PLATAFORMA PARA EL DISEÑO DE DRIVERS SOFTWARE DE ALTO Y BAJO NIVEL PARA DISPOSITIVOS HARDWARE

G. GLEZ. DE RIVERA, R. RIBALDA, E. ANGUIANO Y J. GARRIDO
Departamento de Ingeniería Informática. Escuela Politécnica Superior
Universidad Autónoma de Madrid. España

Se presenta una plataforma hardware que permite la experimentación en varias materias dentro de la electrónica y en el ámbito académico de un centro de enseñanza superior. Por ejemplo materias relacionadas con el control de dispositivos electrónicos desde el PC, el diseño de drivers, la programación de microcontroladores o el uso de protocolos de comunicación serie y paralelo. Con esta plataforma se facilita la modificación de las prácticas a realizar curso tras curso en los diferentes laboratorios donde se aplique y sobre todo sin necesidad de dotar cada año al laboratorio de un nuevo material.

1. Introducción

En la titulación de Ingeniería Informática de la Escuela Politécnica Superior de la Universidad Autónoma de Madrid se imparte, en el primer cuatrimestre del segundo curso, una asignatura denominada Estructura y Tecnología de Computadores II. En esta asignatura se pretende que los alumnos comprendan la arquitectura del PC, el uso de interfaces y el lenguaje ensamblador de la familia xx86 de Intel. Desde el curso 1997-98 en el laboratorio de dicha asignatura [1] las prácticas consisten, no sólo en desarrollar programas en lenguaje ensamblador como había sido hasta entonces, sino en el control de diferentes interfaces y periféricos que, por sencillez, se conectan al puerto paralelo o al serie. Este tipo de prácticas llevan consigo dos inconvenientes claros, por un lado los estudiantes aún no poseen conocimientos suficientes para el diseño del hardware y por otro, este diseño no es el objetivo principal de la asignatura.

Por estas razones en los últimos años se les ha proporcionado dicho material *hardware* ya montado. Cada curso se hace necesario modificar las prácticas con el consiguiente cambio del *hardware*, con lo que en esta situación surge un nuevo inconveniente: el coste, tanto económico como en horas de trabajo en la elaboración de nuevas propuestas. La solución que se presenta en este artículo es el diseño de una plataforma genérica que permita resolver tanto el diseño hardware como la actualización y adición de nuevos dispositivos sobre los que desarrollar las prácticas curso tras curso.

2. Antecedentes

El presente desarrollo nace de la experiencia de un laboratorio de una asignatura relacionada con la arquitectura del PC como sistema digital genérico basado en microprocesador.

El desarrollo de este laboratorio ha pasado por varias etapas. En su primera etapa, las prácticas consistían en la elaboración de una serie de programas escritos en lenguaje ensamblador que utilizaban los diferentes recursos y periféricos disponibles dentro del PC, basados en microprocesadores de la familia 80xx86.

Con este tipo de prácticas los alumnos veían al PC como una “caja negra”, que sólo servía para resolver problemas que estuvieran inicialmente previstos en lugar de verlo como un sistema digital genérico, abierto al exterior a través de sus múltiples buses y puertos.

Por todo lo anterior, en una segunda etapa se decidió realizar prácticas utilizando elementos externos al propio PC, conectando nuevos dispositivos y desarrollando aplicaciones sobre ellos. De esta forma se eliminaba la visión del PC como un sistema cerrado. Desde el principio se descartó el

bus ISA por su complejidad y posible fuente de averías y se eligió el puerto paralelo por su mayor sencillez y accesibilidad.

Así, en el primer curso con este sistema, el desarrollo del laboratorio consistía en la entrega de tres prácticas:

- Manejo de recursos y elementos básicos: Introducción al ensamblador. Realizaban un sencillo programa escrito en lenguaje ensamblador. Les permitía familiarizarse con este tipo de lenguajes y conocer las diferentes herramientas de desarrollo y depuración.
- Control del Hardware, recursos avanzados: Se diseñaron sencillos periféricos que se conectaron al puerto paralelo y se controlaban desde el lenguaje ensamblador. Algunos periféricos fueron pantalla LCD, generador de señal (tarjeta de sonido), controlador de 8 relés, lector de tarjetas chip de Telefónica, control de motores paso a paso y manejo de varios displays de 7 segmentos.
- Interfaz con el lenguaje C. Se desarrollaba una aplicación en lenguaje C que hacía uso de los programa escritos en la práctica anterior en lenguaje ensamblador.

