Diseño mecatrónico de una Shield de arduino para el control de motores DC con escobillas

Mechatronics design of an arduino Shield for controlling brushed DC motors

Eugenio Yime Rodríguez[1], Jheifer Páez Almentero[2]

[1]Ph.D. Automatización y Robótica. Profesor Tiempo Completo del Programa de Ingeniería Mecánica y Mecatrónica de la Universidad Tecnológica de Bolívar.
[2] Ingeniero Mecatrónico. Technical Manager. Polmarine S.A.

Email: eyime@unitecnologica.edu.co

Recibido 22/06/2015
Aceptado 05/11/2015

Cite this article as: E. Yime, J. Páez, "Mechatronics design of an arduino Shield for controlling brushed DC motors", Prospect, Vol 14, N° 1, 73-79, 2016.

RESUMEN

Los motores DC con escobillas son motores de bajo costo comúnmente empleado en diversas aplicaciones donde se requiere de control de movimiento, destacándose entre ellas la Robótica por los requisitos de servo control en posición, velocidad y corriente. Si bien en el mercado existen diversas soluciones para controlar motores DC con escobillas, la propuesta desarrollada está dirigida específicamente al campo académico de la docencia e investigación en Robótica, donde los requisitos usuales son una tarjeta de bajo costo, sencilla de usar, y que permita el control de posición y corriente. La solución planteada es acorde con esos requisitos. El diseño se basó en Arduino por sus características de solución abierta y de bajo costo. Los controles desarrollados son PID en posición y corriente. El control de posición permite desarrollar aplicaciones Robóticas basadas en la cinemática inversa de robots, mientras que el control de corriente permite realizar prácticas de leyes de control basados en la dinámica de los robots. En el artículo se describe el proceso de diseño y construcción del primer prototipo del controlador, así como el desarrollo del software, tanto para Arduino como para MATLAB, el cual se emplea para obtener los resultados experimentales de los controladores PID.

Palabras claves: Controlador PID; Control de movimiento; Robótica.

ABSTRACT

Brushed DC motors are inexpensive motors commonly used in diverse applications where it is required motion control. Robotics stand out among them due to its requirements in servo control of position, speed and current. Although there are several solutions for controlling brushed DC motors on the market, the proposal is specific to the academic field of teaching and research in robotics, where the usual requirements are an inexpensive, and simple to use, card which allows position and current control. The proposed solution is in line with these requirements. The design was based on Arduino for its characteristics of open and inexpensive solution. The developed PID controller are for position and current. Position control allows developing robotic applications based on inverse kinematics, while the current controller allows control practices based on the robot dynamics. In the article the process of design and assembly of the first prototype is covered. It also cover the development of software for both Arduino and MATLAB, which is used to plot and analyze the experimental results of the PID controllers.

Key words: PID controller; Motion control; Robotics.

Doi: http://dx.doi.org/10.15665/rp.v14i1.370

1. INTRODUCCIÓN

El control de motores DC de escobillas es un área madura de la Ingeniería donde existen diversos actores comerciales que ofrecen sus productos a los consumidores. Ejemplo de algunos de estos productos son los ofrecidos por los proveedores: Maxon motors, [1], Faulhaber motors, [2], Baldor motors, [3] , Ingenia Motors Control, [4], Yaskawa Motors, [5], Roboteq motors, [6], Control Techniques, [7], entre otros. Si bien todos estos productos son fiables y fáciles de emplear en una aplicación Robótica, en nuestro medio se transforman en una enorme desventaja cuando se quiere fomentar el uso de aplicaciones robóticas entre estudiantes de pregrado. La razón es simple: costos. De los proveedores mencionados, el producto más económico está en el orden de USD$125 dólares por controlador - amplificador [8], sin incluir el costo del motor DC. Si se piensa en una aplicación de al menos dos grados de libertad, se está hablando de cerca de dos millones de pesos sólo en amplificadores (incluyendo gastos de envío, impuestos y comisión al revendedor) pero sin incluir costos de los motores que pueden oscilar en el mismo valor. Estos costos suelen ser prohibitivos en nuestro medio académico, donde usualmente los estudiantes de pregrado tienen que costear de su propio bolsillo los proyectos. Es por ello que se plantea la propuesta de desarrollar una tarjeta de control de motores DC que se ajuste a los requisitos académicos de un estudiante promedio de pregrado: bajo costo, facilidad de uso, abierta, que sea reprogramable y que los componentes se consigan en nuestro medio.

