Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 2039-2060 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

APLICACIONES DIDÁCTICAS EN CONTROL AUTOMÁTICO CON LEGO® MINDSTORMS NXT

Leonardo Bermeo[a,b], David Herrera[b] y Ricardo Ramirez[c,d]

[a] *LCA - Laboratorio de Controle e Automação, Programa de Engenharia Elétrica, COPPE, Universidad Federal de Rio de Janeiro, Sala I-148, Cidade Universitária, Ilha de Fundão, CEP 21945-970, Rio de Janeiro, RJ, Brasil, lbermeoc@ufrj.br,* www.gmun.unal.edu.co

[b] *Laboratorio de Control, Universidad Nacional de Colombia, Departamento de Ingeniería Eléctrica y Electrónica, Edificio 411, Sala 202, Bogotá, Colombia, lbermeoc@unal.edu.co, dherreraa@unal.edu.co,* http://www.gmun.unal.edu.co

[c] *Grupo de Plataformas Robóticas UnRobot , Universidad Nacional de Colombia,Departamento de Ingeniería Mecánica y Mecatrónica, Of. 453-401, Ciudad Universitaria, Bogotá,Colombia, rramirez@ufrj.br,* http://www.unrobot.unal.edu.co/

[d] *Laboratorio de Robótica y Diseño de Máquinas LabRob, Programa de Ingeniería Mecánica, COPPE, Universidade Federal do Rio de Janeiro, Sala I-101, Cidade Universitária, Ilha de Fundão, CEP 21945-970, Rio de Janeiro, RJ, Brasil, rramirez@ufrj.br,*
http://www.labrob.coppe.ufrj.br

Palabras Clave: Síntesis de controladores, Control PID, Identificación de Sistemas, Educación.

Resumen.

En este trabajo se describe una propuesta didáctica completa para un curso experimental en el tema de control automático, usando los módulos didácticos LEGO NXT. La propuesta incluye software, un conjunto de plantas experimentales y una metodología de trabajo. El software presentado está desarrollado en herramientas GNU. Las plantas mecatrónicas están basadas en LEGO y tienen ventajas importantes en costo, flexibilidad y repetibilidad. Algunas de las plantas presentadas son: servomecanismo, pendulo inverso, robot móvil y sistema de viga y bola.

Complementariamente, se propone una metodología con la cual es posible alcanzar en un mismo curso práctico la integración de conceptos importantes como: simulación, dinámica e identificación de sistemas, diseño mecánico, diseño de algoritmos de control y sistemas operativos de tiempo real.

1. INTRODUCCIÓN

Existe un destacado interés mundial en presentar propuestas y proyectos para realizar en laboratorios de control a nivel de pregrado, teniendo en cuenta limitaciones en costo y espacio (Gawthrop y E.McGookin (2004); Bernstein (2005b,a); K.J. Åström (2004)). La conclusión de quienes trabajan la enseñanza de control es una: en el área del control automático es indispensable la unificación praxis-teoría. El notable profesor Dennis Bernstein de la Universidad de Michigan ha plasmado esta conclusión de manera contundente en la introducción de una edición especial del IEEE Control Magazine dedicada a la educación en control (Bernstein (2005a)):

> "*El control es un tema que tradicionalmente de se enseña de una manera conceptual general. Paradójicamente, es un área que prospera solamente cuándo los estudiantes pueden ver aplicaciones a través casos de estudio puntuales*".

Uno de los problemas relacionados con la enseñanza experimental de control es su costo. Los costos de los sistemas producidos por las empresas dedicadas a desarrollos didácticos hacen difícil su acceso masivo en las universidades.

El propósito fundamental de la enseñanza de control experimental es que un estudiante recorra los pasos metodológicos de síntesis de un controlador (Bermeo y Díaz (2007)) como son:

Paso 1 *Implementación de la planta.* Se realiza la integración de los componentes constitutivos (hidráulicos, mecánicos, electrónicos, etc) del sistema a ser controlado, sensores y actuadores.

Paso 2 *Modelamiento y/o Identificación.* En este paso se obtiene un modelo lineal continuo o discreto del proceso a controlar.

Paso 3 *Diseño preliminar.* Con el modelo obtenido y especificaciones dadas (en el tiempo ó en la frecuencia) para el sistema de lazo cerrado, se hace un diseño preliminar del controlador en el dominio continuo o discreto (Franklin et al. (1997)).

Paso 4 *Discretización.* Si el controlador se diseñó en el dominio análogo se usa genéricaménte la transformación bilineal. El objetivo único de este paso es seleccionar cuidadosamente el tiempo de muestreo T, ponderando el tiempo de cálculo del procesador con la pérdida de margen de fase debida al retardo indroducido por el controlador digital.

Paso 5 *Implementación digital en un lenguage de alto nivel.* Normalmente se usa C y un sistema operativo de tiempo real (RTOS) (Lurie y Enright (2000)).

Paso 6 *Pruebas de Hardware.* En esta parte se realizan directamente las pruebas del controlador diseñado para todas las condiciones de operación previsibles del sistema. La pruebas realizadas determinaran un ciclo de iteraciones en la síntesis del controlador desde el paso 2. Eventualmente, si existe mucha divergencia entre las predicciones del

paso 3 y los resultados experimentales, será necesario rehacer el modelo del sistema desde el paso 1.

El propósito fundamental de la enseñanza de control experimental es que un estudiante recorra los pasos metodológicos de síntesis de un controlador de manera ágil y sin tener que abordar problemas secundarios (calibración de sensores, diseño de acoples eléctricos o mecánicos, etc) que desvían la atención del problema central. La componente experimental de un curso de control debe perseguir dos objetivos claros:

I. Aplicar sobre un mismo sistema varias estrategias de control para establecer comparativamente límites y ventajas de cada una.