Mantener tantas prácticas era costoso y, en años sucesivos, se fueron complicando los diseños pero reduciendo la variedad, hasta llegar a una tercera etapa en la que había un único diseño hardware o proyecto. El desarrollo del curso para la ejecución del proyecto se dividió en las siguientes prácticas:

- Introducción al lenguaje asm: Diseño de programas sencillos, permiten el manejo tanto del ensamblador como de las herramientas. Al alumno se le entrega un emulador del dispositivo hardware, de forma que puedan ir desarrollando el proyecto aunque aún no lo tengan hecho.
- Diseño y control del periférico: El alumno realiza el montaje del hardware a partir de un esquema que se le proporciona y posteriormente desarrolla aplicaciones software para su control y manejo, es decir, se escriben *drivers* que acceden a dicho hardware. Este software tiene la misma estructura que el emulador de la práctica anterior, de esta forma utilizando la práctica primera y sustituyendo el emulador por el HW diseñado, todo debería seguir funcionando.
- Diseño de aplicaciones que usen el HW desde lenguajes de alto nivel: Se estudia la interfaz entre el lenguaje C y el ensamblador. Se desarrollan aplicaciones más complejas y las llamadas al HW se hacen desde ensamblador, llamadas escritas en las prácticas anteriores.

Con esta metodología se asentó la estructura mostrada en la Fig. 1, en la que la única conexión entre el hardware y la aplicación que lo utiliza es un programa intermedio, el *driver*. De esta forma el alumno tiene una idea clara de la verdadera utilidad de un *driver*, hacer que el diseño *hardware* sea independiente de la aplicación que lo utiliza. De esta forma, un cambio o actualización en el HW no implica que se tenga que desarrollar una nueva aplicación o modificar la ya existente, basta con actualizar el *driver*.

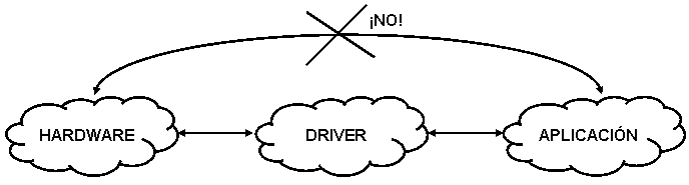

Figura 1. Uso del *Driver* Software.

Se desarrollaron los siguientes proyectos:

- Sistema de adquisición de datos analógicos: Voltímetro digital.
- Sistema de adquisición de datos analógicos: Osciloscopio de 1 canal.
- Sistema de adquisición de datos de 4 canales: Osciloscopio de 4 canales.

Con estas prácticas surgió un nuevo problema: algunos alumnos tenían bastantes problemas en realizar los montajes *hardware* necesarios, pues aún no están suficientemente preparados para ellos y tampoco disponen de las herramientas necesarias al tratarse de una laboratorio de programación. Debido a estos problemas, en la cuarta etapa, en el curso 2003-2004 se propuso la idea de entregar el hardware completamente montado y probado. De esta forma su tarea es exclusivamente desarrollar las aplicaciones y el *driver* que lo controle. Se seguía ofreciendo toda la información necesaria por si algún alumno quería hacer desde el principio, pero ya no era obligatorio.

El proyecto a realizar ese curso fue un analizador lógico con generador de patrones, cuyo esquemática se muestra en la Fig. 2. Se diseñó y fabricó una placa de circuito impreso y a través de los miembros de una asociación de estudiantes, se montó y se hizo llegar a todos los alumnos que la solicitaron.

