Herramienta para la sintonización de las leyes de control clásico e implementación en un balancín.

Pérez López Ricardo (1)
Departamento de Estudios Multidisciplinarios, DCIS,
universidad de guanajuato,
Yurira, Guanajuato, México
rperezlopez@ugto.mx

Rodrigez Doñate Carlos (2)
Departamento de Estudios Multidisciplinarios, DICS,
Universidad de Guanajuato,
Yurira, Guanajuato, México
c.rodriguezdonate@ugto.mx

***Resumen*—La automatización es una parte fundamental del sector industrial, razón por la cual en la literatura se han propuesto diferentes leyes de control que faciliten y mejoren los procesos de manufactura, como las leyes de control clásico, control inteligente, control predictivo, control difuso, entre otros. Sin embargo, las leyes de control clásico siguen siendo las más utilizadas en la actualidad, debido a la baja demanda de recursos computacionales, la robustez ante perturbaciones y la gran variedad de métodos para conseguir su sintonización. En el presente trabajo se muestra una herramienta desarrollada en Matlab para la implementación de las leyes de control clásico, donde la sintonización se realiza a través del método de respuesta en frecuencia utilizando una tarjeta de desarrollo Arduino UNO, además se diseñó un balancín propulsado con hélice para validar de forma experimental este trabajo.**

Palabras Claves— Leyes de control clásico, sintonización, respuesta en frecuencia, Arduino, balancín.

I. INTRODUCCIÓN

Actualmente, el control clásico sigue siendo ampliamente utilizado debido a su gran eficiencia y fácil implementación, donde la mayoría de sus aplicaciones son enfocadas en la rama industrial, desarrollo tecnológico, robótica, automatización e investigación. Existen diversos trabajos donde los sistemas propuestos están gobernados por medio de un controlador proporcional-integral-derivativo (PID). Estos controladores son capaces de manejar sistemas que requieren un alto grado de precisión, por ejemplo, pueden regular la presión sanguínea del cuerpo humano en personas con problemas de hipertensión [1], controlar las trayectorias de movimiento de manipuladores paralelos [2], incluso optimizar procesos de fermentación biológica [3].

En el área de robótica, los controladores PID son utilizados como el cerebro de control de los robots. De esta manera, pueden realizar diversas acciones con un mínimo de error en sus movimientos. Las estructuras y tecnologías empleadas en estos sistemas dependen enormemente de su aplicación, gracias a ello, existe una gran variedad de robots controlados por las leyes de control clásico. Un claro ejemplo de ello son los robots equilibristas o balancines, que pueden mantener una posición fija y permanecer estables frente a perturbaciones externas [4], sin embargo, controlar este tipo de sistemas resulta ser considerablemente complejo, debido a las diferentes variables que intervienen en ello. A raíz de esto, se han propuesto e implementado diversas técnicas, como el uso de filtros Kalman [5], controladores clásicos del tipo PI, PD, PID, etc. que han sido aplicados a prototipos de segways [6] y balancines de un solo disco [7].

No obstante, en la literatura se puede encontrar una gran variedad de métodos para sintonizar un PID o sus variantes, este proceso puede ser complejo, repetitivo y/o tardado, debido a que en la mayoría de las ocasiones se deben de realizar múltiples pruebas experimentales hasta lograr obtener un buen resultado de la variable que se quiere controlar. Por dichas razones, en este trabajo se propone una herramienta para la sintonización e implementación de las distintas leyes de control clásico a través de una interfaz de usuario en el software Matlab, cuya sintonización se llevó acabo por medio de una tarjeta de desarrollo Arduino aplicando el método de respuesta en frecuencia. Además, se diseñó un balancín propulsado por una hélice para comprobar de forma experimental la eficiencia del trabajo desarrollado.

II. MARCÓ TEÓRICO

Los sistemas retroalimentados también conocidos como sistemas de lazo cerrado son altamente utilizados debido a su rápida acción de respuesta frente a perturbaciones externas, ya que su salida y(s) no solo depende de la entrada R(s) sino que también del error de actuación e(s) manteniendo de esta forma una relación entre la salida y la entrada del sistema. En la “Fig. 1” se muestra un diagrama a bloques de un sistema en lazo cerrado para una planta G(s) y un controlador $G_c(s)$ arbitrarios:

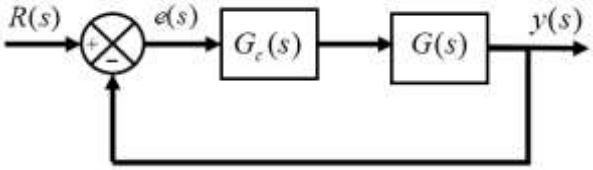

Fig. 1. Sistema de control con retroalimentación unitaria.