II. Realizar controladores para sistemas de diferente dinámica y dominio físico. Al menos se deben incluir experimentos sobre sistemas con polos estables, resonantes, en el origen,en el semiplano derecho, retardos y/o ceros no minifase.

Esta artículo propone un curso experimental, que complementa la componente teórica de los cursos de control. La propuesta incluye software, material didáctico, modelos matemáticos y guias de construcción de los experimentos. Estas prácticas han sido probadas por dos años en los cursos de laboratorio de control del Departamento de Ingeniería Eléctrica de la Universidad Nacional de Colombia.

Este articulo se estructura de la manera siguiente:

En la sección 2 se presentan brevemente las herramientas de software utilizadas y la plataforma Mindstorms. En la sección 3 se presenta una práctica introductoria para demostrar el efecto de las acciones ON-OFF y PID en el control de un servomecanismo. En la sección 4 se describe una práctica más avanzada para el control de un robot móvil que contiene todos los pasos de la metodología de síntesis de un controlador expuesta anteriormente. Estos pasos van de la identificación paramétrica del sistema a la implementación digital en C y el RTOS NXT-OSEK. En sección 5 se describen dos plantas inestables con las cuales se experimenta al final del curso: una versión del pendulo inverso (SEGWAY) y el clásico problema de viga y bola. Finalmente, en la sección 6 se hacen consideraciones acerca de los resultados obtenidos en los cursos impartidos.

2. HERRAMIENTAS USADAS

En esta sección se describen brevemente las herramientas utilizadas en el desarrollo de las prácticas propuestas. A nivel de software, se puede utilizar Matlab, ó, si se prefiere FOSS, Python y Octave para el diseño de controladores e identificación de sistemas. La programación de del sistema embebido de LEGO® se efectúa con las herramientas de programación NXC y NXT-OSEK. A nivel de hardware, los experimentos son construidos únicamente con componentes de LEGO® Mindstorms.

2.1. Matlab® y GNU Octave

Se usan para identificación de sistemas y en diseño, simulación y discretización de controladores.

2.2. Python

Tiene el mismo uso que MATLAB. Parte de este trabajo es el desarrollo de una libreria en Python enlazada con el software OCTAVE para la identificación de sistemas, diseño, simulación y discretización de controladores. Esto para el caso de usar solamente software GNU, que los autores consideran muy importante en el campo educativo.

2.3. NXC

NXC (Not eXactly C) es un lenguaje de programación estándar para los Bricks NXT. Es un conjunto completo de funciones que permiten acceder al hardware del Brick, y que además ofrece una curva de aprendizaje muy rápida. Esta limitado a variables en punto fijo.

Con NXC se pueden escribir rutinas de control básico. La diferencia fundamental con el código C estándar, radica en las funciones creadas para el manejo de los Bricks y el tipo de datos que soporta (solo tipo entero).

2.4. NXT-OSEK

Es un sistema operativo de tiempo real (RTOS) para el Brick NXT basado en el estándar OSEK de la industria automotriz. Este provee un ambiente de programación en C, una API para los sensores y motores NXT así como para otros dispositivos, soporte para operaciones en punto flotante, y capacidad de ejecutar tareas múltiples en tiempo real.

Con nxtOSEK se pueden desarrollar filtros, controladores avanzados, y rutinas de identificación.

2.5. LEGO® Mindstorms

Este es un conjunto de piezas, motores, sensores y actuadores, desarrollado por LEGO, para el diseño e implementación de robots; inicialmente pensado para niños mayores de doce años. Sin embargo, la herramienta ofrece tal flexibilidad en la educación en tecnología, que su uso se ha extendido desde el nivel de secundaria hasta cursos de posgrado en ingeniería.

Las principales ventajas que presenta son su bajo costo y enorme flexibilidad teniendo en cuenta el número de experimentos que pueden obtenerse a partir de un mismo kit. En nuestro caso ha sido usado el kit LEGO MINDSTORMS® para educación. El kit incluye varios sensores, 3 servomotores y más de 400 piezas. El bloque embebido programable, que en el resto del artículo se denomina Brick, es basado en un procesador ARM de 32-bits con una capacidad de cómputo notable. Una descripción completa del kit didáctico se encuentra en la página del fabricante (LEGO (2010)).

Figura 1: kit de educación 9797 de LEGO®. Cortesía del fabricante

3. ACCIONES FUNDAMENTALES APLICADAS EN EL CONTROL DE UN SERVOMECANISMO

Las prácticas propuestas son una introducción a las acciones básicas de control, y un ejercicio de profundización que incluye la identificación del sistema y la aplicación de técnicas de cálculo e implementación de controladores digitales.

3.1. Servomotor: Acciones básicas de control

En esta práctica se busca comprender, en una primera aproximación intuitiva, las acciones básicas de control. Es una convención que la literatura denomine *acciones básicas de control* a la acción *ON-OFF* y las acciones *proporcional (P)*, *integral (I))* y *derivativa (D)* (Åström y Wittenmark (1997)).

El experimento inicial propone el control de posición y velocidad angular de un servomotor de LEGO®. Por ser el ejercicio para comenzar, no se introduce un modelo matemático de la planta (el servomotor) porque esta primera aproximación al problema del control realimentado es de carácter cualitativo. Su propósito es mostrar las acciones básicas de control, el efecto de la realimentación, el error estacionario, los ciclos límite, las perturbaciones, entre muchas otras nociones fundamentales.