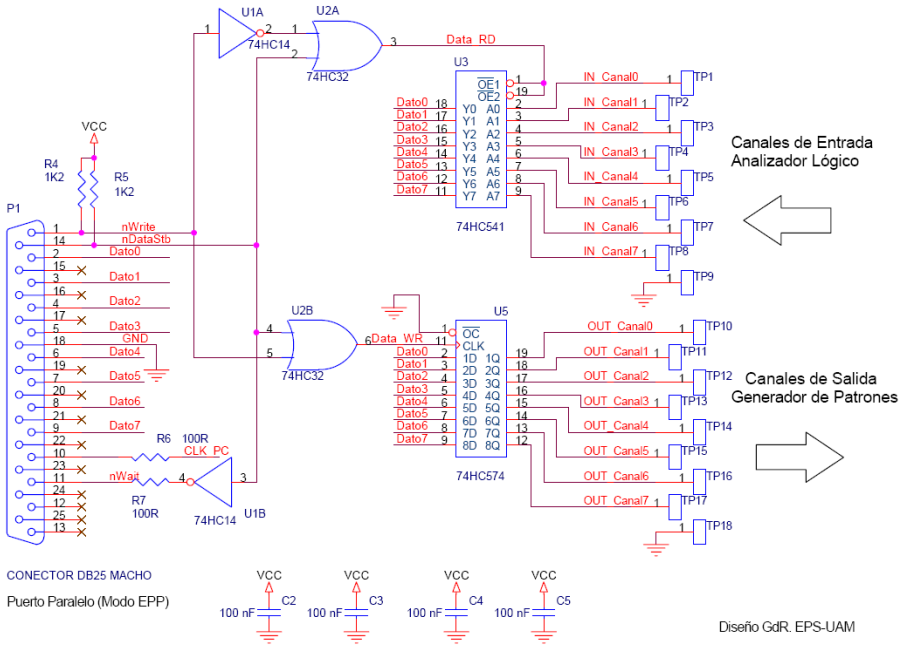

Figura 2. Esquema del circuito del Analizador Lógico y Generador de Patrones.

Tal como se comentó en la introducción, en esta situación surge un nuevo inconveniente: el coste, tanto económico como en horas de trabajo en la elaboración de nuevas propuestas. Cada curso se hace necesario modificar las prácticas con el consiguiente cambio del *hardware*. La solución propuesta es el diseño de una plataforma genérica que permita resolver tanto el diseño *hardware* como la actualización y adición de nuevos dispositivos sobre los que desarrollar las prácticas curso tras curso: la plataforma GDriver.

3. Descripción de la Plataforma GDriver

Como ya se ha mencionado en la introducción, este desarrollo nace de la experiencia en un laboratorio de una asignatura de programación en ensamblador. La idea central de las prácticas es el desarrollo de un *driver* para un dispositivo *hardware* determinado con todo lo que conlleva,

incluyendo un ejemplo de uso, desarrollando una aplicación de usuario que lo utilice. Las Fig. 3 y 4 muestran de forma esquemática esta situación.

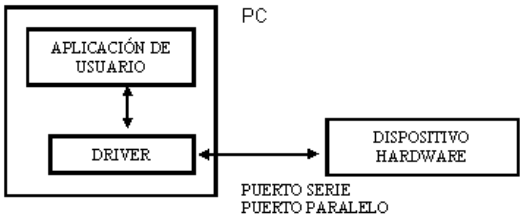

Figura 3. Esquema de trabajo

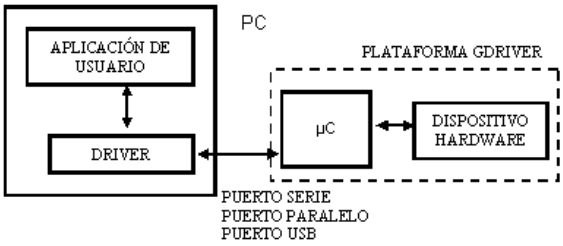

Figura 4. Solución propuesta

La plataforma GDriver, basada en el microcontrolador 68HC908JB16, de la familia 68HC908 de Motorola, de la misma familia al utilizado en [2] usado en otras asignaturas de la titulación, presenta por un lado una comunicación Serie (RS-232 y RS-485), Paralelo (SPP, EPP y ECP) y/o USB y por el otro dispone de una serie de líneas de entrada/salida que permiten conectar cualquier dispositivo. En la propia tarjeta ya se incluyen un conjunto de ellos, como son pantalla LCD de texto y gráfica, displays de 7 segmentos, potenciómetros digitales, conversores A/D y D/A, teclado, *driver* de potencia para motores, registros de entrada/salida, etc. Estos periféricos han sido ya descritos y utilizados en otros trabajos [3].

