MODELADO DE UN MOTOR DC. DISEÑO, SIMULACIÓN Y CONSTRUCCIÓN DEL SISTEMA DE CONTROL DE VELOCIDAD.

F. Espinosa, E. Santiso, E. López, J.J. García, C. Mataix
Dpto. de Electrónica. E.U. Politécnica. Universidad de Alcalá.
Campus Universitario s/n. 28871 Alcalá de Henares (Madrid).
Tfno: 91-8854810, Fax: 91-8854804, E-mail: espinosa@depeca.alcala.es

RESUMEN.- El trabajo presentado refleja la estructura de una práctica de la asignatura Laboratorio de Sistemas Electrónicos de Control Discreto. La práctica se ha planificado para que integre diferentes aspectos que consideramos básicos en la formación de un diseñador de sistemas electrónicos de control: a) manejo de herramientas software que faciliten el diseño y simulación de controladores, b) construcción de la arquitectura hardware que permita aplicar directamente a un proceso real los algoritmos de control validados mediante simulación. El objetivo del sistema es el control en lazo cerrado de la velocidad angular de un motor de continua con encoder óptico.

1.- INTRODUCCIÓN

La enseñanza de Sistemas Electrónicos de Control en el Plan de Estudios, implantado en la E.U.Politécnica de la Universidad de Alcalá desde el curso 93/94, está distribuida en cuatro asignaturas cuatrimestrales obligatorias: una de teoría (4,5 créditos) y otra de laboratorio (3 créditos) en el cuarto semestre orientadas a Control Continuo, y otras dos (con la misma estructura de créditos) en el quinto semestre enfocadas a Control Discreto. Además de una asignatura optativa de Control Avanzado en el sexto semestre (4,5 créditos).
De esta forma, y centrándonos en los créditos obligatorios, el alumno recibe una formación paralela y complementaria sobre:
a) Fundamentos de control, continuo y discreto, fundamentalmente en el dominio del tiempo. Con una introducción final al estudio en el espacio de estados.
b) Soluciones electrónicas prácticas para llevar a cabo el control, la adaptación de la señales que garanticen la excitación del proceso y la construcción de transductores que permita cerrar el lazo de realimentación.
La facilidad de modelado y excitación de motores de continua -DC- de imán permanente -PM-, unido a su amplia utilización en aplicaciones industriales: manipuladores de precisión (robots fijos), vehículos guiados autónomamente (robots móviles), etc., nos han llevado a èlegir este tipo de motores como proceso idóneo para el ensayo de diferentes alternativas de control en el laboratorio.
Cuando el alumno llega a las asignaturas relacionadas con control discreto ya ha conseguido un importante bagaje. Ha extraído el modelo teórico de un sistema real (motor más encoder), apoyado en los principios físicos que lo rigen y en la información comercial que proporciona el fabricante, ha construido un controlador PID analógico tras el correspondiente diseño y ajuste utilizando la herramienta software Matlab/Simulink y su toolbox de control, ha diseñado y

montado el sistema electrónico capaz de excitar al motor a partir de la señal de control, y ha cerrado el lazo de control construyendo un tacómetro analógico basado en un conversor frecuencia/tensión. Además ha estudiado y practicado, en asignaturas obligatorias del mismo Plan de Estudios (Sistemas Digitales I y II, teoría y laboratorio) con microprocesadores y microcontroladores.

Por todo esto hemos preferido introducir otras herramientas que amplíen las perspectivas futuras del alumno en el mundo del control. Concretamente, junto al diseño y construcción de soluciones electrónicas digitales para excitación de motores y registro de las variables a controlar, se le acerca a la utilización de una de las herramientas software profesionales que abarca más ramas dentro del control (continuo o discreto, lineal o no, aplicando métodos transformados o en el espacio de estados, cálculo de estimadores, filtro de Kalman, identificación de sistemas, lógica borrosa, redes neuronales, etc), nos referimos al entorno Matlab/Simulink. Además cuenta con una importante utilidad que permite traducir a código ejecutable en tiempo real el diagrama de bloques del mismo sistema simulado.

