TeCEO
Revista Tecnología, Ciencia y Estudios Organizacionales

TeCEO, 6(12), 2024.
Revista Tecnología, Ciencia y Estudios Organizacionales / ISSN: 2594-1410
https://doi.org/10.56913/teceo.6.12.181-192

Controlador APID subibaja para equilibrio de robot auto-balanceado

Seesaw APID Controller for balancing a Self-Balancing Robot.

- **Juan Pablo Manzo-Hernández,** Centro de Ingeniería y Desarrollo Industrial (CIDESI) (México) (j.manzo@posgrado.cidesi.edu.mx), (https://orcid.org/0009-0005-7464-0670), Máster.
- **Julio César Solano-Vargas,** Centro de Ingeniería y Desarrollo Industrial (CIDESI) (México) (https://orcid.org/0000-0003-4745-0175), Doctor.
- **Juan Manuel Barrera-Fernández,** Centro de Ingeniería y Desarrollo Industrial (CIDESI) (México) (https://orcid.org/0000-0002-2168-3408), Máster.

Resumen: Siendo los algoritmos subibaja y su implementación en el controlador PID de reciente desarrollo, los resultados publicados hasta hoy en día se han limitado a evaluar su desempeño a nivel simulación. En este sentido, aunque se han obteniendo resultados notables, la implementación del controlador PID adaptativo (APID) con algoritmos subibaja en una planta física era una cuestión pendiente. El presente estudio aborda la implementación del controlador APID subibaja en un robot auto-balanceado de dos ruedas (TWSBR). El controlador APID subibaja es logrado mediante el algoritmo subibaja LS (Less Slope – menos pendiente). El TWSBR es el robot comercial "ELEGOO Tumbller Self-Balancing Robot Car", mismo que cuenta con un controlador PID con sintonización de fábrica para su control de equilibrio programado en un Arduino Nano. En general, el TWSBR y todos sus componentes se conservaron inalterados, el algoritmo de control y su programación fue la única variante. La experimentación consistió en comparar el desempeño de los controladores PID y APID subibaja en el control de equilibrio del TWSBR, empleando una sintonización común de ganancias para ambos casos. Una primer comparativa contempla el análisis de las respuestas transitoria y en estado estable para el control de equilibrio. En breve comentario, el APID subibaja logra mantener el control de equilibrio cuando el PID no lo hace. Los resultados físicos confirman los resultados previamente existentes de simulación. El controlador APID subibaja es superior al controlador PID y sencillo de implementar.

Palabras clave: Algoritmos subibaja, Algoritmo de control adaptativo, Control PID adaptativo, Controlador APID subibaja, Control PID, Robot auto balanceado de dos ruedas.

Cómo citar: Manzo-Hernández, J.P.; Solano-Vargas, J.C.; y Barrera-Fernández, J.M. (2024). Controlador APID subibaja para equilibrio de robot auto-balanceado**.** *Tecnología, Ciencia y Estudios Organizacionales*, 6(12), pp. 181-192. https://doi.org/10.56913/teceo.6.12.181-192

Recepción: 03-10-2024

Aprobación: 23-10-2024

Abstract: Since the seesaw algorithms and their implementation in the PID controllers are of recent development, the results published to date have been limited to evaluating their performance at a simulation level. In this context, although notable results have been obtained, the implementation of the adaptive PID (APID) controller with the seesaw algorithms in a physical plant remained as an open question. This study addresses the implementation of the seesaw APID controller in a two-wheeled self-balancing robot (TWSBR). The seesaw APID controller is achieved through the LS

(Less Slope) seesaw algorithm. The TWSBR used is the commercial "ELEGOO Tumbller Self-Balancing Robot Car," which comes with a factory-tuned PID controller for balance control, programmed into an Arduino Nano. In general, the TWSBR and all its components were kept unchanged, programming was the only variant during experimentation. The experimentation involved comparing the performance of the PID and seesaw APID controllers in the balance control of the TWSBR, using the original factory-tuned gains for both cases. The first comparison includes the analysis of transient and steady-state responses for balance control. In brief, the seesaw APID manages to maintain balance control when the PID does not. The physical results confirm the previously existing simulation results. The seesaw APID controller outperforms the PID controller and easy to implement it.

Keywords: Seesaw algorithms, Adaptive control algorithm, Adaptive PID control, Seesaw APID controller, PID control, Self-balancing robot.

