LABORATORIO DE SISTEMAS INFORMÁTICOS EN TIEMPO REAL. UN ENFOQUE MULTIDISCIPLINAR

J.D. MUÑOZ FRÍAS, S. ALEXANDRES FERNÁNDEZ, J.A. RODRÍGUEZ MONDÉJAR, C. FERNÁNDEZ MARTÍNEZ
Departamento de Electrónica y Automática. Escuela Técnica Superior de Ingeniería (ICAI)
Universidad Pontificia Comillas de Madrid. España.

En este artículo se presenta el enfoque seguido en el diseño del laboratorio de Sistemas Informáticos en Tiempo Real impartido en la Escuela Técnica Superior de Ingeniería (ICAI) de la Universidad Pontificia Comillas. La principal innovación docente llevada a cabo ha consistido en usar un autómata programable como hilo conductor de las prácticas.

1. Introducción

Es muy frecuente en los laboratorios docentes el usar una serie de prácticas aisladas entre sí que, aunque se diseñan para afianzar los conceptos estudiados en teoría, suelen ser en cierto modo algo artificiales. Por poner un ejemplo, las prácticas impartidas en esta asignatura en dos universidades españolas elegidas al azar son las siguientes:

Universidad 1:

- Familiarización con el Entorno de Trabajo de las Prácticas (UNIX)
- Prácticas de Programación sobre Linux de Algoritmos Simples Basados en POSIX
- Prácticas de Programación sobre Linux con *threads* de POSIX
- Ejemplos Sencillos de Procesamiento en Tiempo Real
- Diseño e Implementación de una Aplicación de Tiempo Real

Universidad 2:

- Introducción a un sistema operativo de tiempo real
- Procesos
- Planificación de procesos
- Comunicación entre procesos 1: Pipes
- Comunicación entre procesos 2: Semáforos
- Comunicación entre procesos 3: Memoria compartida
- Comunicación entre procesos 4: Mensajes
- Control de tiempo. Temporizadores
- Manejo de Interrupciones (2 prácticas)
- Manejo de la Entrada/Salida serie
- Gestión de Prioridades

Como se puede apreciar, en ambos casos no existe un hilo conductor claro que dé continuidad a las prácticas, ni una aplicación clara que motive al alumno viendo que construye algo útil para el mundo real.

En la presente comunicación se presenta el diseño de un laboratorio en el que se ha buscado una aplicación industrial que sea el hilo conductor de las prácticas. Dicha aplicación ha sido un autómata programable. Así pues, el laboratorio va a consistir en el diseño de un autómata programable, con prestaciones similares a las de uno industrial, a partir de una placa de desarrollo basada en un microcontrolador ColdFire. Las características de la placa usada se muestran en el apéndice.

2. El autómata programable como diseño guía para el laboratorio

Como se ha mencionado en la introducción, el laboratorio diseñado no solo pretende afianzar los conocimientos expuestos en teoría, sino a la vez conseguir motivar al alumno mediante el diseño de un sistema real. Además se pretende inculcar al alumno la idea de que las asignaturas no son "entes" aislados, sino que en el mundo industrial es necesario aplicar varias disciplinas para llegar al producto final.

El sistema elegido como guía para el laboratorio ha sido un autómata programable. Esta elección tiene dos ventajas. Por un lado, es un sistema que se presta fácilmente a la aplicación de las distintas técnicas de programación en tiempo real. Por otro lado, es un sistema que el alumno ya ha estudiado o estudia en paralelo en la asignatura de automatización, lo que permite al alumno profundizar más en esta asignatura, a la vez que motivarlo al ver que es capaz de diseñar un sistema equivalente al que ofrecen fabricantes como Siemens u Omron.

