Simulador de un robot autónomo para las prácticas de ensamblador de la asignatura de Estructura de Computadores

Pedro A. Castillo[1], A. Cañas[1], A. Prieto[1], Juan J. Castillo[2]
[1] Dpto. de Arquitectura y Tecnología de Computadores
Universidad de Granada
18071 Granada
e-mail: {pedro,acanas,aprieto}@atc.ugr.es
[2] Dpto. de Tecnología
IES Europa
Aguilas, Murcia
e-mail: jj__cv@teleline.es

Resumen

La enseñanza del lenguaje ensamblador plantea problemas debido a la dificultad que supone a los alumnos enfrentarse a un lenguaje distante al dominio de las aplicaciones.

Con los sistemas operativos de interfaz gráfica y los lenguajes de programación visuales, cada vez resulta más difícil hacerles ver la necesidad de estudiar a bajo nivel la estructura del computador y más concretamente el lenguaje ensamblador.

La experiencia nos dice que plantear unas prácticas atractivas al alumno les hace interesarse por dicho lenguaje y por la asignatura de Estructura de los Computadores I (en la que se estudia, en la Universidad de Granada).

La programación de un controlador de un robot ha supuesto un incentivo y ha despertado el interés de muchos alumnos del segundo curso de la Ingeniería Informática por dicha asignatura.

1. Motivación

La asignatura de Estructura de los Computadores I, en la Universidad de Granada, estudia el lenguaje máquina, centrándose en la arquitectura de la familia 80x86. Las prácticas de esta asignatura se basan, principalmente, en el estudio del lenguaje ensamblador.

Debido a la facilidad de uso de los sistemas operativos actuales y la potencia de los lenguajes de programación visuales, resulta difícil justificar unas prácticas a nivel de lenguaje ensamblador. A pesar de las ventajas que presenta este lenguaje en cuanto a poder realizar programas completamente adaptados a la máquina y por tanto optimizados en cuanto a velocidad y utilización de recursos, su estudio debe plantearse de forma que sea lo más atractivo y cómodo para el alumno.

En este trabajo se presenta una herramienta software que simula un robot autónomo que debe ser controlado mediante un programa externo, tan complejo como se desee (podría resolver laberintos, esquivar objetos, limpiar residuos, etc).

El robot que se simula posee tres sensores orientados hacia adelante y hacia los dos laterales que calculan la distancia hasta un objeto cercano en las direcciones respectivas (teniendo en cuenta un alcance máximo), y se mueve en su entorno gracias a dos motores independientes que reciben un valor que indica cuánto avanza cada rueda: valores iguales implican un avance en la dirección actual, mientras que valores diferentes implican la realización de un giro.

El programa simula los movimientos a partir de las órdenes que le transmita nuestro programa de control externo. Para ello, ambos programas se sincronizan a través de dos ficheros de texto, donde el simulador escribe los valores leídos por los tres sensores (según la distancia a los obstáculos), y donde nuestro programa escribirá los dos valores de actuación sobre los motores.

Además de la implementación de los accesos a los ficheros y de la sincronización, se han diseñado las reglas que definen cómo actuará el robot dependiendo de los obstáculos detectados.

Este primer año, los alumnos han estado sumamente motivados en la realización de las prácticas, implementando diversas estrategias y comportamientos para el robot dependiendo de su entorno.

El programa de control se ha desarrollado en lenguaje ensamblador, con objeto de que, primero, aprendan el lenguaje y la arquitectura 80x86, y, en segundo lugar, para utilizar dicho programa en un robot real, controlado por un computador basado en esta arquitectura, y que actualmente se encuentra en las últimas fases de desarrollo. Una vez que tienen nociones básicas de ensamblador, pueden programar los puertos de entrada/salida para la lectura de los sensores y la actuación sobre los motores.

El simulador ha sido desarrollado en C++, utilizando la biblioteca de programación gráfica ***gtkmm2*** [1,2], de forma que puede ser compilado y utilizado fácilmente, tanto en Windows como en Linux/Unix.

El resto del artículo está estructurado como sigue: en la siguiente sección se describe el simulador desarrollado; la tercera sección describe el funcionamiento del controlador básico utilizado; y por último, la cuarta sección expone una serie de conclusiones, obtenidas a partir de la experiencia del primer año, y algunas líneas de desarrollo y mejoras.

2. El simulador

El sistema consta de dos programas: un simulador que muestra el entorno del robot (programa gráfico que genera los mensajes de salida en el terminal desde el que se lanzó), y otro programa que realiza el control del primero.

