Herramientas Libres para la Programación de Agentes Físicos

Vicente Matellán
Dept. Ciencias Experimentales e Ingeniería
Universidad Rey Juan Carlos
C/ Tulipán s/n, 28933 Móstoles (Madrid) - España
vmo@gsyc.escet.urjc.es

Jesús M. González
Dept. Ciencias Experimentales e Ingeniería
Universidad Rey Juan Carlos
C/ Tulipán s/n, 28933 Móstoles (Madrid) - España
jgb@gsyc.escet.urjc.es

José Centeno
Dept. Ciencias Experimentales e Ingeniería
Universidad Rey Juan Carlos
C/ Tulipán s/n, 28933 Móstoles (Madrid) - España
jcenteno@gsyc.escet.urjc.es

RESUMEN

La preparación de un curso de robótica para alumnos de informática implica muchas más cosas que el simple diseño del programa a impartir. En particular el diseño de las prácticas implica múltiples decisiones, cómo cuál es el robot más adecuado o cuáles son las herramientas más adecuadas para su programación. En este artículo se analizan las decisiones tomadas en el diseño de la asignatura de Robótica dentro del programa de Ingeniería Técnica en Informática de Sistemas de la Universidad Rey Juan Carlos. El artículo se centra en la descripción de las prácticas, especialmente en los criterios de elección de las herramientas utilizadas, donde el aspecto de ser software libre ha recibido una especial consideración.

1. INTRODUCCIÓN

El *Grupo de Sistemas y Comunicaciones* (GSyC)[1] se encarga de la docencia de las asignaturas que se pueden considerar relacionadas con los agentes físicos en la Universidad Rey Juan Carlos (URJC) de Madrid. Dichas asignaturas se encuadran dentro de la titulación de Ingeniero Técnico en Informática de Sistemas, lo que hace que los objetivos de esta asignatura estén más relacionados con la programación de sistemas, que con otros aspectos que podrían considerarse más "tradicionales" en la docencia de asignaturas relacionadas con la robótica, como la automatización, o los estudios matemáticos relacionados. La misma orientación, centrada en la construcción de software, es la que sigue el grupo en todas las asignaturas impartidas por el GSyC, y en particular con las relacionadas con la construcción de sistemas que interactúan con el mundo real, como "Robótica".

El objetivo principal de este artículo es la descripción de las prácticas de la asignatura, haciendo especial hincapié en el requisito que se impuso de utilizar únicamente para su realización software "libre"[2]. El artículo también describe la organización de las prácticas y del laboratorio en donde se han realizado.

El grupo GSyC ha estado involucrado con actividades relacionadas con el Software Libre desde su fundación en 1994. Una de las facetas en las que sus beneficios son más evidentes es precisamente en la docencia, donde el grupo ha utilizado herramientas libres tanto en asignaturas de Redes de Ordenadores como de Sistemas Operativos[2]. Estas experiencias animaron al grupo a utilizar esta misma aproximación en el diseño de la asignatura "Robótica".

El diseño de la asignatura, desde el punto de vista teórico, se basó en una aproximación constructivista[8], donde se espera de los alumnos adquieran los conceptos básicos mediante la experimentación, lo cual implica en este caso que deberán disponer de elementos que les permitan construir robots y programarlos. Siguiendo esta filosofía, la base teórica está orientada a la construcción de sistemas basados en comportamientos, siendo el libro recomendado el de Ronald C. Arkin [1]. Aunque el resto de las arquitecturas básicas de control, desde las puramente reactivas a las deliberativas, también se analizan en la parte teórica de la asignatura.

Este es el entorno de la asignatura de Robótica en la Universidad Rey Juan Carlos. Se puede obtener información adicional y mas detallada, en las páginas web de la asignatura[3]. La siguiente sección describe el proceso que nos llevó a elegir los LEGO Mindstorms como la plataforma más adecuada para las prácticas. A continuación se analizan las herramientas de programación disponibles para dicha plataforma, así como el resto de aplicaciones utilizadas para el desarrollo de la asignatura. Por último, se resume la organización de las prácticas de la asignatura y su evaluación.

