Tercer Congreso Virtual, Microcontroladores y sus Aplicaciones

Desarrollo de controladores basado en microcontroladores PIC

Ing. Ramón Carrasco Duboué[1]; Msc. Guillermo Álvarez Bestard[2]
[1]Instituto de Geofísica y Astronomía, departamento de Astronomía, Cuba
[2]Instituto de Cibernética, Matemática y Física, departamento de Control Automático, Cuba
Correo-e: ramon@iga.cu; guille@icimaf.cu

***Resumen**: Este proyecto abarca el desarrollo de un controlador PID Profesional y un PID Difuso capaces de regular procesos diversos. Estos se implementan de forma discreta, permitiendo su programación en microcontroladores PIC. Para lograr observar el comportamiento del sistema en tiempo real se ha desarrollado un sistema de adquisición de datos que permite, a través de comunicación por puerto serie, configurar los controladores y monitorear el comportamiento del sistema a controlar.*

Para comprobar el funcionamiento de ambos controladores se emplea un motor de corriente directa, cuya velocidad se regula satisfactoriamente ante cambios en la referencia y en la carga. Esto permite concluir que se justifica la selección de ambos algoritmos, así como el uso de un microcontrolador que cumpla la función de regulador para diversos procesos.

1 Introducción

En la actualidad existen diferentes tipos de controladores con características propias que se diseñan para realizar el control de un proceso específico. Dos de los algoritmos de control más utilizados son el PID (Proporcional-Integral-Derivativo) y el Difuso, y en el laboratorio de control automático del Instituto de Cibernética, Matemática y Física (ICIMAF) es necesario implementar variantes específicas de los mismos para poder comparar su funcionamiento en tiempo real sobre diferentes procesos. Surge entonces la incógnita:

¿La implementación configurable de los algoritmos PID Profesional y PID Difuso sobre un microcontrolador permitirá regular satisfactoriamente en tiempo real un proceso determinado?

Los reguladores PID, a pesar de la gran gama de controladores existentes actualmente y de su antigüedad, se han ganado un puesto entre todos los controladores, siendo probablemente los más usados y los más adecuados para la mayoría de los casos, lo que explica la importancia de su estudio.

Por las razones antes aludidas se han diseñado distintas variantes del PID con el propósito de eliminar problemas que presentan las acciones proporcional, integral y derivativa, tales como el *Proportional Kick*, el *Integrator Windup* y el *Derivative Kick* respectivamente; así como otros comportamientos indeseados como el *Bumpless Transfer* y el ruido en la medición.

Uno de los estudiosos de estos algoritmos en Cuba es el Dr. Alberto Aguado Behar quien, basado en su experiencia, diseñó un PID con características particulares que le permitieron eliminar una gran cantidad de esos problemas, al cual llamó PID Profesional.

La lógica difusa fue introducida por el profesor Lofti Zadeh en el año 1965 y sus antecedentes fueron las ciencias lógicas. Después de 1991 la tecnología difusa pasó, de estudiarse solamente en laboratorios, a ser una fuerte herramienta en las industrias. Algunos ejemplos son:

- Control simplificado de robots (*Hirota, Fuji Electric, Toshiba, Omron*).
- Autoenfoque de cámaras para la transmisión de eventos deportivos (*Omron*).
- Control de trayectoria para automóviles (*Nissan, Subaru*).
- Sistemas de predicción de terremotos (*Seismology Bureau of Metrology, Japan*).

Teóricamente, los controladores difusos pueden cubrir un rango de operación mucho más amplio que los controladores PID clásicos, ya que pueden operar con ruidos y perturbaciones de diferente naturaleza. El desarrollo de estos controladores resulta, además, mucho más barato que el de otros, principalmente que otros controladores inteligentes. Estas y muchas otras ventajas rodean a los controladores basados en lógica difusa, lo que justifica su uso en situaciones diversas y su estudio en la rama de la ingeniería de control. Estos algoritmos se pueden implementar en su forma discreta, lo que permite su programación en un microcontrolador que, además de ser un dispositivo de bajo costo, permite una mayor velocidad de procesamiento y le da portabilidad y

adaptabilidad al controlador. También ofrecen la posibilidad de comunicarse a través de puerto serie con el usuario para variar diferentes parámetros a través de un software, mediante el cual se pueden almacenar datos y observar en tiempo real el comportamiento del sistema.

La novedad de esta investigación consiste en desarrollar, sobre un mismo dispositivo electrónico, un PID Profesional y un PID Difuso, preparados para regular diversos procesos en tiempo real; y un sistema de adquisición de datos que permita comparar su comportamiento sin requerir computadoras de altas prestaciones o un sistema operativo en tiempo real.

2 Materiales y métodos

2.1 Controladores PID

El regulador Proporcional-Integral-Derivativo (PID) es un controlador de tres términos que ha sido utilizado a lo largo de la historia en el campo del control automático [1]. Este regulador constituye, es sus distintas versiones, la solución más aceptada para los problemas de control en la industria. Esto se debe a que es considerado un regulador "natural" por excelencia, ya que su comportamiento imita al de un ser racional ante una toma de decisiones (la acción proporcional ofrece una estimación del estado actual, la integral de la historia pasada y la derivativa permite un pronóstico del futuro) [2]. El triunfo de estos controladores también está dado por el hecho de que en muchas ocasiones son el componente fundamental de reguladores más sofisticados [1].

Entre sus principales ventajas están su simpleza, su capacidad de lograr una respuesta adecuada en gran variedad de sistemas, y la posibilidad de ser implementados tanto analógica como digitalmente [3].

2.2 Variante de PID: PID Profesional

El objetivo de esta configuración es tratar de eliminar la mayor cantidad de problemas que presentan las tres acciones del PID, los cuales no se solucionan con los algoritmos de posición o velocidad [4] y que ocurren en los procesos con frecuencia.

