y vel. tray. cuadrada
      ini_ref_cuadrado();
      // lectura de encoders de las ruedas
      ruedaAnue=nMotorEncoder[motorA];
      ruedaBnue=nMotorEncoder[motorB];
      // cálculo velocidades angulares (rad/s) de las ruedas
      vel_ang_d=pul2rad*(ruedaAnue-ruedaAant)/Ts;
      vel_ang_i=pul2rad*(ruedaBnue-ruedaBant)/Ts;
      // cálculo velocidades lineales (m/s) ruedas
      vel_lin_d=vel_ang_d*radiorueda;
      vel_lin_i=vel_ang_i*radiorueda;
      // cálculo velocidad lineal del robot
      velLin_robot=(vel_lin_d+vel_lin_i)/2;
      // cálculo la velocidad angular del robot
      velAng_robot=(vel_lin_d-vel_lin_i)/(2*b);
      // cálculo posición X-Y y la orientación del robot
      x=x+velLin_robot*Ts*cos(theta);
      y=y+velLin_robot*Ts*sin(theta);
      theta=theta+velAng_robot*Ts;
      // cálculo expresión control cinemático robot
      xpunto=kvel*velxref+kpos*(refx-(x+e*cos(theta)));
      ypunto=kvel*velyref+kpos*(refy-(y+e*sin(theta)));
      // cálculo modelo cinemático inverso del robot
      ecos_theta=e*cos(theta);
      esin_theta=e*sin(theta);
      bsin_theta=b*sin(theta);
      bcos_theta=b*cos(theta);
      vref_d=((ecos_theta-bsin_theta)*xpunto+(esin_theta+
        bcos_theta)*ypunto)/e;
      vref_i=((ecos_theta+bsin_theta)*xpunto+(esin_theta-
        bcos_theta)*ypunto)/e;
      // cálculo velocidades angulares de referencia
      wref_d=vref_d/radiorueda;
      wref_i=vref_i/radiorueda;
      // cálculo errores de la velocidad angular de las ruedas
      error_vel_d=wref_d-vel_ang_d;
      error_vel_i=wref_i-vel_ang_i;
      // cálculo acciones de control (ctrl P)
      acontrolA=error_vel_d*q0;
      acontrolB=error_vel_i*q0;
      // proporcionar acciones de control a motores
      motor[motorA] = acontrolA;
      motor[motorB] = acontrolB;
      // almacenar valores encoder para la próxima iteración
      ruedaAant=ruedaAnue;
      ruedaBant=ruedaBnue;
      // escribir valores en fichero de datos
      writeFloatNumber(refx);
      writeFloatNumber(refy);
      writeFloatNumber(x);
      writeFloatNumber(y);
      writeNewLine();
      // verificar periodo de muestreo
      wait1Msec(Ts*1000);
      i++;
    }
}

```

A cada iteración del bucle de control se debe calcular primero las velocidades angulares de las ruedas derecha (conectada en el puerto A) e izquierda (conectada en el puerto B). Esto se hace a partir de la estimación con los valores de las posiciones obtenidos a partir de los encoders.

Considerando el modelo cinemático diferencial del robot, a partir de las velocidades angulares se calcula la velocidad angular y lineal del robot, obteniendo posteriormente una estimación de la posición $X-Y$ y de la orientación θ del robot, de forma que se puede calcular a continuación el control cinemático del robot.

A continuación, considerando el modelo cinemático inverso del robot, se puede calcular las velocidades lineales de referencia para las ruedas, y a partir de éstas, las velocidades angulares de referencia.

Una vez se tienen las velocidades angulares de referencia se puede establecer el control abordado en el apartado 5.2. Aunque se podría plantear un control más complejo, en este algoritmo se ha implementado un control proporcional.

Lo único que falta es proporcionar las acciones de control a los motores, escribir los valores en un fichero de texto y verificar el periodo de muestreo.

La figura siguiente muestra la trayectoria de referencia (un cuadrado de 1m de lado) y la seguida por un robot Lego con configuración diferencial. La posición inicial de éste era (-0.01,-0.01), y se puede observar como el robot sigue de una forma muy buena a la referencia.

