PLATAFORMA PARA LA EXPERIMENTACIÓN EN CONTROL DIGITAL DIRECTO

Esteban del Castillo Pérez.
Departamento de Ingeniería Electrónica, Eléctrica i Automática.
Escuela Técnica Superior de Ingeniería. Universitat Rovira i Virgili, URV.
43006-Tarragona. E.Mail: ecastill@etse.urv.es. Tef: 977-559629

RESUMEN

La implementación de los sistemas digitales de control exige un manejo eficiente del sistema de interrupciones del que deberá estar dotado, necesariamente, el equipo de control. Es ésta una parte crítica que debe someterse a experimentación y que, bajo mi criterio, no conviene exclusivamente simular.

Seguidamente se presenta una plataforma concebida para la enseñanza de los sistemas digitales de control sobre la que, desarrollando una serie de prácticas, facilitaremos el acercamiento de los alumnos al mundo de la automática. Se comenta un software elaborado para esta plataforma que implementa los controles de velocidad y posición

1. INTRODUCCIÓN.

En esta comunicación presentamos un sistema, ya implementado y experimentado, orientado a la dotación de un puesto de trabajo en laboratorio, con el objetivo de desarrollar en él prácticas de control digital directo. El puesto de trabajo incluye un PC, un pequeño hardware externo y un software de control.

Nuestro propósito es implementar un control de la velocidad y posición del eje de un motor de corriente continua atendiendo a las siguientes consideraciones: control PID, excitación por PWM, interfaz a través del puerto paralelo, información de velocidad y posición proveniente de un encoder relativo. El software se ha desarrollado para dos sistemas operativos: MS-DOS y RTOS OS-9000. Este último es una versión portable del conocido OS-9 de Microware, muy extendido en aplicaciones de control industrial. Todas las condiciones de funcionamiento podrán ajustarse antes del ensayo: constantes del controlador, consigna, tiempo de muestreo, frecuencia del PWM, etc.

Para hacer uso de esta plataforma se deberán establecer una serie de prácticas de laboratorio concretas, atendiendo al nivel de conocimientos de los alumnos y a los objetivos que se pretendan alcanzar. Como objetivos, podemos apuntar los siguientes:

- Familiarizarse con los controladores PID digitales (1 ciclo):
  1. Determinando las constantes del controlador a partir de la respuesta del Sistema en lazo abierto (Ziegler-Nichols), y en lazo cerrado haciendo al sistema inestable.
  2. Observando los efectos que sobre un sistema real tiene la actuación sobre las condiciones de funcionamiento del controlador.
  3. Procediendo a la actualización, parcial o total, del algoritmo de control.

- Elaborar o actualizar el software de presentación (interfaz con el usuario) (1 ciclo).
- Elaborar o actualizar los drivers para el control del hardware (2° ciclo).
- Elaborar el software para un control autoajustable (2° ciclo).

Ya que nuestro propósito es establecer un sistema de control totalmente digital que no requiera la adición de ningún tipo de hardware interno al PC, se hace necesario usar técnicas que permitan comunicar al ordenador, como elemento de control, con el sistema a controlar, haciendo uso exclusivo de los recursos básicos de cualquier PC. Concretamente, los requerimientos son tres:

1. Posibilidad de modificar suavemente la tensión de excitación en el motor.
2. Obtener información precisa de la velocidad y posición de su eje.
3. Dotar a la etapa de potencia de un aislamiento galvánico, como medida de seguridad.