Como parte del trabajo en laboratorio, los estudiantes construyen la planta (su construcción toma unos 10 minutos aproximadamente); adicionando un componente lúdico a la sesión práctica. Una secuencia detallada de pasos de construcción puede ser generada mediante el software CAD LEGO® Digital Designer, provisto por el fabricante. Mediante el mismo software se produce la figura 2 que muestra el ensamble mecánico necesario para el primer experimento. El servomecanismo es estético, robusto y satisface los requerimientos para ser la planta de este experimento.

Figura 2: Servomecanismo para el primer experimento

3.1.1. Control ON-OFF

Se trata de la estrategia más simple de control en lazo cerrado (conocida desde los griegos), en la que se usa toda la energía correctiva disponible para llevar la salida del sistema al punto deseado.

$$u(t) = \begin{cases} u_{max} & \text{si } e(t) > 0 \\ u_{min} & \text{si } e(t) < 0 \end{cases} \tag{1}$$

Esta estrategia se programa fácilmente en un lenguaje de alto nivel; requiriendo un conocimiento elemental de programación. El código del algoritmo de control ON-OFF en NXC se muestra en el algoritmo 1, y en NXT-OSEK en el algoritmo 2. Tales códigos hacen parte del material didáctico entregado a los estudiantes para que, durante la sesión, ellos solo tengan que hacer algunas modificaciones de los parámetros; por ejemplo, el porcentaje de energia entregada al motor. Los estudiantes hacen las modificaciones, compilan y programan el Brick NXT en un proceso que tarda un par de minutos, casi como si el ejercicio fuera hecho solo a nivel de simulación.

Un factor que merece consideración es que los estudiantes, desde esta fase inicial, están conscientes de que la realización de controladores no está un mundo pitagórico intangible de transformadas z o de Laplace, sino que es un problema práctico de programación que requiere de compiladores de alto nivel e incluso sistemas operativos en tiempo real como el NXT-OSEK.

Las implementaciones mostradas en el algoritmo 1 y en el algoritmo 2 son programadas en C. Solo que en el segundo caso se un sistema operativo de tiempo real que involucra definiciones adicionales como la duración y ejecución de las tareas. Desde un punto de

Algorithm 1: Controlador ON-OFF en NXC.

```
long r,e,umax,umin;

task main()
{
 umax=100;
 umin=-100;
 r=180;
 while(true)
        {
        y = MotorRotationCount();
        e = r-y;
        if(e>0){u=umax;}
        if(e<0){u=umin;}
        OnFwd(OUT_A,u);
        }
}
```

vista pedágogico es mejor iniciar con la implementación en NXC (algoritmo 1), pero introducir el uso de una herramienta avanzada como el NXT-OSEK muestra las ventajas de tener un RTOS para hacer control. Asi mismo, los estudiantes van siendo preparados en el manejo de una herramienta que necesitarán para las prácticas más avanzadas en el final del curso.

Como un ejemplo de los comportamientos dinámicos obtenidos en esta práctica, en la Figura 3 se muestran respuestas para diferentes valores máximo y mínimo en la energia de la señal de control (notada u_{max}, y u_{min}), y sus efectos sobre el ciclo límite y el tiempo de respuesta del sistema. Aqui es importante resaltar que, por medio del software desarrollado, el alumno puede obtener directamente una gráfica como la mostrada; sin embargo, lo más importante en esta etapa es la percepción directa de los fenómenos involucrados.

3.1.2. Control PID

La estructura de Control PID es predominante en las aplicaciones industriales. Más del 90 % de las aplicaciones de control de procesos en la industria están basadas en el PID (Åström y Hagglund (2006)). Esta realidad impone que los futuros ingenieros que tengan que tratar con procesos controlados tengan conceptos muy elaborados de los efectos producidos por cada una de las acciones de control en un PID. El hecho de poder percibir y prever a un nivel intuitivo los efectos de cada acción, permite que se inicie adecuadamente el desarrollo de estos importante conceptos.

El control PID implementado en el laboratorio es de la forma no interactuante (llamada a veces de textbook (Åström y Hagglund (2006)):

Algorithm 2: Controlador ON-OFF en NXC.

```
#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"
#include "math.h"

DeclareCounter(SysTimerCnt);
DeclareTask(Task1);

void user_1ms_isr_type2(void)
{
  StatusType ercd;
  ercd = SignalCounter(SysTimerCnt);
  if (ercd != E_OK)
  {
    ShutdownOS(ercd);
  }
}

int y=0;
int ref=180;
double e,u,umin,umax;
umin=-100;
umax=100;

TASK(Task1)
{
  y = nxt_motor_get_count(NXT_PORT_A);
  e = (double)ref-(double)y;
  if(e>0){u=umax;}
  if(e<0){u=umin;}
  nxt_motor_set_speed(NXT_PORT_A,(int)u,0);
  TerminateTask();
}
```

$$u = k_p e + k_i \int_0^t e\,dt + k_d \frac{d}{dt} e \tag{2}$$

Esta no es la implementación usual en un controlador industrial que debe incluir muchos otros detalles, pero es la más adecuada para entender los efectos individuales de cada acción.