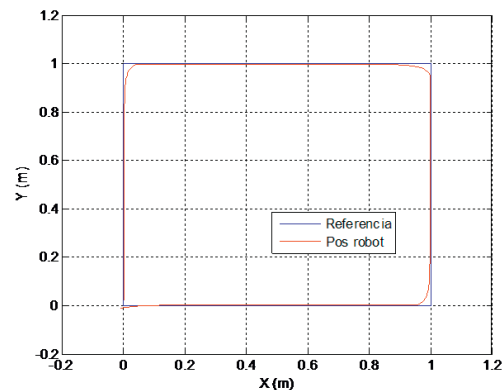


Figura 11: Respuesta control cinemático robot diferencial Lego.

5.5. Control de Robots con Sensores Avanzados

Como se ha comentado anteriormente, además de los sensores originales suministrados por Lego, existe una gran variedad de sensores completamente compatibles con las unidades de control (ladrillos inteligentes). Gracias a estos dispositivos electrónicos se puede abordar el desarrollo de una gran variedad de actividades basadas en robots móviles, como por ejemplo:

- *Robots limpiadores*: los alumnos deben desarrollar un robot que sea capaz de limpiar de objetos una determinada zona en el menor tiempo posible. Para detectar los objetos se utiliza

un sensor de distancia, y para detectar los límites del área, un sensor luz.

- *Robots luchadores de sumo*: basándose en la actividad anterior, es bastante rápido generar un robot que pueda competir al sumo. En este caso los alumnos pueden utilizar, además de los sensores anteriores, sensores de contacto.

- *Robots jugadores de fútbol/Rugby*: es este caso se puede utilizar una cámara de visión artificial para detectar la pelota y una brújula para detectar la línea de marcación.

- *Generación de convoyes de vehículos*: utilizando también la cámara, se puede programar un robot que debe seguir al robot que le precede. De esta forma únicamente el primer robot debe tener programada la trayectoria de movimiento, mientras que el resto de vehículos se deben encargar de mantener una distancia constante con el que le precede.

Para ello lo único que hay que hacer es configurar los sensores indicando el puerto de entrada y el tipo de sensor conectado. Si se utiliza el entorno de RobotC con el controlador original de Lego, esto se puede hacer mediante la ventana *Motors and Sensors Setup* del menú *Robot* de dicho entorno. Si por el contrario se utiliza la unidad de control BrickPi, la configuración de los sensores se realiza con el comando `BrickPi.SensorType[número_puerto]`, tanto si se programa con C como con Python. En caso de utilizar el firmware leJOS existe una clase para cada tipo de sensor. Para el sensor de luz, por ejemplo, su clase es *LightSensor*, cuyo constructor recibe el puerto al que está conectado (`new LightSensor(SensorPort.S1)`).

La lectura de los valores de los sensores se realiza invocando un comando `SensorValue(nombre_sensor)` en el caso de Robot C, `BrickPi.Sensor[numero_puerto]` en el caso de BrickPi y mediante el método definido en cada clase de tipo sensor `.readValue()` en el caso del firmware leJOS.

La figura siguiente muestra unas fotografías de distintas actividades relacionadas con estos sensores avanzados llevadas a cabo por los alumnos.



Figura 12: Diferentes actividades desarrolladas por los alumnos con los equipos de Lego.

En A.Valera (2014) se pueden encontrar algunos un conjunto de vídeos donde se muestra el trabajo realizado con estos equipos.

Por supuesto, gracias a la gran variedad de sensores electrónico, también se puede plantear desarrollar cualquier actividad que no tenga nada que ver con la robótica.

5.6. Control de Organizaciones de Robots basado en Sistemas Multi-agente

En los últimos años, la investigación y el trabajo en el campo de la robótica móvil cooperativa han crecido notablemente. En este caso se suelen tener entornos dinámicos con elementos móviles que deben organizarse y coordinarse entre ellos para cumplir las misiones encomendadas (Jiménez-González et al. (2014)).