2.- OBJETIVOS DE LA PRÁCTICA

La práctica que presentamos se ha planificado para conseguir, a la par que el control de velocidad de un motor de continua en lazo cerrado de acuerdo al esquema de la Figura 1, los siguientes objetivos:

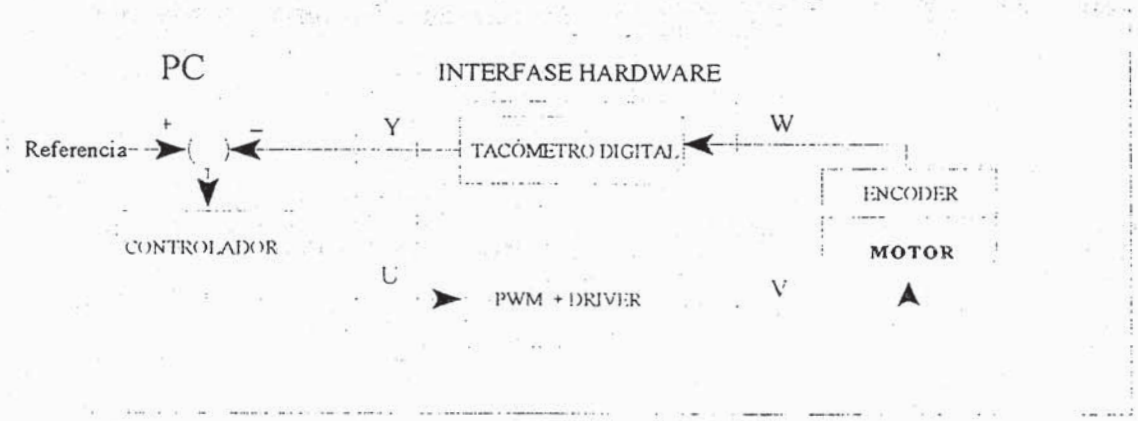

Figura 1.- Diagrama de bloques del sistema de control a realizar en la práctica.

a) Diseño de la interfase hardware que reciba la información, vía puerto paralelo, de la señal de actuación de un motor y devuelva la velocidad real alcanzada por el mismo.

b) Modelado del sistema a controlar. A partir del ensayo del mismo en lazo abierto y del registro del comportamiento transitorio, se obtienen las muestras de las variables de entrada (tensión) y de salida (velocidad angular) que precisa la herramienta software de identificación paramétrica de Matlab.

c) Aplicación directa de los fundamentos teóricos de control sobre el modelo obtenido para proceder al diseño, ajuste y simulación de un controlador PID.

d) Traducción a lenguaje C del algoritmo obtenido en el punto anterior. Proceso que se realiza automáticamente con la utilidad Real-Time Workshop de Matlab. De esta manera se consigue un programa ejecutable en tiempo real, bajo DOS en el propio PC.

e) Puesta a punto del control en lazo cerrado representado en la Figura 1, y contrastación de los resultados reales con los obtenidos por simulación.

3.- METODOLOGÍA Y FASES DE REALIZACIÓN

A continuación describimos el proceso a realizar de acuerdo con los objetivos planteados, indicando el material y las claves de las diferentes etapas.
El objeto de control es un motor de continua, con escobillas e imán permanente, de baja potencia (2225-006S de Micromotor) dotado de encoder óptico (HEDS-5540A) [1].
Dado que el laboratorio en el que se realizan las prácticas cuenta con un ordenador 486 por puesto, aprovechamos los mismos tanto para la etapa de tratamiento software (identificación, diseño y simulación) como para la de control en tiempo real ejecutando el programa diseñado, bajo DOS en el propio PC en conexión con el motor a través de la interfase hardware construida.

3.1.- Diseño y construcción de la interfase motor-PC

En esta interfase hardware se desarrollan los dos bloques resumidos en la Figura 1 como generador de PWM más excitador electrónico del motor y como tacómetro digital. Su arquitectura está basada en los bloques indicados en la Figura 2.