Otro punto a tener en cuenta en diseño de la tarjeta controlador de motores DC es que realmente permita realizar un trabajo académico con componente investigativo. En esta línea hay que resaltar que desde el punto de vista de investigación las publicaciones de robótica internacionales tienen componente de control no lineal basado en el modelo dinámico del robot [9], [10], [11], [12]. Para poder incentivar trabajos estudiantiles con objetivos similares es necesario que el controlador pueda realizar un control en lazo cerrado de la corriente que circula por el motor. Sin ello no es posible realizar control por par calculado, el más básico de las leyes de control no lineal para robots [13], [14], [15].

Por último, si bien existen soluciones de bajo costo para el control de motores DC como son: Pololu, [16], Arduino Motor Shield, [17]. Estas soluciones tienen inconvenientes para el problema planteado. En el caso de Pololu, si bien es un PID en lazo cerrado, el feedback es análogo, lo cual reduce la precisión deseada y el rango de movimiento permitido para el lazo de posición. En el caso del Arduino Motor Shield, es un amplificador en lazo abierto que debe cerrarse con Arduino, lo cual disminuye la capacidad de este último para controlar de manera simultánea varios motores DC, ya que debe dedicar tiempo de procesamiento a la decodificación de los encoders de cuadratura y además Arduino no tiene suficientes canales para controlar simultáneamente seis motores DC, máximo grados de libertad de un cuerpo en el espacio.

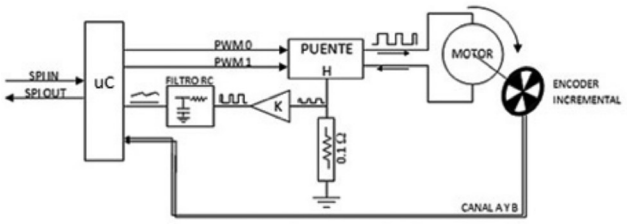
Lo anterior conlleva a definir claramente las especificaciones técnicas deseadas de la tarjeta Shield que la haga diferente a las existentes y por lo tanto justifique su diseño y construcción. Estas características son: capacidad para controlar de manera autónoma un motor DC haciendo lazo cerrado con un PID en posición o corriente, comunicación rápida entre la tarjeta y arduino, protocolo de comunicación sencillo para facilitar su programación desde arduino, capacidad de ser apilable para poder controlar hasta seis motores DC con una sola tarjeta arduino, capacidad de controlar hasta 2 amperios de corriente, capacidad para decodificar encoders de cuadratura con los tres canales comunes, canal A, canal B e índice. Estas especificaciones dan inicio al proceso de diseño que se tratará en la siguiente sección.

2. METODOLOGÍA

2.1 Diseño mecatrónico de la Shield

El diseño conceptual de la tarjeta está basado en el prototipo desarrollado en [18]. Es decir, se tiene un microcontrolador PIC18F2431 el cual recibe un feedback de posición y corriente. El feedback de posición se realiza a través del módulo interno del PIC el cual decodifica por hardware los pulsos del encoder y los multiplica por 4. El feedback de corriente se hace a través de una lectura ADC del PIC donde la señal proviene de una resistencia de 0.1 Ohmios a la salida de un puente H. El PIC manda señales PWM al puente H para mover el motor, y está en comunicación con arduino por medio del bus SPI. La figura 1 ilustra el concepto del diseño.