Estos entornos originan típicamente un contexto muy heterogéneo a todos los niveles: diferentes plataformas robóticas con diferentes capacidades sensoriales, áreas de conocimiento distintas (robótica, inteligencia artificial, tecnologías del habla, etc.). Por ello, a la hora de realizar una coordinación de estos equipos, los sistemas jerárquicos tradicionales típicamente tienen una estructura rígida que les impide reaccionar de una manera ágil ante variaciones en el sistema y/o el entorno (Valero-Gómez et al. (2013)).

Como alternativa se puede utilizar una solución basada en sistemas holónicos, que es una organización altamente distribuida, donde la inteligencia se distribuye entre las entidades individuales desarrollada a partir de sistemas multiagente.

Para los Lego, el entorno de desarrollo más aconsejable es JADE (Java Agent Development Framework). Se trata de una plataforma de software libre para el desarrollo de agentes implementada en Java. La plataforma JADE soporta la coordinación de múltiples agentes FIPA (Foundation for Intelligent Physical Agents) y proporciona una implementación estándar del lenguaje de comunicación de agentes FIPA-ACL.

El trabajo que tiene que realizar cada uno de los agentes se encuentra dentro de sus comportamientos (*behaviours*), que representa una tarea que un agente puede llevar a cabo. De esta forma un agente puede ejecutar varios comportamientos concurrentemente.

Para la programación de los Lego se ha utilizado el firmware leJOS, el cual permite la programación en lenguaje Java, ofreciendo librerías orientadas a la navegación, a la comunicación Bluetooth, al uso de trigonometría y funciones matemáticas complejas, a la programación multithreading, al soporte de programación orientada a objetos, etc.

Con el entorno JADE-leJOS se han desarrollado varias aplicaciones basadas en sistema multiagente utilizando la arquitectura mostrada en la Figura 13. Se trata de una red en la que se tienen varios robots y computadores. Además, se dispone de una cámara cenital que está encargada de procesar las imágenes para obtener las posiciones de los robots, la ubicación de obstáculos, posiciones y orientaciones finales a las que deben ir, etc. Para las comunicaciones entre los robots y los computadores se utilizan comunicaciones inalámbricas Wi-Fi, XBee y Bluetooth, en función de la tarjeta de control con que se equipan los robots móviles.

Basado en Lego y en esta arquitectura multi-agente se han desarrollado varios demostradores:

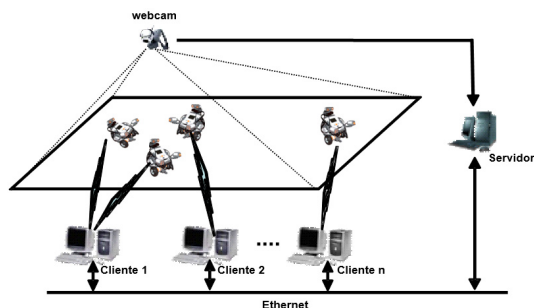


Figura 13: Arquitectura cliente-servidor utilizada con los Lego.

- *Evitación de colisiones*: la aplicación tiene 3 fases: detección de la colisión entre robots móviles, negociación y resolución de la colisiones. En la negociación se consideran cuestiones relacionadas con cuánto se debe desplazar cada robot para evitar la colisión, la prioridad, la velocidad y la maniobrabilidad de cada robot, etc.

- *Control distribuido de tareas industriales*: se simula un proceso industrial compuesto por un vehículo móvil tipo carretilla, una cinta transportadora, un brazo robot de 4 grados de libertad y un semáforo. A diferencia de la programación secuencial utilizada en entornos industriales, la operatividad de esta aplicación es completamente distribuida, y cada agente ejecuta sus propios comportamientos que definen las acciones que se pueden llevar a cabo en todo momento.

- *Equipo de robots de desactivación de minas*: aplicación que emula el proceso de desminado. Para ello se ha definido tres modelos de robots, cada uno caracterizado por su sensorización y capacidad de actuar con el entorno: robot explorador (dotado de sensores que ofrecen la detección de las minas), robot manipulador (que simula realizar la desactivación de las minas) y por último, el robot recolector (equipado con un actuador lineal que le permite recoger las minas).