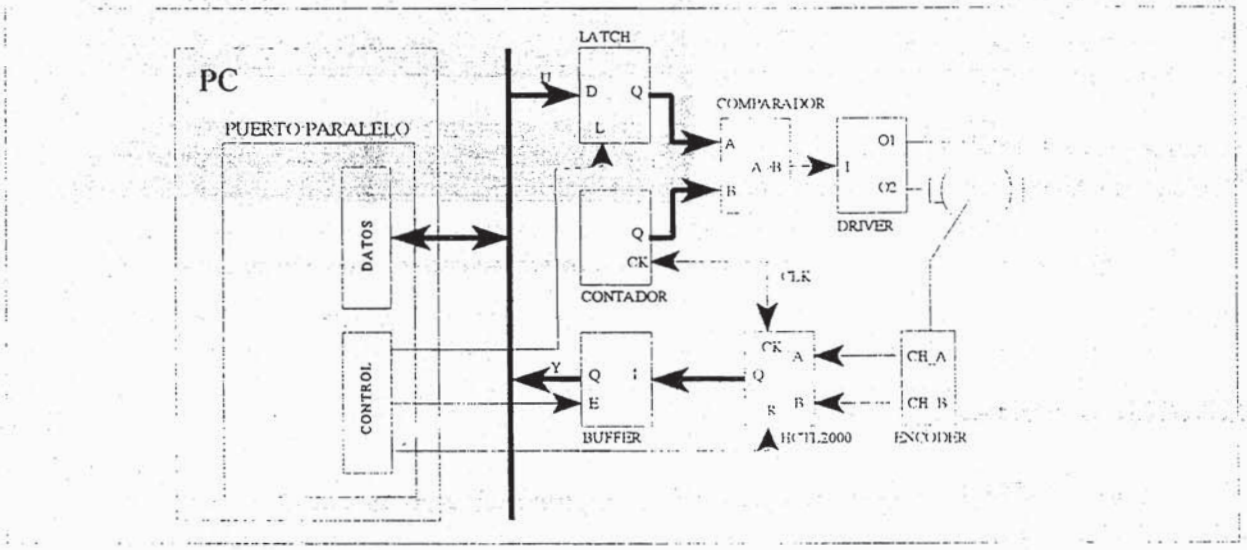

Figura 2.- Esquema de la interfase hardware motor-PC.

La actuación de la señal U de control de velocidad sobre el motor se consigue variando el ciclo de trabajo de una señal PWM que se genera como se explica a continuación. El PC saca un dato U de ocho bits por el bus de datos del puerto paralelo y a continuación lo valida a través de una línea del bus de control del mismo puerto. Este dato queda almacenado en el latch, y se compara con la salida de un contador binario. El resultado de la comparación genera una señal cuyo tiempo a nivel alto es proporcional al dato U sacado. Esta señal se amplifica en el excitador (driver en puente en H) y se aplica al motor.
Para realizar la lectura Y de la velocidad, las salidas del encoder se conectan al contador de estados HCTL2000 [2], el cual es leído de forma periódica por el PC y reseteado inmediatamente después. De esta manera el código Y es directamente proporcional a la velocidad del motor. La lectura se lleva a cabo habilitando el buffer a través de una línea del bus de control, se lee el bus de datos, y a continuación se deshabilita el buffer y se activa momentáneamente otra línea de control para provocar el reset del HCTL2000.

3.2.- Modelado del sistema a controlar

Conocida la función de transferencia simplificada del motor: velocidad del eje rotor W en función de la tensión de armadura aplicada V, reflejada en la ecuación (1), donde τ (s) es la constante de tiempo electromecánica y A (rpm/V) la ganancia en continua del mismo, nos proponemos identificar los parámetros característicos A y τ a partir del ensayo del motor en lazo abierto. Para ello recurrimos a herramientas software disponibles para la identificación de sistemas aprovechando la interfase motor-PC diseñada.

$$\frac{W(s)}{V(s)} = \frac{A}{\tau s + 1} \tag{1}$$

De las funciones del System Identification Toolbox de Matlab [3], ensayamos una de las estructuras de identificación paramétrica más simples: ARX. Según ésta, la salida y(k) está ligada con la entrada u(k) y con la componente de ruido e(k), supuesto blanco y gaussiano, según la relación (2), donde "r" representa el número de retardos entre la salida y la entrada. A y B son polinomios en el operador retardo "q^{-1}" de grado "na" y "nb" respectivamente.