El sistema funciona de la siguiente forma:

1. La sincronización entre el programa-control y el simulador se hace a través de dos ficheros (S y A).
2. El simulador inicializa la ventana, dibuja el robot, calcula los niveles de excitación de los 3 sensores, distancia y ángulo respecto al objetivo (en verde), y los escribe en el fichero S.
3. El programa-control, que estaba en un bucle de espera hasta que hubiera datos en el fichero S, lee los 5 valores (activación de los tres sensores, distancia y ángulo respecto al objetivo). Se trata de acceder al fichero en modo de texto e ir leyendo carácter a carácter hasta encontrar un espacio, lo que indicará que se ha leído un número más. Seguiremos leyendo esa línea hasta haber leído los 5 números; después se pasan esas cadenas leídas a números enteros.
4. El control procesa esos valores para generar una orden al robot según dónde estén los obstáculos (se han construido una serie de reglas que tienen en cuenta la cercanía de los obstáculos). La orden al robot consta de dos valores: las cantidades que deben avanzar la rueda izquierda y la derecha.
5. Una vez generados los valores citados, se guardan en el fichero A (en la primera línea, separados ambos números por un espacio). Para ello se transfiere a una cadena cada uno de los números, y se escriben en el fichero en formato texto.

La sincronización por ficheros hace más fácilmente portable el sistema, y al tiempo delimita los aspectos que el alumno debe estudiar para hacer la práctica:

- E/S por pantalla
- E/S por fichero
- Estructuras de control complejas para la sincronización
- Cálculos matemáticos con números enteros
- Transformación de datos entre diversas representaciones
- Toma de decisiones para establecer la estrategia de control a implementar

3. ¿Cómo usar el simulador en prácticas?

Hemos desarrollado una versión del simulador para Linux y otra para Win32, por lo que la práctica puede plantearse en ambos sistemas operativos. De hecho, este primer año, varios grupos de prácticas desarrollaron controladores bajo Linux, mientras que otros utilizaron Windows.

Las versiones del simulador para los diferentes sistemas pueden descargarse de la URL *http://atc.ugr.es/~pedro/docencia/simulador/*

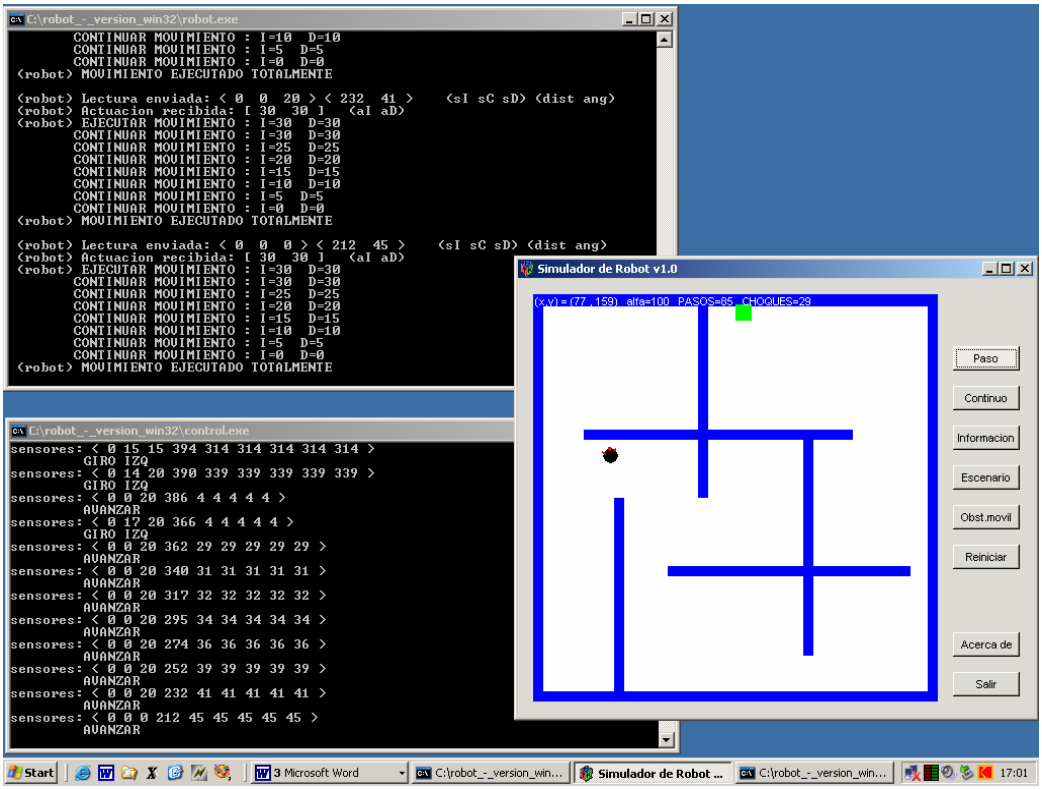

Figura 1. Captura de pantalla en la que el programa-control (ventana de comandos, abajo) envía órdenes al simulador del robot. En este caso, el entorno en el que se está moviendo el robot corresponde a un laberinto. El simulador muestra información detallada de su estado, de las órdenes que recibe, y cómo las ejecuta, continuamente. El cuadro verde corresponde al objetivo al que hay que llevar al robot.

El simulador tiene varios botones para establecer el modo de funcionamiento, cambiar el escenario o hacer que ciertos obstáculos se muevan.