Las acciones integral y derivativa son aproximadas usando diferencias finitas:

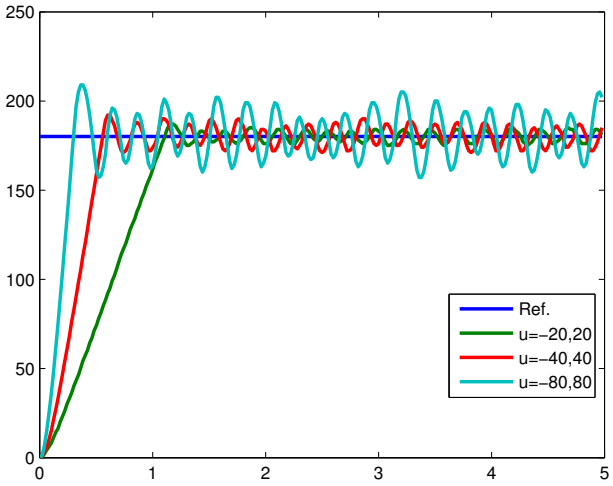

Figura 3: Respuesta del servomecanismo con control ON-OFF

$$u_d^k = k_d \Delta t \left[e^k - e^{k-1} \right] \tag{3}$$

$$u_i^k = k_i \Delta t \, e + u_i^{k-1} \tag{4}$$

Al momento de realizar esta práctica, no es necesario que los estudiantes tengan conocimientos en control digital. Los conocimientos necesarios para entender estas implementaciones ya han sido desarrollados plenamente en los cursos de cálculo diferencial e integral de cualquier programa de ingeniería. En el primer caso (ecuación 3), por la aproximación de la derivada por la recta secante; y en el segundo (ecuación 4), mediante el cálculo de integrales por métodos numéricos, usando la suma de Riemman. La acción integral está representada por la acumulación de la variable u_i que representa la señal de control. La acción derivativa u_d por la diferencia del error entre dos instantes de tiempo que pueden considerarse "cercanos" en la aproximación de la derivada. *Este es un buen momento y lugar para ver la acción, importancia y potencia de esos conocimientos del cálculo elemental, en el mundo real.*

La implementación del control PID en NXC se muestra en el algoritmo 3, y en NXT-OSEK en el algoritmo 4. Estos programas son entregados a los estudiantes como parte del material de la guía. Nótese que sintonizar unas nuevas constantes del PID es solo cambiar los números en las primeras líneas de código.

En la experimentación del servomecanismo controlado por un PID se explora el efecto conjunto y separado de cada una de las acciones de control . En la metodología propuesta, los estudiantes modifican cada una de las constantes, verifican la dinámica y en la misma aula de laboratorio se discuten los resultados obtenidos.

Debe comentarse que reprogramar el Brick NXT para lograr un controlador PID con nuevas constantes tarda menos de 30 segundos. Esto permite que en la misma sesión se proponga sintonizar el PID para obtener diferentes comportamientos dinámicos. Dada la

Algorithm 3: Controlador PID en NXC.

```
long kp,ki,kd,up,ui,Ts;
long ud,uiant,e,eant,r,y;

sub inicializar()
{
 SetSensor(S1,SENSOR_TYPE_ROTATION);
 ClearSensor(S1);
 r = 180;
 Ts = 20;
 kp = 2;
 ki = 1;
 kd = 1;
}

sub control()
{
 y = MotorRotationCount(OUT_A);
 e = r-y;
 up = kp*e;
 ui = uiant + ki*Ts*e;
 ud = kd*(e-eant)/Ts;
 u = up + ui + ud;
 if (u > 100) {u = 100;}
 if (u < -100) {u = -100;}
 OnFwd(OUT_A,u);
 eant  = e;
 uiant = ui;
}

task main()
{
 inicializar();
 while(true)
        {
        control();
        Wait(Ts);
        }
}
```

facilidad de programación y reprogramación del Brick, esto se hace con el mismo nivel de dificultad de un simulador, pero con la diferencia cognitiva profunda de percibir el fenómeno real.

Las figuras 4, 5, y 6 muestran las respuestas del motor usando controladores P, PI, y

Algorithm 4: Controlador PID en nxtOSEK.

```
#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"
#include "math.h"

DeclareCounter(SysTimerCnt);
DeclareTask(Task1);

void user_1ms_isr_type2(void)
{
  StatusType ercd;
  ercd = SignalCounter(SysTimerCnt);
  if (ercd != E_OK)
  {
    ShutdownOS(ercd);
  }
}

int y=0;
int ref=180;
double e,up,ui,ud,u,uiant=0;
kp=2; ki=0.5; kd=0.1;

TASK(Task1)
{
  y = nxt_motor_get_count(NXT_PORT_A);
  e = (double)ref-(double)y;
  up = kp*e;
  ui = uiant+ki*Ts*e;
  ud = kd*(e - eant)/Ts;
  u = up + ui + ud;
  if(u>100){u=100;}
  if(u<-100){u=-100;}
  eant = e
  uiant = ui;
  nxt_motor_set_speed(NXT_PORT_A,(int)u,0);
  TerminateTask();
}
```

PID, respectivamente. En la misma sesión de sesión de laboratorio puede verse el efecto sobre el servomecanismo de aplicar una acción de control integral (error de estado estacionario cero, incremento del sobrepico), y una acción de control derivativa (reducir las oscilaciones, mejorar el tiempo de respuesta). También se pide a los estudiantes reproducir comportamientos no deseables; como por ejemplo, aumentar la ganancia integral y pro-

porcional hasta llevar el sistema a la estabilidad marginal o la inestabilidad. El objetivo es caracterizar experimentalmente tanto los comportamientos deseables, como aquellos que deben evitarse.