2. LOS "AGENTES FÍSICOS"

Respecto de a los robots a utilizar en las prácticas, la primera opción que se consideró para las prácticas fue usar el minirobot Khepera [10] con el que el grupo tenía experiencia previa, aunque en trabajos de investigación relacionados con la

[1] http://gsyc.escet.urjc.es

[2] Por "libre" no se debe entender "gratis". La idea es disponer de software libremente modificable y redistribuible

[3] http://gsyc.escet.urjc.es/docencia/asignaturas/robotica

competición denominada RoboCup [9]. A la vez se consideró la posibilidad de utilizar el propio entorno de la RoboCup como entorno de prácticas, como se viene haciendo en otros lugares [4, 7]. Este robot fue diseñado en el laboratorio suizo del EPFL y en la actualidad es fabricado y distribuido por la empresa K-Team.

Sin embargo, se descartó esta posibilidad (Khepera + fútbol). La razón principal para descartar este robot fue su elevado precio (alrededor de los 2000 Euros) [4]. En cuanto al uso del fútbol como entorno de prácticas, aún reconociendo su potencial motivador y la gran cantidad de desafíos que proporciona, se consideró como demasiado complejo para su estudio dentro de la enseñanza reglada de primer ciclo.

La segunda alternativa que se consideró, también por experiencias previas, fue el RugWarrior[5]. Este es un robot basado en el chip 68HC11 de Motorola, uno de los más extendidos para este tipo de hardware. En este caso la razón para descartarlo no fue el precio, que podría considerarse dentro de los rangos aceptables, sino el tipo de robot. Es un robot diseñado para que los alumnos comprendan la arquitectura interna de un robot móvil, desde los problemas de control, hasta las consideraciones más próximas a los sistemas operativos. El objetivo de la asignatura está más orientado a los problemas de control, por lo que se consideró poco apropiado. En cualquier caso, el libro en el que se describe la arquitectura del RugWarrior[5] se ha recomendado como bibliografía adicional como referencia para esos asuntos.

Otras opción que se ha tenido en cuenta es la tarjeta Handyboard[5], una tarjeta basada también el micro-controlador Motorola 68HC11, diseñada específicamente para la construcción de robots, pero que tiene el mismo problema que el RugWarrior y es que está orientada a la enseñanza de conceptos de arquitectura más que de control.

Además se considerói el EyeBot[6]. Se trata de una tarjeta, originalmente diseñada en Australia, pensada para el procesamiento de imágenes. Esto hace que disponga del sistema sensorial más rico de los posibles, esto es una cámara, lo cual es su principal ventaja. Su mayor problema es que los robots basados en ella son aún más caros que el Khepera.

La elección definitiva fue el LEGO Mindstorms[7]. A pesar de tratase de un producto diseñado como producto de consumo, lo que le dota de una gran calidad, es de una enorme flexibilidad. Permite construir una enorme variedad de robots, desde un brazo hasta un robot recolector, sin necesitar ninguna soldadura. Además, permite una gran libertad a la hora de colocar los sensores y los motores. Por otro lado, los bloques de LEGO son muy conocidos para la mayoría de los alumnos. El cerebro de estos robots está basado en un Hitachi H8/300 con 32 Kbytes de RAM, con tres puertos de salida para conectar motores y tres puertos de entrada para conectar sensores, además de un puerto de infra-rojos para comunicarse con el ordenador de desarrollo. Los sensores disponibles son el de luz ambiente, de luz reflejada, de colisión, de temperatura y de rotación.

Los Mindstorm se venden como *kits* completos formados por más de 700 piezas tradicionales de LEGO. Cada *kit* además incluye dos motores, dos sensores de contacto, uno de luz y el bloque del micro-procesador que se denomina habitualmente "ladrillo" o RCX. Además, el *kit* incluye un entorno gráfico de desarrollo y el equipo necesario para descargar el software desde un ordenador personal al ladrillo.

Los estudiantes se agruparon en parejas cada una de las cuales recibió un *kit* Mindstorm y adicionalmente otro sensor de luz, un sensor de rotación y un motor extra para permitir diseños más complejos.

