ScienceDirect
Disponible en www.sciencedirect.com
Revista Iberoamericana de Automática e Informática industrial 13 (2016) 228–237
www.elsevier.es/RIAI

# Sistema integral para el diseño e implementación de control asistido por computadora

David L. Leal[a,1,*], Miguel R. Celi[b], Esteban Alvarez[c]

[a] *Departamento de Formación General y Ciencias Básicas, Universidad Simón Bolívar, Caracas, Venezuela.*
[b] *Departamento de Tecnología, Universidad Simón Bolívar, Caracas, Venezuela.*
[c] *Facultad de Ciencias, Universidad Central de Venezuela, Caracas, Venezuela.*

**Resumen**

En este trabajo se presenta el desarrollo de un sistema de adquisición de datos y control altamente interactivo, de bajo costo, y basado en un computador personal. El sistema está constituido por un hardware basado en un microcontrolador y un software basado en estándares abiertos. El software utiliza programación de diagramas de bloques como base para el diseño teórico e implementación de un sistema de control, con la finalidad de hacer que la programación sea sencilla y rápida. Una característica importante del sistema es la capacidad de ejecutar control embebido sin la intervención del computador personal, lo cual permite control en tiempo real. Adicionalmente, tiene capacidad de ampliación para crear un sistema de monitoreo y control distribuido por medio del Bus CAN, como interfaz de comunicación. En este trabajo también se expone la implementación del sistema desarrollado en diferentes plantas.

*Palabras Clave:*
Entornos de experimentación, Sistemas embebidos o empotrados, Microcontroladores, Sistemas de tiempo real.

## 1. Introducción

Con los grandes avances en la microelectrónica, la adquisición de datos, y sobre todo por la disminución de los costos, se está volviendo más común el control asistido por computadora de los sistemas industriales y comerciales modernos. El uso de la computadora se extiende desde el diseño, el modelaje, la simulación y la experimentación, hasta llegar a su final implementación. Muchas son las razones que existen para el diseño e implementación de sistemas de control asistido por computadora, entre las que destacan; la naturaleza cíclica del proceso del diseño de control (Uran and Šafarič, 2009), los contenidos visuales que tiene las ideas, conceptos y métodos de control automático (Dormido et al., 2005), el mejoramiento de las interfaces hombre-máquinas (Rimvall and Jobling, 1999), y la facilidad de programación y la versatilidad del software realizado (facilidad de modificación, visualización de resultados, parámetros y variables, comunicación externa, entre otras).

Esta nueva metodología involucra otros campos de competencia en el diseño de sistemas de control, como son la adquisición de datos, la ingeniería de software, la ingeniería de hardware, entre otras. Por consiguiente, existen una heterogeneidad de herramientas y de métodos, tanto como para el software como para el hardware, que dificultan y prolongan el proceso de diseño (Perko et al., 2011). Generalmente, el ingeniero de control define las leyes de control, que para ser implementadas debe generar un código para su simulación, y posteriormente la conversión del diseño en un código que ejecute todas las operaciones necesarias sobre un computador y hardware de adquisición específico. Rimvall & Jobling (Rimvall and Jobling, 1999), plantean que la generación manual de códigos representó un cuello de botella en los procesos de diseños, hasta la llegada de las herramientas basadas en la autogeneración de código a partir de diagramas de bloques. Este tipo de herramienta disminuye el tiempo de diseño e implementación, primordialmente debido a la reducción de errores generados con la codificación manual. En los últimos años, se han desarrollado varias herramientas comerciales basadas en diagramas de bloques, entre las que destacan Simulab/Simulink de Mathworks, Xcos, SystemBuild™ de Integrated Systems Incorporated, VisSim™, Visual ModelQ y SimApp. Algunas de estas herramientas sirven de interfaces para paquetes de análisis como MATRIXx™ (SystemBuild™), Scilab (Xcos) y MATLAB (Simulink).

Existen diferentes compañías que desarrollan tanto el software como el hardware para la implementación de sistemas de control, como por ejemplo: Mathworks, Quanser, National Instruments, dSpace, Visual Solutions, entre otras. Sin embargo, para el sector académico, estas alternativas pueden resultar cos-

*Autor en correspondencia.
*Correos electrónicos:* dleal@usb.ve (David L. Leal), mirodriguez@usb.ve (Miguel R. Celi), ealvarez@fisica.ciens.ucv.ve (Esteban Alvarez)


http://dx.doi.org/10.1016/j.riai.2015.05.007

tosas, lo cual dificulta su adopción, y adicionalmente, en ciertos casos son diseñados con sistemas no abiertos, lo cual podría impedir su utilización en algunos tipos de plantas.

En los últimos tiempos se han desarrollado hardwares de adquisición basados en sistemas embebidos, como una alternativa de bajo costo. Además, los mismos eliminan las restricciones que se obtienen con el uso de un computador personal para la adquisición de datos y control de alto rendimiento. Microcontroladores, Procesadores de Señales Digitales (DSP, Digital signal processing) y dispositivos compuestos por bloques de lógica programables, mejor conocidos como FPGA por sus siglas en inglés (Field Programmable Gate Array), son ejemplos de hardware embebidos usados en la actualidad para los sistemas de adquisición de datos.

Los microcontroladores fueron uno de los primeros en ser utilizados para esta tarea, y los más populares hoy en día por su bajo costo (Demirtas et al., 2008; Katrancıoğlu et al., 2010). Otra razón por la cual se utilizan los microcontroladores en aplicaciones de adquisición de datos y control, es la amplia gama de periféricos incrustados en el mismo chip, facilitando y acelerando el proceso de diseño e implementación de un sistema, ya que disminuye el número total de partes o integrados a ser utilizados. Entre los periféricos destacan: I/O digitales, PWM, timers, comparadores analógicos, conversores A/D y D/A, interfaces de comunicación para USB, Ethernet, CAN, entre otros. Adicionalmente, están disponibles plataformas de hardware libre como Arduino, Pingüino, OLinuXino, Raspberry PI, entre otras, que facilitan aún más la creación de proyectos con microcontroladores.

Lo anteriormente expuesto nos ha motivado a desarrollar un sistema integrado para el diseño e implantación de sistemas de control asistido por computadora, altamente interactivo y de bajo costo. El sistema está constituido por un hardware y un software de adquisición basado en un microcontrolador y en estándares abiertos. El mismo constituye un sistema alternativo que interviene en todos los pasos del diseño e implementación de sistemas de control, incluyendo el modelaje, el propio diseño, la simulación, la experimentación y su final implementación, eliminando así la dependencia de softwares comerciales y el uso de diversas herramientas acopladas. Una de las características más importantes del sistema es la posibilidad de implementar control embebido y control distribuido. Esta capacidad no es excluyente, es decir, el sistema puede ejecutar un control embebido distribuido. Para su utilización no se requiere tener grandes conocimientos en programación ya que el software incluye también mecanismos que permiten realizar el diseño e implementación de un sistema de control en forma gráfica, utilizando diagramas de bloques como base para la creación del código de simulación y final implementación.

