CONTROL DE POSICIÓN PID PARA MOTOR DE CD CON RASPBERRY PI 3 SIN CARGA MECÁNICA

PID POSITION CONTROL FOR DC MOTOR WITH RASPBERRY PI 3 WITHOUT MECHANICAL LOAD

Marcos Gutiérrez López
Tecnológico Nacional de México / IT de Celaya, México
marcos_22_11_88@hotmail.com

Alejandro Israel Barranco-Gutiérrez
Cátedras CONACyT - Tecnológico Nacional de México / IT de Celaya, México
Israel.barranco@itcelaya.edu.mx

Recepción: 5/mayo/2020 **Aceptación:** 29/octubre/2020

Resumen

En la actualidad el diseño e implementación de controladores PID es primordial en la industria eléctrica y electrónica debido a la necesidad de ejecutar y supervisar procesos automáticamente. Este trabajo presenta el control PID de un motor CD de 24 V, que será el encargado de manipular el volante de un vehículo eléctrico que se encuentra en desarrollo. A través de un encoder absoluto se determinó la posición del rotor del motor eléctrico, y mediante un algoritmo en Raspberry se realizó la adquisición de la posición y la determinación de las acciones de control correspondientes. El sistema de control PID implementado en Raspberry Pi 3 funciona con una baja cantidad de error.

Palabras Clave: Control PID, Motor eléctrico CD, Raspberry Pi.

Abstract

Currently the design and implementation of PID controllers is primordial in the electrical and electronics industry due to the need to automatically run and monitor processes. This work presents the PID control of a 24 V DC motor, which will be in charge of manipulating the steering wheel of an electric vehicle that is under development. The position of the rotor of the electric motor was determined through of an absolute encoder, and by means of an algorithm in Raspberry the acquisition

of the position and the determination of the corresponding control actions were carried out. The PID control system implemented in Raspberry Pi 3 works with a low amount of error.

Keywords**:** *DC electric motor, PID control, Raspberry Pi.*

1. Introducción

En la actualidad el diseño e implementación de controladores PID es primordial en la industria eléctrica y electrónica debido a la necesidad de ejecutar y supervisar procesos automáticamente. Ejemplos de sistemas que utilizan este concepto son: refrigeradores, hornos de microondas, automóviles, aviones, y máquinas especializadas. Por otro lado, la minicomputadora de bajo costo Raspberry PI es una tecnología de código abierto y hardware libre que ha cobrado relevancia debido a las prestaciones que ofrece el ámbito de la programación de sistemas digitales [1], esta tarjeta se muestra en la figura 1. Recientemente, estos minicomputadores son ampliamente utilizados por su portabilidad, bajo costo y funcionalidad [2]. Sin embargo, existe un compromiso entre su facilidad de uso y la cantidad de operaciones que puede procesar por segundo [3-4].

Figura 1 Vista superior de la tarjeta Raspberry PI 3.

Gran parte de la teoría y tecnología de sistemas automáticos descansan en el control tipo PID. Este se caracteriza en utilizar la multiplicación por un escalar de la señal de error (Ecuación 1), la integral de la señal de error (Ecuación 2) que multiplica una constante y la derivada de la señal de error (Ecuación 3) por una

constante. Éstos son sumados para obtener una señal de excitación de la planta y aplicarla para operar al sistema (Ecuación 4) [5].

$$P(t) = K_p e(t) \tag{1}$$

$$I(t) = \int_{t=0}^{t} K_i e(t)\ dt \tag{2}$$

$$D(t) = K_d \frac{d}{dt} e(t) \tag{3}$$

$$PID(t) = P(t) + I(t) + D(t) \tag{4}$$

Respecto al control proporcional es conveniente indicar que es la acción de control lineal más importante, debido a que cambia la rapidez con la cual se llega a la referencia, lo que también se torna delicado debido a que una ganancia alta provoca que el sistema se torne inestable y dentro de la región estable se presenta la imposibilidad de corregir algunos errores en el régimen permanente. La acción integral (Ecuación 2) ayuda a vencer la inercia de un sistema en la etapa inicial del control y en la etapa de estado estable a reducir su error. Mientras la acción derivativa indica al control la tendencia del error y con esto se pueden reducir las oscilaciones menores en estado estable y en el arranque del proceso de control [5-8].

