Programación de LEGO MindStorms bajo GNU/Linux

Vicente Matellán Olivera, Jesús M. González Barahona,
Pedro de las Heras Quirós, José Centeno González*
{vmo,jgb,pheras,jcenteno}@gsyc.escet.urjc.es
Departamento de Ciencias Experimentales e Ingeniería
Universidad Rey Juan Carlos

11 de noviembre de 2000

Esta ponencia forma parte del III Congreso Hispalinux[1] virtual, celebrado en Noviembre de 2000. Puedes participar en él en la sección de eventos de BarraPunto[2]

Resumen

GNU/Linux sobre un ordenador personal es la opción libre preferida por muchos desarrolladores de aplicaciones, pero también es una plataforma de desarrollo muy popular para otros sistemas, incluida la programación de robots, en particular es muy adecuada para *jugar* con los LEGO Mindstorms. En este artículo se presentaremos las dos opciones más extendidas a la hora de programar estos *juguetes*: NQC y LegOS. NQC es una versión reducida de C que permite el desarrollo rápido de programas mientras que LegOS es un sistema operativo completo que permite la programación en lenguajes tradicionales como C o C++. Además, presentaremos algunas herramientas para GNU/Linux relacionadas con LegOS como simuladores o compiladores web.

1 Introducción

Antes de comenzar a analizar las herramientas de programación para los LEGO Mindstorms[3] conviene hacer una pequeña recapitulación sobre el estado de la robótica a nivel comercial de forma que se puedan comprender las razones del éxito que ha supuesto este "juguete". Se puede afirmar que el LEGO Mindstorms constituye uno de los primeros intentos de difundir los robots a gran escala. De hecho se pueden considerar equivalentes en cuanto a difusión, a lo que fue la familia de los ZX de Sinclair en su día en el mundo de los ordenadores. Al igual que ocurrió entonces, donde habían existido computadores que se vendían en forma de *kit*, como por ejemplo el MIT Altair 8800 (basado en el intel 8080) [5], han existido otros robots que han tenido difusión comercial, como por ejemplo el RugWarrior[2], quizá el más conocido. Este robot, que todavía se comercializa, está basado en el chip 68HC11 de Motorola, uno de los más extendidos en el mundo de la electrónica aplicada al control y de la robótica. Otro ejemplo también basada en el chip 68HC11 es la tarjeta Handyboard[4], diseñada específicamente para la construcción de robots, o la tarjeta española de Mibrobótica[5] que usa el mismo chip. De igual forma, han existido y siguen apareciendo, muchos otros robots "en *kit*" basados en este y en otros chips.

Todos estos robots comparten con los primeros ordenadores el "espíritu de garaje", es decir, el programador tiene que montar su hardware y luego hacer sus programas. Este espíritu es atractivo para muchos de nosotros, pero a la vez hace que muchos potenciales usuarios queden fuera del mundo de robótica por la complejidad y dedicación que implica. En general tener que usar un soldador, un polímetro, etc. son condiciones inaceptables para el gran público.

[1] `http://congreso.hispalinux.es`

[2] `http://barrapunto.com/eventos`

[3] `http://www.legomindstorsm.com`

[4] `http://el.www.media.mit.edu/projects/handy-board/`

[5] `http://www.microbotica.es`

Por supuesto, existen robots que se venden como productos comerciales finales. Estos robots generalmente están destinados a la investigación y su precio es tan elevado (casi siempre por encima de los 6.000 Euros) que los hacen inaceptables para el gran público. Ejemplos típicos de estos robots son los de Real World Interface[6] con robots como el B21, Magellan o la familia ATRV; o los de ActiveMedia Robotics[7] con productos como el Pioneer, Peoplebot, o Amigobot. En esta categoría de robots para centros de investigación también hay algunos más pequeños y asequibles como los suizos de K-Team[8] (Khepera y Koala) o el australiano Eye-Bot[9], pero con el coste de ser productos menos profesionales.

El siguiente estadio tras los garajes en la evolución del mercado de los robots dirigido al gran público es el paso del *kit* de auto-montaje, al producto final, el equivalente al del ZX en el mundo de los ordenadores. En este paso el caso de los LEGO Mindstorms puede resultar engañoso, de hecho es un producto comercial que precisamente está pensado para que los usuarios "construyan" sus modelos. Sin embargo, el aspecto electrónico del producto es el de producto final, es decir, no hay que soldar ningún elemento, simplemente comprar unas pilas y encenderlo. De la misma forma la parte de programación del robot es un producto final, se entrega un entorno de programación completamente visual muy orientado a los niños, que hace que cualquier usuario, sin ninguna formación informática pueda programar fácilmente el robot.