$$A(q)\ y(k) = B(q)\ u(k-r) + e(k) \tag{2}$$

Un proceso iterativo de identificación, conduce a la obtención de los parámetros de A y B más significativos, junto al posible retardo. A partir del modelo discreto se obtiene el equivalente en el dominio continuo utilizando las opciones adecuadas de la función de Matlab "d2cm".
Los resultados alcanzados se han de ajustar a los previstos en la ecuación (1). Para ello basta tener en cuenta que la variable de entrada registrada para la identificación U está ligada a la tensión V aplicada al motor según la relación (3). Siendo V_H la tensión que proporciona el puente en H de la etapa excitadora al motor para el máximo valor de U (255).

$$V = \frac{V_H}{255} U \tag{3}$$

A su vez, el código Y leído del tacómetro está relacionado con la velocidad angular del motor W(rpm) mediante la expresión (4), en la que T_{res} es el periodo de la señal con que se ha de resetear el contador de estados (HCTL2000) para obtener la velocidad digitalizada del motor.

$$Y = \frac{100}{3} \cdot T_{res} \cdot W \tag{4}$$

3.3.- Diseño y simulación del controlador

Obtenido el modelo del sistema, recurrimos a Simulink para diseñar el controlador de interés, en nuestro caso un PID, tal y como se indica en la Figura 3.
El siguiente paso consiste en la puesta en práctica del control simulado. Para ello, caben diversas alternativas. Si descartamos la opción de utilizar dispositivos electrónicos digitales como microprocesadores o microcontroladores, una solución pasaría por la ejecución desde el propio PC de un controlador programado en alto nivel. Sin embargo, ésta no resulta tan trivial cuando se trata de algoritmos de control más avanzados, o cuando se trata del control de un sistema multivariable. De ahí, el interés de introducir una nueva herramienta que facilite esta tarea.

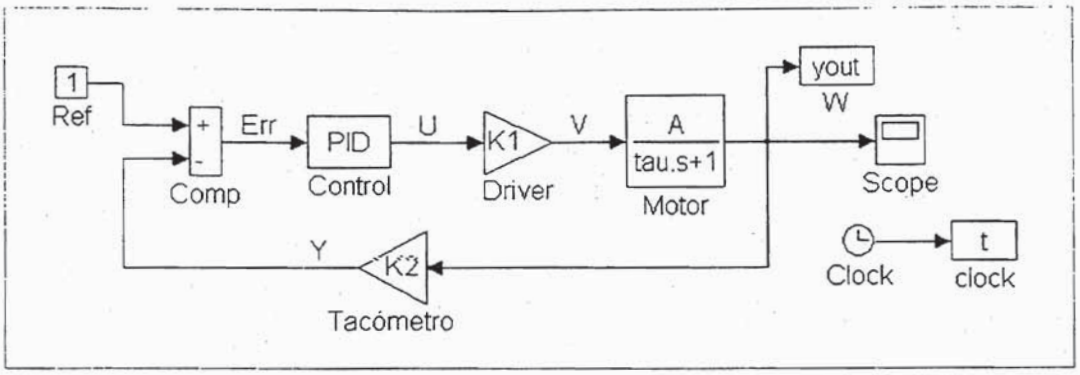

Figura 3.- Diagrama de bloques para el diseño y simulación del sistema de control con Simulink.

3.4.- Traducción automática del algoritmo de control simulado a código ejecutable en tiempo real

De entre las posibilidades que ofrece la expansión Real-Time Workshop -RTW- [4] de Simulink, cabe destacar aquélla que permite la traducción de los diagramas de bloques construidos con Simulink a código C ejecutable en tiempo real bajo distintos entornos y sistemas operativos.
La implementación del controlador utilizando esta herramienta se realiza en las siguientes etapas:
1) Personalización de plantillas ("templates"). RTW dispone de una serie de plantillas que permiten adaptar el proceso de compilación y linkado a la configuración de nuestro sistema.
2) Programación de los bloques "drivers", cuya misión es establecer la comunicación entre el programa ejecutándose en tiempo real y el hardware asociado al mismo. Los drivers deben ser escritos como ficheros .S [5] y compilados como ficheros .MEX [6], y se linkan automáticamente con el resto del diagrama de bloques Simulink. RTW dispone de una serie de plantillas que facilitan su programación. Una vez generados pueden incluirse en las librerías de Simulink para ser utilizados en otros modelos que necesiten el mismo tipo de acceso a la interfase hardware.
3) Construcción del diagrama de bloques en Simulik, a partir de los diferentes bloques incluidos en sus librerías y de los drivers construidos específicamente para nuestra aplicación.
4) Generación automática de código C mediante Real-Time Workshop. Diseñado el diagrama de bloques, se selecciona la opción "Real Time Options..." del menú "Code" de Simulink, y se completa el cuadro de diálogo que aparece, sirva de ejemplo el de la Figura 4.
5) Ejecución del modelo en tiempo real. El resultado del paso anterior es un fichero ejecutable que puede operar en distintos entornos en función de la configuración de la plantilla utilizada. En nuestro caso el resultado es un ejecutable de 32 bits para DOS.