El entorno para el desarrollo de programas incluido en el *kit* Mindstorm 1.5 (versión utilizada en las prácticas) es resumidamente un lenguaje gráfico de programación pensado fundamentalmente para niños o adultos sin ninguna preparación informática. Se trata de un entorno muy limitado, y no cumple el requisito de ser software libre. Afortunadamente, los Mindstorms han recibido mucha atención por parte de la comunidad *open source*[8], con lo cual han aparecido muchas opciones diferentes para su programación, las cuales se analizan en la siguiente sección.

3. HERRAMIENTAS LIBRES UTILIZADAS

Una vez que el robot se ha elegido, lo siguiente es elegir el entorno de programación que usarán los alumnos. Ese entorno incluye el sistema operativo que se ejecutará en el robot, la plataforma de desarrollo, el entorno de compilación, las herramientas de depuración y los simuladores. En esta sección se analizarán las herramientas de software libre que se usarán en la asignatura de *Robótica* para ello. Así mismo, se analizarán algunas otras herramientas para proporcionar información a los estudiantes, o para mantenerse en contacto con ellos.

3.1 Herramientas de programación

Las prácticas de la asignatura de *Robótica* se componen de dos partes, la primera y menos importante, es la construcción de robots. La segunda es la programación de dichos robots. Para construir un robot basado en LEGO se necesitan conocimientos básicos de mecánica, de los que los alumnos en general carecen. Sin embargo, la experiencia ha demostrado que los alumnos son capaces de adquirirlos fácilmente, sin más que recurrir a pequeñas guías, como la *Constructopedia* que incluye el *kit*, o la guía de Fred Martin[9].

Para programar los LEGO Mindstorm existen diversas opciones, muchas de las cuales se analizan en el libro de Jonathan B. Knudsen [6]. La primera opción es utilizar el entorno gráfico de desarrollo proporcionado por LEGOS, que es una herramienta muy intuitiva, pero muy limitada y además no es software libre. Esto quiere decir que al igual que en el caso de la plataforma hardware, hay que evaluar diversas posibilidades. En el caso del entorno de desarrollo se eval-

[4]http://www.k-team.com
[5]http://el.www.media.mit.edu/projects/handy-board/
[6]http://www.ee.uwa.edu.au/∼braunl/eyebot/
[7]http://www.legomindstorms.com

[8]Open Source Definition es un término acuñado por la Open Source Initiative para certificar las licencias de software que cumplen las condiciones para poder considerarse software libre.
[9]http://el.www.media.mit.edu/groups/el/projects/constructopedia/

uaron fundamentalmente dos opciones: NQC y legOS, cada una de las cuales se describe en las siguientes dos secciones.

La opción que se recomendó a los alumnos fue el sistema operativo legOS. Las razones para esta decisión se analizan al final de la sección 3.1.1. El uso de este sistema operativo implica también la necesidad de instalar un sistema de compilación cruzada, para lo que se emplearon las `binutils` y `egcs`. Aunque estas herramientas también son parte del entorno de programación no serán objeto de análisis en este artículos por ser bien conocidas para cualquier programador.

En este entorno de prácticas, esto es, los Mindstorms más legOS, es a veces difícil depurar un programa, por eso se instalaron dos simuladores diferentes para nuestros estudiantes: Legosim y Emulegos, ambos programas libres. Las diferencias entre ambos se estudian en las secciones 3.2.1 y 3.2.2.

3.1.1 NQC

El acrónimo NQC [3] significa *Not Quite C*. Se trata de un simple lenguaje con una sintaxis muy similar a C que se puede usar para programar el ladrillo. Es, desde luego, la alternativa más sencilla al lenguaje de programación basado en iconos arrastrables que proporciona LEGO.