A la vez que los LEGO Mindstorms han aparecido otros robots destinados al consumo. Por ejemplo, el RL-500[10] diseñado para cortar el césped, Cye[11] capaz de pasar la aspiradora o de llevar una bandeja de una habitación a otra, la mascota de Sony, el famoso perrito AIBO[12], etc. Sin embargo, ninguno de ellos ha alcanzado los miles de unidades vendidos por LEGO, ni ha conseguido la enorme atención de la comunidad del software libre que ha conseguido el MindStorms.

Una vez analizada brevemente la situación de la robótica en el aspecto comercial, la siguiente sección analizará las alternativas de programación de los Lego Mindstorms bajo GNU/Linux, que es el objetivo de este artículo.

2 La programación de los LEGO Mindstorms

LEGO Mindstorms es un producto diseñado como producto de consumo, lo que le dota de una gran calidad y además 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 de los robots. El cerebro de estos robots está basado en un microprocesador Hitachi H8/300 con 32 Kbytes de RAM. Dispone de 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 que sirve de plataforma de desarrollo. Los sensores disponibles son el de luz ambiente, de luz reflejada, de colisión, de temperatura y de rotación.

Los LEGO Mindstorms 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 microprocesador que se denomina habitualmente "ladrillo" o RCX (con la memoria y los puertos de entrada/salida). Además, el *kit* incluye un entorno gráfico de desarrollo y el equipo necesario para descargar el software desde un ordenador personal al ladrillo.

El entorno para el desarrollo de programas incluido en el *kit* Mindstorms 1.5 es un lenguaje gráfico de programación pensado fundamentalmente para niños o adultos sin ninguna preparación informática. Se trata de un entorno muy limitado, y que además no es requisito de ser software libre. Afortunadamente, los LEGO Mindstorms han recibido mucha atención por parte de la comunidad de software libre, con lo cual han aparecido varias opciones diferentes para su programación muchas de las cuales se analizan en el libro de Jonathan B. Knudsen [3]. Entre ellas destacan dos, NQC y LegOS, que se analizan en las siguientes secciones.

2.0.1 NQC

El acrónimo NQC [1] 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.

[6] `http://www.irobot.com/`
[7] `http://www.activrobots.com/`
[8] `http://www.k-team.com`
[9] `http://www.ee.uwa.edu.au/~braunl/eyebot/`
[10] `http://friendlyrobotics.com/`
[11] `http://www.personalrobots.com`
[12] `http://www.sony.com/aibo`

El aspecto de un programa en NQC es el siguiente:

```
/* Ejemplo en NQC */
int tiempo, giro;

task main() {
  SetSensor(SENSOR_1,SENSOR_TOUCH);
  SetSensor(SENSOR_2,SENSOR_TOUCH);
  start avanzar;

  start evitar;
}

task avanzar(){
  while(true){
    OnFwd(OUT_A+OUT_C);
  }
}

task evitar(){
  while(true) {
    if (SENSOR_1 ==1) {
       stop avanzar;
       OnRev(OUT_C); Wait(50);
       start avanzar;
    }
    if (SENSOR_2 ==1) {
       stop avanzar;
       OnRev(OUT_A); Wait(50);
       start avanzar;
    }
  }
}
```

Este programa realiza el control del robot de la Figura 1. Este robot tiene dos "palpos" que al tropezar con algún escalón presionan un sensor, detectando el peligro. El objetivo es conseguir que el robot pasee aleatoriamente por una zona, por ejemplo sobre una mesa, sin caerse por ningún posible escalón.

El programa consta de tres tareas, una principal que configura el sistema (indica en que puertos están conectados los sensores) y arranca las otras dos (`main`). La segunda tarea hace avanzar en línea recta al robot (`avanzar`) y la tercera detecta los choques, y según el sensor que se haya activado hace girar al robot hacia atrás durante medio segundo.

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.

Figura 1: Robot usado para los ejemplos

A pesar de estas limitaciones, NQC es un lenguaje muy popular entre los usuarios de baja formación informática, debido sobre todo a su simplicidad, pero también a la abundante documentación y a la existencia de herramientas como el *RCX Command Center* para MS-Windows que facilitan el desarrollo y la descarga de programas. En el caso de GNU/Linux, se dispone de un compilador que produce código directamente descargable en el robot, pudiendo utilizarse como editor cualquiera que edite texto ASCII, como por ejemplo Emacs.

Sin embargo, si se quieren realizar programas que requieren cierto almacenamiento de datos, por ejemplo crear un mapa, o simplemente que sean más grandes, NQC no es opción y hay que utilizar un lenguaje de programación real (C, C++, Forth, Ada, etc.) lo cual obliga a cambiar el sistema operativo del ladrillo y en consecuencia a usar LegOS.