En este trabajo se presenta el desarrollo del sistema antes expuesto. El trabajo se encuentra dividido en dos secciones. En la primera se explica el funcionamiento del sistema desarrollado, el cual se divide en el hardware y en el software de adquisición como se indicó anteriormente. En la segunda sección se describe la implementación de diferentes tipos de controles sobre diferentes plantas. Finalmente, las conclusiones relevantes de este trabajo son presentadas.

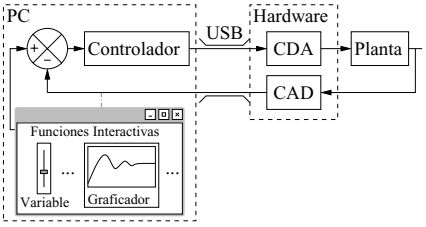


Figura 1: Diagrama de bloques simplificado de sistema en modo compuesto.

## 2. Descripción del sistema

El sistema está constituido por un hardware embebido y un software para adquisición de datos y control, que utiliza las capacidades de un PC para crear una herramienta que interviene en todos los pasos del diseño e implementación de sistemas de control, incluyendo el modelaje, el diseño, la simulación, la experimentación y su final implementación.

El sistema desarrollado tiene dos modos de operación: el compuesto y el embebido. En el modo compuesto, el lazo de control se cierra por medio de la PC, la cual ejecuta todos los cálculos de realimentación del control. En el modo embebido, tanto los cálculos, como la adquisición y actuación son ejecutadas sobre el hardware embebido, mientras que la PC solo monitorea y envía consignas al proceso. Es importante destacar que en este modo no es indispensable la conexión con la PC, lo que implica que el hardware podría ser desconectado de la misma sin alterar el proceso de ejecución.

### *2.1. Modo compuesto*

En la figura 1 se muestra un diagrama de bloques simplificado del funcionamiento del sistema en el modo compuesto, ejecutando un control de lazo cerrado simple. Los cálculos de la retroalimentación son llevados a cabo en la PC, quedando el funcionamiento del hardware enmarcado solamente para la adquisición y manipulación de los voltajes de salida.

El sistema permite la implementación de sistemas de control que no requieran un tiempo de muestreo pequeño, teniendo en consideración que tanto las latencias producidas en la comunicación con la PC, en el software y en el sistema operativo en sí, interfieren con el lazo de control. A pesar de esto, el sistema intenta corregir este inconveniente sincronizando el procesamiento de la PC con el reloj del hardware, asegurando así una adquisición en tiempo real. El tiempo mínimo de muestreo es de 5 ms.

### *2.2. Modo embebido*

La figura 2 muestra el diagrama de bloques simplificado del sistema en modo embebido ejecutando el mismo control de lazo cerrado ejemplificado en el modo anterior. Observe que ahora los cálculos de la retroalimentación son llevados a cabo en el hardware, y la PC solamente modifica el valor de referencia y monitorea las variables del proceso.

Este mecanismo ayuda a superar la limitación propia de los computadores personales para el manejo o monitoreo de plantas que requieren respuestas veloces, eliminando la necesidad

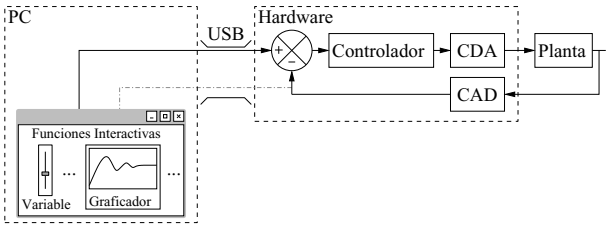


Figura 2: Diagrama de bloques simplificado de sistema en modo embebido.

de incorporar costosas computadores con capacidad de procesamiento en tiempo real.

El proceso de control embebido se basa en la descarga de un programa diseñado en el software a la memoria de hardware, el cual ejecutará hasta que la PC lo indique o hasta que se reinicie el microcontrolador. El programa puede utilizar las funciones principales del hardware (entradas y salidas, tanto analógicas como digitales y comunicación por medio del bus CAN) y un agregado de funciones de cómputo (por ejemplo: suma, resta, multiplicación, filtrado, control PID, control Borroso, aritmética lógicas, entre otras). Mientras el hardware esté ejecutando el control embebido, la PC puede monitorear, o enviar consignas al mismo. Esta característica permite al usuario final interactuar en el proceso de control.

En este modo, el tiempo de muestreo debe ser seleccionado para que la respuesta del sistema sea estable, normalmente se toma un valor pequeño en función del tiempo de respuesta del sistema a controlar. Además, debe considerarse el retardo entre leer las variables y aplicar la acción de control (tiempo total de ejecución del programa descargado en el hardware), si este retardo es menor a la décima parte del tiempo de muestreo se puede descartar y con esto se asegura la estabilidad cuando se aplica el control. Por ejemplo, para un controlador PID básico, el menor tiempo de muestreo que el hardware puede alcanzar es de 0,32 ms (diez veces el tiempo de retardo). Este ejemplo puede considerarse como el control más simple que el hardware puede implementar, así que la interfaz solo podrá asegurar el control para sistemas con tiempo de respuesta mucho mayor que este tiempo. Por otro lado, el tiempo de monitoreo no necesariamente debe ser el mismo que el tiempo de muestreo al cual se esté ejecutando el control embebido. Debido a las mismas limitaciones explicadas en el modo anterior, el menor tiempo de monitoreo es 5 ms.

### *2.3. Hardware de adquisición*

El hardware embebido se basa en un microcontrolador de la familia PIC32MX, el cual tiene un núcleo MIPS32® M4K® 32-bit. En la figura 3 se muestra el esquema simplificado del hardware de adquisición, donde se observa que el microcontrolador se encarga de la mayor parte de las tareas del hardware, como la conversión análogo digital (CAD), la comunicación con la PC por medio de bus USB, el bus CAN y las entradas/salidas digitales. La conversión digital análogo (CDA) y el acondicionamiento de las señales (CAS, circuito de acondicionamiento de señal) son implementadas por circuitos externos al microcontrolador. El sistema por ende, tiene la capacidad de

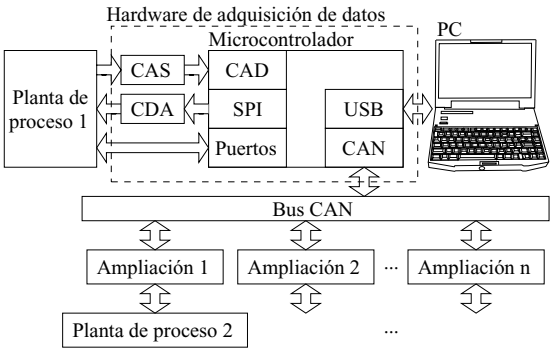


Figura 3: Esquema simplificado del hardware de adquisición y control desarrollado.

interactuar con el mundo externo de dos formas diferentes: la primera es la forma directa, en la cual utiliza una serie de entradas y salidas analógicas con diferentes voltajes de referencia, y también entradas y salidas digitales (ej.: la Planta de proceso 1 de la figura 3). La segunda forma es por medio del Bus CAN que permite ampliar el alcance y las posibilidades de implementar un sistema distribuido de control (ej.: la Planta de proceso 2 de la figura 3).