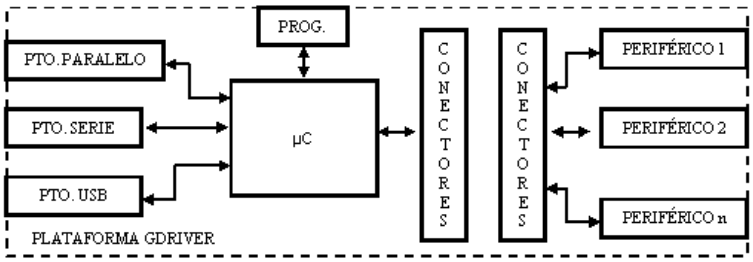

Figura 5. Diagrama de bloques de la plataforma GDriver

La Fig. 5 presenta el diagrama de bloques de la solución propuesta para el laboratorio descrito al comienzo [5], [6] y [7]. Consta de una serie de periféricos que no están conectados entre sí, ni tampoco están conectados al bus de I/O. Sólo están conectados los pines de alimentación, el resto de líneas están disponibles a través de una serie de pines de *wrapping* de manera que el alumno los puede conectar donde él decida: bien entre ellos o bien a los pines que seleccione de los puertos libres del microcontrolador.

Los elementos o periféricos que se han incluido (ver Fig. 6) son:

- Pantalla de cristal líquido (LCD). Dispone de 2 líneas y 12 columnas. Los terminales del contraste están disponibles, de forma que se puede conectar un potenciómetro clásico o bien uno programable (en la placa descrita se encuentran ambos). El controlador del LCD es compatible con el estándar LSI HD44780.
- Array de 4 displays de 7 segmentos, dispuestos en una configuración de ánodo común.
- 16 pulsadores configurados como una matriz de 4x4.
- 4 pulsadores independientes.
- 4 interruptores.
- 8 diodos LED.
- 8 microinterruptores en formato DIL.
- 2 circuitos integrados MAX6817, de Maxim [4]. Cada uno de ellos integra dos eliminadores de rebotes, útiles para filtrar las salidas de interruptores y pulsadores.

- Un circuito integrado DS1806, de Dallas [4]. Está compuesto por 6 potenciómetros de 64 posiciones, controlados por un bus de 3 hilos.
- Dos circuitos integrados DS1804, de Dallas [4]. Cada uno de ellos es un potenciómetro digital de 100 posiciones, no volátil. El control es por incremento/decremento a través de dos líneas.
- Dos amplificadores operacionales, en una configuración de amplificador de ganancia variable. Se puede conectar una resistencia fija o cualquiera de los potenciómetros descritos anteriormente.
- Dos señales analógicas, que pueden variar entre 0 y 2.5 voltios a través de sendos potenciómetros.
- Dos *drivers* de potencia del tipo L293D, útiles para controlar motores.
- Una zona de *wrapping*, para incluir cualquier otro periférico. Esto permite al alumno el poder incluir cualquier otro elemento de manera que no le limite su diseño.

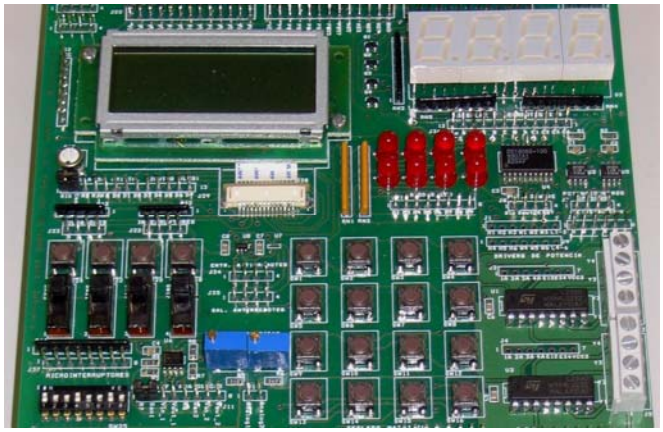

Figura 6. Detalle de algunos periféricos de la tarjeta.