- *Equipo multirobot de rescate*: combina y coordina tres tipos de robots con distintas arquitecturas. Éstos, mediante la ayuda de una cámara cenital, colaboran para llevar a cabo un objetivo: el rescate de uno de ellos.

En COBAMI (2014), la página web del proyecto de investigación del Plan Nacional del Ministerio de Economía y Competitividad DPI2011-28507-C02-01/02 se puede encontrar un conjunto de vídeos con el trabajo desarrollado.

6. Conclusión

Se ha presentado una plataforma hardware/software orientada hacia la educación en el campo de la automática en institutos y universidades. El sistema reúne elementos típicos estudiados en problemas de control y automatización como los sensores electrónicos, la identificación de sistemas, el control de motores, el control cinemático de robots móviles etc.

La plataforma propuesta está basada en Lego Mindstorms. Se trata de un equipamiento de bajo coste que proporciona, además de las típicas piezas de construcción, un conjunto de

sensores, actuadores eléctricos y una unidad de control potente. Gracias a él se pueden plantear toda una batería de actividades muy motivadoras para el alumnado de contenidos relacionados con el control por computador.

El trabajo ha presentado también una alternativa a las unidades de control originales de Lego. Actualmente hay disponibles una gran cantidad de tarjetas embebidas que permiten realizar la programación y la ejecución de cualquier algoritmo de control. Estas tarjetas tienen dos ventajas principales: la primera es su bajo coste, y la segunda es que se pueden programar con lenguajes, compiladores y entornos de desarrollo de software libre.

Esto permite equipar laboratorios de prácticas de laboratorio con prototipos interesantes y motivadores con un precio muy bajo, lo que repercute de una forma directa y positiva en la calidad docente.

English Summary

Low-Cost Platforms for Realization of Mechatronics and Robotics Practical Works.

Abstract

As occurred with the introduction to programming, learning automation, computer control, robotics and mechatronics systems in general, are contents that are being taught more and more to college and high school students. For this reason, the choice of appropriate laboratory platforms becomes a critical decision to encourage theoretical concepts with motivating experimentation. Thanks to advances in technology, today there are many options available, both at the hardware and programming level. This paper presents a multidisciplinary platform for low cost which covers various issues related to the realization of practical work related to automatic control.

Keywords:

Computer Control Systems, Robot Control, Multi-agent Systems, Embedded Systems, Mobile Robots, Control Education.

Agradecimientos

Los autores desean expresar su agradecimiento al Ministerio de Economía y Competitividad España por la financiación parcial de este trabajo bajo los proyectos de investigación DPI2011-28507-C02-01 y DPI2013-44227-R

Referencias

- ADEPT, 2014. Adept Mobile Robots. <http://activrobots.com/>, [Último acceso: 01.09.2014].
- Al-Busaidi, A. M., 2012. Development of an educational environment for on-line control of a biped robot using matlab and arduino. *Mechatronics (MECATRONICS)*, 337–344.

- Araújo, A., Portugal, D., Couceiro, M. S., Sales, J., Rocha, R. P., 2014. Desarrollo de un robot móvil compacto integrado en el middleware {ROS}. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)* 11 (3), 315–326.
DOI: <http://dx.doi.org/10.1016/j.riai.2014.02.009>
- Arduino, 2014. Sitio web de Arduino. <http://www.arduino.cc>, [Último acceso: 01.09.2014].
- A. Valera, 2014. Sitio web de Ángel Valera. <http://avalera.ai2.upv.es/docencia/>, [Último acceso: 01.09.2014].
- Baum, D., 2000. *Extreme MINDSTORMS: an advanced guide to LEGO MINDSTORMS*. Apress.
- Baum, D., Zurcher, R., 2003. *Definitive Guide to Lego Mindstorms*. Vol. 2. Apress.
- BeagleBoard, 2014. Sitio web de BeagleBoard. <http://www.beagleboard.org>, [Último acceso: 01.09.2014].
- Bradley Valdenebro, P., Puente Alfaro, J. A. d. I., Zamorano Flores, J. R., Brosnan Blazquez, D., 2012. A platform for real-time control education with lego mindstorms.