Este trabajo presenta el control PID de un motor CD de 24 V, que será el encargado de manipular el volante de un vehículo eléctrico que se encuentra en desarrollo dentro de las instalaciones del Instituto Tecnológico de Celaya, en el laboratorio de autotrónica del departamento de ingeniería electrónica. El objetivo es llevar al motor a una posición deseada por el usuario con un mínimo error y una acción rápida y controlada. El trabajo es del tipo experimental, donde las ganancias del PID fueron sintonizadas por prueba y error. Se obtuvo un sistema de control PID implementado en Raspberry Pi 3 que funciona con una baja cantidad de error.

2. Material y métodos

Respecto a los materiales, el motor que se utilizó es refacción de scooter eléctrico y tiene como características: Potencia eléctrica de consumo de 250 W, trabaja a 24 VCD con un máximo de 2750 rpm (revoluciones por minuto). Intensidad de corriente:

15.4 A. Longitud total: 101,6 mm, ancho total: 90 mm, altura total: 76 mm, diámetro del eje: 12 mm y longitud del eje OA: 1 pulgada. Por otra parte, el encoder absoluto utilizado es el AMT203-V que tiene una resolución de 0.2 grados y comunicación SPI (Figura 2). El puente H utilizado es de tipo MOSFET con entrada PWM, corriente pico de salida de 30 A y corriente máxima de seguridad de 20 A (Figura 3).

Figura 2 Motor de corriente directa conectado mecánicamente a un encoder absoluto.

Figura 3 Puente H dual MOSFET para control de motor de corriente directa de 3 a 36 V.

En la integración de componentes se tiene el objetivo de posicionar el rotor del motor en la posición deseada, también llamada posición de referencia [9-10]. El motor es alimentado por el circuito puente H, lo que provoca que el rotor se mueva y el encoder registre ese movimiento que envía a un microcontrolador por comunicación SPI, además que alimenta al encoder con 5 V. El microcontrolador envía la posición a la Raspberry por comunicación serial implementada en el puerto USB. En la minicomputadora se compara la posición del encoder con la posición deseada por medio del cálculo del error, el cual se utiliza para calcular la salida del controlador PID. Una vez calculada la salida del PID, se envía al puente H en forma de una señal PWM y una señal de dirección que se traduce como el signo de la salida del PID, así como se ilustra en la figura 4. El objetivo de utilizar Arduino nano para la comunicación SPI con el encoder, es debido a que se planea conectar multiples

tarjetas Arduino que estarán monitoreando y procesando parte de la información de otros sensores que serán implementados en el vehículo eléctrico. Con lo cual la Rasberry realizará la toma de decisiones basada en la información que los Arduinos le envíen, bajo un esquema de comunicación maestro-esclavo.

En la figura 5 se muestra un diagrama de flujo del algoritmo implementado en la Raspberry Pi 3.

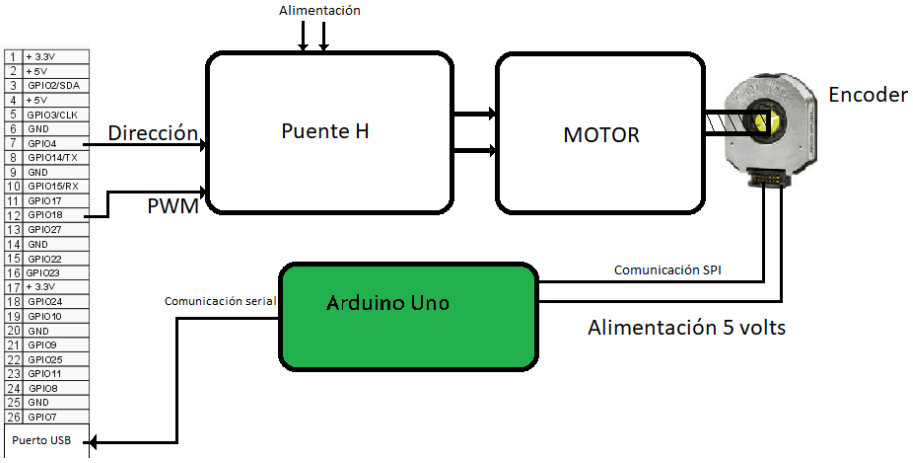

Figura 4 Esquema de flujo de información en los componentes del sistema de control.