El planteamiento que se ha seguido para diseñar las prácticas ha sido el de partir de un diseño sencillo e ir añadiendo funcionalidades que obligan a mejorar las técnicas de programación en tiempo real aplicadas. En los siguientes apartados de detallan las prácticas realizadas junto con los conceptos de programación en tiempo real que pretenden ilustrar. Los enunciados completos de las prácticas así como el resto del material de la asignatura están disponibles en la página web de la asignatura [Muñoz06].

2.1 Práctica 1: Diseño de sistemas basados en bucle de *scan*

El bucle de *scan* es la técnica más básica para programar un sistema en tiempo real [Simon99]. En esta práctica el alumno toma contacto con esta técnica implantando un bucle de *scan* que se encarga de leer el valor de las entradas, interpretar el programa del PLC y actualizar las salidas.

Para especificar el programa del PLC se ha usado un lenguaje basado en texto similar al usado por los antiguos TSX de Telemecánica. En la actualidad los PLCs se suelen programar usando lenguajes gráficos, pero implantar un sistema de este tipo se sale obviamente de los objetivos de este laboratorio introductorio. El lenguaje usado se basa en usar un acumulador de forma que todas las instrucciones realizan una operación entre un operando y el acumulador, guardando el resultado en el acumulador. Esto tiene como resultado el que las instrucciones tengan un solo operando, lo cual facilita su codificación. Para no complicar en exceso el PLC, en la práctica se implantan solo 5 instrucciones, las cuales se detallan en la siguiente tabla:

Mnemónico	***Argumento***	***Funcionamiento***
L arg	E0-E7, S0-S7	Carga el valor de la entrada o salida indicada en el argumento en el acumulador.
LN arg	E0-E7, S0-S7	Carga el valor **negado** de la entrada o salida indicada en el argumento en el acumulador.
A arg	E0-E7, S0-S7	Realiza un AND entre el argumento y el acumulador.
O arg	E0-E7, S0-S7	Realiza una OR entre el argumento y el acumulador.
= arg	S0-S7	Escribe el valor del acumulador en la salida indicada en el argumento.

La forma de codificar cada instrucción es mediante un Byte, usando los 4 bits más significativos para codificar la instrucción y los 4 menos significativos para codificar el argumento. Esto permite introducir dos nuevos conceptos:

- Cómo se codifican las instrucciones en código máquina, ya que el mecanismo es similar.
- Cómo interpretar campos de bits, ya que el programa tendrá que extraer cada uno de los campos de la instrucción mediante desplazamientos y máscaras.

A modo de resumen, lo que aporta esta práctica al alumno es por un lado el cómo implantar un sistema en tiempo real sencillo usando un bucle de *scan* y por otro lado cómo manipular campos de bits.

2.2 Práctica 2: Diseño de sistemas basados en interrupciones.

La siguiente técnica, según su grado de complejidad, para implantar un sistema en tiempo real es mediante interrupciones: se mantiene un bucle de *scan* para las tareas que no tienen estrictos requerimientos de latencia y se usan interrupciones para aquellas tareas que necesitan responder con una baja latencia a sucesos del *hardware*. Se construye así lo que en inglés se denomina un sistema

foreground/background. En esta práctica se añaden al PLC unos interruptores horarios[1] para obligar al alumno a usar esta técnica para implantarlo. En la práctica es necesario ahora usar una interrupción de tiempo para actualizar un reloj y en *background* se mantiene el PLC diseñado en la práctica anterior con unos pequeños cambios para incluir el tratamiento de los interruptores de nivel en el intérprete del programa del PLC.

La problemática que se añade, aparte de la gestión del *hardware* a bajo nivel para tratar las interrupciones, es la comunicación entre las rutinas de primer plano (interrupción) y las de segundo plano (bucle de *scan*). En esta primera práctica sobre el tema se usan variables globales compartidas para ello. Esto introduce el problema de la coherencia de datos, solucionándose en este caso mediante la inhabilitación de las interrupciones dentro de las zonas críticas.