Los distintos botones y su utilidad quedan descritos a continuación:

- "***Paso***": al ejecutar el simulador se encontrará en estado de espera, hasta que decidamos si ejecuta un paso (una orden del programa-control) o se ejecuta de forma continua. En el caso de ejecutar un solo paso, llevará a cabo una comunicación con el control para enviar los valores de los sensores y recibir una orden de actuación. Una vez ejecutada la actuación (se moverá el robot), pasará al estado de espera.
- "***Continuo***": las comunicaciones se llevarán a cabo continuamente, sin pasar al estado de espera (a no ser que pulsemos sobre el botón de "Paso" o el programa control envíe un código de control para detener la simulación).
- "***Información***": el simulador mostrará por la ventana de comandos desde la que se ejecutó cierta información sobre su posición en el área de simulación, la posición del objetivo (en verde), distancias y ángulos entre ellos, el tipo de escenario, si los obstáculos son móviles o no, etc.
- "***Escenario***": cambia el escenario, colocando los obstáculos y el objetivo en diferentes sitios. Los obstáculos móviles en cada escenario son diferentes.
- "***Obst. móvil***": los obstáculos estarán fijos; si pulsamos este botón, algunos de ellos pasarán a desplazarse de su posición original, dificultando el avance del robot.

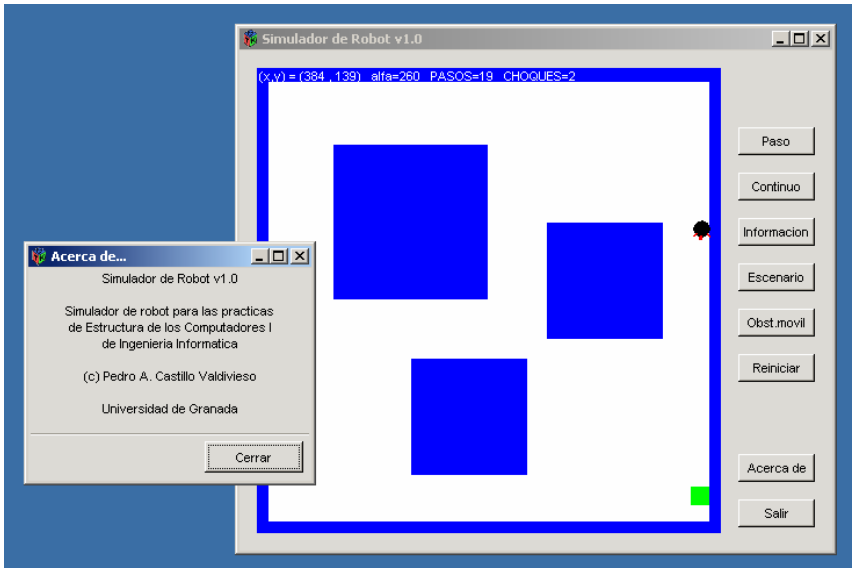

Figura 2. Captura de los controles del simulador.

- "***Reiniciar***": si queremos probar diferentes programas-control bajo las mismas condiciones, el estado del simulador debería ser exactamente el mismo. Así, pulsando en este botón, hacemos que el número de pasos vuelva a cero, el robot a su posición inicial, y los obstáculos a su posición fija.
- "***Acerca de***": muestra información sobre la versión del programa simulador.
- "***Salir***": termina el programa.

4. Conclusión

El aprendizaje del lenguaje ensamblador suele resultar complicado a los alumnos, así como llegar a hacerles ver la necesidad de su estudio. Sin embargo, plantearles unas prácticas amenas y atractivas puede motivarlos, evitando que les resulte tan engorroso el estudio a tan bajo nivel de la programación.

Este trabajo presenta un simulador de robot desarrollado con esta idea. El alumno debe desarrollar un programa de control que reciba los valores obtenidos por los sensores del robot, y le envíe órdenes para que se desplace por su entorno, de acuerdo a la estrategia implementada en el programa de control.

Las experiencias este primer año han sido muy positivas: un alto porcentaje de los grupos de prácticas de la asignatura de Estructura de Computadores I desarrollaron programas de control para el simulador utilizando el lenguaje ensamblador, tanto bajo Linux como bajo Windows. Es significativo que muchos alumnos decidieran hacer la práctica en Linux (a pesar de no tener buenos conocimientos de este sistema), ya que supuso un mayor esfuerzo por su parte. Sin embargo, les resultó muy estimulante y motivador ver cómo sus programas controlaban el robot y resolvían laberintos, o detectaban y esquivaban objetos.

5. Trabajos y líneas de desarrollo futuras

Está en desarrollo un robot real (no simulado), basado en un microprocesador Intel 8086, para implementar el sistema presentado. En ese robot se podrán ejecutar los programas de control desarrollados por los alumnos.

Referencias

[1] http://www.gtkmm.org
[2] http://gtkmm.sourceforge.net/gtkmm2