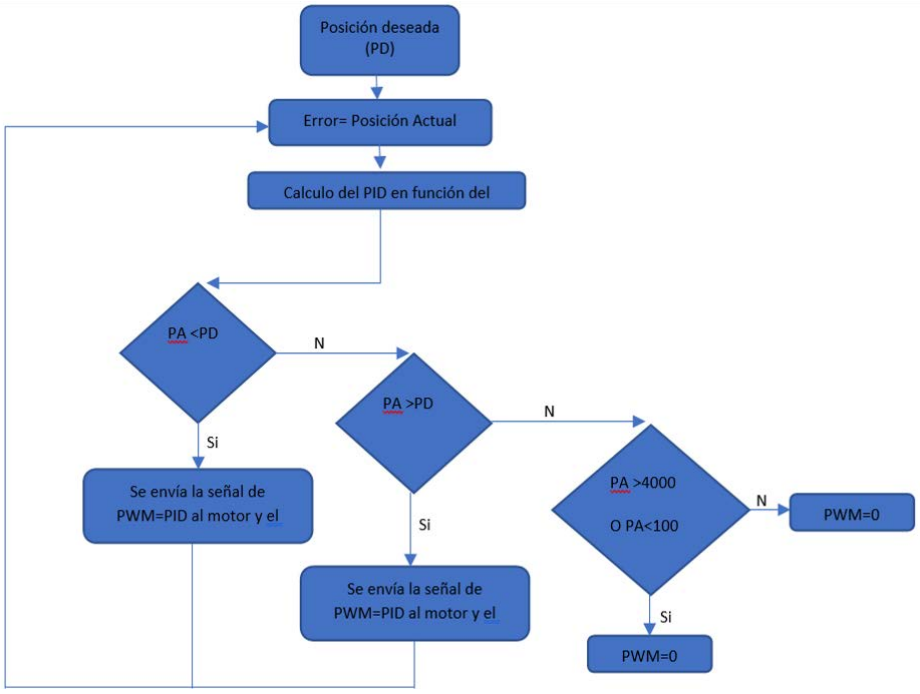

Figura 5 Diagrama de flujo del algoritmo programado en la Raspberry Pi 3.

El algoritmo espera a la entrada de la posición deseada, la cual es suministrada por el usuario. Después adquiere la medición de la posición actual del encoder para calcular el error, a partir del cual se calcula el PID. Posteriormente se entra a una serie de condiciones para tomar la decisión del sentido de giro del motor, dependiendo los valores de la posiciona actual y la posición deseada. La última condición tiene una doble función, ya que el objetivo del control de la posición del motor tendrá una aplicación en el volante de un vehículo eléctrico, se debe evitar que la posición del motor sobrepase uno de los límites establecidos (>4000 o <100) que podrían poner en riesgo la integridad de la dirección mecánica del vehículo eléctrico. La segunda función de la última condición es detener el motor una vez la posición actual es la posición deseada.

El PID trabaja de acuerdo con el error entre el setpoint y la señal de salida del sistema, en este caso la posición del encoder. En la ecuación 5 se establece el error (Er) en un determinado momento (tiempo discreto):

$$Er(z) = posX(Z) - PosD \tag{5}$$

Donde $PosD$ es la posición deseada y $posX$ es la posición actual del encoder.

Para calcular el control proporcional se emplea la ecuación 6, para el control integral se toman los errores de la muestra actual y las dos muestras anteriores, tal como lo indica la ecuación 7 y por último la ecuación 8 muestra la acción de control derivativa, en función de la diferencia entre el erro actual menos el error anterior.

$$P(z) = Er(z) \tag{6}$$

$$I(z) = Er(z-2) + Er(z-1) - Er(z), \quad z \geq 0 \tag{7}$$

$$D(z) = Er(z) - Er(z-1) \tag{8}$$

Una etapa de suma importancia en el diseño del control; es la sintonización del PID. Una vez se tiene un código que permita un control de las salidas y entradas del sistema, lo siguiente es la elección de las ganancias P, I y D. Debido a que se realizó de manera experimental, lo primero que se hizo fue posicionar manualmente al motor en 2000, que es aproximadamente la mitad del rango del encoder, lo que permite la elección de valores superiores o inferiores a este, sin riesgos de perder el control del motor por salir de los límites establecidos para el encoder (100 y 4000)

cuando se energiza. Después se establecieron todas las ganancias con un valor de 0.1 y se ejecutó el código para observar la respuesta del PID.

Como el motor no tiene carga, no requirió de una ganancia proporcional superior a uno para llegar con rapidez a valores cercanos al valor de referencia; el valor final de esta ganancia fue de 0.9. Por otro lado, la ganancia integral finalizó con un valor de cuatro, debido a que presentaba oscilaciones en estado estable. Aunque por definición esta ganancia es la sumatoria de todos los errores, se limitó a que solo fuese la suma del error actual y dos anteriores. Esta decisión se tomó al observar que el error puede ser muy grande cuando la posición deseada está muy alejada de la posición actual del encoder, lo que implica oscilaciones muy abruptas de la acción de control y mayor tiempo para la estabilización.