Las características del hardware son las siguientes:

*Entradas analógicas:*

- 8 canales de entrada de 10-bit de resolución
- Rango de voltajes: 0-5 Vdc, 0-10 Vdc, ±5 Vdc y ±10 Vdc
- Protección contra sobre voltajes

*Salidas analógicas:*

- 4 canales de salida de 12-bit de resolución
- Rango de voltajes: 0-5 Vdc, 0-10 Vdc, ±5 Vdc y ±10 Vdc
- Límite de corriente de 20 mA por canal

*Entradas/Salidas Digitales:*

- 8 canales de entrada TTL
- 8 canales de salida TTL
- Un frecuencímetro

*Comunicación con el computador:*

- Interfaz: USB 2.0
- Velocidad de transmisión: 12 Mbps/s máx (Full-Speed)

*Ampliación del sistema de adquisición:*

- Interfaz: CAN Estándar (Versión 2.0A)
- Velocidad: 1 Mbps/s máx

Los rangos de voltaje, tanto para las entradas como para las salidas analógicas, son seleccionados por software, es decir, al momento de la adquisición el usuario puede escoger el rango que desee para cada entrada o salida por separado.

El costo aproximado para la elaboración del hardware es de $75.

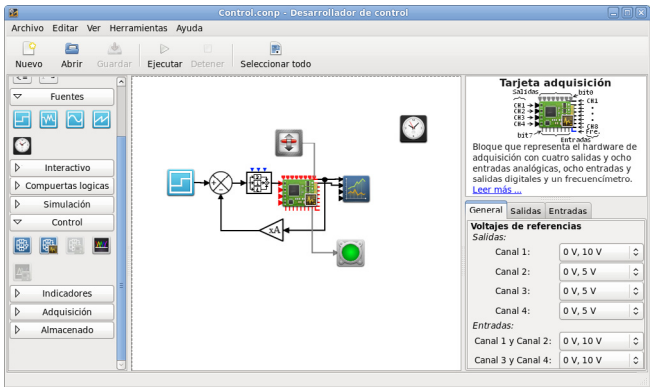


Figura 4: Interfaz del software de adquisición de datos y control desarrollado.

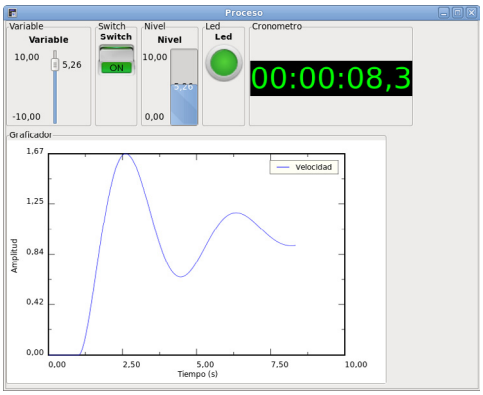


Figura 5: Ventana empotrada con cinco de funciones interactivas.

### 2.4. *Software de adquisición*

El software multiplataforma elaborado (*ConDP*) está desarrollado bajo librerías pre-escritas de licencia pública, entre las que destacan: GTK+, GLib, LibUSB y libXML. El proceso de diseño e implementación de sistemas de control se realiza de forma gráfica. La programación se basa en la interconexión de objetos en forma de diagramas de bloques. Esta característica permite a un ingeniero de control desarrollar muy rápidamente controladores complicados, ya que tanto el análisis como la simulación del código son automáticamente generados a partir de los diagramas de bloques. Así, se elimina el tiempo que se usaría con la codificación manual y evita la introducción de depuradores de código, reduciendo así los costos finales de producción. Adicionalmente, no se requiere un alto conocimiento en programación para la elaboración de estos controladores.

Para la interacción con el usuario, el software posee una interfaz gráfica formada de un panel de bloques pre-establecidos, un área de dibujo, tablas para la modificación de los atributos de los bloques, zona de ayuda y un menú con elementos básicos y herramientas adicionales (Ver figura 4).

El funcionamiento del software se basa en la construcción de un diagrama de bloques, que será luego simulado o implementado en conjunto con un sistema físico externo. Esta construcción se lleva a cabo sobre un área de dibujo, agregando y conectando bloques. Cada bloque tiene características que pueden ser modificadas por el usuario. Una vez que se termine esta construcción, el usuario puede simular el sistema construido. El proceso de simulación es muy parecido al proceso que ejecuta el programa Simulink (Mosterman et al., 2005).

Cada bloque consiste en una función de entradas, estados y salidas, donde las salidas, a su vez, dependen del tiempo de muestreo, de las entradas y de los estados de la misma función. Con la información del programa creado por el usuario, el software crea primero una lista de las funciones que se ejecutarán, luego enlaza las salidas y entradas correspondientes, y finalmente inicializa y copia los datos de las funciones que los requieran. Una vez realizado esto, el software procede a ejecutar función por función en el orden impuesto hasta finalizar la lista. Este paso de ejecución se denomina paso del lazo de control y continuará hasta que la ejecución sea suspendida, u ocurra un error.

Con este mecanismo es posible simular sistemas físicos, como también crear lazos de ejecución para un monitoreo o control de una planta real. Las características del proceso de ejecución dependen solamente de los bloques que sean incluidos en el proyecto y sus atributos.

Algunos bloques tienen funciones interactivas. Esto quiere decir que intercambian información con el usuario. Unos exponen información de forma gráfica, como por ejemplo el bloque “Graficador”, y otros captan información del usuario para luego introducirla en el proceso de simulación. Cada uno de estas funciones despliega una ventana nueva con las características especiales de su función. Puede suceder que muchos de los bloques utilizados en un proyecto sean interactivos. En este caso se abrirán una cantidad considerable de ventanas, lo cual puede ser engorroso a la hora de trabajar. Una forma de solucionar este problema es empotrar todas estas ventanas en una gran ventana. Esta gran ventana debe ser habilitada antes de iniciar la simulación. En la figura 5 se muestra un ejemplo de la forma de la ventana con una cantidad de funciones interactivas, la primera (superior izquierda) en una variable interactiva que permite modificar un dato mientras se ejecuta el proceso. La segunda en un switch que permite modificar un valor binario. La tercera y cuarta son indicadores de variables diferentes, uno real y el otro binario. Y por último (parte inferior) una gráfica que representa las variables introducidas en la misma en función del tiempo de ejecución.

Para la adquisición de datos o manipulaciones de voltajes de salida, se incluye, dentro del stock de bloques, un bloque que representa el hardware de adquisición. Este bloque posee todas las características virtuales que posee el hardware real.