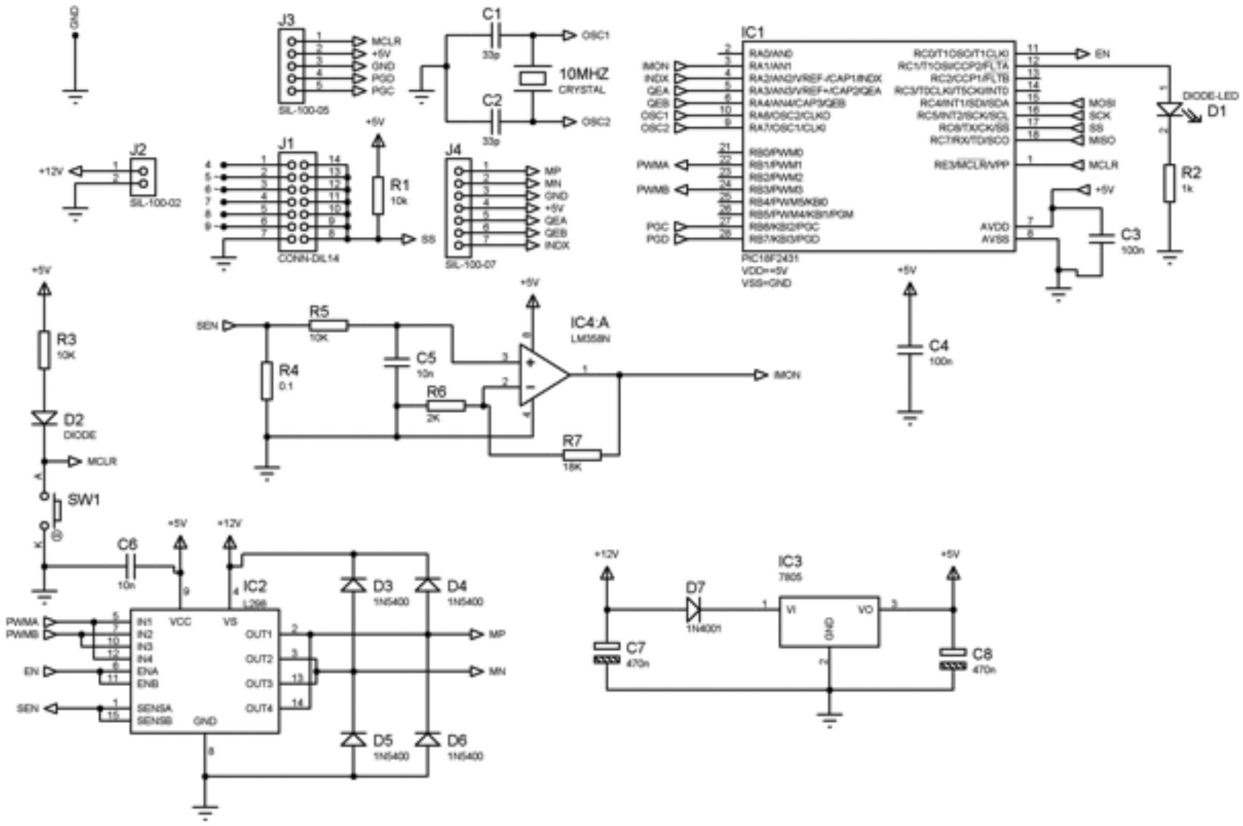
Figura 1. Diseño conceptual del controlador de motores DC.
Figure 1. Conceptual design of a brushed DC motor controller.



El la figura 2 se ilustra el diseño electrónico de la tarjeta basado en el circuito ilustrado en la figura 1. En este diseño electrónico cabe destacar la presencia de cuatro circuitos integrados básicos: el PIC18F2431, IC1, el puente H L298N, IC2, el amplificador operacional LM358N, IC4, y el regulador de voltaje 7805, IC3. Se incluye un regulador de voltaje para efectos de hacer la tarjeta independiente

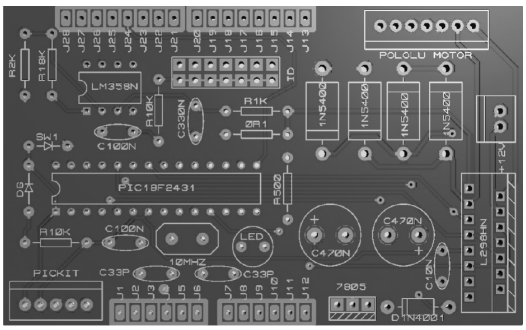
de manera que pueda usarse con arduino o con otro tipo de controladores, por ejemplo raspberry PI. El objetivo es que la tarjeta se alimente al nivel de voltaje de operación del motor DC y de allí se obtenga la alimentación para los integrados. El puente H, L298N, se selecciona por su disponibilidad en nuestro medio, lo cual facilita su adquisición y uso. Este puente H puede controlar hasta dos motores DC, o puede usarse en paralelo para controlar un solo motor que demande más corriente. Esta última opción fue la seleccionada. Se puso en la configuración en paralelo para poder alimentar hasta 4 amperios en continuo y 8 amperios en pico un motor DC. Sin embargo, para poder lograr ese nivel de corriente se debe cumplir ciertos requisitos de disipación de calor del integrado, en caso contrario este se recalienta y por seguridad la corriente disminuye. Los diodos D3 a D6 se incluyen para compensar el efecto de la inductancia de la bobina de los motores, tal como se especifica en la hoja de datos del L298N. El circuito IC4 tiene la función de amplificar la señal de voltaje que se genera cuando la corriente del motor pasa por la resistencia R4. La ganancia está ajustada a 10, de manera que es posible detectar hasta 5 amperios de corriente en el motor. Entre la resistencia y el circuito amplificador se coloca un filtro RC para convertir dicha salida en un voltaje constante. El integrado principal es el IC1, PIC18F2431, el cual se seleccionó por tener un decodificador hardware para los encoders incrementales y además poseer salidas PWM de control de motores DC. Los encoders incrementales se conectan en los terminales J4, el cual está especificado para motores Pololu por su bajo costo y facilidad de adquisición por los estudiantes. Sin embargo, esto no es impedimento para conectar cualquier otro tipo de motor DC. El PIC18F2431 opera a 40Mhz, para ello requiere un cristal de 10Mhz e internamente está configurado para utilizar el PLL que multiplica por cuatro la frecuencia del cristal. El circuito del PIC permite la programación en circuito del mismo, para ello se coloca el conector J3, donde se acoplaría el programador de PIC, PICKIT-2/3. El conector J2 es para alimentar toda la tarjeta. Por último, el conector J1 es para configurar el selector de chip del bus SPI. El objetivo es que las tarjetas puedan ser apilables hasta llegar a tener seis. Con el conector J1 se logra ese objetivo, así cada tarjeta seleccionará un PIN de arduino como el selector de tarjetas (chip select) a través del uso de "jumpers". Los posibles pines de arduino son el 4, 5, 6, 7, 8 y 9. Cada tarjeta selecciona un solo pin y este pin se conecta al pin 17 del PIC, es decir el chip select. La comunicación SPI utiliza los pines por defecto de arduino, PIN 11 como MOSI (Master Output, Slave Input), PIN 12 como MISO (Master Input, Slave Output), PIN 13 como reloj.