NQC es software libre, distribuido bajo la Licencia MPL (*Mozilla Public License*). Sin embargo, usa el sistema operativo original de LEGO, lo que hace que no se cumpla el requisito previamente impuesto de emplear sólo software libre en las prácticas. Sin embargo esa no es la principal razón por la que se escogió legOS. NQC se diseñó para ser muy simple y para ser utilizable por personas con limitados conocimientos de programación. Todo ello se ha conseguido, pero a costa de una serie de limitaciones, de entre las cuales las más importantes son:

- Las subrutinas no admiten parámetros, por lo que no es posible emplear realmente los principios de la programación estructurada.
- Sólo se pueden usar variables globales, es decir, el espacio de nombres es único, lo cual complica enormemente el desarrollo y mantenimiento de programas complejos.
- Las subrutinas no pueden devolver valores, lo cual las limita a ser subconjuntos de código, sin poder aportar funcionalidades nuevas ocultando su implementación.
- El número de variables está enormemente limitado, de hecho sólo se dispone de 32.
- No existen las estructuras de datos, ni las estáticas ni las dinámicas, lo que imposibilita casi cualquier tipo de aproximación que necesite almacenar cualquier tipo de estado.

Como los alumnos de la asignatura se supone que tienen fuertes conocimientos de programación (no en balde "Robótica" es una asignatura optativa de tercer curso), se decidió que NQC era demasiado limitado como para que los alumnos pudiesen generar comportamientos sofisticados para los robots.

3.1.2 LegOS

LegOS es un sistema operativo libre diseñado para el LEGO Mindstorms, diseñado e implementado fundamentalmente por Markus Noga [11]. Comparado con el sistema operativo original de LEGO, ofrece muchas ventajas además de mejores prestaciones y mayor flexibilidad. Entre las características más importantes de la versión legOS 0.2 se pueden destacar:

- La carga dinámica de programas y módulos.
- El protocolo de comunicación basado en el transmisor infra-rojo.
- La posibilidad de realizar programas multitarea.
- La gestión de memoria dinámica.
- La existencia de *drivers* para todos los subsistemas del ladrillo.
- El uso de la velocidad nativa del micro-procesador, esto es 16 MHz.
- El acceso a los 32K de memoria RAM.
- Permitir el uso completo del lenguaje de programación elegido, como por ejemplo C. Lo cual implica que se pueden usar punteros, estructuras de datos, etc.

El entorno de desarrollo se basa en ordenadores personales (Intel x86) bajo GNU/Linux donde se compilan los programas y que se comunican con los Mindstorms, tanto para descargar programas como para depurarlos. La versión empleada de legOS ha sido la 0.2.0 que es bastante estable, tanto si se usan distribuciones comerciales de GNU/Linux como RedHat o SuSE, como bajo Debian 2.1, que es la que se instaló en el laboratorio para los alumnos

LegOS es solo el sistema operativo, lo que quiere decir que se necesita el soporte de algún lenguaje de programación. En la asignatura se utilizó C, pero se pueden utilizar otros lenguajes de programación si se dispone de una versión cruzada que genere código para el Hitachi H8. Por ejemplo, muchos desarrolladores utilizan C++ en vez de C.

Alrededor del sistema operativo legOS se han desarrollado múltiples herramientas auxiliares, como por ejemplo simuladores que hacen más fácil la depuración al permitir ejecutar programas en la propia plataforma de desarrollo usando un depurador tradicional de GNU/Linux como por ejemplo `gdb`. Algunas de estas herramientas se analizan en la próxima sección.

3.2 Herramientas relacionadas con legOS

Se eligió legOS como base para la programación del RCX, lo que obliga a utilizar otras herramientas, algunas evidentes como el entorno de compilación cruzado. En este caso se ha utilizado el entorno de la FSF[10] basado en las `binutils` y el compilador `gcc`. También se han utilizado otras herramientas como los simuladores. Algunas de estas herramientas son:

[10] http://www.fsf.org

3.2.1 LegoSim

LegoSim es un simulador para legOS cuya principal virtud, que le diferencia de Emulegos (descrito en la siguiente sección), es que el interfaz gráfico de usuario (GUI) se puede separar del simulador propiamente dicho. Esta separación permite utilizar el GUI como unidad de control no sólo como simulador, permitiendo por ejemplo, conectarlo a otros RCX vía infra-rojos. Por supuesto, también permite ejecutar el GUI en una máquina distinta de la del simulador.