- Busquets, J., Busquets, J. V., Tudela, D., Pérez, F., Busquets-Carbonell, J., Barberá, A., Rodríguez, C., García, A. J., Gilbert, J., 2012. Low-cost auv based on arduino open source microcontroller board for oceanographic research applications in a collaborative long term deployment missions and suitable for combining with an usv as autonomous automatic recharging platform. In: *Autonomous Underwater Vehicles (AUV), 2012 IEEE/OES. IEEE*, pp. 1–10.
- Chaos, D., Chacon, J.n, J., Lopez-Orozco, J. A., Dormido, S., 2013. Virtual and remote robotic laboratory using ejs, matlab and labview. *Sensors* 13 (2), 2595–2612.
URL: <http://www.mdpi.com/1424-8220/13/2/2595>
DOI: 10.3390/s130202595
- Chin, K.-Y., Buhari, S., Ong, W.-H., Feb 2009. Impact of lego sensors in remote controlled robot. In: *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pp. 1777–1782.
DOI: 10.1109/ROBIO.2009.4913271
- COBAMI, 2014. Sitio web del proyecto COBAMI. <http://wks.gii.upv.es/cobami/webfm>, [Último acceso: 01.09.2014].
- Cuellar, M. P., Pegalajar, M. C., 2014. Design and implementation of intelligent systems with lego mindstorms for undergraduate computer engineers. *Computer Applications in Engineering Education* 22 (1), 153–166.
URL: <http://dx.doi.org/10.1002/cae.20541>
DOI: 10.1002/cae.20541
- Danahy, E., Wang, E., Brockman, J., Carberry, A., Shapiro, B., Rogers, C. B., 2014. Lego-based robotics in higher education: 15 years of student creativity. *Int J Adv Robot Syst* 11, 27.
- DexterIndustries, 2014. Sitio web Dexter Industries. <https://www.dexterindustries.com>, [Último acceso: 01.09.2014].
- EJS, 2014. Sitio web de Easy Java Simulations. <http://fem.um.es/Ejs/>, [Último acceso: 01.09.2014].
- epuck, 2014. e-puck education robot. <http://www.e-puck.org>, [Último acceso: 01.09.2014].
- Evans, B., 2012. *Practical 3d printers: The science and art of 3d printing*. Apress, Berkely, CA, USA.
- Faludi, R., 2010. Building wireless sensor networks: with ZigBee, XBee, arduino, and processing. O'Reilly Media, Inc."
- Gawthrop, P. J., McGookin, E., 2004. A lego-based control experiment. *Control Systems, IEEE* 24 (5), 43–56.
- Greenwald, L., Kopena, J., 2003. Mobile robot labs. *Robotics & Automation Magazine, IEEE* 10 (2), 25–32.
- Grega, W., Pilat, A., 2008. Real-time control teaching using lego® mindstorms® nxt robot. In: *Computer Science and Information Technology, 2008. IMCSIT 2008. International Multiconference on*, IEEE, pp. 625–628.
- Guo, J., Kettler, D., Al-Dahhan, M., 2007. A chemical engineering laboratory over distributed control and measurement systems. *Computer Applications in Engineering Education* 15 (2), 174–184.
- HiTechnic, 2014. Sitio web HiTechnic. <https://www.hitechnic.com>, [Último acceso: 01.09.2014].
- Ionescu, C. M., Fabregas, E., Cristescu, S. M., Dormido, S., De Keyser, R., 2013. A remote laboratory as an innovative educational tool for practicing control engineering concepts. *Education, IEEE Transactions on* 56 (4), 436–442.
- J. Henry, H. S., 2005. International cooperation in control engineering education using remote laboratories. *European Journal Of Engineering Education* 30, 265–274.
- Jiménez-González, A., de Dios, J. R. M., de San Bernabé, A., Núñez, G., Olle-ro, A., 2014. Un banco de pruebas remoto para experimentación en robótica ubicua. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)* 11 (1), 68–79.
DOI: <http://dx.doi.org/10.1016/j.riai.2013.09.007>
- Jung, S., 2013. Experiences in developing an experimental robotics course program for undergraduate education. *Education, IEEE Transactions on* 56 (1), 129–136.