- Algunas de las ventajas del PID Profesional [2] son: Al filtrarse la variable de control se elimina el ruido provocado por la acción derivativa.
- Al tener el integrador aparte permite iniciar este término con la posición del actuador y verificar su valor constantemente (para ver si está dentro del rango de control), evitando así que ocurra el efecto de *wind-up*.
- Al depender la acción derivativa del incremento de la salida en vez del error se elimina el *derivative kick* pues ya no será afectada por cambios bruscos en la referencia.

Esta variante PID se modifica de forma que quede en su forma paralela [4], permitiendo analizar las tres acciones por separado y trabajar con las variables típicas del PID, tales como Kp, Ki y Kd (ganancias proporcional, integral y derivativa respectivamente). Así, las 3 acciones en variables separadas quedan:

$$Proporcional(kT) = K_p e_n(kT) \quad (1)$$

$$Integral(kT) = Integral(kT-1) + T_c e_n(kT) K_i \quad (2)$$

$$Derivativo(kT) = K_d * \frac{y_f(kT) - y_f(kT-1)}{T_c R_y} \quad (3)$$

Quedando el cálculo de la variable de control como sigue:

$$u(kT) = Proporcional(kT) - Derivativo(kT) + Integral(kT) \quad (4)$$

- e_n: Error normalizado. Se calcula como sigue:

$$e_n(t) = \frac{y_{rf}(t) - y_f(t)}{R_y} \quad (5)$$

- T_c: Período de control.
- R_y: Rango del transmisor con que se mide la variable en unidades físicas (unidades de ingeniería).
- y_{rf}: Valor de referencia, eventualmente filtrado mediante un filtro exponencial de primer orden.
- y_f: Valor de la salida filtrada.

El diagrama en bloques se muestra en la fig. 1.

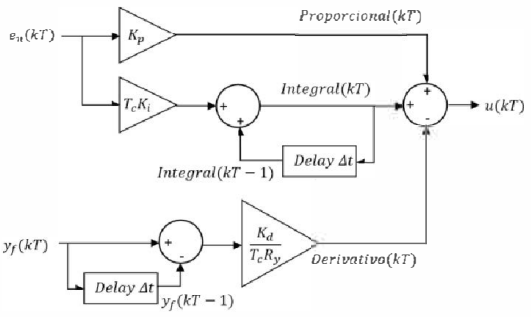

Fig. 1: *Diagrama en bloques correspondiente al PID Profesional en su forma paralela*

Estas modificaciones permiten ajustar los parámetros de forma empírica ya que, al poder analizar las

acciones por separado, se puede apreciar claramente el efecto que ejerce cada una sobre el proceso. Además, permite trabajar ya sea con un controlador P como con un I, un PI o un PD. Para esto solo es necesario hacer una ganancia u otra equivalente a cero.

Para el buen funcionamiento de este algoritmo de control digital se requiere el filtraje de las variables controladas, de manera que el ruido presente en las señales no provoque un comportamiento excesivamente oscilatorio en el control [2].

Independientemente de los filtros de hardware existentes, en general es conveniente el uso de dos tipos de filtros por software:

- Filtro para ruidos de alta frecuencia (inducción electromagnética, ruidos térmicos, etc.).
- Filtro para ruidos de frecuencia media y baja (ruidos en el proceso, variaciones en la composición de los materiales, falta de homogeneidad de los mismos, etc.).

Para el filtraje de los ruidos de alta frecuencia se utiliza un filtro promedio de varias réplicas de la medición en cada período de muestreo y para el filtraje de ruidos de frecuencia baja y media, se utiliza el concepto de período de control, que puede coincidir con el período de muestreo o ser un múltiplo del mismo. Este se define como el intervalo de tiempo que transcurre entre 2 acciones sucesivas de control [2].

2.2.1 Recursos empleados en el microcontrolador

Las funciones dedicadas al diseño de los controladores se deben ejecutar cada un período de control específico, por lo que son ejecutadas a través de los temporizadores del microcontrolador. El mínimo período de control que puede alcanzar el controlador PID Profesional es de 1 *ms* (aunque se restringe a 10 *ms* para cederle tiempo al resto de las tareas del microcontrolador), dando la posibilidad al mismo de tener una respuesta rápida ante cambios en el sistema.

Para la implementación de este algoritmo fue necesario el almacenamiento de algunas variables en memoria EEPROM, en este caso, los parámetros del controlador. El espacio total que ocuparon estos parámetros fue 14 *bytes*.

Este algoritmo, como se ha explicado previamente, se ejecuta cada un período de control (Tc) determinado, el cual está restringido por el tiempo que demora el microcontrolador en efectuar todas las operaciones correspondientes al cálculo de la variable de control. El cálculo del filtro promedio y lectura del A/D demora 154.40 *us* en ejecutarse, el cálculo del error 18 *us* y el cálculo del algoritmo de control 235.40 *us*. En total, la función del PID Profesional demora 786.20 *us*. Por tanto, el Tc mínimo sería de 1 *ms*.

En cuanto a memoria de programas (Flash) y memoria RAM, este algoritmo requiere 920 (2% del total) y 16 *bytes* respectivamente.

2.3 Controladores Difusos y PID Difuso

Los sistemas expertos de control difuso basados en reglas, conocidos como controladores difusos o FLC (*Fuzzy Logic Controllers*), o también sistemas de inferencia difusa o FIS (*Fuzzy Inference Systems*), son la aplicación más extendida de la lógica difusa [5].

Los FLC se utilizan para reducir el tiempo de desarrollo o para mejorar el rendimiento de un PID existente, y en el caso de sistemas muy complejos los FLC pueden llegar a ser la única solución. Estos son particularmente buenos para el control de procesos no lineales difíciles de controlar mediante métodos convencionales [6]. El objetivo principal de su uso es poder crear un modelo lingüístico basado en el conocimiento de un experto.