Adicionalmente, el software posee bloques para la comunicación CAN (SargunaPriya.N et al., 2014), que permiten el envío y la recepción de datos provenientes del bus. Cada bloque del bus CAN tiene un número definido de entradas y salidas. Cada una de ellas puede ser enviada o recibida con un identificador diferente (en el hardware elaborado solo se utilizan identificadores estándares), o todas las entradas o salidas puede encapsularse en una sola trama con un único identificador. La función del bloque es encapsular y des-encapsular tramas entre

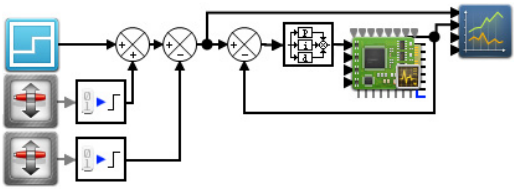

Figura 6: Diagrama de bloques del sistema de control de Velocidad de un Motor de corriente continua.

la simulación en el sistema y el bus.

Finalmente, como se mencionó anteriormente, la comunicación entre el hardware y la PC se realiza por medio del bus de datos USB, siendo esta comunicación transparente para el usuario, de forma que el usuario solo se preocupe de solicitar o enviar datos.

## 3. Diseño e implementación de controladores

### 3.1. Controlador PID de velocidad

En la teoría clásica de control los controladores PID son los más utilizados. Existen muchos métodos para hallar los parámetros de estos controladores, el más básico es el método de sintonización de primer orden de Ziegler y Nichols (Ogata, 2003), el cual consiste en comparar la salida del proceso con respecto a la respuesta al escalón de un sistema de primer orden.

#### 3.1.1. Descripción y modelado matemático de la planta

El experimento se ejecuta sobre un motor de corriente continua conectado a un juego de engranajes y sensores analógicos de posición y de velocidad integrado todo en un módulo de aprendizaje de sistemas de control (Elettronica Veneta S.p.A., n.d.). En (Leal, 2012) se procedió a caracterizar la planta y los traductores con la finalidad de conseguir una respuesta lineal del sistema con respecto al voltaje aplicado a la entrada. En esta caracterización se aplicó un escalón a la entrada y se ajustó la respuesta a un sistema de primer orden obteniendo la siguiente relación entrada-salida en la variable de Laplace.

$$\frac{Y(s)}{U(s)} = \frac{940e^{-0{,}006s}}{0{,}072s + 1} \tag{1}$$

Donde $U(s)$ es la entrada del sistema, en voltios. Y la salida $Y(s)$ representa la velocidad del motor en revoluciones por minutos.

#### 3.1.2. Resultado del experimento

Se procedió a montar el experimento conectando el modulo del motor al sistema de adquisición. En este caso usamos una sola salida, la velocidad del motor, y una sola variable a manipular, el voltaje de entrada.

Para la implementación de este controlador se utilizó el sistema de adquisición en modo compuesto, es decir, se ejecuta sobre el computador.

En la interfaz del software de adquisición se procedió a conectar el bloque de adquisición con los diferentes bloques de funciones (Figura 6). La función PID se basa en un algoritmo implementado en (Ledin, 2004), el cual incluye mecanismos de saturación para los limites del actuador y anti-windup para evitar la saturación en el término integral. Los parámetros del controlador PID encontrados con la sintonización de Ziegler y Nichols son $K_p = 0,012766$, $t_d = 0,003$ y $t_i = 0,012$, parámetros que fueron configurados en el bloque PID.

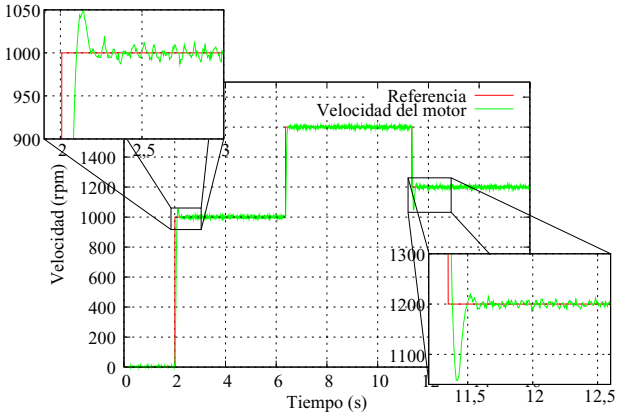


Figura 7: Respuesta de la velocidad con un controlador PID.

Además de los bloques de sistemas de adquisición y control, se colocaron bloques sumadores, generadores de escalón, funciones de gráfico y registro de las variables, y pulsadores para cambios manuales de la referencia durante la ejecución del experimento, todos con la finalidad de crear un sistema de control. El experimento estuvo basado en tres cambios de las referencias en el tiempo, el primer cambio fue realizado por el bloque generador de la señal escalón a los 2 segundos de inicializado con una magnitud de 1000 rpm. Aproximadamente a los 6 y 11 segundos se hicieron cambios manuales de la referencia de 1600 rpm y 1200 rpm respectivamente. Con estos cambios se esperaba ver la respuesta asimétrica del sistema, el cual fue linealizado para un punto de operación particular. En la figura 7 se puede observar la respuesta de la planta. La velocidad del motor va desde 0 rpm hasta 1000 rpm, por los valores escogidos de 1000 y 1600 rpm. En la ampliación de las gráficas observamos que los sobrepicos de la señal en aumento y reducción de velocidad son diferentes, pero ninguno sobrepasa de un 10 % de tolerancia. Además, se observa un tiempo de asentamiento menor a 0,1 segundo.

### 3.2. Controlador con Lógica Borrosa de velocidad

La Lógica Borrosa o Difusa ha tenido una buena aceptación en el campo de la automatización debido a que es una extensión de la lógica tradicional humana, la cual utiliza conceptos subjetivos del pensamiento humano. En vez de realizar cálculos complicados para una acción de control, si el operador conoce el sistema a controlar, realizara cambios relativos a un conjunto de funciones de pertenencia del sistema que el operador genere. De esta manera, el operador podría abrir “un poco” la válvula, o calentar “mucho” el agua, etc. Por lo tanto, la Lógica Borrosa no utilizará valores exactos, sino valores difuminados entre las

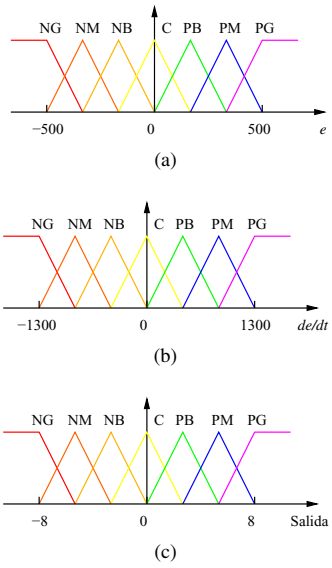


Figura 8: Funciones de pertenencia para el control PD difuso de un motor DC. (a) Función de pertenencia de la variable *error* ($e$), (b) Función de pertenencia de la variable *cambio de error* ($de/dt$) y (c) Función de pertenencia de la variable de salida del controlador (*Salida*).