2. DESCRIPCIÓN DEL SISTEMA.

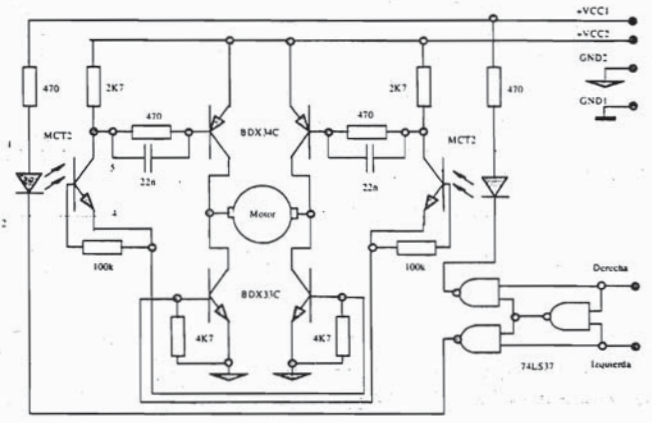
Hacemos una pequeña descripción atendiendo separadamente a cada una de las cuatro partes principales en que podemos dividir nuestro sistema: potencia, sensores, recursos del PC y software de control.

2.1 Etapa de Potencia.

Se ha recurrido al uso de la técnica de Modulación de la Anchura de Pulsos -PWM- para el ataque del motor a través de un puente de transistores. Esto posibilita la inversión de polaridad, necesaria para el control de posición, además de permitir la implementación del aislamiento galvánico con suma facilidad. La interfaz con el PC puede hacerse a través de sólo dos líneas de uno cualquiera de los puertos paralelo de los que siempre hay, cuando menos, uno instalado. Una crítica que podemos hacer a esta técnica es el considerable uso de CPU requerido para poder obtener prestaciones similares a las de una implementación por hardware a partir del uso de una tarjeta DA. En nuestro caso, para una frecuencia de 100Hz podemos fijar la tensión de salida con un error de ±0.6% y un consumo de CPU de tan sólo 14 µs por periodo (0.14%).

El circuito de interfaz, aislamiento y excitación es el siguiente:

Etapa de potencia



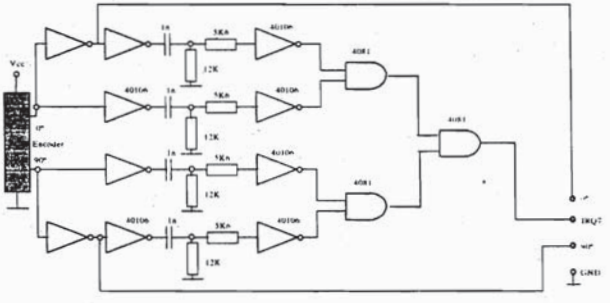
2.2 Sensores.

La información necesaria para los controles de velocidad y posición se obtiene de un encoder relativo. Éste deberá entregar dos señales en cuadratura para poder detectar el sentido de giro e implementar así el algoritmo para el control de posición. El PC extrae la información requerida vía interrupción. Para el control de posición se requiere generar una interrupción a cada flanco de cada una de las dos señales en cuadratura, es decir, cuatro veces más que las necesarias para el control de velocidad. Así pues, se requiere una electrónica que acondicione la información proveniente del encoder con el fin de poder solicitar los servicios del PC vía interrupción.

En el puerto paralelo del PC se dispone de un pin (pin 10) a través del cual puede provocarse una interrupción a la CPU (IRQ7). Se requieren dos líneas más para obtener información del estado de las dos señales en cuadratura para poder determinar el sentido de giro.

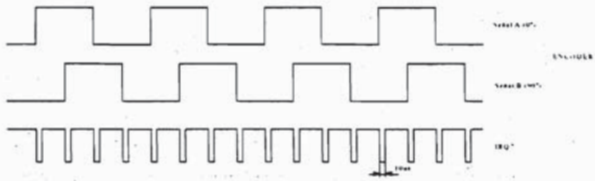
El circuito de acondicionamiento para adecuar la información del encoder al PC es el siguiente:

Generador de interrupciones



El siguiente diagrama muestra la evolución temporal de las señales en los puntos clave. Observar la cuadratura de las dos señales provenientes del encoder así como la dependencia del flanco ascendente de la señal IRQ7 con cada uno de los flancos de éstas. Como ejemplo, para una velocidad de 1000 r.p.m. se generará una petición de servicio, vía interrupción, cada 75 µs. Petición que habrá que atender necesariamente.