Figura 2. Diseño electrónico de la Tarjeta Shield.
Figure 2. Electronic design of the Arduino Shield.

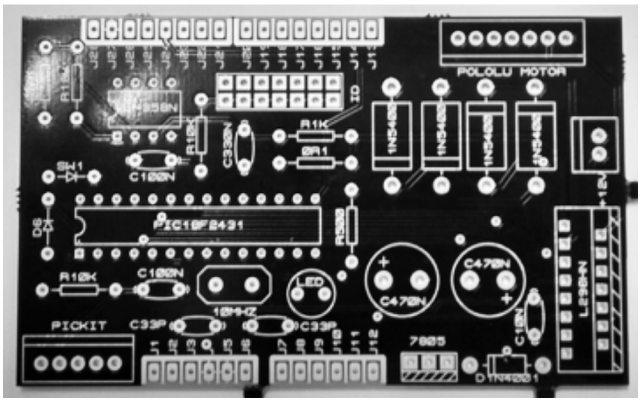


Una vez realizado el circuito electrónico, este se simuló en Proteus para validar el funcionamiento y la programación del PIC. Posteriormente se diseñó la PCB utilizando el mismo software y se generaron los archivos GERBER para mandarla a fabricar en la tienda en línea OSHPARK, [19]. La figura 3 ilustra el diseño electrónico de la PCB y la fotografía de la misma una vez recibida.

Figura 3. Diseño y construcción de la PCB.
Figure 3. PCB Pictures after designed and built.



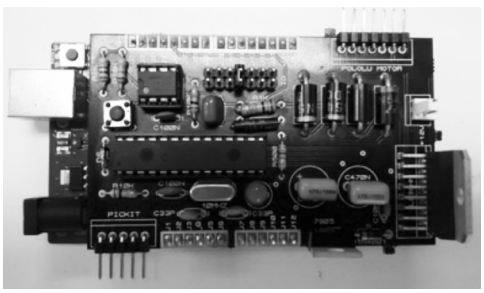
(a) PCB Diseñada en Proteus / PCB designed with Proteus



(b) PCB construida / PCB built.

La figura 4 es una fotografía de la shield terminada con los componentes soldados y montada sobre un arduino. En la siguiente sección se discute el software desarrollado para el PIC, arduino y MATLAB.

Figura 4. Tarjeta finalizada y montada sobre Arduino UNO.
Figure 4. Finished card upon an Ardunio UNO.



2.2 Firmware de la Shield

La tarjeta diseñada al tener un microcontrolador PIC18F2431 que realiza el control PID de corriente o posición, debe programarse para que el software embebido, también conocido como firmware, se ejecute en el PIC. En esta sección se discute los aspectos más relevantes de dicho firmware.

Figura 5. Diagrama de flujo del firmware de la tarjeta shield.
Figure 5. Arduino Shield firmware f l lowchart.