Con el objetivo de lograr abarcar una mayor diversidad de procesos y que el controlador difuso se adapte a estos mediante su configuración, se decidió implementar un controlador PID basado en lógica difusa, o sea, un *regulador difuso con estructura PID*. Al realizar un PID Difuso se aprovechan tanto las ventajas del algoritmo PID sobre sistemas lineales como las de un controlador difuso sobre sistemas no lineales y, además, al diseñarlo se conocen de antemano las variables del proceso y de control, así como la base de reglas y lo que ocurre cuando se lanza alguna regla.

El controlador difuso diseñado está conformado por 2 controladores difusos en paralelo, uno PD y el otro PI (en su forma incremental), como se muestra en la figura 2.

De esta manera el número de reglas disminuyen de n 3 a 2n [4]. Utilizando esta variante se obtienen 4 ganancias en vez de 3, ya que la ganancia proporcional afecta a cada controlador de manera diferente. Esto ocurre producto de que el controlador PI difuso se realiza en su forma incremental y su ganancia proporcional se multiplica con la *variación de error*, a diferencia del PD que se multiplica con el *error*.

Las funciones de membresía empleadas son las *triangulares*, *fusificador singleton*, *conjuntos normalizados*, *defusificador por centro de área* y el cálculo de *máximos* y *mínimos* frente a multiplicaciones (buscando eficiencia computacional). Cada controlador está compuesto por *25 reglas*, haciendo un total de *50 reglas*. En la figura

3 se muestra la base de reglas que emplean ambos controladores.

Estas reglas fueron conformadas priorizando la rapidez del sistema pero intentando mantener el *%MP* lo más bajo posible. Por ejemplo, si el error es **PB**, aunque la variación de error sea **DB**, la salida de control se mantiene alta, con el objetivo de alcanzar más rápidamente la referencia. Sin embargo, para valores más cercanos a la referencia, la acción de control será más suave (**NS**, **Z**, **PS**), con el objetivo de buscar un bajo error de estado estacionario. En la tabla **N** significa *negative*, **Z** *zero*, **P** *positive*, **S** *small* y **B** *big*.

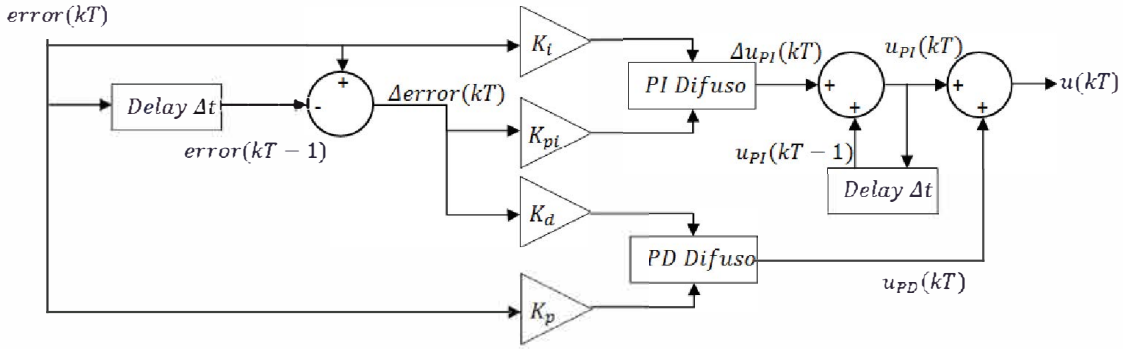

Fig. 2: *Diagrama en bloques correspondiente al controlador PID Difuso*

				de		
		DB	DS	Z	IS	IB
	NB	NB	NB	NB	NB	NB
	NS	NS	NS	NS	NS	NB
e	Z	Z	Z	Z	Z	Z
	PS	PS	PS	PS	PS	PB
	PB	PB	PB	PB	PB	PB

Fig. 3: *Base de reglas para los controladores PD y PI difusos*

El método de inferencia utilizado es el Mandani, ya que las reglas se formularon basándose en el conocimiento del comportamiento de los controladores PID, y como método de defusificación el "centro de área" o "centro de gravedad", el más utilizado mundialmente por su simplicidad y bajo costo computacional [7]. Además, la velocidad del algoritmo se puede cambiar ajustando el paso con que se recorre el universo de discurso.

En la figura 4 se muestran las funciones de membresía de entrada y salida de ambos controladores difusos, donde el **eje X** corresponde al universo de discurso y el **eje Y** al valor entre 0 y 1 que toma la función de membresía de cada una de las variables, las gráficas *a* y *b* corresponden a las variables de entrada de ambos *error* y *variación de error*, *c* a la variable de salida del controlador *PD* y *d* a la variable de salida del controlador *PI*, diseñado en su forma incremental.

Se eligen funciones de membresía triangulares con los puntos de cruce en 0.5 tanto para las entradas, buscando eficiencia computacional, como para las salidas, con el objetivo de que el controlador funcione para la mayor variedad de procesos. Si se emplean *singletons* aumenta la sensibilidad del regulador y, por lo tanto, para los sistemas más complejos el control resultará más difícil de realizar. Además, funciones de membresía más estrechas producen respuestas más rápidas, pero también mayor oscilación, pico máximo y tiempo de establecimiento; y aunque en algunos sistemas pueden lograr obtener un menor error de estado estacionario, si es demasiado estrecha puede ocurrir que no se logre siquiera alcanzar la referencia [8].

El universo de las entradas depende en este caso de los valores máximo y mínimo que pueden tomar el error y la variación de error, los cuales a su vez dependen del rango de la salida del sistema o variable sensada. Como el sensor está conectado al conversor A/D del microcontrolador, y este es de 10 *bits*, el rango de las variables de entrada será de -2^{10} (-1023) hasta 2^{10} (1023).