El GUI es un *applet* de Java que se parece realmente al RCX. El simulador es sólo una biblioteca (librería). Ambos componentes se relacionan mediante un conjunto de *scripts* en Perl. El simulador es una biblioteca que reemplaza la parte de legOS que se enlaza con cualquier aplicación en el proceso de compilación, generando una aplicación completa y ejecutable en la máquina de desarrollo. En la simulación, las tareas (*tasks*) de legOS se traducen en *threads* POSIX. Las entradas y salidas, que resultan vitales, se simulan mediante cadenas de texto sobre `stdin` y `stdout` siguiendo una sintaxis particular.

LegoSim se distribuye bajo licencia MPL, es decir, es software libre. Sus diseñadores e implementadores principales han sido Frank Mueller, Thomas Röblitz, y Oliver Bühn[11].

3.2.2 EmuLegOS

EmuLegOS [12] es otro simulador de legOS. Su objetivo de diseño fue proporcionar un entorno más confortable para probar y depurar programas. EmuLegOS es, en esencia, un conjunto de código escrito en C++ que se puede compilar y enlazar junto con cualquier aplicación para legOS, generando como resultado de ese proceso una aplicación que emula el comportamiento de ese código al que tuviese si estuviese ejecutándose en un RCX real. El nivel de programación o API (*Application Program Interface*) emula las rutinas de legOS. La mayoría de legOS está implementado en EmuLegOS, incluyendo por ejemplo el soporte de tareas, o la comunicación por infra-rojos.

El aspecto externo de una aplicación para legOS ejecutando en EmuLegOS es el de la Figura 1. EmuLegOS permite al usuario configurar los sensores e interaccionar con ellos mientras el programa está en ejecución, por ejemplo simulando eventos externos. El interfaz también muestra el estado de los hasta tres motores que se pueden "pinchar virtualmente" en los puertos A, B y C del RCX.

EmuLegOS también permite que se emule el mundo real, proporcionando un lugar donde insertar código que imite algunas de las características físicas del robot. Por ejemplo, se puede incorporar un sensor de rotación que gire mientras un determinado motor virtual está en marcha, o hacer que un sensor de colisión se active pasada una cantidad de tiempo desde que se arrancó el motor.

La mayor utilidad de los simuladores es la posibilidad de depurar. En ambos casos (EmulegOS y LegoSim), se pueden utilizar todas las herramientas disponibles en el entorno de desarrollo, ya que el programa para legOS se ejecuta dentro del simulador, y éste dentro de la plataforma.

[11] http://www.informatik.hu-berlin.de/~mueller/legosim/
[12] http://www.geocities.com/~marioferrari/emulegos.html

Figure 1: Simulador Emulegos

Otra categoría de herramientas relacionadas con legOS son las que se pueden catalogar como herramientas de terceras partes. Así por ejemplo, existen compiladores para legOS accesibles vía Web, como el que se analiza en la siguiente sección.

3.2.3 WebLegos

Web-LegOS es un interfaz escrito en HTML para compilar programas escritos para el sistema operativo legOS del LEGO Mindstorm RCX.

El uso de Web-LegOS es sencillo bastando con cortar y pegar un fichero fuente en una caja de una página web, elegir el lenguaje de programación y seleccionar la forma en que se quiere recibir el fichero que se genera. A continuación, con pulsar el botón de "compilar" se produce el envío del fichero fuente y su compilación cruzada remota a legOS. El fichero obtenido está en formato `S-record` (un formato diseñado para para permitir la descarga de datos desde un ordenador a otro diferente). Una vez que el fichero `S-record` se ha obtenido a través del compilador web es posible descargarlo en el RCX utilizando el canal de infra-rojos habitual.