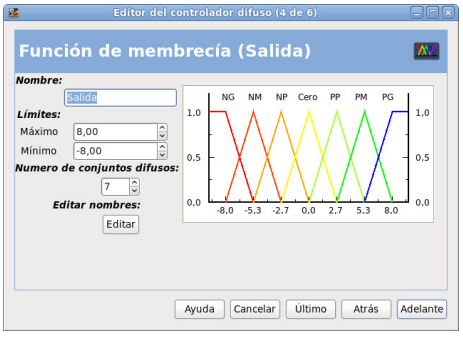


Figura 9: Asistente para la configuración del controlador la función de pertenencia de una de las variables difusas. El usuario puede modificar los límites de universo de discurso, el número de divisiones que componen dicho valor difuso y los nombres de cada función de pertenencia.

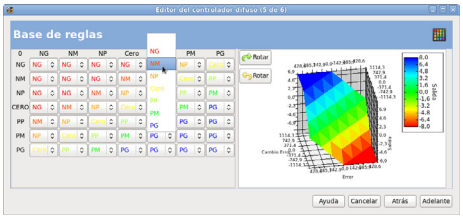


Figura 10: Asistente para la configuración de la base de reglas del controlador difuso. En cada selector aparecen las sub-funciones de la salida del controlador, permitiendo la configuración de todas las reglas posibles. En la parte derecha se muestra la superficie del control.

posibles gamas de valores que se necesiten para resolver el problema (Wang, 1997). Una de las ventajas que presenta utilizar un controlador con Lógica Borrosa es la baja complejidad matemática para hallar el controlador, ya que si el sistema posee respuesta no-lineal o es muy complicado, hallar o sintonizar un controlador PID, o por realimentación de estado no es sencillo. Así, el diseño a base por Lógica Borrosa se basa en definir las funciones de pertenencia para la entrada y salida del controlador, las cuales son generadas a partir de los conocimientos subjetivos del operador del sistema. Un control directo borroso permite una interpretación por parte del usuario final y una mejor integración con otras etapas superiores en un marco único (Albertos and Sala, 2004). Ahora bien, si la sencillez para hallar el controlador es una ventaja, lo complicado que puede ser representar todas las funciones de pertenencia si el sistema posee varias entradas y varias salidas es una desventaja. No es que los controladores borrosos estén limitados a sistemas de una entrada y una salida, pero para sistemas de múltiples entradas y salidas puede ser muy difícil su representación. Otra de las mayores desventajas es la desconfianza que existe sobre su funcionamiento, ya que no son soportados por una matemática sólida, que pueda comprobar la robustez del controlador.

### *3.2.1. Descripción de la planta y el controlador borroso*

El sistema a controlar consiste en el motor de corriente continua utilizado en el experimento anterior. En este caso, no se caracterizó el motor, sino que se procedió a montar las funciones de pertenencia a partir del funcionamiento de un controlador PD, es decir, estudiamos las acciones que se deberían tomar en función del valor difuso de la señal de error $e(t)$ y su derivada $de(t)/dt$. Los valores que se tomaron para generar la funciones de pertenencia fueron los siguientes: NG representando negativo grande, NM representando negativo medio, NB representando negativo bajo, C representando cero, PB representando positivo bajo, PM representando positivo medio y PG representando positivo grande. Se utilizaron funciones de membrecía continua del tipo Mamdani triangulares, las cuales son fáciles y rápidas de implementar (Ver Figura 8). Su implementación se logra de manera sencilla utilizando las herramientas para configurar las funciones de pertenencias de las variables difusas, figura 9, y las reglas del controlador, figura 10. Estas herramientas permite de manera gráfica configurar el controlador PD-Borroso fácilmente.

Una vez definidas las funciones de pertenencia, es decir la lógica lingüística, se procede a colocar las reglas para el diseño del controlador PD. Algunos ejemplos de diseño de controladores PID se pueden observar en (Passino and Yurkovich, 1998) y en (Rodríguez et al., 2004). En la tabla 1 se pueden observar las reglas para el controlador PD, donde se decide el tipo de variación de la acción de control, $\Delta u$, que se tomará en cada uno de los casos. Así por ejemplo, si la derivada del error es CERO y el error es CERO, $\Delta u$ tomará el valor de CERO.

El diseño gráfico del controlador consta de los bloques necesarios para hacer cambios al valor de referencia, de un filtro representado con su función de transferencia, que permitirá hallar la derivada del error, el bloque del controlador borroso, un integrador continuo para $\Delta u$, el bloque del módulo de adqui-

Tabla 1: Reglas del control difuso para el motor DC.

| $\Delta u$ | | Error ($e$) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | **NG** | **NM** | **NB** | **C** | **PB** | **PM** | **PG** |
| $\frac{de}{dt}$ | **NG** | NG | NG | NG | NG | NM | NB | C |
| | **NM** | NG | NG | NG | NM | NB | C | PB |
| | **NB** | NG | NG | NM | NB | C | PB | PM |
| | **C** | NG | NM | NB | C | PB | PM | PG |
| | **PB** | NM | NB | C | PB | PM | PG | PG |
| | **PM** | NB | C | PB | PM | PG | PG | PG |
| | **PG** | C | PB | PM | PG | PG | PG | PG |

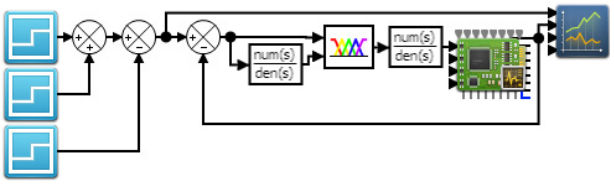


Figura 11: Diagrama de bloques del proyecto para el control de la planta de proceso de velocidad utilizando control de lógica borrosa.

sición, y el bloque de registro de información (Ver figura 11). Debido a que nuestro controlador borroso suministra la variación de la acción de control, el integrador es el encargado de hallar la acción de control $u$. En el caso del ejemplo de la tabla 1, cuando se haya cambiado la referencia y el error y su derivada sean CERO, $\Delta u$ resultará también CERO, pero el integrador estará suministrando la suficiente energía al motor para mantener una velocidad continua.

### 3.2.2. *Resultado del experimento*

El experimento se diseñó para tener tres cambios del valor de referencia en un periodo de 14 segundos. Estos cambios son los siguientes: 1000, 1600 y 1200 revoluciones por minutos. En la figura 12 se observa la respuesta del sistema controlado. Comparado con el controlador PID del primer experimento podemos observar que la respuesta es más lenta, característica normal de un controlador PD. Este tipo de respuesta tiene como ventaja que el sobre-pico de la señal sea más pequeño. El controlador fue ejecutado sobre el computador usando un periodo

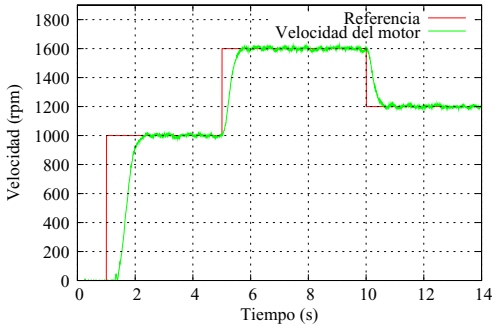


Figura 12: Respuesta de la velocidad con un controlador de lógica borrosa.