El universo de las variables de salida depende del sistema a controlar y puede ser ajustado como otro parámetro más. Sus máximos valores de mínimo y máximo dependen del rango de la variable de control. En este caso, como esta variable la proporciona un PWM del microcontrolador cuya resolución es de 10 *bits*, al igual que en las entradas, estos valores podrán configurarse en el rango desde -1023 hasta 1023.

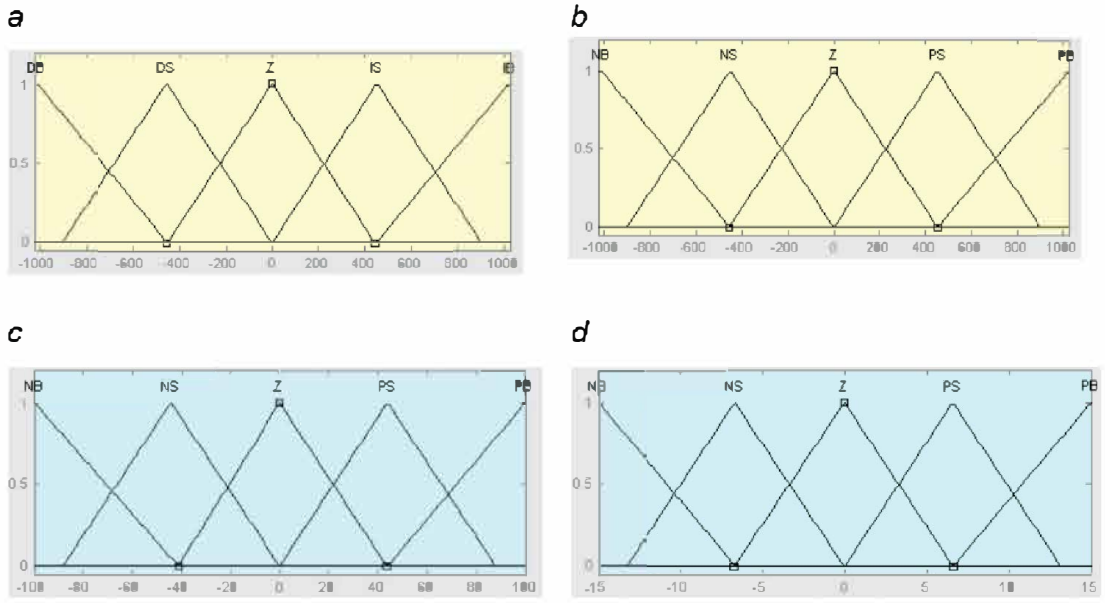

Fig. 4: *Funciones de membresía de entrada y salida de ambos controladores difusos* ***a)*** *Entrada "de" (variación del error de control),* ***b)*** *Entrada "e" (error de control),* ***c)*** *Salida "u" (variable de control),* ***d)*** *Salida "du" (variación de la variable de control)*

2.3.1 Recursos empleados en el microcontrolador

Este controlador, al igual que el analizado anteriormente (PID Profesional), está asociado a una interrupción por temporizador. La diferencia de este algoritmo con el anterior es que el tiempo de ejecución varía en función de diferentes parámetros configurables del controlador Difuso, por lo que el mínimo período de control dependerá del tiempo que se demore el microcontrolador en realizar esta función. Este período, para los parámetros definidos por defecto, es de 70 *ms*, mucho mayor que el del controlador PID Profesional, por lo que la respuesta ante cambios en el sistema podría ser más lenta.

Para la implementación de este algoritmo, al igual que con el PID Profesional, fue necesario el almacenamiento de sus parámetros en memoria EEPROM. El espacio total que ocupan estos parámetros es de 30 *bytes*.

A diferencia del PID Profesional, este requiere un mayor Tc pues las funciones referentes al mecanismo de inferencia requieren un mayor tiempo para ejecutarse, ya que todo el proceso de fusificación y defusificación lleva varias funciones y cada una demora un tiempo determinado. El cálculo del filtro promedio y lectura del A/D demora 154.40 *us*, el cálculo del error 15 *us* (este cálculo es más rápido que el del PID Profesional ya que no se calcula el error nominal, por lo que solamente se efectúa una resta) y el cálculo del algoritmo de control 41.873 *ms*, ya que cada función de inferencia demora 21 ms aproximadamente. En total, la función del PID Difuso demora alrededor de 42.3 *ms*. Por tanto, el Tc mínimo sería de 43 *ms* (aunque se restringe a 20 *ms* por encima de Tc para cederle tiempo al resto de las tareas del microcontrolador, al igual que se hace con el PID Profesional). Esto dependerá en gran medida del valor de los parámetros del regulador pues, mientras menor sea el paso con que se recorren los universos de discurso, más demorará el cálculo de las funciones correspondientes al mecanismo de inferencia. Por tal razón, el Tc de este controlador se autoajusta cada vez que los parámetros varían.

En cuanto a consumo de memoria Flash y RAM, hay que analizar cuanto utiliza individualmente cada una de las funciones que evalúan el algoritmo. En total ocupa aproximadamente 9720 *bytes* de Flash (30% del total) y 444 *bytes* de RAM. Este es el espacio en memoria que abarca este algoritmo pero, para la implementación del PID Difuso, fue necesario utilizar otras funciones, que son las encargadas de definir el rango de cada uno de los universos de salida así como el de sus conjuntos, las cuales se ejecutan tanto al inicio del programa como cuando se cambian los parámetros del controlador. Las mismas utilizan 5228 *bytes* de Flash (16% del total) y 243 *bytes* de RAM.

2.4 Obtención de las funciones dedicadas a realizar el mecanismo de inferencia

El *software* utilizado para el desarrollo de los dos controladores difusos fue el *xFuzzy* [4]. Después de diseñar cada controlador este *software* permite exportarlo a código *C*. Este código necesita modificarse para que el compilador *PIC C* lo pueda interpretar.