Algunos numerales extractados de la guía de acompañamiento de este experimento son:

- "Ajustar un control proporcional para una referencia de 180°. Aumentar k_p hasta llevar el error de posición al 5 %."
- "Ajustar un controlador PI para una referencia de 720°, de manera que la respuesta en lazo cerrado sea suave (sin oscilaciones)."
- "Ajustar un controlador PID para una referencia de −180, de manera que la respuesta en lazo cerrado sea rápida y con muy poca oscilación."
- "Verificar y explicar el comportamiento si la realimentación fuera de signo contrario, es decir cuando $e = r + y$"

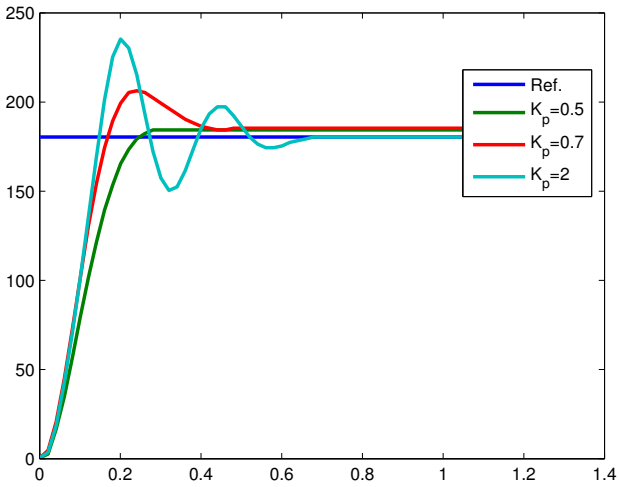

Figura 4: Respuesta del servomecanismo para diferentes valores de k_p

3.2. Control de Velocidad

Ahora se busca controlar la velocidad angular del servomotor. Esto podría hacerse con el mismo ensamble mecánico anterior; sin embargo, para visualizar los efectos de los controladores en una aplicación algo más entretenida, se construye el pequeño robot mostrado en la Figura 7. Nuevamente la elaboración del robot agrega una componente lúdica a la realización del experimento.

El robot incluye un sensor de ultrasonido para detectar obstaculos. Las variables de salida son la velocidad lineal estimada del robot, y la posición medida con respecto a una pared fija que se detecta con el sensor de ultrasonido.

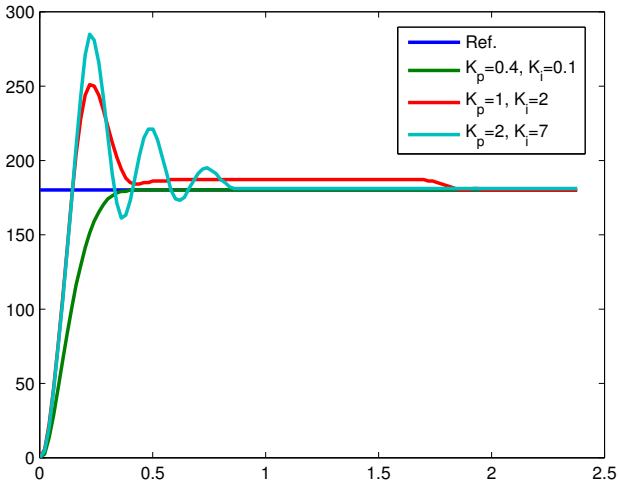

Figura 5: Respuesta del sistema con control PI

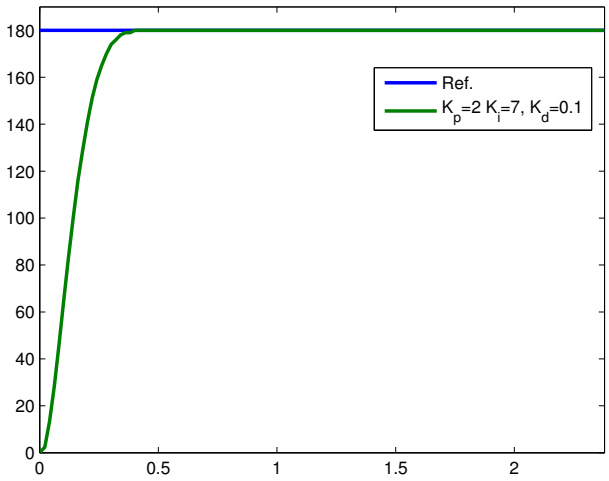

Figura 6: Respuesta del sistema con control PID

La velocidad se estima a partir de las medidas de posición angular:

$$\omega^k = \frac{\theta^k - \theta^{k-1}}{\Delta t} \tag{5}$$

Al hacer la aproximación de la ecuación 5, se producen errores numéricos típicos en la estimación numérica de derivadas. Por esta razón se espera una ligera oscilación en la respuesta de estado estable. Los autores consideran de gran utilidad la baja calidad de la medición de la velocidad por aproximación de la derivada para ilustrar uno de los principios fundamentales de la teoría de la realimentación, conocido desde Black:

> *"El desempeño de un sistema de lazo cerrado esta condicionado y limitado por la calidad del sensor".*

Figura 7: Pequeño robot para control de Velocidad

La Figura 8 muestra la respuesta del sistema con un controlador PI para velocidad. La oscilación sobre el estado estable es una consecuencia de la calidad de la estimación de la velocidad a partir de la derivada de la ecuación 5.