3.3 Herramientas de información

Además de las herramientas previamente descritas, muy específicas de la programación de agentes físicos, se han utilizado algunas otras además de las típicas de cualquier entorno de trabajo (editores, navegadores, etc.). Especialmente importantes desde nuestro punto de vista son aquellas que promueven la colaboración entre los estudiantes y facilitan su relación con los profesores. Entre ellas hay que citar las tradicionales de Internet. Así, se crea una lista de correo[13] para cada asignatura impartida por el GSyC, existe otra para que los alumnos puedan dirigirse a todos los profesores del grupo[14], un grupo de noticias[15] para cada asig-

[13] robotica@gsyc.escet.urjc.es
[14] gsyc-profes@gsyc.escet.urjc.es
[15] news://es.urjc.robotica

natura, y páginas web (http://gsyc.escet.urjc.es/docencia) donde los alumnos pueden encontrar información relacionada con cada asignatura: descripción, material utilizado en las clases como transparencias o ejercicios, enunciados de las prácticas, enlaces a sitios útiles, etc. Se utiliza `mailman` [16] como software de manejo de las listas, `inn` [17] como software para los grupos de noticias y `Apache` [18] como servidor web. Todas ellas ejecutando sobre un servidor Debian GNU/Linux [19] basado en un ordenador personal.

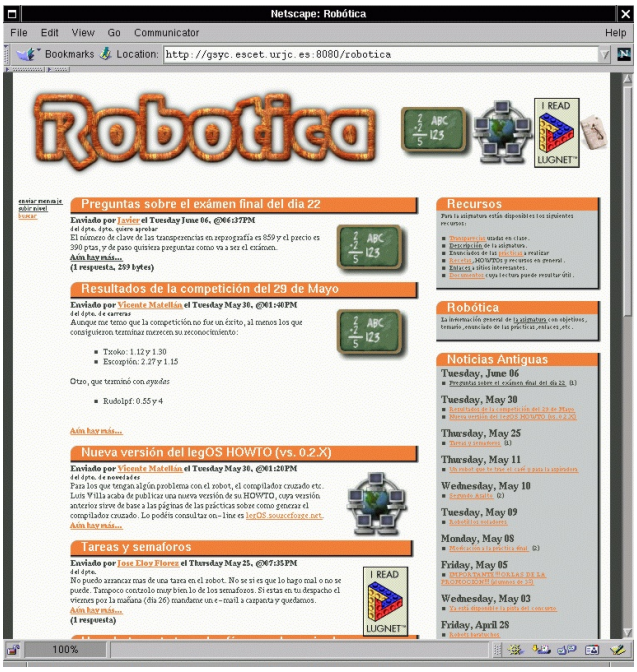

Figure 2: Weblog de la asigntaura de Robótica

La mayor novedad del último curso ha sido utilizar *weblogs* para las asignaturas. La idea de un *weblog* es dar a la gente interesada en un determinado área las noticias relevantes de dicha comunidad, siendo la fuente de dichas noticias la propia comunidad y permitiendo además que dichas noticias sean comentadas. Uno de los *weblogs* más conocidos es SlashDot[20] que se ha convertido en el sitio de referencia entre los desarrolladores y usuaros de software libre, GNU/Linux, etc. El aspecto de esta clase de herramientas se puede ver en la Figura 2. El instalado para la asignatura de Robótica se puede consultar en la siguiente URL: `http://gsyc.escet.urjc.es/foros/robotica`.

El software utilizado para construir el *weblog* de la asignatura es una herramienta libre llamada SquishDot[21], que a su vez está construida utilizando un entorno de desarrollo de aplicaciones web también libre denominado ZOPE[22] (*Z Object Publishing Environment*) que permite fácilmente construir aplicaciones que integran bases de datos, plantillas, etc. y que ha desarrollado y mantiene la empresa Digital Creations[23].

4. ORGANIZACIÓN DE LAS PRÁCTICAS

El número de créditos asignados a la asignatura de robótica es de 6, lo que quiere decir que el curso debe constar de 60 horas. La mitad de dichos créditos son teóricos y la mitad prácticos. La parte teórica se imparte en forma de clase magistral utilizando las transparencias que se pueden encontrar en el sito web de la asignatura.