También se programaron funciones para poder variar algunos parámetros de los controladores difusos, tales

como los universos de salida y el paso que utilizan los algoritmos de defusificación para recorrer cada uno de los universos. Después de cambiar las dimensiones de un universo las funciones de membresía se reajustan sin perder la proporción o los puntos de cruce en 0.5.

Posteriormente se desarrolló un código en *C* basado en el exportado por el programa (*xFuzzy*), el cual agrupa los 2 controladores y está optimizado para minimizar el consumo de memoria en el microcontrolador.

2.5 Microcontroladores PIC

Los microcontroladores son un componente imprescindible de todos los sistemas de medición modernos, sistemas de accionamiento, controladores empotrados o embebidos y de cualquier sistema de medición y/o control. Estos tienen bajas y medias capacidades de procesamiento, reducido tamaño, comunicación a través de buses de campo, alta fiabilidad y bajo consumo de energía, y además trabajan en tiempo real. Por tanto, son dispositivos de bajo costo relativamente fáciles de programar [9].

El propósito del desarrollo de una teoría de control digital es proporcionar la capacidad de entender, diseñar y desarrollar sistemas de control donde una computadora sea utilizada como el controlador de un sistema, esto se aplica de igual modo en los microcontroladores. Además de realizar el control del sistema, este dispositivo puede realizar funciones de supervisión, tales como la posibilidad de comunicarse con un usuario.

Tradicionalmente, estos se han programado en lenguaje ensamblador, el cual presenta la desventaja de que el código se extiende mucho en dependencia de la complejidad del programa, lo que hace la confección y mantenimiento de un programa escrito en ensamblador relativamente difícil. Además, los microcontroladores fabricados por diferentes firmas tienen un diferente repertorio de instrucciones por lo que el usuario tiene que aprender un nuevo repertorio de instrucciones por cada microcontrolador que utilice. Afortunadamente, estos pueden ser programados en un lenguaje de más alto nivel, como el lenguaje *C*. Esto permite el desarrollo de programas más complejos a mayor velocidad y, además, es un lenguaje mucho más sencillo de aprender [10].

Los microcontroladores PIC son una familia manufacturada por *Microchip Technology Inc.* Actualmente los PIC son uno de los microcontroladores más populares, utilizados para la educación, aplicaciones industriales, domótica, equipos médicos y electrodomésticos [9]. El PIC elegido fue el 18F4550, ya que cumple con las características mínimas necesarias para el desarrollo del proyecto: el conversor análogo digital se puede emplear para la conexión del sensor, los temporizadores para calcular los algoritmos de control en períodos de tiempo específicos y el PWM se puede utilizar para aplicar la acción de control; además, se puede utilizar la memoria EEPROM para almacenar parámetros específicos de cada controlador y datos de interés.

2.6 Software de adquisición de datos y configuración

La función principal de este *software* [4] es la de permitir una interacción entre un usuario y el microcontrolador. Este tiene básicamente 2 modos de trabajo:

- Adquisición de datos: Este modo está previsto para que el usuario interactúe con el microcontrolador y cambie algunas de las características de los controladores. En este modo deben existir opciones relacionadas a la configuración del microcontrolador y a monitorear el estado actual de sus puertos.
- Control: Permite elegir entre los controladores PID Profesional y PID Difuso y observar el funcionamiento de los mismos sobre diferentes sistemas en tiempo real. También permite variar los parámetros de estos controladores para que funcionen correctamente en un proceso determinado.

Además, presenta una ventana historial que da la posibilidad al usuario de interactuar con la gráfica obtenida, permitiendo así un mejor análisis de la respuesta del sistema ante los reguladores. Los resultados se van almacenando en diferentes ficheros, por lo que se pueden analizar las diferentes variables utilizando el *software* MATLAB.

Para la selección del lenguaje a utilizar y del entorno de desarrollo se hace uso de la comparación desarrollada en [11] y se analizan diferentes criterios de comparación:

- Portabilidad.
- Capacidades 2D/3D.
- Matemática de precisión compleja.
- Gestión de memoria.
- Velocidad de ejecución.
- Licencia.
- Eficiencia.
- Modularidad.

Basado en los resultados obtenidos de esta comparación se escoge el lenguaje C++ con el objetivo de aprovechar su velocidad de ejecución, eficiencia y todas las potencialidades que ofrece de manera general.

Para la selección del entorno de desarrollo se tuvo en cuenta que fuera una aplicación sobre *software* libre por las ventajas que conlleva, tales como:

- Evita la dependencia tecnológica de empresas foráneas.
- Ahorros por pagos de licencias de *software*.
- Posibilidad de revisar el código fuente.

Basándose en estas necesidades se seleccionó el *Qt Creator*. Este es un IDE completamente integrado para el desarrollo de proyectos basados en las librerías *Qt* mediante el lenguaje *C++*, y está disponible tanto para plataformas *Windows* como *Mac OS X* y *Linux* [11]. Gracias al uso de estas librerías este IDE presenta alta portabilidad ya que permite utilizar el mismo código sobre diferentes plataformas sin realizar grandes cambios. La ventana de históricos y la de operación del software se muestra en la las figuras 5 y 6 respectivamente.