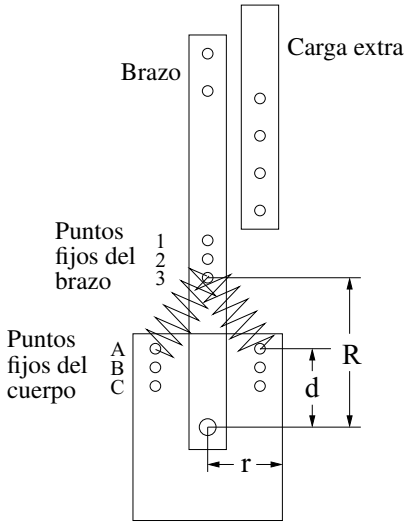


Figura 13: Módulo de la articulación flexible rotatoria.

de muestro de 5 ms sin que se produjeran pérdidas de datos en el registro, y sin que se produjeran errores en la acciones de control.

## 3.3. *Control por realimentación de estado para una articulación rotatoria flexible*

### 3.3.1. *Descripción del experimento*

Para mostrar la operatividad del sistema de adquisición en modo embebido, se utilizó la planta piloto de articulación flexible rotatoria. Este módulo educacional es distribuido por Quanser y emula los efectos que ocurren en un brazo mecánico al girar para cambiar de posición, y se puede describir como una articulación flexible compuesta por dos resortes iguales que unen a un rotor y un brazo que representa la carga. La flexibilidad de la articulación se puede variar cambiando la posición de los resortes. Un pequeño brazo puede ser unido al final del brazo principal para cambiar la inercia de la carga (Ver figura 13).

El propósito del experimento es diseñar un controlador que ajuste el brazo a un ángulo deseado con la menor vibración por parte de la articulación. Este ángulo está compuesto por el ángulo de salida del motor ($\theta$) sumado al ángulo correspondiente al movimiento de la articulación flexible, representado éste por el ángulo ($\alpha$), medido por un potenciómetro engranado en la articulación.

### 3.3.2. *Modelo Matemático*

El modelo matemático es derivado a partir de la formulación del Lagrangeano. Este es usado para derivar la energía cinética y potencial del sistema. Las ecuaciones dinámicas son así derivadas y entonces un modelo lineal se obtiene al linealizar en la proximidad del punto de equilibrio (Quanser Inc., 2011). Sustituyendo los parámetros del sistema, se obtiene el siguiente modelo lineal:

$$\frac{dx}{dt} = Ax + Bu \quad (2)$$

$$y = Cx = \theta + \alpha \quad (3)$$

donde

$$A = \begin{vmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 767 & -53 & 0 \\ 0 & -1040 & 53 & 0 \end{vmatrix}, \quad B = \begin{vmatrix} 0 \\ 0 \\ 98 \\ -98 \end{vmatrix},$$
$$C = \begin{vmatrix} 1 & 1 & 0 & 0 \end{vmatrix}, \quad x = \begin{vmatrix} \theta \\ \alpha \\ \dot{\theta} \\ \dot{\alpha} \end{vmatrix} \tag{4}$$

### 3.3.3. *Diseño del controlador*

Para este tipo de experimento se optó por diseñar un controlador de realimentación de estado asociado a la ecuación de Riccati (Sontag, 1998):

$$A^T S + SA - (SB + N) R^{-1} \left(B^T S + N^T\right) + Q = 0 \tag{5}$$

donde

$$K = R^{-1} \left(B^T S + N^T\right) \tag{6}$$

Reescribiendolo ahora en tiempo continuo, la ley de control de realimentación $u = -Kx$ minimiza la función de costo cuadrática:

$$\begin{aligned} J(u) = & \int_0^\infty \left(x^T Q x + u^T R u + 2x^T N u\right) dt, \\ \text{s.t.} \quad & \dot{x} = Ax + Bu \end{aligned} \tag{7}$$

Esta minimización es realizada por la función "lqr" de MATLAB. Para ello utilizaremos los siguientes valores para los pesos en los estados y las entradas:

$$Q = \begin{vmatrix} 1000 & 0 & 0 & 0 \\ 0 & 4500 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}; \quad R = 10; \quad N = 0 \tag{8}$$

Observe que se penaliza el movimiento elástico de la articulación ($\alpha$) con 4500. La finalidad es disminuir lo más posible este ángulo. La penalización sobre la posición del motor y su velocidad es la necesaria para encontrar un mejor ajuste con respecto al tiempo de estabilización y el error de estado estacionario. El controlador de realimentación de estado $K$ es el siguiente:

$$K = \begin{vmatrix} 10 & -13,7 & 1,2 & 0,42 \end{vmatrix} \tag{9}$$

La implementación de este controlador se realizó utilizando un equipo comercial y el sistema de adquisición y control integrado. En primer lugar, se utilizaron el módulo "Rotary Flexible Joint" de Quanser junto con el sistema de adquisición proporcionado por la misma compañía. Además se implementó el modelo en Simulink. En la figura 14 podemos observar el diagrama en bloque del sistema de control de la articulación. La tarjeta de adquisición está representada por bloques de entradas y salidas, y después de haber montado el controlador se debe realizar una compilación para poder hacer funcionar el experimento a tiempo real (WinCon). Se utilizó un tiempo de muestreo de 1 ms para ejecutar el control.

Debido a que en nuestro sistema de adquisición y control el hardware y el software están unificados, se procedió a conectar los sensores del módulo "Rotary Flexible Joint" a las entradas analógicas y digital de nuestro hardware. En el software,

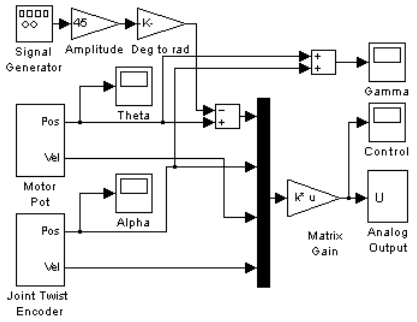


Figura 14: Diagrama en bloque del sistema de control de la articulación bajo Simulink.