Las prácticas están divididas en tres, dos de las cuales implican la construcción de robots. Las dos primeras no son obligatorias, sólo recomendadas. Éstas no deberían consumir más de un tercio del tiempo de prácticas (10 horas) y deberían permitir a los alumnos adquirir gradualmente las conocimientos necesarios para realizar la tercera práctica. Esta tercera práctica es la que se utiliza para calificar a los estudiantes, conjuntamente con un examen teórico.

La primera práctica consistió en la elección del entorno de programación. La parte más importante de esta práctica no es hacer la elección correcta, que debería ser legOS, sino dar los motivos adecuados.

La segunda práctica requirió construir un robot capaz de moverse y "sobrevivir" en un entorno desconocidos. Es decir, ser capaz de moverse aleatoriamente sin golpearse ni bloquearse en los obstáculos y sin caerse por ningún escalón. El objetivo de esta práctica es familiarizar a los alumnos con la API de legOS, conocer las herramientas que necesitan utilizar, los procesos de descarga de programas en el robot, las técnicas básicas de depuración y la utilidad de los simuladores.

La tercera práctica se organizó en forma de competición, donde los robots debían encontrar la salida de dos laberintos en el menor tiempo posible. El laberinto consistía básicamente en paredes de madera con diferentes ángulos, con callejones sin salida y obstáculos aislados. La salida del laberinto se marcaba mediante una luz que se podía utilizar también para orientar al robot.

Los robots debían resolver dos laberintos diferentes, el primero conocido a-priori y el segundo desconocido. La idea es motivar en los alumnos el uso de arquitecturas generales en vez de soluciones locales. Así, los alumnos que intentaron utilizar recorridos pre-compilados para el primer laberinto encontraron serias dificultades para enfrentar el segundo, mientras que los que utilizaron su tiempo en perfeccionar una arquitectura general que solventase el laberinto conocido no tuvieron en general problemas en el segundo. El resultado de la competición puede comprobarse en el web. En una primera competición participaron 8 robots y en la definitiva 15.

5. CONCLUSIONES

Este artículo ha presentado el entorno que se ha elegido para impartir la asignatura de robótica en la Universidad Rey Juan Carlos. Como principal característica de dicho entorno se debe resaltar que todo el software empleado, tanto

[16] http://http://www.list.org
[17] http://www.isc.org/products/INN
[18] http://www.apache.org
[19] http://www.debian.org
[20] http://slashdot.org
[21] http://www.squishdot.org
[22] http://www.zope.org
[23] http://www.digital.com