3.5.- Puesta a punto y ensayos finales

Llegados a esta fase no queda sino ejecutar el programa de control bajo DOS y contrastar los resultados reales y simulados obtenidos. A este respecto cabe comentar la posibilidad que Simulink ofrece, trabajando en modo "external" (comunicación cliente-servidor entre Simulink y el programa ejecutable en tiempo real), de presentar gráficamente la evolución "on line" de las variables de interés.

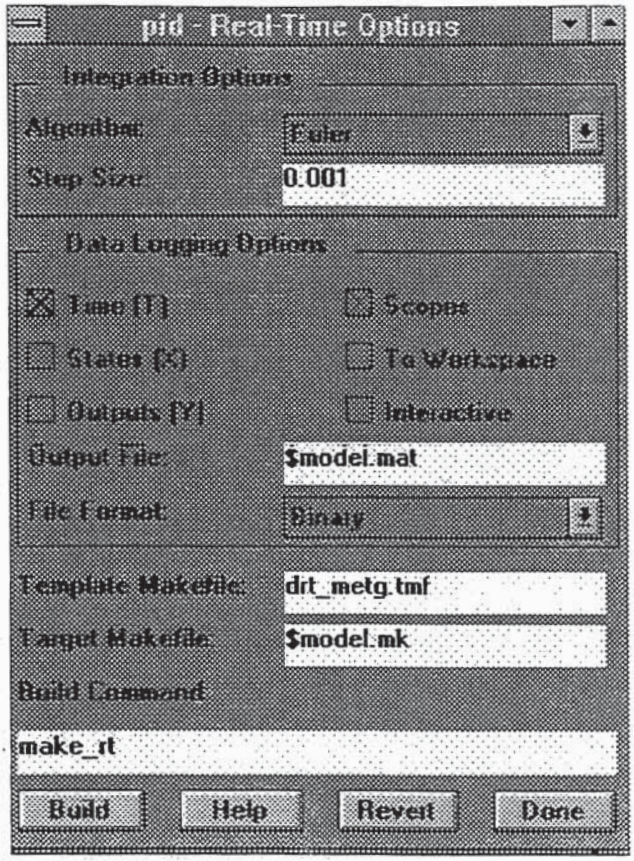

Figura 4.- Ejemplo de cuadro de diálogo para generación de código con RTW.

4.- CONCLUSIONES

Tras la realización de la práctica propuesta el alumno ha podido profundizar en dos aspectos que consideramos importantes para su formación:

* Utilización de herramientas software profesionales que facilitan la puesta en práctica de los fundamentos teóricos de diseño de controladores. Concretamente, de las posibilidades que ofrece el entorno Matlab/Simulink hemos trabajado con las utilidades de identificación, diseño interactivo, simulación y traducción a código que asegura la ejecución en tiempo real del sistema simulado.
* Construcción de soluciones hardware ligadas a las etapas de conexión con el PC, excitación del motor y registro de la evolución de la variable a controlar.

5.- REFERENCIAS

[1] Minimotor SA. DC Motors. Small Compact Powerful. 92/93.
[2] Hewlett Packard. Optoelectronics Designer's Catalog. 1993.
[3] Ljung, L. "System Identification Toolbox for Use with Matlab". The Math Works. 1991.
[4] Real-Time Workshop for use with Simulink. User's Guide. The Math Works Inc. 1994.
[5] Simulink. Dynamic System Simulation Software. User's Guide. The Math Works. 1993.
[6] Matlab. High Performance Numeric Computation and Visualization Software. External Interface Guide. The Math Works Inc. 1993.