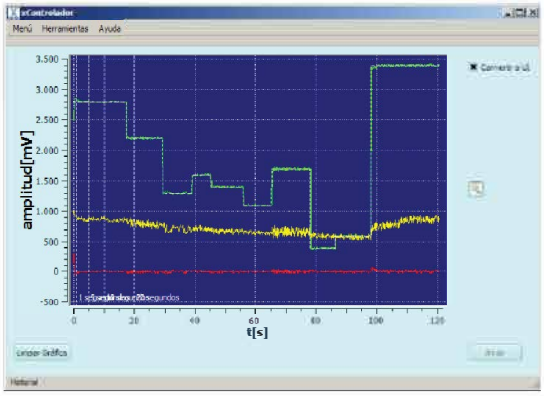

Fig. 5: *Ventana de históricos del software*

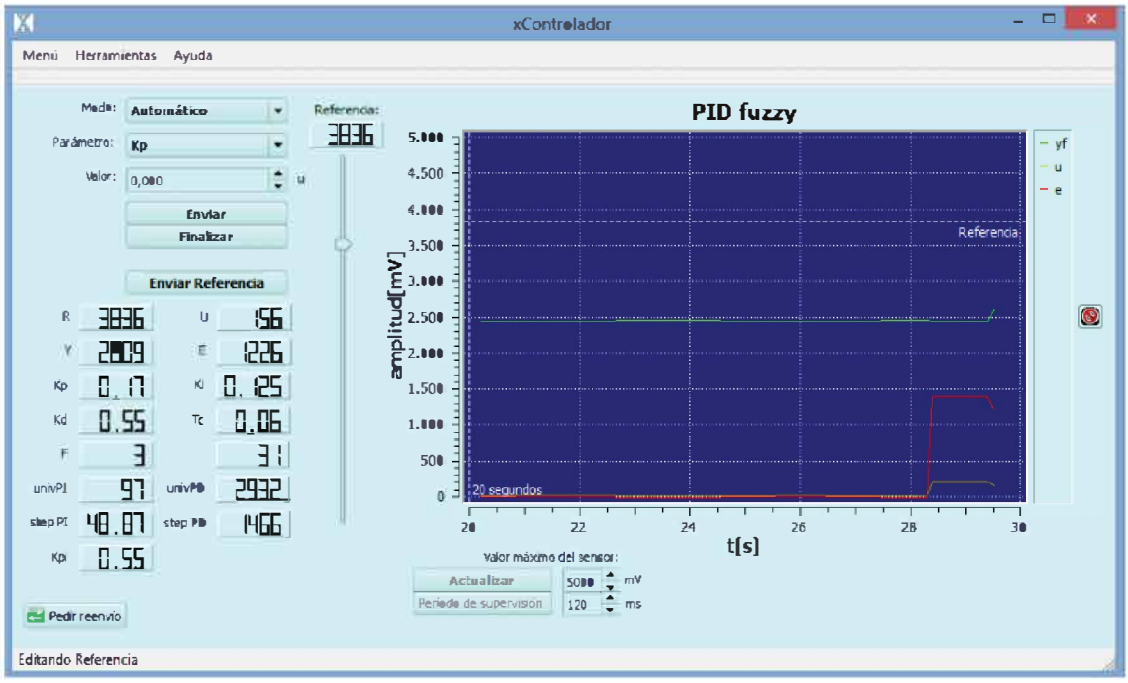

Fig. 6: *Ventana principal del software*

Para la comunicación con el microcontrolador se conforman tramas específicas, tratando de evitar pérdidas de información. Para esto se decidió utilizar un caracter de inicio de trama ('<') y uno de fin de trama (13 o 0x0D).

Una vez que el microcontrolador haya recibido y analizado la trama, devolverá el mismo caracter de inicio de trama con el objetivo de informar al software que se recibió correctamente. Si este no recibe este caracter en un tiempo determinado se mostrará un mensaje de error para alertar al usuario que ocurrió algún problema en la comunicación. En la figura 7 se muestra la ventana de configuración y donde también se pueden apreciar las tramas recibidas.

Fig. 7: *Ventana configuración y diagnóstico del software*

3 Resultados y discusión

Para probar los controladores desarrollados se decidió regular la velocidad de un motor de corriente directa. Para ello fue necesario el uso de un microcontrolador PIC18F4550 en la tarjeta de adquisición de datos MPIC1 de la Maqueta Motor [12], según se aprecia en la figura 8.

Fig. 8: *Interfaz MPIC1 y motor de corriente directa*

El microcontrolador constituye el elemento fundamental del circuito y es el encargado de realizar los algoritmos de control y la comunicación con la computadora.

Para esta prueba se utilizó una tarjeta de adquisición de datos que tiene incorporado el hardware necesario para la comunicación por puerto serie y para la manipulación del motor, incluyendo el sensor (tacómetro) con que se mide la velocidad del motor, la cual constituye la salida del proceso.

3.1 Diseño del experimento

Al programar el microcontrolador y conectarlo mediante el puerto serie con el software se realizó el ajuste de ambos controladores de forma experimental. Posteriormente se realizaron una serie de experimentos con el objetivo de comprobar el comportamiento de cada regulador sobre el sistema, entre los que se encuentran:

- Variaciones en la referencia: Se aplican estímulos tipo paso escalón en la referencia.
- Variaciones en la carga: Se realizaron variaciones sobre la fuente de alimentación, lo cual es equivalente a una perturbación en la carga al variar la velocidad del motor sin variar la acción de control.

3.2 Comprobando los controladores

Al aplicar una serie de pasos escalón en la referencia de igual amplitud para los controladores PID Profesional (PIDPR) y PID Difuso (PIDFR), se obtuvo la respuesta de la figura 9.

Se puede apreciar que la respuesta del sistema con el regulador PIDPR es mucho menos oscilatoria, estabiliza a mayor velocidad, y presenta menor %MP (porciento de pico máximo) y error de estado estacionario. Aun así, se puede decir que el PIDFR también presenta un comportamiento rápido y los resultados obtenidos con el mismo son satisfactorios. Si se calcula el promedio de los tiempos de establecimiento del sistema ante cada controlador, con los mostrados en la tabla I, se obtiene que para el PIDPR es de 0.1094 *s* (una respuesta muy rápida) y para el PIDFR es de 0.8324 *s*.