Otro concepto importante que se introduce en esta práctica es el de la programación modular. Para ello se obliga al alumno a implantar toda la gestión de los interruptores horarios en un módulo aparte (`tiempo.c`). Con esto se consiguen dos cosas:

- Encapsular datos: La comunicación entre las rutinas de interrupción y las de segundo plano se realiza mediante variables globales compartidas. En lugar de hacer estas variables globales públicas para todo el programa, se hacen privadas al módulo (definiéndolas como `static`) y se incluyen en el módulo todas las funciones que trabajan con estas variables. Si es necesario acceder desde fuera del módulo a una variable global basta con crear una función que devuelva el valor de la variable. El objetivo perseguido es el lograr una encapsulación de los datos similar a la obtenida en los lenguajes orientados a objetos.
- Facilitar las futuras ampliaciones. Dado que el laboratorio se basa en una mejora continua de un programa, es conveniente facilitar lo máximo posible las ampliaciones, lo cual se consigue usando estas técnicas de programación modular.

2.3 Práctica 3: Diseño de sistemas basados en interrupciones II.

En esta práctica se añade al PLC una comunicación con un host mediante RS-232 para ilustrar la problemática asociada a la programación de tareas de primer plano con distintos niveles de prioridad. Además se introducen las colas para comunicar rutinas de interrupción con rutinas de segundo plano.

En esta práctica se vuelve a hacer énfasis en la programación modular, obligando al alumno a crear un nuevo módulo (`serie.c`) en donde se introduce toda la gestión de las comunicaciones. De esta forma para añadir este módulo al programa desarrollado en la práctica anterior, lo único que hay que hacer es insertar en el bucle de *scan* la llamada a la función de segundo plano encargada de interpretar las órdenes que llegan por la línea serie.

2.4 Práctica 4: Diseño de sistemas basados en RTOS.

Las técnicas de programación en tiempo real usadas en las prácticas anteriores sólo son válidas para sistemas sencillos. En cuanto se empiezan a añadir tareas con distintos requisitos de prioridad y distintas necesidades de sincronización y comunicación, se hace necesario el uso de un sistema operativo en tiempo real (RTOS). En esta práctica se hace uso de un sistema operativo en tiempo real de *software* libre denominado FreeRTOS [Barry06], que ha sido portado por el autor para el microcontrolador ColdFire MCF5282 usado en el laboratorio. El objetivo de la práctica es mostrar al alumno los servicios que aporta un RTOS y sus ventajas frente al uso de las soluciones usadas en las prácticas anteriores. En concreto, la labor que debe realizar el alumno es la de adaptar la práctica 3 para usar el RTOS. Para ello los cambios que ha de realizar son:

1. Las tareas de segundo plano de las prácticas anteriores eran funciones "normales" que se llamaban desde el bucle de *scan*. Ahora estas funciones han de convertirse en tareas que serán llamadas por el planificador del RTOS. Para ello son necesarios dos pasos: a) inicializar la tarea dentro del RTOS y b) modificar la función para adaptarla a los requisitos del RTOS, que consiste básicamente en crear un bucle sin fin dentro del cuál existirá una llamada al operativo para la sincronización.
2. Se usará un semáforo para sincronizar la interrupción de tiempo con las tareas que tienen que ejecutarse sólo cuando éste cambia, que son la tarea de impresión en el display de la hora y la tarea encargada de actualizar los interruptores horarios.

---

1 Un interruptor horario se conecta automáticamente a una hora y se desconecta a otra. Ambas horas son programables.

3. Se usarán las colas suministradas por FreeRTOS para comunicar y sincronizar la rutina de interrupción del puerto serie con la tarea encargada de procesar la información que llega por el puerto. De esta forma la rutina de segundo plano sólo se ejecuta cuando llegan nuevos caracteres a la cola.
4. La tarea encargada de interpretar el programa del PLC se ejecuta periódicamente con un periodo de 20 ms.