Por último, la ganancia derivativa mantuvo el valor de 0.1, debido que valores superiores de esta ganancia alentaban demasiado la acción de control y la llegaban a desestabilizar. En la ecuación 9 se presenta el controlador PID y las ganancias de sus respectivos controles.

$$PID(z) = 0.9 * P(z) + 4 * I(z) + 0.1 * D(z) \quad (9)$$

Los códigos que se utilizaron en la Rasberry Pi y en Arduino, están disponibles en https://drive.google.com/drive/folders/11-hFzjQyMEJ8n8rlg4qwzplF-rrYLD9L

3. Resultados

Para observar las cualidades de la propuesta, se presentan las gráficas de la posición del encoder respecto al tiempo de muestreo, que permite ver como se alcanza la posición deseada del rotor del motor. La tabla 1 muestra los errores que nos permiten ilustrar las capacidades del sistema de control.

Se presentan cuatro experimentos representativos que ilustran las diferentes respuestas del sistema de control a diferentes condiciones iniciales y finales. Las posiciones se expresan en términos de los pasos del encoder que van de 0 a 4096 posiciones y un tiempo de muestreo aproximado promedio de 0.002 segundos. El primer experimento, analiza una ventana de 63 muestras en 0.1124 segundos. Logra llevar al rotor de la posición 2000 a la 3000 en 0.04106 segundos con un sobre impulso de 37%, como lo muestra la figura 6.

El segundo ensayo, examina una ventana de 111 muestras en 0.1167 segundos. Consigue transportar al rotor de la posición 3500 a la 2000 en 0.0525 segundos con un sobre impulso de 29.3%, como lo muestra la figura 7.

Tabla 1 Error acumulativo en los cuatro diferentes experimentos.

Experimento	Error acumulativo
2000 a 3000	125.28
3700 a 2000	256.55
3200 a 3650	78.04 en la ventana de tiempo
806 a 430	40.36

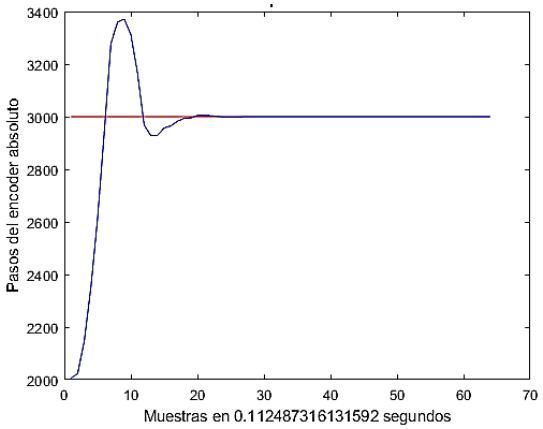

Figura 6 Posición entregada por el encoder respecto de las muestras tomadas en el tiempo cuando se lleva al rotor a la posición 3000.

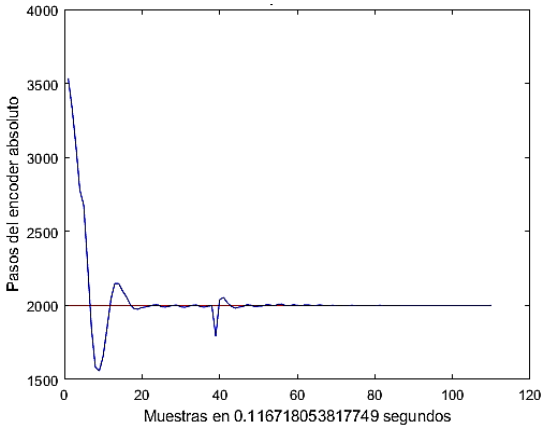

Figura 7 Posición entregada por el encoder respecto de las muestras tomadas en el tiempo cuando se lleva al rotor a la posición 2000.

La tercera prueba, estudia una ventana de 205 muestras en 0.1137 segundos. Alcanza a trasladar al rotor de la posición 3200 a la 3650 en 0.0113 segundos con

un sobre impulso de 42%, como lo muestra la figura 8. Finalmente, se examina una ventana de 146 muestras en 0.1124 segundos. Esta lleva al rotor de la posición 806 a la 430 en 0.0103 segundos con un sobre impulso de 21%, como muestra figura 9.