A. Controlador PID

En la teoría de control clásico se encuentran diversas leyes de control como es el controlador proporciona (P), integral (I), Proporcional-Integra (PI), Proporcional-Derivativo (PD) y Proporcional-Integral-Derivativo (PID). Donde, cada una de estas leyes puede mejorar el estado estable y/o transitorio del sistema a controlar [8]. La función de transferencia de estos controladores se puede obtener a partir de la suma de cada una de las acciones como en la ecuación (1) que representa la función de transferencia del controlador PID.

$$G_c(s) = k_p + \frac{k_p}{T_i s} + k_p T_d s \quad (1)$$

Donde, K_p es la ganancia proporcional, T_i es el tiempo integral y T_d el tiempo derivativo. Para obtener estos parámetros se utiliza el método de respuesta en frecuencia, donde se deben de cumplir las siguientes condiciones:

Condición de magnitud:

$$\left|G(jw_c)G_c(jw_c)\right| = 1 \quad (2)$$

Condición de fase:

$$\angle G(jw_c)G_c(jw_c) = MF - 180 \quad (3)$$

De esta manera es posible calcular los diferentes parámetros que definen cada una de las leyes de control, donde $G(jw_c)$ son los ángulos aportados por la planta y $G_c(jw_c)$ los ángulos contribuidos por el controlador del sistema. Los parámetros de diseño son el margen de fase (MF) y tiempo de respuesta (tr) el cual se puede aproximar a la frecuencia de cruce de ganancia w_c [8].

Por otro lado, para poder llevar acabo la implementación de cualquier ley de control se debe de obtener su ecuación en diferencial de cada una de las leyes de control P, I, PI, PD y PID. Para obtener la ecuación en diferencia de cuada una de las leyes de control se puede utilizar el método de la transformada bilineal [9], donde al aplicarla se obtiene una ecuación en diferencias general definida por (4).

$$y(k) = b_0 y(k-1) + a_0 e(k) + a_1 e(k-1) + a_2 e(k-2) \quad (4)$$

Siendo b_0, a_0, a_1, a_2, los coeficientes que definen la ley de control y toman los valores que se muestran en la tabla 1.

Tabla I. Coeficientes de las ecuaciones en diferencia por cada uno de los controladores.

Controlador	Coeficientes Para Ecuación De Diferencias			
	b_0	a_0	a_1	a_2
P	0	k	0	0
I	1	$\frac{k_i T}{2}$	$\frac{k_i T}{2}$	0
PD	1	$k_p + \frac{k_d}{T}$	$k_p + \frac{k_d}{T}$	0
PI	1	$k_p + \frac{k_i T}{2}$	$\frac{k_i T}{2} - k_p$	0
PID	1	$k_p + \frac{k_i T}{2} + \frac{k_d}{T}$	$\frac{k_i T}{2} - k_p - \frac{2k_d}{T}$	$\frac{k_d}{T}$

III. METODOLOGIA

En este trabajo se propone una herramienta para la sintonización e implementación de controladores basados en las leyes de control clásico. La etapa de sintonización se desarrolló mediante una interfaz grafica de usuario en el software Matlab y la implementación del controlador se realizó en la plataforma Arduino. Con esta herramienta se logra tener un sistema en software capaz de reducir el tiempo invertido para el ajuste de los diferentes parámetros de sintonización, ya que facilita las pruebas experimentales que por lo general suelen ser tardadas y repetitivas. Además, brinda una serie de graficas con información relevante sobre el sistema a controlar, en las cuales se puede observar el tiempo de respuesta, la estabilidad y el comportamiento de las señales de entrada y salida. Con la finalidad de comprobar la eficiencia de la interfaz, se diseñó e implementó un balancín propulsado por una hélice "Fig.6" el cual fue sintonizado con un controlador PID, logrando controlar la posición en grados del brazo móvil con hélice respecto a la base de su plataforma.

A. Sintonización de las leyes de control en Matlab

La interfaz desarrollada en Matlab para la sintonización de controladores a través del método de respuesta en frecuencia acepta los siguientes parámetros de entrada: tiempo de respuesta (tr), margen de fase (MF), planta del sistema, tiempo de muestreo (T), tipo de aproximación para discretización, tipo de controlador clásico y en caso de un controlador PID el zero propuesto.

A partir de estos parámetros, la interfaz es capaz de generar el tiempo derivativo (T_d) e integral (T_i) los cuales son necesarios para obtener los coeficientes de la ecuación de diferencias (b_0, a_0, a_1, a_2). Además, muestra el diagrama de Bode de la planta ingresada con información de magnitud y fase, así como su respuesta a un escalón unitario con y sin la acción del controlador. También, grafica en tiempo real la salida y(s) lo que permite visualizar como responde el sistema con el controlador frente a perturbaciones externas y monitorear el tamaño del error. En la "Fig.2" se muestra el diseño de la interfaz desarrollada.