Señales



2.3 Recursos del PC.

Para la implementación efectiva de un controlador PID digital se requiere un estricto manejo de tiempos. El intervalo de muestreo deberá poder establecerse para cada ensayo, ser constante y con muy buena precisión si se requieren buenos resultados. La determinación de la velocidad exige una muy fina medida del tiempo si el error ha de ser pequeño, y la implementación de la técnica PWM requiere manejar tiempos muy breves si se desean frecuencias elevadas y un manejo suave de la tensión de ataque al motor.

Nuestro sistema manejará dos interrupciones: una para la medida del tiempo (real time clock 146818 si del OS-9000 se trata o del timer 0 del 8254 si es del MS-DOS), canalizada por la línea IRQ8 (IRQ0), y otra para el seguimiento de los pulsos proveniente del encoder relativo (puerto paralelo), a través de la línea IRQ7.

Se dispone de dos relojes de frecuencias bien diferentes: 1.19MHz para el timer 0 (8254) y 32.7KHz para el real time clock. El timer 0 nos permite interrupciones con una cadencia del orden de 1 µs y el RTC de 122 µs. Obviamente el timer 0 resulta más adecuado al permitirnos *hilar muy fino* (pensemos que el sistema tolera perfectamente interrupciones con periodos inferiores a 10µs para un procesador Pentium a 120MHz, periodos de muestreo próximos a los 40µs y frecuencias para PWM del orden de varios KHz). No obstante, este timer es el responsable de garantizar el procesamiento concurrente y, aún cuando posible, no parece muy adecuado intervenirlo si la aplicación ha de correr bajo OS-9000 (este problema no se da bajo MS-DOS). Nos queda el RTC que sí puede ser intervenido sin problemas y, aún cuando su periodo mínimo (122µs) puede no ser adecuado para ciertas aplicaciones, para nuestros objetivos no supone ningún problema serio (pensemos que la constante de tiempo de un pequeño motor de corriente continua en vacío puede aproximarse a los 100 milisegundos). Donde más puede acusarse esta baja resolución temporal es en la determinación de la velocidad de giro del motor. Si se asignan periodos de muestreo muy pequeños pueden cometerse errores considerables, lo que debería obligarnos a replantear el algoritmo de forma que uno de los parámetros a considerar fuese el del máximo error permitido.

Nuestro código, para el procesador indicado, nos permite la captura de pulsos del encoder cuyo periodo sea del orden de 5 µs, lo que para un encoder de 200 p.p.r. como el empleado, supone la medida de una velocidad de giro para nuestro motor de hasta 15.000 r.p.m.

Al usar la IRQ7 se nos plantea el problema de que no podrá hacerse uso efectivo de ninguna impresora gobernada por esta interrupción mientras dure el ensayo.

2.4 Software de control.

El manejo efectivo de interrupciones en un entorno multiprogramado requiere la implementación de los *drivers* oportunos, dejando a éstos el trato directo con el hardware. En nuestro caso el *driver* debe manejar dos interrupciones (IRQ7 + IRQ8/IRQ0) y despertar al proceso de usuario, que estará a la espera de la ocurrencia de algún evento. El evento coincide con el periodo de muestreo, justamente con la conclusión de la rutina de atención. En ella de evalúa el algoritmo PID y se actualiza la información del controlador (duty cycle del PWM). Para darle máxima rapidez al algoritmo, las operaciones aritméticas se realizan en coma fija (sólo bajo OS-9000). El tiempo de procesamiento requerido en esta rutina, en las condiciones antes citadas, es de unos 30 µs. Este tiempo incluye el requerido para colocar al proceso de usuario a la cola de procesos activos. Siendo el tiempo empleado por la rutina de gestión del PWM de unos 7 µs, en el peor de los casos el *driver* se apropiará de la CPU durante unos 37 µs por muestra.