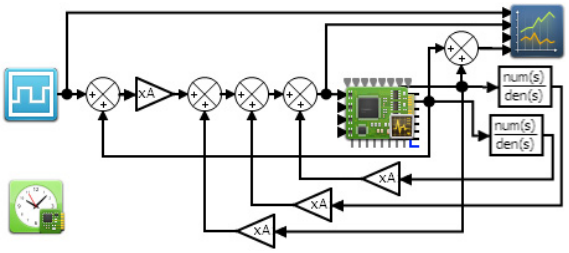


Figura 15: Diagrama en bloque del sistema de control de la articulación bajo el software desarrollado.

se procedió a implementar el diagrama en bloques equivalente, considerando la diferencia de que la tarjeta está representada por un solo bloque de entrada-salidas. Se calcularon las derivadas de las dos variables medidas utilizando filtros representados con sus respectivas funciones de transferencias. Finalmente, se realizaron la realimentación de cada estado (Ver figura 15). A diferencia de los ejemplos anteriores, el sistema de control estará embebido en el sistema de adquisición. Este procedimiento es sencillo, simplemente hay que configurar el diagrama de bloques para operar embebido y verificar que el tiempo del cálculo del control sea mucho menor al periodo de muestreo. Esta configuración se encuentra representada en el bloque verde, con un reloj y una tarjeta como icono. La información gráfica saldrá del sistema a través del puerto USB con un periodo de muestreo gráfico superior al periodo de muestreo de control, en este caso se ajustó con periodo de muestreo de 1 ms, igual que la plataforma Quanser, y un tiempo de monitoreo de 5 ms.

### 3.3.4. *Resultados de los experimentos*

En primer lugar, se realizó el experimento utilizando la plataforma Quanser-MATLAB, la respuesta al desempeño del controlador se puede observar el la figura 16. Observe que el controlador es capaz de seguir el valor de referencia, con un tiempo de respuesta menor a medio segundo. El error en estado estacionario es muy pequeño y no hay sobre-pico ni oscilaciones en la respuesta.

En la figura 17 podemos observar las respuestas del sistema para la plataforma presentada en este artículo. Si comparamos las dos respuestas, se puede decir que las dos plataformas son capaces de controlar el sistema, y que el desempeño de cada plataforma es similar. La diferencia que se puede observar en

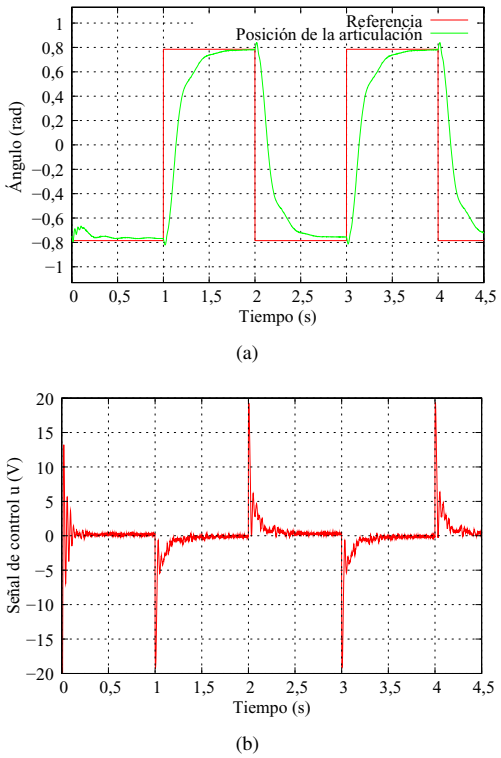


Figura 16: Resultado del experimento realizado con la tarjeta de adquisición de Quanser y Simulink de MATLAB: (a) ángulo del brazo y referencia. (b) señal de control.

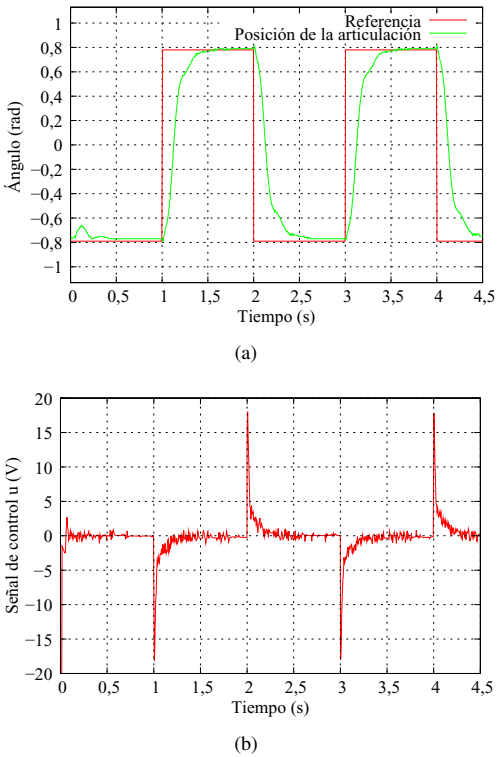


Figura 17: Resultado del experimento realizado con el Sistema de Adquisición y Control desarrollado: (a) ángulo del brazo y referencia. (b) señal de control.

las gráficas es debido al monitoreo, el cual hace un sobremuestreo 5 veces mayor que el tiempo de muestreo, es por ello que los desempeños son similares pero las gráficas presentan diferencias. En la primera parte del experimento podemos observar una pequeña oscilación debido a las condiciones iniciales de los filtros, los cuales no poseen datos correctos para ajustar el controlador.

Se puede observar una respuesta similar al mostrado con la otra plataforma, un tiempo de asentamiento menor de 0,5 segundos. Podemos notar que el error de estado estacionario tiene un valor que siempre es mayor a la referencia. Esto es debido a la cuantificación del sensor de posición, que aunque la señal de control solicita disminuir dicho valor, el motor no es capaz de desplazarse con el mismo. En la figura 17(b) se puede observar que la señal de control intenta ajustar el error de estado estacionario haciendo que la señal oscile con la rapidez del tiempo de muestreo del sistema de adquisición.

Sin embargo, hay algunas pequeñas diferencias, debidas principalmente a los niveles de cuantificación e implementación computacional de los filtros. Por ejemplo: en cuantificación, la tarjeta de Quanser trabaja con 16 bits, mientras que nuestro sistema trabaja con 10 bits de entrada y 12 bits de salida.

## 4. Conclusiones

El diseño e implementación del control asistido por computadora ha permitido la creación de sistemas de control a un bajo costo. Esto es cierto, no solo cuando se consideran los costos de los elementos involucrados, sino por la simplificación y aceleración de los procesos de diseño, en los cuales se debe iterar en ciclos de modelaje, simulación y experimentación hasta lograr los resultados esperados. Para lograr incorporar una computadora en este proceso es necesaria la utilización de un sistema de adquisición (hardware y software) como interfaz entre el computador y un sistema físico real. El objetivo central de este trabajo ha sido el desarrollo de una interfaz de bajo costo y de fácil manipulación para la construcción e implementación de un sistema de adquisición de datos y control.

Entre los resultados obtenidos destacan la capacidad de monitoreo y control de diferentes tipos de procesos. Se logró implementar tres tipos de controles sobre plantas con diferentes requerimientos de tiempo de respuesta. Es importante destacar la implementación de un control en tiempo real sobre una articulación rotatoria flexible, la cual fue comparada con herramientas de software y hardware de empresas como Mathworks y Quanser. Los resultados son similares, donde el beneficio es el ahorro en el costo del software y del hardware. Además, las experiencias construidas se realizaron de una forma fácil y rápida con el uso de los diagramas de bloques como base para el diseño y final implementación.