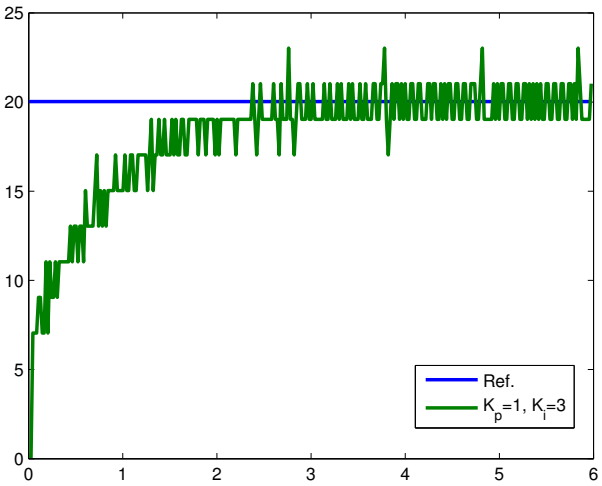

Figura 8: Vehículo con control de Velocidad (PI)

4. EXPERIMENTOS MÁS AVANZADOS

Para desarrollar controladores más avanzados que un PID, es necesario obtener un buen modelo de la planta. El método ARX de identificación de sistemas puede ser usado. Los fundamentos para usar esta herramienta son los conceptos de control digital y la aproximación de minimos cuadrados de un sistema de ecuaciones lineales sobredeterminado. Con estos conceptos se puede abordar la identificación de sistemas para obtener un modelo discreto de buena calidad $G(z)$.

La entrada $u(t)$ de la planta es el porcentaje de voltaje aplicado ($u(t) \in [0, 100]$) y la salida $y(t)$ es la velocidad del robot, estimada a través de la posición detectada por el sensor de ultrasonido. Es parte de este trabajo la programación de rutinas que entregan una señal binaria pseudoaleatoria o una onda "chirp" al robot, detectan la salida y envian los datos a un computador via USB o bluetooth, para su posterior procesamiento.

4.1. Identificación de Sistemas

En este experimento se usa una señal binaria seudo-aleatoria-RBS (Ljung (1987)), que varia entre los niveles máximo y mínimo de PWM del motor NXT y con una frecuencia inferior a unos 5 Hz. La señal de identificación se muestra en la figura 9. El tiempo para ejecutar este experimento es cerca de un minuto. La señal se almacena en un archivo de texto en el Brick para ser enviada al servomotor, y al mismo tiempo el programa descargado crea otro archivo de texto con las lecturas de posición del robot. En este caso, el uso del Brick para obtener los datos de entrada-salida del servomotor, elimina la necesidad de usar tarjetas de adquisición de datos dedicada. Los archivos de texto con los datos de entrada-salida de la planta se transfieren a un computador por USB o BLUETOOTH.

Los datos se capturan y envian al PC en un archivo de texto CSV, que puede ser facilmente leido en MATLAB u OCTAVE para usar las funciones especilizadas de identificación de sistemas.

En el laboratorio se hace uso del toolbox de identificación de sistemas de Matlab (ident) o del System Identification Toolbox de Octave. Este software permite obtener diferentes tipos de modelos a partir de los datos de entrada y salida del sistema.

Elegimos un modelo sencillo y que se ajuste a la forma de la función de transferencia típica de un sistema mecánico sin componente elástica. La función de transferencia en posición es

$$G(s) = \frac{0{,}18536}{s(1 + 0{,}062695\, s)} \tag{6}$$

4.2. Implementación del Controlador

Se diseña un controlador basado en el modelo de la ecuación 6 para cumplir características de sobrepico y tiempo de establecimiento. Las especificaciones de diseño toleran un sobrepico y tiempo de establecimientos grandes (para poder observarlos en el vehículo). El controlador obtenido es

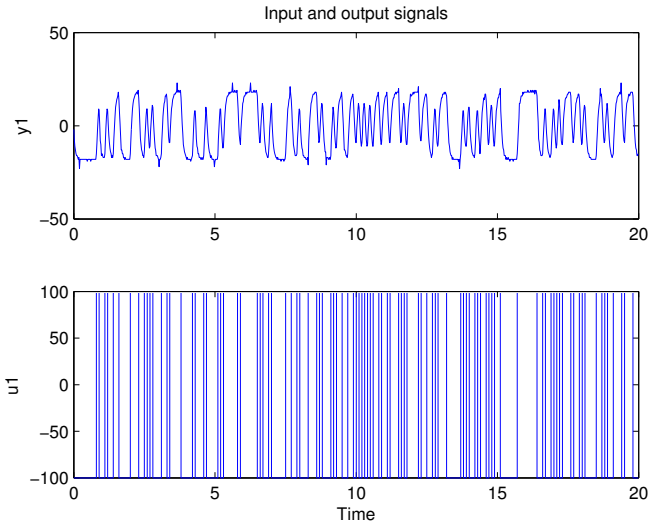

Figura 9: Parte inferior: onda RBS usada para identificación. Parte superior salida $y(t)$ del sistema

$$C(s) = \frac{1{,}1224(s + 0{,}4294)(s + 7{,}117)}{s(s + 2{,}591)}$$

Seguidamente se discretiza el controlador y se lleva a su representación en variables de estado para convertirlo en un algoritmo en C. El sistema discreto obtenido es

$$C(z) = \frac{1{,}1224(z - 0{,}9914)(z - 0{,}8609)}{(z - 1)(z - 0{,}9495)}$$

La representación en variables de estado discreta para el controlador $C(z)$ está dada por

$$\begin{aligned} x_1[k+1] &= x_1[k] + 1{,}4\, e[k] \\ x_2[k+1] &= 0{,}9495\, x_2[k] - 0{,}2874\, e[k] \\ u[k+1] &= 0{,}001891\, x_1[k] - 0{,}2874\, x_2[k] + 1{,}122\, e[k] \end{aligned}$$

Para evitar errores de cuantificación de coeficientes es mejor que la realización digital del controlador sea hecha en punto flotante. Para ello se usa el sistema operativo de tiempo real NXT-OSEK que incluye tipos de datos float de 32 bits. El código es muy sencillo y es equivalente para cualquier controlador implementado con esta herramienta (Algoritmo 5). El estudiante puede modificar facilmente las lineas que definen el controlador y el tiempo de muestreo para probar muchos otros controladores. Esto toma apenas unos minutos, no implica un conocimiento profundo de programación y deja claros los problemas y las herramientas necesarias que se deben tener en cuenta al enfrentar un problema de control.