Como puede observarse en la práctica se utilizan todos los modos posibles para controlar la ejecución de tareas de segundo plano que proporciona FreeRTOS: periódica, sincronizada mediante semáforos o sincronizada mediante una cola.

2.5 Integración de sistemas analógicos y digitales

En la última práctica se vuelve a hacer énfasis en la multidisciplinariedad de las asignaturas de la carrera. En concreto se añade al PLC un módulo de interruptores de nivel para controlar el llenado de depósitos, de forma que se cierra el interruptor cuando el nivel es inferior a un umbral programable por el usuario. Para medir el nivel se diseña, en conjunto con el laboratorio de instrumentación, un detector de nivel por ultrasonidos. En el laboratorio de instrumentación se diseña la parte analógica y en el de tiempo real el *software* que gobierna el sensor. Este *software* consta de una tarea periódica que se encarga de activar el sensor y medir la distancia del líquido, abriendo o cerrando el contacto en función de esta distancia.

Esta práctica sirve también para ilustrar la problemática que surge cuando hay que desarrollar en paralelo el *hardware* y el *software* de un sistema, siendo necesario recurrir a simulación en las primeras fases del diseño para suplir el *hardware* que no está disponible en ese momento.

3. Desarrollo temporal

Todas las prácticas tienen una duración de dos sesiones de dos horas, si bien es cierto que requieren de una preparación por parte del alumno de unas cuatro horas. Esto da lugar a 10 sesiones de laboratorio, que junto con dos sesiones de dos horas usadas para introducir la plataforma[1] y que no se han discutido en esta comunicación por ser muy específicas, hacen un total de 12 sesiones de laboratorio; que es el número de sesiones en un cuatrimestre para un laboratorio de 3 créditos.

4. Conclusiones

Aunque el laboratorio lleva funcionando sólo dos cursos, el resultado en cuanto a motivación del alumnado y aprendizaje obtenido ha sido satisfactorio. En una encuesta realizada a final de curso, un 93 % de los alumnos consideraban buena la idea de usar un PLC como diseño guía para el laboratorio y un 72 % consideraba que el nivel de las prácticas era asequible.

5. Agradecimientos

Los autores agradecen a Freescale Semiconductor el apoyo prestado, tanto en *hardware* como *software*, sin el cual este laboratorio no hubiera sido posible.

6. Apéndice. Placa de desarrollo usada en el laboratorio

La placa usada en el laboratorio para realizar las prácticas está basada en el microcontrolador ColdFire MCF8252 de Freescale Semiconductor. Como se puede apreciar en la Figura 1, el sistema consta de una placa de evaluación suministrada por Freescale (M5282LITE), la cual se acopla a una placa de entrada/salida desarrollada por el autor, la cual aporta los siguientes elementos de entrada/salida:

- Dos puertos de entrada de 8 bits.
- Dos puertos de salida de 8 bits.
- Un teclado matricial de 16 teclas.
- Dos entradas analógicas mediante potenciómetros.
- Dos entradas analógicas para señales externas.
- Dos salidas analógicas (los conversores A/D se comunican con el procesador mediante SPI).

1 En el laboratorio se usa una tarjeta basada en el microcontrolador ColdFire. Para programarlo se usa el entorno CodeWarrior. Ambos sistemas son de la firma Freescale Semiconductor.