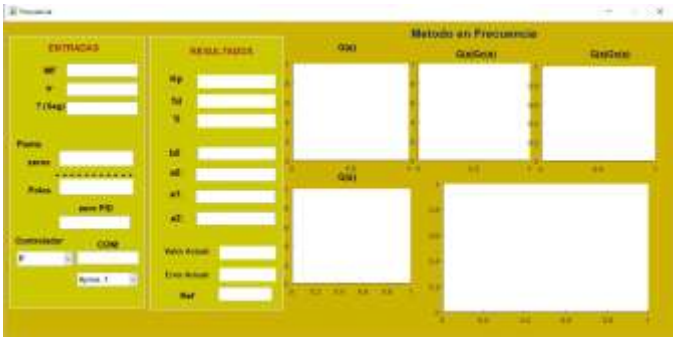

Fig. 2. Interfaz en Matlab para la sintonizacion de las leyes de control.

IV. RESULTADOS

A. *Experimentación*

El sistema de experimentación propuesto consiste en cuatro secciones interconectadas: la interfaz desarrollada en Matlab para la sintonización de controladores, una placa de Arduino, la estructura del balancín y alimentación externa. La estructura del balancín se diseñó mediante el uso del software Fusion 360, el cual permite hacer modelados virtuales para impresión en 3D. Para la parte mecánica se utilizó un balero y un tornillo con la finalidad de reducir la fricción generada por el eje de movimiento. Además, se acoplo un sensor MPU6050 para medir los grados de inclinación de la hélice de drone en el brazo giratorio del balancín "Fig.3".

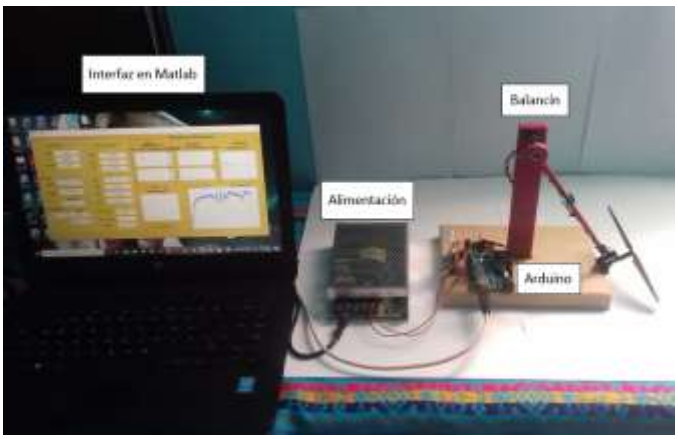

Fig. 3. Sistema de experimentación.

B. *Validación*

Para realizar la identificación de la planta se utilizó el método experimental; como estímulo de entrada del sistema se envió diferentes valores para el PWM y la respuesta del sistema se tomó a partir de la medición de inclinación que genera por las señales del sensor MPU6050, la cual se muestra en la "Fig 4". A partir de la señal de entrada y la salida, se utilizó el Toolbox Ident de Matlab, para estimar la función de transferencia del sistema, la cual se describe en (5).

$$G(s) = \frac{1.7027}{1 + 69.8379S} \quad (5)$$

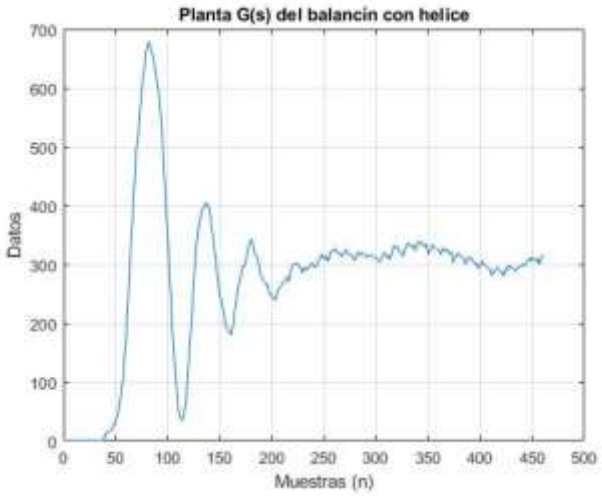

Fig. 4. Respuesta del sensor para identificar.