En la figura 10 se aprecia la comparación entre la respuesta del sistema, comparada con la repuesta simulada. El sistema responde razonablemente de acuerdo con lo proyectado. Es destacable la rápidez y facilidad de implementación digital. Por otra parte, se comprueba experimentalmente la validez del modelo de $G(s)$ en la expresión 6, obtenido en

Algorithm 5: Control PID en NXT-OSEK.

```
#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"
#include "math.h"


DeclareCounter(SysTimerCnt);
DeclareTask(Task1);

void user_1ms_isr_type2(void)
{
  StatusType ercd;
  ercd = SignalCounter(SysTimerCnt);
  if (ercd != E_OK)
  {
    ShutdownOS(ercd);
  }
}

int y=0;
int ref=180;
double e,x1,x2,x1ant,x2ant,u;

TASK(Task1)
{
\\ Controller code
  y=nxt_motor_get_count(NXT_PORT_A);
  e=(double)ref-(double)y;
  x1=x1ant + 1.4*e;
  x2=0.9495*x2ant - 0.2874*e;
  u=0.0019*x1ant - 0.2874*x2ant + 1.122*e;
  nxt_motor_set_speed(NXT_PORT_A,(int)u,0);
  x1ant = x1;
  x2ant = x2;
  TerminateTask();
}
```

el procedimiento de identificación. Según el punto de vista de los autores de este artículo, es vital para los estudiantes que inician su camino en la teoría de control, el hecho de obtener concordancia entre teoría-práctica y simulación-realidad. Esto crea confianza en los métodos de la teoría de control automático.

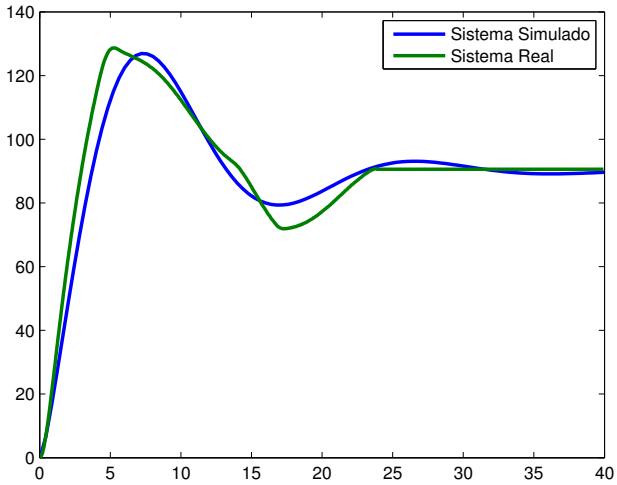

Figura 10: Respuesta con Controlador PID

En el material didactico que acompaña estos experimentos se pide diseñar redes de adelanto, de atraso, controladores algebraicos y de realimentación de estado. El ciclo completo de identificación, diseño y realización de controladores se puede cumplir eficientemente usando el Brick de LEGO® con con las herramientas NXC y NXT-OSEK y los programas para control e identificación desarrollados en esta propuesta.

Finalmente, se muestran 2 experimentos que pueden ser usados para cursos de control avanzados y cursos de posgrado. Estos son implementados en NXT-OSEK debido a su capacidad de trabajar en punto flotante. Estos ejercicios buscan el desarrollo de las habilidades en el análisis y modelado de sistemas, estabilización de sistemas con polos en el semiplano derecho, diseño e implementación de controladores digitales y filtrado digital.

4.3. Ball & Beam

La figura 11 muestra una planta avanzada: la bola y la viga, donde se busca controlar la posición de la bola a lo largo de la viga. En este ejercicio se desarrolla una rutina de identificación de la planta y diferentes tipos de controladores. La implementación del controlador en NXT-OSEK se muestra en el algoritmo 6.

4.4. Segway

Otra planta interesante es el segway mostrado en la figura 12. La implementación se muestra en el algoritmo 7. El diseño mecánico de esta planta y software mucho más avanzado que el presentado en este artículo, ha sido hecho por Takashi Chikamasa [1].

[1] http://lejos-osek.sourceforge.net/

Algorithm 6: Control del Ball & Beam en NXT-OSEK.

```
#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"
#include "math.h"

DeclareCounter(SysTimerCnt);
DeclareTask(Task1);

void user_1ms_isr_type2(void)
{
  StatusType ercd;
  ercd = SignalCounter(SysTimerCnt);
  if (ercd != E_OK)
  {
    ShutdownOS(ercd);
  }
}

int y=0;
int ref=25;
double e,x1,u;

TASK(Task1)
{
  y = ecrobot_get_sonar_sensor(NXT_PORT_S1);
  e=r-y;
  x1=0.8007*x1+1.561*e
  u=-1.561*x1+14.52*e
  nxt_motor_set_speed(NXT_PORT_A,(int)u,0);
  TerminateTask();
}
```