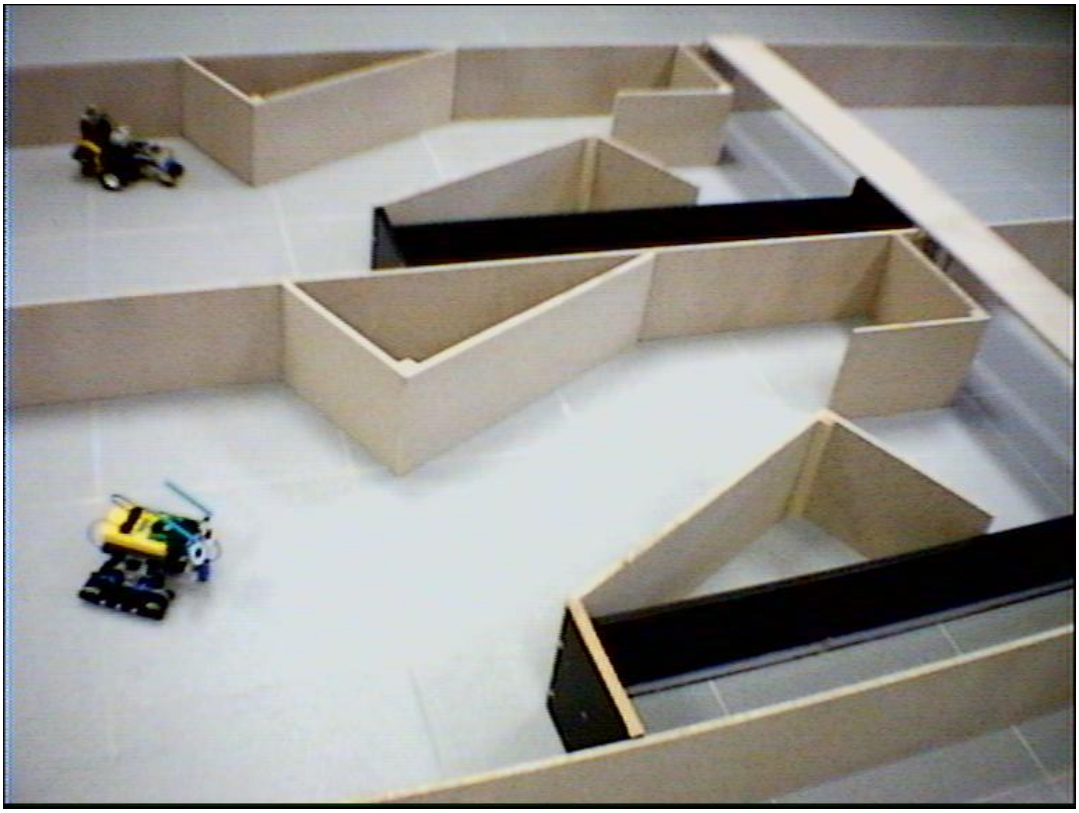

Figure 3: Practical assignment maze

en los equipos de desarrollo (ordenadores personales bajo GNU/Linux) como en el utilizado en el robot, ha sido software libre. Igualmente, el compilador, el depurador o los simuladores. Así mismo se han descrito la mayoría de dichas herramientas, así como otras opciones disponibles, tanto en el caso del hardware como el del software.

En resumen, se puede considerar que utilizar *sólo* software libre como es el aspecto más innovador descrito en este artículo. Una contribución secundaria descrita son las herramientas auxiliares que se han instalado para ayudar al desarrollo de las clases, destacando las herramientas relacionadas con Internet.

6. AUTORES ADICIONALES

Otros autores: Pedro de las Heras Quirós. Departamento de Ciencias Experimentales e Ingeniería. Universidad Rey Juan Carlos. E-mail: `pheras@gsyc.escet.urjc.es`.

7. REFERENCIAS

[1] R. C. Arkin. *Behavior based robotics*. MIT Press, 1998.

[2] J. Barahona, J. Centeno, P. de las Heras, V. Matellán y F. Ballesteros. Libre Software in CS Practice Teaching (The experience at Carlos III University). *IEEE Software*, 17(3), 2000.

[3] D. Baum. *Dave Baum's Definitive Guide to LEGO Mindstorms*. APress, 1999.

[4] A. O. J. L. d. l. R. J.A. Ramon, A. Figueras. Plataforma docente de robots móviles, cooperantes y autónomos. En el *1er Concurso Iberoamericano de Tecnologías Aplicadas a la Enseñanza de la Electrónica (CITA'98)*, 1998.

[5] J. L. Jones, A. M. Flynn y B. A. Seiger. *Mobile Robots: Inspiration to Implementation (2nd Edition)*. A. K. Peters, Wellesley, Massachusetts (USA), 1998.

[6] J. B. Knudsen. *The Unofficial Guide to LEGO MINDSTORMS Robots*. O'Reilly & Associates, 1999.

[7] H. H. Lund. Robot soccer in education. *Advanced Robotics Journal*, número especial sobre la RoboCup, 1999.

[8] F. Martin. *Ideal and Real Systems: A study of notions of control in undergraduates who design robots*. MIT Press, 1994.

[9] V. Matellán, D. Borrajo y C. Fernández. Using ABC^2 in the Robocup Domain. En *RoboCup-97: Robot Soccer World Cup I*, páginas 475–483, editado por H. Kitano. Springer Verlag, 1998.

[10] F. Mondada and F. P. Ienne. Mobile robot miniaturization: A tool for investigation in control algorithms. En *Proceedings of the Third International Symposium on Experimental Robotics*, Kyoto, Japan, 1993.

[11] M. L. Noga. Legos: Open-source embedded operating system for the lego mindstorms. `http://www.noga.de/legOS/`.