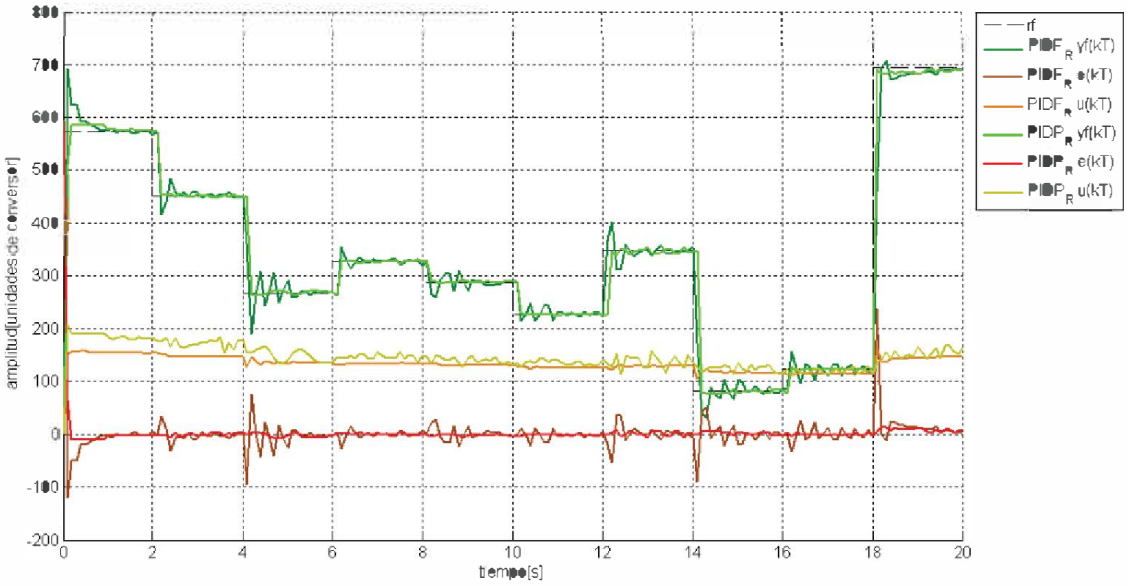

Fig. 9: *Gráfica de ambos controlador ante varios cambios en la referencia*

TABLA I

Tiempos de establecimiento del sistema (expresados en segundos) ante 10 pasos escalón aplicados en la referencia utilizando ambos controladores

Pasos	1	2	3	4	5
$PIDP_R$	0.1737	0.0907	0.0945	0.0912	0.0870
$PIDF_R$	0.7800	0.4000	1.0875	0.4500	0.9952
Pasos	**6**	**7**	**8**	**9**	**10**
$PIDP_R$	0.0991	0.0966	0.0991	0.1643	0.0974
$PIDF_R$	0.8463	0.6091	1.7725	2.0000	0.3838

Al realizar la prueba ante variaciones en la carga (figura 10) se puede apreciar que cuando ocurren las perturbaciones, los picos del sistema con el regulador PIDFR fueron mucho mayores que con el PIDPR, pero a pesar de esto el tiempo de establecimiento es relativamente pequeño, considerando las fuertes variaciones aplicadas. Estos valores se aprecian en la tabla II.

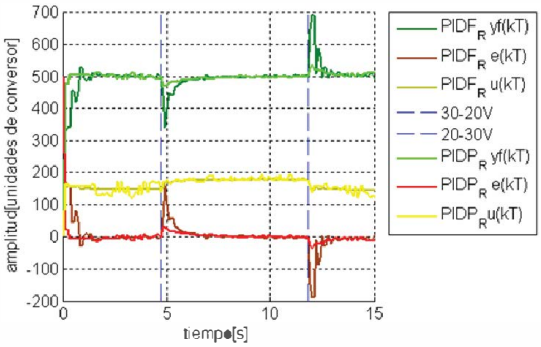

Fig. 10: *Respuesta de ambos controladores ante variaciones de 10V en la fuente de alimentación*

TABLA II

Tiempos de establecimiento del sistema (expresados en segundos) ante las perturbaciones aplicadas a la carga utilizando ambos controladores

Controlador	30-20V	20-30V	Promedio [seg.]
$PIDP_R$	1.5000	1.1125	1.0234
$PIDF_R$	2.1369	1.1453	1.3154

3.3 Comparación de ambos controladores con las respuestas obtenidas en simulación utilizando tanto Matlab como Proteus

Si se comparan las respuestas obtenidas mediante la simulación del sistema utilizando primeramente MATLAB (PIDPM y PIDFM) y después Proteus (PIDPP y PIDFP) con las obtenidas en el proceso real, se puede apreciar una notable similitud entre estos resultados. Para realizar la simulación se emplea la función transferencial del proceso, obtenida previamente en otras investigaciones.

En la figura 11 se puede apreciar que la salida obtenida utilizando el controlador PIDFR, presenta el mayor pico máximo. A pesar de esto, ambos controladores lograron alcanzar tiempos de establecimiento casi tan pequeños como los alcanzados en simulación utilizando MATLAB, y son más rápidos que los simulados sobre Proteus.

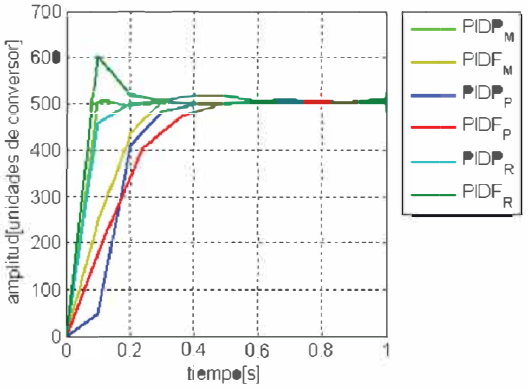

Fig. 11: *Salidas (yf) de los controladores*

En las tablas III y IV se muestran las características de estado estacionario de las 6 gráficas. **Yee** representa el valor en que se estabiliza la salida, **Eee** es el error de estado estacionario, **mp** es el pico máximo, **tp** es el tiempo en que se alcanzó el pico máximo, **%MP** es el porciento de pico máximo, **td** representa el tiempo de demora, **tr** el de subida y **ts** el tiempo que demoró en estabilizarse (utilizando el criterio del 2%). Todos los tiempos se expresan en segundos, y **Yee**, **Eee** y **mp** en unidades del conversor.