- Kim, S., Oh, H., Choi, J., Tsourdos, A., 2014. Using hands-on project with lego mindstorms in a graduate course. *INTERNATIONAL JOURNAL OF ENGINEERING EDUCATION* 30 (2), 458–470.
- Klassner, F., Anderson, S. D., 2003. Lego mindstorms: Not just for k-12 anymore. *IEEE Robotics & Automation Magazine* 10 (2), 12–18.
- Klassner, F., Continanza, C., 2007. Mindstorms without robotics: an alternative to simulations in systems courses. In: *ACM SIGCSE Bulletin*. Vol. 39. ACM, pp. 175–179.
- Labview, 2014. Librería Labview para LEGO Mindsensors. <http://www.ni.com/academic/mindstorms/>, [Último acceso: 01.09.2014].
- LeJOS, 2014. Sitio web del firmware LeJOS. <http://lejos.org>, [Último acceso: 01.09.2014].
- Lim, H., Park, J., Lee, D., Kim, H. J., 2012. Build your own quadrotor: Open-source projects on unmanned aerial vehicles. *Robotics & Automation Magazine, IEEE* 19 (3), 33–45.
- MathWorks, 2014. Sitio web de Matlab-Simulink. <http://www.mathworks.es>, [Último acceso: 01.09.2014].
- Mindsensors, 2014. Sitio web Mindsensors. <https://www.mindsensors.com>, [Último acceso: 01.09.2014].
- Mindstorms, L., 2014. Sitio web de LEGO Mindstorms. <http://mindstorms.lego.com>, [Último acceso: 01.09.2014].
- Moor, S. S., Piergiovanni, P. R., Metzger, M., 2013. Process control kits: a hardware and software resource. In: *Computer Applications in Engineering Education*, pp. 21:491–502.
DOI: 10.1002/cae.20495
- Moway, 2014. Moway Robots. <http://moway-robot.com>, [Último acceso: 01.09.2014].
- Márquez, J. A., Sanguino, T. M., 2010. Diseño de laboratorios virtuales y/o remotos. un caso práctico. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)* 7 (1), 64–72.
DOI: [http://dx.doi.org/10.1016/S1697-7912\(10\)70009-1](http://dx.doi.org/10.1016/S1697-7912(10)70009-1)
- Papert, S., 2000. What's the big idea? toward a pedagogy of idea power. *IBM Systems Journal* 39 (3.4), 720–729.
- Raspberry, 2014. Sitio web de Raspberry. <http://www.raspberrypi.org>, [Último acceso: 01.09.2014].
- Resnick, M., Martin, F., Sargent, R., Silverman, B., 1996. Programmable bricks: Toys to think with. *IBM Systems journal* 35 (3.4), 443–452.
- RobotC, 2014. Sitio web de RobotC. <http://www.robotc.net>, [Último acceso: 01.09.2014].
- Saleiro, M., Carmo, B., Rodrigues, J. M., du Buf, J. H., 2013. A low-cost classroom-oriented educational robotics system. In: *Social Robotics*. Springer, pp. 74–83.
- Schmalzhaus, 2014. Sitio web de big easy driver. <http://www.schmalzhaus.com>, [Último acceso: 01.09.2014].
- Valera, A., Díez, J. L., Vallés, M., Albertos, P., 2005. Virtual and remote control laboratory development. *Control Systems, IEEE* 25 (1), 35–39.
- Valero-Gómez, A., de la Puente, P., Rodríguez-Losada, D., Hernando, M., Segundo, P. S., 2013. Arquitectura de integración basada en servicios web para sistemas heterogéneos y distribuidos: aplicación a robots móviles interactivos. *Revista Iberoamericana de Automática e Informática Industrial (RIAI)* 10 (1), 85–95.
DOI: <http://dx.doi.org/10.1016/j.riai.2012.11.008>
- Weinberg, J. B., Yu, X., 2003. Robotics in education: Low-cost platforms for teaching integrated systems. *Robotics & Automation Magazine, IEEE* 10 (2), 4–6.
- Zachariadou, K., Yiasemides, K., Trougakos, N., 2012. A low-cost computer-controlled arduino-based educational laboratory system for teaching the fundamentals of photovoltaic cells. *European Journal of Physics* 33 (6), 1599.