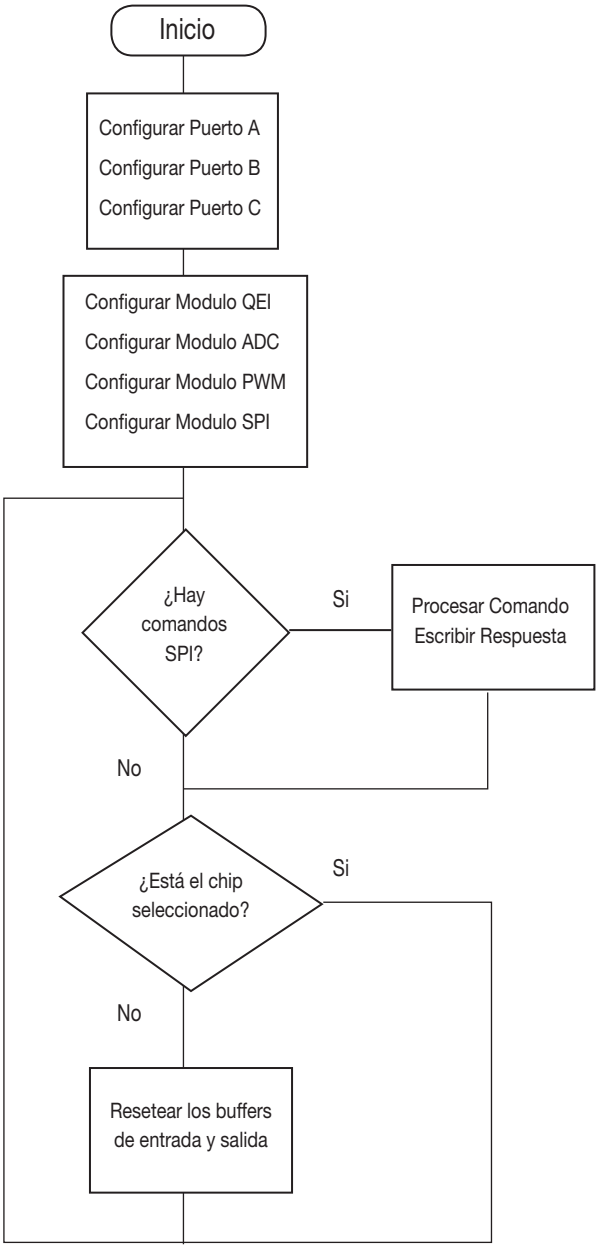


Figura 6. Interrupción de alta prioridad.
Figure 6. High priority interruption.

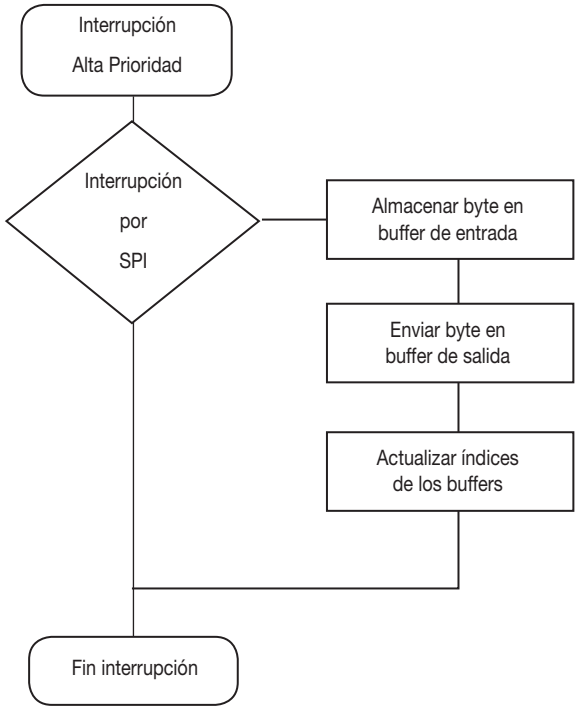
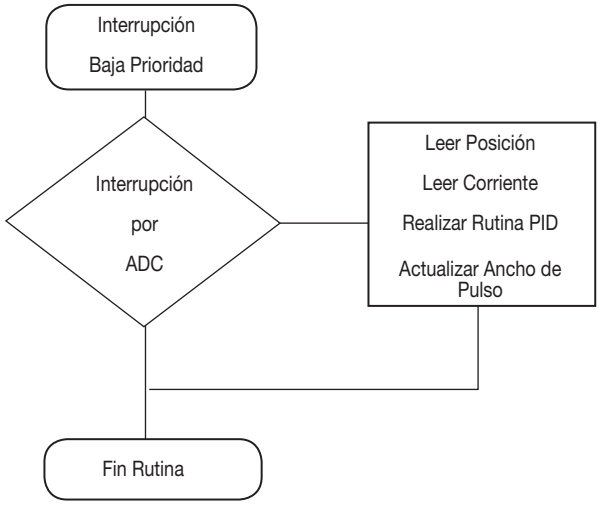


Figura 7. Interrupción de baja prioridad.
Figure 7. Low priority interruption.