2.0.2 LegOS

LegOS es un sistema operativo libre diseñado para el LEGO Mindstorms, diseñado e implementado fundamentalmente por Markus Noga [4]. 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.

LegOS es solo el sistema operativo, lo que quiere decir que se necesita el soporte de un compilador capaz de generar código para el H8 a partir del lenguaje de programación que deseemos y para el que existan las librerías adecuadas de LegOS. Para ilustrar el aspecto de dichas librerías se incluye a continuación un ejemplo del mismo programa anterior realizado en esta ocasión en C:

```
#include <conio.h>
#include <unistd.h>
#include <dsensor.h>
#include <dmotor.h>

/*Declaro una función que se ejecutará al detectar la colisión*/
wakeup_t colision(wakeup_t dato);

int main(int argc, char *argv[]) {
  int dir=0;

  while(1) {
    /* arranco
       - Fijo velocidad*/
    motor_a_speed(MAX_SPEED);
    motor_c_speed(MAX_SPEED);
    /* - Avanzo*/
    motor_a_dir(fwd);
    motor_c_dir(fwd);

    /* Espero */
    wait_event(&colision,0);
    /* Me apunto en que lado fue la colisión*/
    if(SENSOR_1<0xf000)
      dir=0;
    else
      dir=1;

    /* reculo */
    motor_a_dir(rev);
    motor_c_dir(rev);

    /* - el ratito que reculo*/
    msleep(500);

    motor_a_speed(MAX_SPEED);
    motor_c_speed(MAX_SPEED);

    /* Una vez reculado, ahora giro un pelín*/
    if(dir==1) {
      motor_c_dir(fwd);
    } else {
      motor_a_dir(fwd);
    }
    /* - ratito para girar */
    msleep(500);
  }
}

wakeup_t colision(wakeup_t dato) {
        lcd_refresh();
        /* Los sensores están conectados a los puertos 1 y 2 */
        return SENSOR_1<0xf000 || SENSOR_2<0xf000;
```

```
}
```

La estructura del programa es equivalente a la del anterior en NQC, pero en esta ocasión sólo se utiliza una tarea que espera en un bucle infinito a que se produzca un evento, la pulsación del sensor, para realizar el giro. Se han añadido suficientes comentarios al código como para que no merezca la pena entrar en más detalles de su descripción.

El entorno de programación bajo GNU/Linux incluye el compilador de C de GNU (gcc) compilado como cruzado para el Hitachi H8, para lo que hace falta usar las `binutils`. La distribución para GNU/Linux de LegOS incluye varias herramientas que permiten descargar el código de forma dinámica, descargar el *firmaware* o sistema operativo, así como varios ejemplos.

Además, 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 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[13] basado en las `binutils` y el compilador `gcc`. También se han utilizado otras herramientas como los simuladores. Algunas de estas herramientas son:

3.1 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[14] se distribuye bajo licencia MPL, es decir, es software libre. Sus diseñadores e implementadores principales han sido Frank Mueller, Thomas Röblitz, y Oliver Bühn.

3.2 EmuLegOS

EmuLegOS[15] 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 2. 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.

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

[14] `http://www.informatik.hu-berlin.de/$\sim $mueller/legosim/`

[15] `http://www.geocities.com/$\sim $marioferrari/emulegos.html`

Figura 2: Simulador Emulegos

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.

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.

3.3 WebLegos

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

4 Conclusiones

Este artículo se centra en presentar la programación mediante software libre de los robots LEGO Mindstorms. Para ello comienza dando un repaso a la situación emergente de la robótica comercial, centrándose especialmente en el papel de los LEGO Mindstorms. A continuación se aborda su programación bajo GNU/Linux, presentando el lenguaje NQC y el sistema operativo LegOS haciendo especial énfasis en las herramientas relacionadas utilizables bajo GNU/Linux. Como conclusión, se puede afirmar que LegOS es la opción más recomendable, en primer lugar por ser una opción completamente libre, en el caso de NQC el sistema operativo sigue siendo el propietario de LEGo. Pero además porque aporta la flexibilidad suficiente y necesaria en cualquier desarrollo. Por último, el soporte bajo GNU/Linux es el más adecuado.

Referencias

[1] **Dave Baum**. *Dave Baum's Definitive Guide to LEGO Mindstorms.* Apress, USA, 1999.

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

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

[4] **Markus L. Noga**. *LegOS: Open-source embedded operating system for the LEGO Mindstorms.* http://www.noga.de/legOS/.

[5] **M. V. Wilkes**. *Computing Perspectives.* Morgan Kaufmann Publishers Inc., 1995.