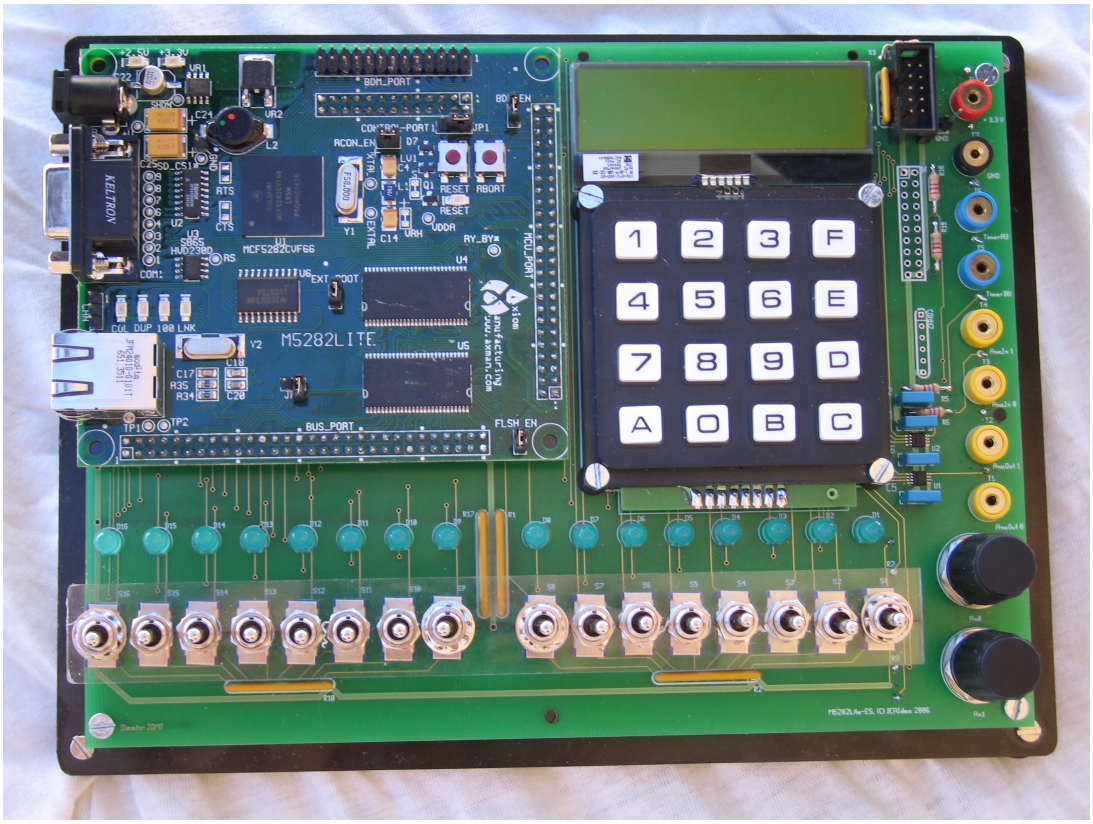

Figura 1: Placa de desarrollo

- Dos salidas PWM.
- Un display de 2 lineas de 16 caracteres (se comunica con el procesador mediante I2C)

Conviene destacar que la placa desarrollada está pensada para poder usarse en un laboratorio de programación de sistemas empotrados, por lo que se ha dotado de una abundante periferia para poder realizar prácticas típicas en este tipo de laboratorios como entradas/salidas digitales, muestreo de teclado, entradas analógicas, etc. Además se han añadido periféricos que usan buses serie estándar (I2C y SPI) para poder introducir al alumno en la problemática de este tipo de comunicaciones, tan usuales hoy en día en los sistemas basados en microcontrolador.

Para realizar las prácticas mostradas en esta comunicación bastaría cualquier placa con 8 entradas digitales y 8 salidas digitales, un puerto serie y un display, lo cual es muy común en el mercado. La última práctica necesita entradas analógicas, aunque si la placa disponible en el laboratorio no dispone de ellas, es posible idear otro tipo de aplicación que mezcle sistemas analógicos y digitales.

7. Referencias

[Muñoz06] José Daniel Muñoz Frías. Página web de la asignatura: Sistemas Informáticos en Tiempo Real. http://www.dea.icai.upco.es/daniel/asignaturas. Última visita: 17/06/2006.

[Simon99] David E. Simon. An Embedded Software Primer. Addison-Wesley. 1999.

[Barry06] Richard Barry. FreeRTOS. A Free RTOS for small embedded real time systems. http://www.freertos.org. Última visita 17/06/2006.