Las figuras 5, 6 y 7, ilustran los diagramas de flujo del firmware desarrollado. En la figura 6 se muestra el proceso realizado por la función principal, main. En este caso, el software empieza configurando los módulos QEI, ADC, PWM y SPI del PIC18F2431, posteriormente configura las interrupciones de alta y baja prioridad. El módulo QEI se configura para que trabaje a 4X y genere interrupciones de baja prioridad en los desbordamientos de buffers. Estos desbordamientos se pueden emplear para aumentar el tamaño de la variable de registro de posición del encoder, con lo cual se podría aumentar hasta 32 bits si se desea. En el software inicial se utiliza la opción por defecto de 16 bits para posición. El módulo ADC se configura para que haga una conversión automática cuando se genere un pulso PMW y el módulo PWM se configura para que trabaje a 10Khz y dispare la conversión ADC con cada nuevo pulso PWM. El módulo SPI se configura como esclavo y la interrupción se configura como de alta prioridad para no perder comunicación con el maestro. Una vez configurado los módulos de entrada / salida, se procede a entrar a un ciclo infinito donde se valida si existen comandos del maestro y en caso positivo se procede a escribir la respuesta.

La figura 6 ilustra el algoritmo para la comunicación SPI, la cual se configura como interrupción de alta prioridad para no perder información proveniente del maestro. La interrupción almacena el byte entrante en una variable buffer de entrada y si existe información en un buffer de salida procede a actualizar el byte que va a responder al maestro. De esta forma se garantiza que lo que ingresa será procesado posteriormente en la función principal main y lo que se haya escrito para enviarse al maestro se enviará a través del buffer de salida.

La figura 7 muestra el algoritmo para las interrupciones de baja prioridad, en este caso la interrupción por ADC. Cuando se genera una interrupción por ADC, se procede a verificar si el PID está en ejecución, si es así, se llama a la rutina del PID, la cual calcula el ancho del puso PWM acorde con el error y las constantes Kp, Ki y Kd. La salida del PID oscila entre los valores -4096 y 4096, donde 4096 corresponde a un pulso del 100% a 5 voltios, y el valor negativo significa un cambio en el sentido de giro del motor. Un aspecto importante del PID es que hace sus cálculos usando precisión de punto flotante, mientras que la comunicación está basada en número enteros, lo que hace que los cambios de las ganancias sean multiplicados por un factor de $1e^{-7}$, ya que el periodo del PID es del orden de microsegundos.

El PIC responde a 16 posibles comandos que puede enviar le maestro: entrar a modo corriente, entrar a modo posición, activar el PID, desactivar el PID, ajustar el setpoint, ajustar la ganancia Kp, ajustar la ganancia Ki, ajustar la ganancia Kd, obtener posición actual, obtener corriente actual, definir la posición del encoder, probar PWM con giro derecho, probar PWM con giro izquierdo, calcular la zona muerta del conjunto Motor – Puente H, ajustar la zona muerta hacia la derecha, ajustar la zona muerta hacia la izquierda. De estos comandos el que tal vez requiera un poco más de explicación es el cálculo de la zona muerta. La razón de este comando es que debido a la fricción interna

de los motores DC, sobre todo si tienen reductores, y a la histéresis del puente H, existe un rango de pulsos PWM en donde los motores DC no se mueven a pesar de que se les está enviando una señal. El comando "calcular zona muerta" calcula el rango de los PWM que existe en dicha zona muerta y la respuesta se utiliza para ajustar la salida del PID de manera que se salte la no linealidad debido a la zona muerta. Si no se hace eso, la sintonización del PID se transforma en una tarea bastante complicada. Un factor importante a tener en cuenta, es que en algunos motores la zona muerta es variable, es decir depende de la posición del motor y si el mismo está en reposo o en marcha. La tarea consiste en hacer varias pruebas sobre el motor y copiar los valores más pequeños de la zona muerta para minimizar el efecto de la nolinealidad en el desempeño del PID.

3. RESULTADOS Y DISCUSIÓN

Los resultados experimentales se realizaron con MATLAB, para ello se intentó utilizar el código desarrollado por MATHWORKS [20], para comunicarse con arduino. Sin embargo, tocó crear un código nuevo desde cero por cuanto [20] no cobija la comunicación SPI. El código desarrollado se dividió en dos partes, un código en arduino, que no es más que un servidor que espera que lleguen los comandos por USB y los reenvía al PIC, espera la respuesta del mismo y la envía de vuelta a MATLAB. El código en MATLAB el cual no es más que una comunicación por RS-232 utilizando el puente USB a 232 de arduino. Los comandos desarrollados para tal fin fueron: OpenCommunication, para abrir la comunicación con arduino, CloseCommunication, para cerrar el puerto y finalizar la comunicación y SendCommand, para dialogar con el PIC.