TABLA III

Características de estado estacionario de la variable de salida yf ante la acción de los 6 controladores

Controlador	Yee	Eee	mp	tp
$PIDP_M$	500.0011	0.0011	504.0704	0.1146
$PIDF_M$	500.1831	0.1831	502.3537	0.3300
$PIDP_P$	500.0000	0.0000	500.0000	1.6000
$PIDF_P$	497.0000	-3.0000	504.0000	0.6000
$PIDP_R$	497.0000	-3.0000	507.0000	1.1000
$PIDF_R$	498.0000	-2.0000	603.0000	0.1000

Con esta respuesta se puede ver que, a pesar de que virtualmente el controlador PID Difuso tiene un comportamiento más lento que el PID Profesional, en la realidad no necesariamente tiene que ser así, aunque para lograrlo haya que comprometer otros parámetros.

TABLA IV

Características de estado estacionario de la variable de salida yf ante la acción de los 6 controladores (continuación)

Controlador	%MP	td	tr	ts(2%)
PIDP$_M$	0.8139	**0.0621**	**0.0848**	**0.0812**
PIDF$_M$	**0.4340**	0.1096	0.1995	0.2751
PIDP$_P$	**0.0000**	0.1555	**0.1528**	**0.3571**
PIDF$_P$	1.4085	**0.1407**	0.2864	0.4368
PIDP$_R$	**2.0121**	0.0543	0.0868	0.3000
PIDF$_R$	21.0843	**0.0412**	**0.0660**	**0.2842**

Las salidas del sistema PIDPR y PIDFR son sin duda las de mayor interés, ya que se obtuvieron sobre el sistema real y se puede apreciar que los resultados son muy buenos. Ambos controladores lograron alcanzar bajos tiempos de establecimiento, de subida y de demora, al igual que bajos valores de error de estado estacionario. La única característica que sobresale es el pico máximo del controlador PID Difuso que, como se explicó anteriormente, se compromete para lograr rapidez en la respuesta. Por tanto, se puede decir que las respuestas obtenidas son satisfactorias.

4 Conclusiones

En este proyecto se realizó el estudio sobre diferentes técnicas de control utilizando reguladores PID y basados en inteligencia artificial, como los controladores difusos. El estudio y la selección de ambos controladores se realizó buscando en todo momento poder regular la mayor cantidad de sistemas, variando solamente algunos parámetros de cada controlador.

Se logró diseñar un controlador PID Profesional discreto en el microcontrolador, así como un controlador PID Difuso, obteniendo resultados satisfactorios y demostrando la potencia de ambos reguladores, de manera que se justificó su selección.

El uso de un microcontrolador permitió tener un controlador muy poderoso sin estar atado a una máquina de escritorio, y brinda una alta precisión y rapidez a la hora de realizar los cálculos correspondientes a cada algoritmo. Haciendo uso del puerto serie se permite la comunicación con una PC, dando la alternativa de que un usuario pueda hacer cambios sobre los parámetros del regulador seleccionado y comunicarse con el mismo. Esto permite observar el funcionamiento del controlador sobre diferentes sistemas en tiempo real. Además, da la posibilidad de almacenar datos utilizando memoria EEPROM, por lo que los parámetros correspondientes a cada controlador una vez actualizados, no se borrarán.

Con las modificaciones realizadas en las tramas utilizadas para la comunicación serie se logró una correcta interacción con la PC, ya que mediante el software se puede actuar tanto sobre la configuración del sistema como sobre los parámetros de cada controlador: Esto permite el ajuste de los reguladores en dependencia del proceso a controlar, pues mediante el uso de una gráfica se puede ver fácilmente y en tiempo real el comportamiento de las variables principales del sistema.

Es importante resaltar que esta aplicación puede ser utilizada tanto como herramienta docente como en una industria para supervisar el proceso controlado y permite ajustar los controladores en dependencia de las características del mismo.

Referencias

[1] A. Visioli, Practical PID Control. England: Springer-Verlag, 2006.

[2] A. Aguado, Identificación y control adaptable de procesos, Segunda. La Habana: Academia, 2010, ch.4.

[3] J. Miklěs and M. Fikar, Process Modelling, Identification and Control. Berlín: Springer-Verlag, 2007.

[4] R. Carrasco, Diseño e implementación de un PID Profesional y un PID Difuso utilizando un microcontrolador PIC18F4550. Tesis de Diploma, ISPJAE, 2013.

[5] B. M. del Brío and A. S. Molina, Redes Neuronales y Sistemas Difusos, Segunda. España: Alfaomega, 2001.

[6] A. M. Ibrahim, Fuzzy Logic for Embedded Systems Applications. USA: Newnes, 2004.

[7] Z. Kovačić and S. Bogdan, Fuzzy Controller Design. Theory and Applications. USA: Taylor & Francis Group, LLC, 2006.

[8] L. Reznik, Fuzzy Controllers. Australia: Newnes, 1997.

[9] D. Ibrahim, Microcontroller Based Applied Digital Control. England: John Wiley & Sons, Ltd, 2006.

[10] D. Ibrahim, Advanced PIC Microcontroller. Projects in C. USA: Newnes, 2008.

[11] J. J. Travieso and Y. M. Ortega, "Programación de un Planificador Semafórico. Trabajo de Diploma para optar por el título de Ingeniero en Automática." 2010.

[12] ICIMAF, "Documentación de la Interfaz MPIC-1 para la Maqueta Motor." 2004.