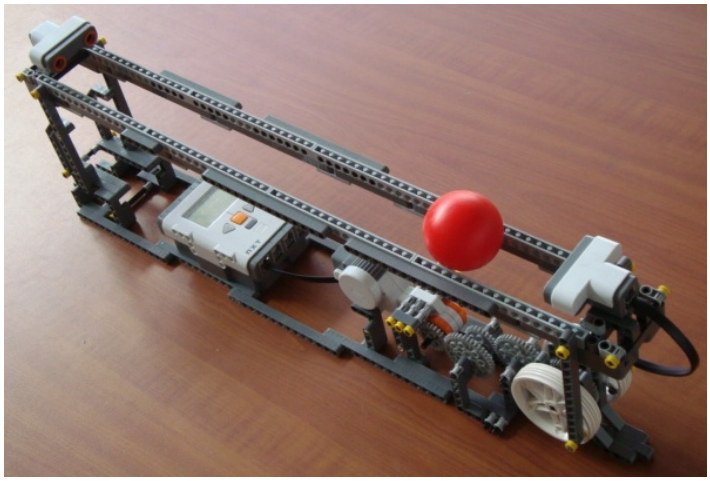

Figura 11: Ball & Beam con LEGO ®

En la página del autor se incluyen diseños realmente impresionantes como una bicicleta autoestable.

Figura 12: Segway con LEGO ®

Algorithm 7: Control del Ball & Beam en NXT-OSEK.

```
#include "kernel.h"
#include "kernel_id.h"
#include "ecrobot_interface.h"
#include "math.h"

DeclareCounter(SysTimerCnt);
DeclareTask(Task1);

void user_1ms_isr_type2(void)
{
  StatusType ercd;
  ercd = SignalCounter(SysTimerCnt);
  if (ercd != E_OK)
  {
    ShutdownOS(ercd);
  }
}

int y=0;
int ref=25;
double e,x1,u,linear_pos,ang_velocity,angle,scale;

TASK(Task1)
{
  scale=0.85
  y1 = ecrobot_get_sonar_sensor(NXT_PORT_S1);
  y2= ecrobot_get_gyro_sensor(U8 port_id);
  linear_pos = nxt_motor_get_count(NXT_PORT_A);
  linear_vel = estimate_vel(linear_pos);
  ang_velocity = passbandfilter(y2);
  angle=estimate_angle(ang_velocity);
  u=(0.8344*linear_pos -38.1749*linear_vel -3.55*angle -1.09*ang_velocity)*s
  if(u>100){u=100;}
  if(u<-100){u=-100;}
  nxt_motor_set_speed(NXT_PORT_A,(int)u,0);
  nxt_motor_set_speed(NXT_PORT_B,(int)u,0);
  TerminateTask();
}
```

5. CONCLUSIONES Y TRABAJO FUTURO

Se reporta satisfactoriamente, una alternativa de bajo costo para la experimentación en problemas de control automático. Mediante el uso de esta herramienta, un estudiante de un curso de experimental de control puede hacer todos los pasos de un ciclo de diseño: *construcción de la planta, modelamiento e identificación, diseño e implementación de controladores y pruebas de hardware.*

Se ha diseñado una serie de plantas para trabajar en un curso experimental de control, cuyos planos y guias de construcción están disponibles (consultar a los autores). Las plantas diseñadas pueden ser realizadas con una caja estándar del set educativo de LEGO ®Mindstorms. El uso de LEGO ®Mindstorms ha mostrado ser una alternativa viable por costo y calidad para hacer cursos de control experimental.

Las prácticas mencionadas son realizadas actualmente en los cursos de control para pregrado de la Universidad Nacional de Colombia. La constante realimentación de información y un continuo esfuerzo en el mejoramiento de estas prácticas han generado un incremento significativo en el interés por parte de los estudiantes hacia la teoría de control y una mejor comprensión de los conceptos claves en el estudio de sistemas dinámicos.

La propuesta aqui presentada es un proyecto permanentemente activo cuyos objetivos son aumento en la cobertura, calidad y número de experimentos por semestre realizados en el curso de laboratorio de control.

REFERENCIAS

Bermeo L. y Díaz H. Sideco: una propuesta para la enseñanza de control experimental. *Memorias del VII congreso de la Asociación Colombiana de Automática*, 2007.

Bernstein D.S. Innovations in undergraduate control education. *Control Magazine*, 25/1, 2005a.

Bernstein D.S. The quanser d.c. motor control trainer. *Control Magazine*, 25/1:90–93, 2005b.

Franklin G., Powell J., y Workman M. *Digital Control of Dynamic Systems*. Adisson Wesley, Sand Hill Road, U.S., third edición, 1997.

Gawthrop P. y E.McGookin. A lego-based control experiment. *Control Magazine*, 24/5:43–56, 2004.

K.J. Åström J.A. A laptop servo for control education. *Control Magazine*, 24/5:70–73, 2004.

LEGO. Lego® mindstorms® education base set. Website, 2010. http://www1.lego.com/education/search/default.asp?l2id=0_1&page=7_1&productid=9797.

Ljung L. *System Identification Theory For User*. Prentice Hall, 1987.

Lurie B. y Enright P. *Classical Feedback Control with Matlab*. Marcel Dekker, New York, 2000.

Åström K. y Hagglund T. *Advanced Pid Control*. ISA - Instrumentation, Systems ans Automation Society, Research Triangle Park,NC 27709, U.S., first edición, 2006.

Åström K. y Wittenmark B. *Computer Controlled Systems*. Prentice Hall, Upper Saddle River, U.S., third edición, 1997.