La idea es entregar esta tarjeta al alumno con el microcontrolador ya programado para un dispositivo concreto. Dicho programa lo que hará es hacer que la tarjeta GDriver aparezca como "transparente", como si no estuviera. El alumno debe programar el *driver* como si fuera el dispositivo el que estuviera conectado al puerto directamente. En este sentido la plataforma dispondría de dos conjuntos de elementos claramente diferenciados: los dispositivos transparentes (que podrían estar directamente conectados a los interfaces del PC) y los no transparentes.

En el caso de un laboratorio de microprocesadores, la plataforma se entregaría sin programar. El objetivo sería controlar desde dicha tarjeta un *hardware* determinado, bien que ya estuviera en la placa o bien que se pudiera conectar externamente a través de los buses de expansión. En este caso no estaría conectada a un PC.

4. Uso de la Plataforma

Se ha identificado claramente que esta plataforma será una herramienta útil, no sólo para la asignatura descrita, sino para todas aquellas en las que estén involucrados los siguientes elementos:

- Desarrollo de *drivers* de dispositivos electrónicos desde el PC en ensamblador o en lenguajes de alto nivel. El microcontrolador ya estará programado por el profesor.
- Desarrollo de *drivers* desde el PC bajo diferentes sistemas operativos (MS-DOS, Linux, Windows, etc). En este caso el microcontrolador ya estará programado por el profesor.
- Desarrollo de *drivers* desde un microcontrolador, prácticas con microcontroladores.
- Manejo de diferentes tipos de puertos: serie, paralelo y USB
- Pruebas de dispositivos con diferentes protocolos serie: RS-232 y RS-485

- Pruebas de dispositivos con diferentes protocolos paralelo: SPP, Bi-direccional, EPP y ECP. Emulación de estos protocolos en un microcontrolador que no dispone de ellos.

5. Conclusiones

Mediante el uso de la plataforma GDriver se pretende ofrecer una gran diversidad de prácticas así como de materias a impartir, en función de las necesidades de cada uno.

Tras una primera inversión, se puede utilizar en cursos sucesivos sin necesidad de reinvertir en nuevos diseños y con la posibilidad de implementar un conjunto amplio de prácticas y para diferentes asignaturas.

Se pueden hacer desarrollos que no sólo utilicen un dispositivo sino un conjunto de ellos de tal forma que el abanico de posibilidades es muy amplio.

Se puede utilizar en diferentes asignaturas relacionadas con la programación de microcontroladores, instrumentación electrónica, gestión de protocolos de comunicación, etc.

Referencias

[1] http://www.eps.uam.es/~gdrivera/labetcii/labetcii.htm

[2] G. González de Rivera, S. López-Buedo, I. González, C. Venegas, J. Garrido y E. Boemo *GP_BOT: Plataforma Hardware para la enseñanza de Robótica en Ingeniería Informática*. Tecnologías Aplicadas a la Enseñanza de la Electrónica (TAEE'02). Pg 67-70 Univ. de las Palmas de Gran Canaria, España 2002

[3] G. González de Rivera, J. Garrido. *Diseño de un laboratorio de Sistemas Electrónicos Digitales*. Tecnologías Aplicadas a la Enseñanza de la Electrónica (TAEE'04). Universidad Politécnica de Valencia, España. Julio de 2004.

[4] http://www.maxim-ic.com

[5] A. Santos, E. Boemo, J. Faura, y J. Meneses, "Microcontrollers in Education", *Proc. IEEE 24th Frontiers on Education Conference*, San Jose, 1994. Disponible en http://www.ii.uam.es/~ivan.

[6] M. Barrón y J. Martinez, "Equipo Didáctico para la Familia de uC 8051", *Actas TAEE´98*, Publicaciones UPM, 1998.

[7] B. Martín y C. Bernal, "Visual 11 y Kit11: Herramientas para el aprendizaje del MC86HC11 de Motorola", *Actas TAEE´98*, Publicaciones UPM, 1998.