Una vez obtenida esta función de transferencia, se realizó la sintonización del sistema utilizando un PID, con un margen de fase de 70º, un tiempo de respuesta de 0.9961 s, un periodo de muestra de 0.003 s y el cero se ubicó en -25. A partir de estos parámetros se estimó un Td = 0.02996 s, Ti = 0.15936 s, Kp = 6.5364, Ki = 41.0165, Kd = 0.1958, b0 = 1, a0 = 71.8748, a1 = -137.0286 y a2 = 65.2769. Esta información se envió a la placa de desarrollo de Arduino con una referencia de 50º y se monitoreo la salida del balancín en tiempo real para comprobar su correcto funcionamiento, lo cual se puede ver en la "Fig. 5".

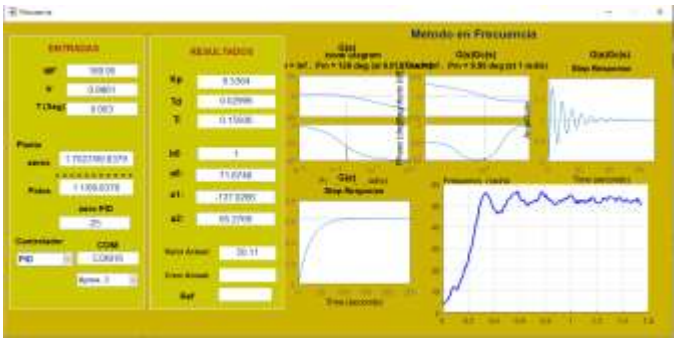

Fig. 5. Sintonización y control del balancín a 50°.

Por otro lado, la "Fig. 6." Muestra como el balancín logra mantenerse en la posición de 50 grados. Al igual que esta prueba se realizaron diversas pruebas con distintas posiciones logrando obtener un resultado satisfactorio.

Fig. 6. Balancin pocisionado a 50°.

V. CONCLUSION

Se diseñó una herramienta en Matlab para la sintonización de las leyes de control clásico a través del método en frecuencia, la cual optimiza el tiempo invertido en el ajuste de los parámetros de sintonización para obtener las ganancias k_p, k_i y k_d, así como los coeficientes de la ecuación de diferencias facilitando las pruebas experimentales. Además, permite observar el comportamiento en magnitud y fase de la planta con y sin controlador mediante los diagramas de Bode, la respuesta del sistema frente a un escalón unitario y la salida y(s) en tiempo real. De esta manera, se puede tener un panorama exacto y preciso del comportamiento del sistema bajo la acción del controlador especificado. También, se controló la posición en grados de un balancín propulsado por una hélice, donde la sintonización del controlador PID utilizado fue a través de la interfaz desarrollada, obteniendo como resultado un buen control del sistema con un error máximo de 2 grados en estado estable.

REFERENCIAS

[1] Kumar, H., Kumar, R., Yadav, J., Rani, A., & Singh, V. (2016, July). Genetic Algorithm based PID controller for blood pressure and Cardiac Output regulation. In 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES) (pp. 1-6). IEEE.

[2] Chaudhary, G., & Ohri, J. (2016, July). 3-dof parallel manipulator control using pid controller. In 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems (ICPEICES) (pp. 1-6). IEEE.

[3] Khan, O., Madhuranthakam, C. M. R., Douglas, P., Lau, H., Sun, J., & Farrell, P. (2018). Optimized PID controller for an industrial biological fermentation process. Journal of Process Control, 71, 75-89.

[4] Kang, Y., She, J., Xu, W., Li, Q., & Jin, H. (2018, August). Design of Three-ring Nesting Cascade PID Controller of Full-auto Two-wheel Balance Vehicle. In 2018 IEEE International Conference on Mechatronics and Automation (ICMA) (pp. 364-369). IEEE.

[5] Prasetio, B. H. (2015, September). Ensemble Kalman filter and PID controller implementation on self balancing robot. In 2015 International Electronics Symposium (IES) (pp. 105-109). IEEE.

[6] Pratama, D., Ardilla, F., Binugroho, E. H., & Pramadihanto, D. (2015, August). Tilt set-point correction system for balancing robot using PID controller. In 2015 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC) (pp. 129-135). IEEE.

[7] Riattama, D., Binugroho, E. H., Dewanto, R. S., & Pramadihanto, D. (2016, September). PENS-wheel (one-wheeled self balancing vehicle) balancing control using PID controller. In 2016 International Electronics Symposium (IES) (pp. 31-36). IEEE.

[8] O. Katsuhiko, "Modern control engineering," Prentice Hall, Fifth Edition, New Jersey, pp. 308-330, pp. 491-511, ISBN: 978-01-3300-225-6, 2010.

[9] Ogata, K. (1995). Discrete-time control systems (Vol. 2, pp. 446-480). Englewood Cliffs, NJ: Prentice Hall.