Tomando en cuenta los resultados anteriormente mencionados, podemos concluir que se logró desarrollar un sistema en un ambiente unificado para modelar, diseñar, simular, experimentar e implementar sistemas de control asistido por computado-

ra. El sistema permite además implementar controles en modo compuesto, en conjunto con una PC, y en modo embebido, donde las acciones de realimentación de control son ejecutadas sobre el hardware de adquisición. El uso de diagramas de bloques facilita la implementación de sistemas de control, ya que no se requiere un alto conocimiento en programación para la elaboración de un proyecto sobre el mismo. Esta característica, además acelera el proceso de desarrollo de lazos de ejecución para diversos objetivos. El sistema de adquisición de datos elaborado constituye un desarrollo de muy bajo costo, en comparación con muchos desarrollos de sistemas similares. Esta afirmación, no solo se fundamenta por considerar los elementos físicos que componen el sistema, sino que además, el mismo elimina la posible dependencia con algún software privativo en todo el proceso de diseño e implementación de sistemas de control. El software permite su expansión para posteriores incorporaciones de nuevos elementos, ya que fue desarrollado bajo Estándares Abiertos.

Por los momentos la interfaz está en la etapa inicial y en la siguiente dirección Web `https://sites.google.com/a/usb.ve/david-leal/home/descargas/condp` se puede obtener un material de apoyo y descargar el software para las plataformas Windows 32-bits y Linux 64-bits, como también el código fuente. La próxima etapa del desarrollo es incluir el diagrama esquemático del circuito para permitir la reproducción del hardware. Con esta etapa se espera que la comunidad pueda mejorar diferentes aplicaciones del sistema desarrollado.

## English Summary

### Integral computer-aided system for the control design and implementation

**Abstract**

In this paper, the development of a highly interactive, low-cost, and personal computer based data acquisition and control system is presented. The system includes an embedded microcontroller based hardware and an open-standards software. The software uses block diagram programming as the base for the theoretical design and implementation of the control system, which makes the programming simple and fast. An important feature of this system is that it can execute an embedded control without the personal computer intervention, which facilitates the implementation of real-time control. Additionally, the system has the capacity to expand in order to create a distributed monitoring and control system by using a CAN bus as a communication interface. It is also shown in this paper the implementation of the developed system in different plants.

*Keywords:*

Experimentation environments, embedded or built-in systems, microcontrollers, real-time systems.

## Agradecimientos

Agradecemos al Decanato de Investigación y Desarrollo de la Universidad Simón Bolívar por el soporte económico mediante el proyecto S1-IN-CBTLI-008-10 para realizar esta investigación. A los Laboratorios “C” y “G” adscritos a la Unidad de Laboratorios de la Universidad Simón Bolívar, por facilitar la realización de los experimentos. Y por último, al post-grado de Instrumentación de la Facultad de Ciencias de la Universidad Central de Venezuela por su apoyo académico e institucional.

## Referencias

Albertos, P., Sala, A., 2004. El control borroso: Una metodología integradora. Revista Iberoamericana de Automática e Informática Industrial 1 (2), 22–31.

Demirtas, M., Sefa, I., Irmak, E., Colak, I., june 2008. Low-cost and high sensitive microcontroller based data acquisition system for renewable energy sources. In: Power Electronics, Electrical Drives, Automation and Motion, 2008. SPEEDAM 2008. International Symposium on. pp. 196–199. DOI: 10.1109/SPEEDHAM.2008.4581303

Dormido, S., Dormido-Canto, S., Dormido, R., Sánchez, J., Duro, N., 2005. The role of interactivity in control learning. Int. J. Engng Ed. 21 (6), 1122–1133.

Elettronica Veneta S.p.A., n.d. Transductor y control de velocidad y posición (Mod. G36A/EV). [Documento en línea] Consultado en junio de 2014 en: `http://www.elettronicaveneta.com/`.

Katrancıoğlu, S., Savaş, K., Erdal, H., 2010. A modular and low-cost data acquisition card design with multitasking support. Procedia Social and Behavioral Sciences 2, 5266–5270.

Leal, D., 2012. Diseño y construcción de sistema de adquisición de datos y actuación basado en pc para la enseñanza de diferentes estrategias de control. Trabajo de grado de maestría, Universidad Central de Venezuela, Caracas.

Ledin, J., 2004. Embedded Control Systems in C/C++: An Introduction for Software Developers Using MATLAB. CMP Media LLC, San Francisco.

Mosterman, P., Prabhu, S., Dowd, A., Glass, J., Erkkinen, T., Kluza, J., Shenoy, R., 2005. Embedded real-time control via matlab, simulink, and xpc target. In: Hristu-Varsakelis, D., Levine, W. (Eds.), Handbook of Networked and Embedded Control Systems. Birkhäuser, Boston, pp. 419–446.

Ogata, K., 2003. Ingeniería de Control Moderna, 4th Edition. Prentice Hall, Madrid.

Passino, K., Yurkovich, S., 1998. Fuzzy Control. Addison-Wesley.

Perko, K., Kocik, R., Hamouche, R., Trost, A., 2011. A modelling-based methodology for evaluating the performance of a real-time embedded control system. Simulation Modelling Practice and Theory 19 (7), 1594 – 1612.

Quanser Inc., 2011. Rotary Flexible Joint Workbook. [Documento en línea] Consultado en julio de 2014 en: `http://eecs.ucf.edu/~abehal/eel4612/lab/General/Flexible_Joint_Workbook.pdf`.

Rimvall, C. M., Jobling, C. P., 1999. Computer-aided control systems design. In: Levine, S. W. (Ed.), The Control Handbook. Vol. I. CRC Press, Boca Raton, Ch. 23, pp. 429–442.

Rodríguez, M., Perdomo, J., Strefezza, M., Colmenare, W., 2004. Control de una extrusora de plástico usando un control pi difuso adaptado con el error de predicción del modelo. Ciencia e Ingeniería 25, 61–66.

SargunaPriya.N, Mani.M, J., Amudha.P, March 2014. Monitoring and control system for industrial parameters using can bus. International Journal of Engineering Trends and Technology (IJETT) 9 (10), 479–484.

Sontag, E., 1998. Mathematical Control Theory: Deterministic Finite Dimensional Systems, sagunda edición Edition. Springer.

Uran, S., Šafarič, R., 2009. Web based control teaching. In: Tzafestas, S. (Ed.), Web-Based Control and Robotics Education. Vol. 38 of International Series on INTELLIGENT SYSTEMS, CONTROL AND AUTOMATION: SCIENCE AND ENGINEERING. Springer, Zographou, Ch. 3, pp. 61–82.

Wang, L.-X., 1997. A course in fuzzy systems and control. Prentice Hall.