Se han implementado controles de velocidad y posición, siendo ambos del tipo PID incrementales no saturables. Las funciones presentadas son: velocidad/posición = f(kt) y salida al controlador = f(kt) siendo k un múltiplo del periodo de muestreo, ajustable por la opción *presenta resultados* del menú de configuración.

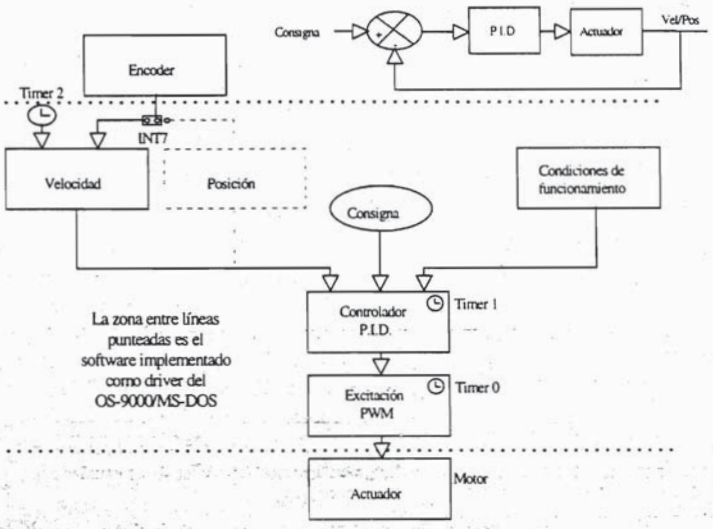
Se puede analizar la respuesta del sistema a un cambio de consigna en escalón (en lazo abierto o cerrado) o en rampa (lazo cerrado). Esto nos ayudará a obtener la información de planta necesaria para modelar el sistema. La magnitud del escalón o la pendiente de la rampa son ajustables desde el menú de configuración.

La *presentación gráfica continua* informa continuamente del estado del sistema permitiéndonos observar su comportamiento ante cualquier perturbación que podamos provocarle. La *presentación gráfica transitoria* es útil para observar los detalles de cómo responde ante un transitorio. En ambas opciones se dispone del efecto zoom. La *presentación alfanumérica* nos informa del estado del sistema de forma alfanumérica. Esta presentación es útil solamente para aquellos entornos no gráficos. Con la *presentación diferida* ahorramos tiempo de CPU en beneficio de menores intervalos de muestreo.

Todas las opciones se presentan en forma de menús, configurando un entorno agradable.
Para cada ensayo se genera un archivo report.txt que contiene toda la información manejada, permitiéndonos un estudio diferido de los resultados.

El diagrama funcional del driver es el siguiente:

Software de control: diagrama funcional del driver



3. CONCLUSIONES

Esta plataforma se ha venido usando durante los dos últimos cursos en la asignatura de Informática Industrial correspondiente a la titulación de Ingeniería Técnica Industrial en Electrónica Industrial. Se han elaborado una serie de prácticas que culminan con la implementación, por parte del alumno, de un control PID para velocidad o posición. Estas prácticas incluyen: el tratamiento de interrupciones, la gestión de la temporización, el control de potencia (PWM) y, por último, la elaboración del algoritmo de control. Todas estas prácticas se enlazan, finalmente, en un miniproyecto de control de la velocidad o posición del motor DC. Los alumnos tienen el aliciente de *ver* las consecuencias físicas derivadas de su actuación en el software. El resultado es muy positivo.

4. BIBLIOGRAFÍA

[1] David M. Auslander y Cheng H. Tham. "Real-Time Software for Control. Program Examples in C"
Prentice Hall (1.989)

[2] Stuart Bennett. "Real-Time Computer Control. An introduction"
Prentice Hall(1.994)

[3] Tecnical Reference Manual IBM-PC

[4] Colección de manuales del RTOS OS-9000