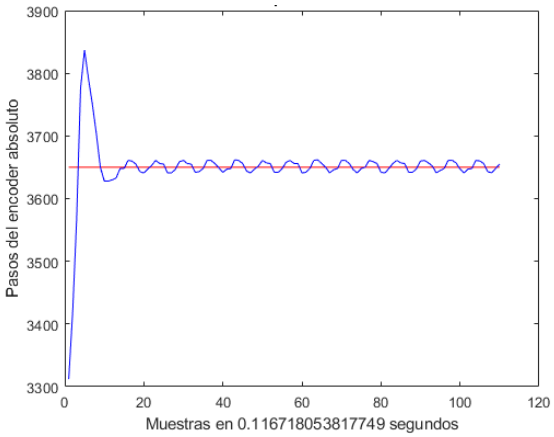

Figura 8 Posición entregada por el encoder respecto de las muestras tomadas en el tiempo cuando se lleva al rotor a la posición 3650.

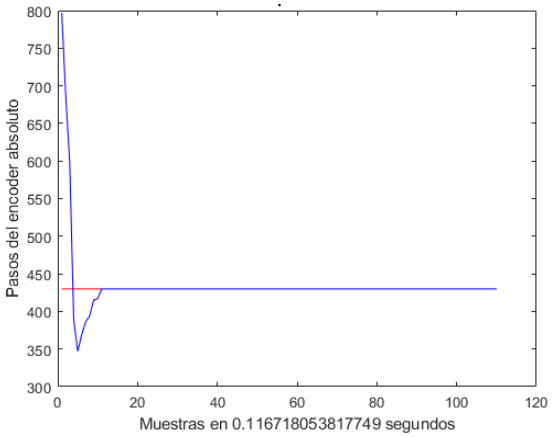

Figura 9 Posición entregada por el encoder respecto de las muestras tomadas en el tiempo cuando se lleva al rotor a la posición 430.

4. Conclusiones

De acuerdo con los resultados se obtuvo un control de posición del rotor de un motor de corriente directa satisfactorios, aunque en la zona más alejada al origen del encoder, se conserva una oscilación en estado estable, lo cual no es deseable, sin embargo, en zonas cercanas al origen se obtuvieron buenos resultados. La Raspberry Pi 3 como centro de control de posición y comunicación entre

componentes, es suficiente para lograr un posicionamiento del rotor medido por un encoder absoluto. Esta plataforma demostró funcionar para probar diferentes controladores que mejoren la exactitud, precisión y velocidad con que se alcancen las posiciones deseadas.

5. Bibliografía y Referencias

[1] Mahesh S. S., Joshitha C., Reddy M., Reddy S., Yaswanth S., Facial Detection and Recognition System on Raspberry pi with Enhanced Security, International Conference on Emerging Trends in Information Technology and Engineering, 2020.

[2] Gutiérrez López M., Barranco Gutiérrez A. I., Implementación de un sistema difuso en VHDL, AMITE 2016. Coatzacoalcos, noviembre 2016.

[3] Monroy Sahade E. A., Lázaro Mata D., Vázquez Rodríguez E. A., Barranco Gutiérrez A. I., Padilla Medina J. A., Fuzzy color description on Raspberry PI 3, Congreso Internacional de Robótica y Computación, 2019.

[4] Cárdenas León A., Pérez Pinal Francisco Javier, Barranco Gutierrez Alejandro Israel, Implementación de sistema difuso en Arduino Uno, Academia Journals, Celaya, noviembre 2016.

[5] K. Ogata, "Sistemas de control en tiempo discreto", Primera edición, Prentice Hall Hispanoamericana SA, 2002, pp. 53-213.

[6] C. R. Dorf, "Sistemas de Control Moderno", Segunda edición, Pearson Education, 2005, pp. 158-188.

[7] García Martínez E., Desarrollo de un controlador PID industrial de bajo coste mediante raspberry pi para control de temperature, Tesis de la Escuela técnica superior de Ingenieros Industriales Valencia, 2016.

[8] L. G. Ramirez, "Sensores y actuadores: Aplicaciones con Arduino", Primera edición, Grupo Editorial Patria, 2016, pp. 37-88.

[9] W. Cheney, D. Kincaid, “Métodos numéricos y computación”, 6a edición, Cengage Learning, 2012, pp. 124-179.

[10] Raspberry pi fundation, Going Straight with PID: https://projects.raspberrypi.org/en/projects/robotPID.