Una vez se pudo comunicarse con el PIC, se procedió a sintonizar el PID, para ello se calculó inicialmente la zona muerta del motor. El motor utilizado, un motor pololu con reductor 70:1 [21], la zona muerta fue del 4.9% del ancho del pulso para ambos sentidos del motor. Es decir que si el pulso es igual o menor a ese porcentaje el motor no se moverá debido a la fricción y la histéresis del puente H. Posteriormente se procedió a sintonizar el PID. La figura 8 muestra la salida para el lazo de posición cuando se requiere un escalón de 6400 pulsos de encoders. Las constantes empleadas durante la prueba fueron una ganancia Kp de 5000 y una ganancia Kd de 500. La figura 9 ilustra la salida para el lazo de corriente con un setpoint de 800 mA y ganancias Kp de 500, Ki de 50 y Kd de 100. En este caso se puede apreciar que la respuesta no es continua, la razón está relacionada con los retardos en la comunicación entre el PC y la shield. En ambas figuras, los datos se tomaron con un intervalo de 50 ms desde MATLAB. Sin embargo, la respuesta en corriente es más rápida que en posición, por ello se aprecia la discontinuidad.

Figura 8. Salida del PID para posición.
Figure 8. PID Position output.

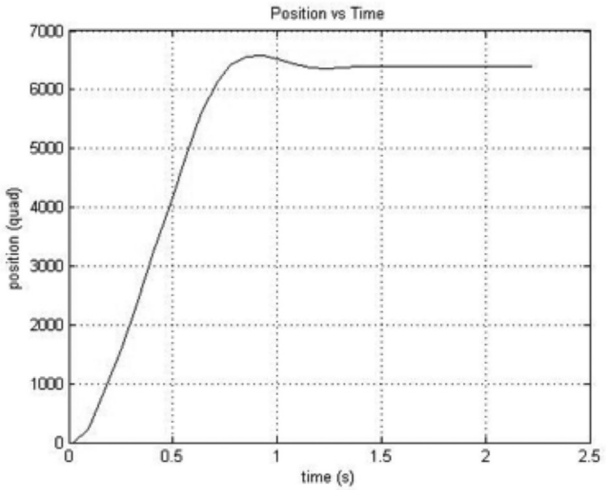
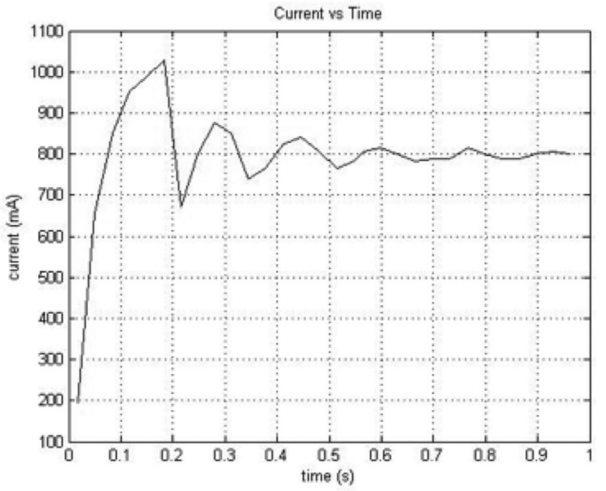


Figura 9. Salida del PID para corriente.
Figure 9. PID Curent output.



4. CONCLUSIONES

• A través de los resultados experimentales se puede demostrar que se diseñó, construyó, programó y validó una tarjeta shield de arduino para el control de motores DC de escobillas. Estas a su vez demuestran que el sistema es capaz de controlar de manera efectiva la posición y la corriente del motor DC. Una característica relevante de la shield es que su firmware puede ser actualizable por los estudiantes, lo que a su vez abre la posibilidad de adaptar el mismo a las necesidades de la aplicación donde se vaya a utilizar. Otra característica importante de la shield es que puede ser apilable, gracias a una serie de "jumpers" que permiten a Arduino seleccionar la tarjeta a comunicarse entre las varias apiladas. Esta funcionalidad posibilita el control de varios motores DC de manera simultánea, aspecto muy importante al momento de querer controlar un robot con varios grados de libertad.

• Si bien los resultados experimentales se resumen en dos gráficas de salidas de los PID, para que estas gráficas fueran posibles de obtener se tuvo que realizar un trabajo a bajo nivel. El principal aspecto que permitió la generación de las gráficas fue la comunicación entre MATLAB y arduino, y la comunicación entre este último y la Shield. El software que permitió esto fue creado desde cero por cuanto el código existente en MATHWORKS no permite el uso del bus SPI. Debido a esto, se creó un software en Arduino que facilita la comunicación entre MATLAB y las diferentes tarjetas, y en MATLAB se creó un software para poder controlar los motores ya sea en corriente o posición.

REFERENCIAS

[1] M. Motors, «Maxon Motors» [Internet]. Disponible desde <http://maxonmotorusa.com> [Acceso: Junio 2015].

[2] Faulhaber, «Faulhaber» [Internet]. Disponible desde <http://www.faulhaber.com> [Acceso: 1 Junio 2015].

[3] Baldor, «Baldor» [Internet]. Disponible desde <http://www.baldor.com> [Acceso: 1 Junio 2015].

[4] I. MC, «IngeniaMC» [Internet]. Disponible desde <http://www.ingeniamc.com> [Acceso: 1 Junio 2015].

[5] Yaskawa, «Yaskawa» [Internet]. Disponible desde <http://www.yaskawa.com> [Acceso: 1 Junio 2015].

[6] Roboteq, «Roboteq» [Internet]. Disponible desde <http://www.roboteq.com> [Acceso: 1 Junio 2015].

[7] C. Techniques, «Emerson,» [Internet]. Disponible desde <http://www.emersonindustrial.com>. [Acceso: 1 Junio 2015].

[8] SDC1130, «Roboteq» [Internet]. Disponible desde <http://www.roboteq.com/index.php/roboteq-products-and-services/brushed-dc-motor-controllers/sdc1130-detail>. [Acceso: 1 Junio 2015].

[9] B. Rim, N. Habib, B. Hala, M. Nacer K., A. Adel M., N. Aziz, "Dynamic Modeling and Control of a Multi-Fingered Robot Hand for Grasping Task", Procedia Engineering, 41, 923-931, 2012.

[10] L. Jaeoh, H. Seongik, L. Jangmyung, "Decoupled Dynamic Control for Pitch and Roll Axes of the Unicycle Robot", IEEE Transactions on Industrial Electronics, 60 (9), 3814 - 3822, 2013.

[11] H. Jin, J. Hwang, J. Lee, "A Balancing Control Strategy for a One-Wheel Pendulum Robot Based on Dynamic Model Decomposition: Simulations and Experiments", IEEE/ASME Transactions on Mechatronics, 16 (4), 763 - 768, 2011.

[12] S. Han, J. Lee, "Balancing and Velocity Control of a Unicycle Robot Based on the Dynamic Model", IEEE Transactions on Industrial Electronics, 62 (1), 405 - 413, 2015.

[13] F.-J. Lin, Y.-S. Lin, S.-L. Chiu, "Slider-crank mechanism control using adaptive computed torque technique", IEE Proceedings on Control Theory and Applications, 145 (3), 364 - 376, 1998.

[14] M. A. Llama, R. Kelly, V. Santibanez, "Stable computed-torque control of robot manipulators via fuzzy self-tuning", IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 30 (1), 143-150, 2000.

[15] A. Constaín, K. Torres, J. Arango, A. Vivas. Modelado, identificación paramétrica y control del robot Scorbot ER 5 plus. Presentado en el VIII Congreso de la Asociación Colombiana de Automática, Cartagena, Colombia, 2009.

[16] Jrk21V3, «Pololu» [Internet]. Disponible desde <https://www.pololu.com/product/1394>. [Acceso: 1 Junio 2015].

[17] A. M. Shield, «Motor Shield For Arduino» [Internet]. Disponible desde <http://www.arduino.cc/en/Main/ArduinoMotorShieldR3>. [Acceso: 1 Junio 2015].

[18] E. Yime, J. L. Villa, J. Paez. Design of a brushed DC motors PID controller for development of low-cost robotic applications. Presentado en el III International Congress of Engineering Mechatronics and Automation (CIIMA), Cartagena, Colombia, 2014.

[19] Oshpark, «OSHPARK» [Internet]. Disponible desde <https://oshpark.com/>. [Acceso: 1 Junio 2015].

[20] Mathworks, «MATLAB Support for Arduino» [Internet]. Disponible desde <http://www.mathworks.com/matlabcentral/fileexchange/32374-matlab-support-for-arduino--aka-arduinoio-package->. [Acceso: 1 Junio 2015].

[21] pololu, «Metal Gear Motor 70:1» [Internet]. Disponible desde <https://www.pololu.com/product/1445.> [Acceso: 1 Junio 2015].