Introducción

El controlador PID es la estrategia de control más implementada a nivel industrial abarcando al menos un 90% de las aplicaciones, incluyendo las más contemporáneas (Åström & Hägglund, 2001; Díaz-Rodríguez et al., 2019). El controlador PID ofrece una solución simple y eficiente capaz de lograr la estabilidad del sistema y reducir el error en estado estable (Åström & Hägglund, 2001; Johnson et al., 2005) a partir a su estructura de tres términos compuesta por el termino proporcional P, el término integral I y el término derivativo D. Cada término involucra una ganancia fija, por lo que una sintonización competente resulta crucial para lograr un desempeño adecuado (Ang et al., 2005; Johnson et al., 2005). Si se considera que un controlador adaptativo es aquel capaz de ajustar sus coeficientes acordes a los cambios del sistema (Swarnkar et al., 2014), debido a sus ganancias fijas, el controlador PID carece de adaptabilidad (Manzo Hernández et al., 2024).

La necesidad de adaptabilidad surge debido a que de manera predeterminada o no, eventualmente se suscitarán cambios o perturbaciones (momentáneas o permanentes) en el sistema que, si el controlador no es capaz de compensar, el desempeño o el sistema mismo se puede ver comprometido, generando incertidumbre y una baja en la confiabilidad del sistema, aspectos que pueden proyectarse inclusive en pérdidas económicas o riesgos.

En este sentido y considerando que el controlador PID no augura perder vigencia (a pesar del auge de novedosas y vanguardistas estrategias de control) existen diversos esfuerzos que buscan mejorar el desempeño del controlador PID. Una parte de estos desarrollos se ha enfocado en combinar el PID con otras estrategias de control ya sea para conseguir una sintonización óptima o para incorporar dinámica (adaptabilidad) a las ganancias del controlador PID.

Existen diversos casos, por ejemplo, el desarrollo de un controlador $APID$ para control de velocidad en vehículos autónomos (AGV's) que incorpora algoritmos genéticos (GA) y redes neuronales (NN) (Kebbati et al., 2021); el desarrollo de un controlador APID basado en aprendizaje profundo (DL) mediante redes neuronales (NN) para el control de vehículos en caravana (de Zarzà et al., 2023) y el desarrollo de un controlador $APID$ que combina modos deslizantes (SMC) y lógica difusa (FL) para el control de altitud de drones (Noordin et al., 2023), entre otros.

Si bien el desempeño de estos controladores no fuese cuestionado, es de resaltar que la combinación de dos o más controladores indudablemente incrementa la complejidad en relación a

 ISSN: 2594-1410

su entendimiento, implementación y procesamiento computacional (Manzo Hernández et al., 2024), en este tenor se desarrollaron los algoritmos subibaja (Manzo Hernández et al., 2024), que esencialmente aportan adaptabilidad con menor complejidad si se le compara con las tendencias actuales.

Los algoritmos subibaja consisten en generar comportamientos alternativos a partir de alterar la pendiente o derivada del error actual $de(t)/dt$ en una señal de control. Dichos comportamientos alternativos se insertan dentro del controlador PID, lo que le añade características adaptativas al mismo, logrando así un controlador PID adaptativo basado en los algoritmos subibaja ($APID_{subibaja}$) con prestaciones considerables.

Si bien el desarrollo y testeo de estos algoritmos fue favorable, estos sólo habían sido probados a nivel simulación, por lo que un paso natural y obligado complementario en su desarrollo e investigación era dar profundidad al uso de estos algoritmos mediante su implementación en una planta física. Para esto se ha seleccionado un robot auto-balanceado de dos ruedas (comercial), que es una variante del péndulo invertido, lo que lo hace una planta de prueba idónea para algoritmos de control ya que su naturaleza hace evidente la acción del controlador al conseguir o no su equilibrio.

Es de mencionar que bien si pueden existir diversas maneras de alterar una pendiente, hasta el momento sólo se han formalizado dos estrategias particulares como algoritmos subibaja, el algoritmo subibaja MS (más pendiente) y el algoritmo subibaja LS (menos pendiente). El presente trabajo solo aborda al algoritmo subibaja LS ya que se anticipó que su programación fuese relativamente sencilla.

Para un amplio entendimiento de los algoritmos subibaja, el trabajo original (Manzo Hernández et al., 2024) aborda una aproximación gráfica, su formulación matemática y demostración en extenso, así como la formulación matemática correspondiente para su implementación en un controlador PID y testeo con diversas funciones de transferencia (todo a nivel simulación).

Método

El presente apartado explora el concepto del algoritmo subibaja LS, su formulación y programación para adicionarlo al controlador PID. De igual manera se incluye una breve descripción del robot y se aborda la programación particular de la tarjeta Arduino perteneciente al mismo y se plantea la experimentación.

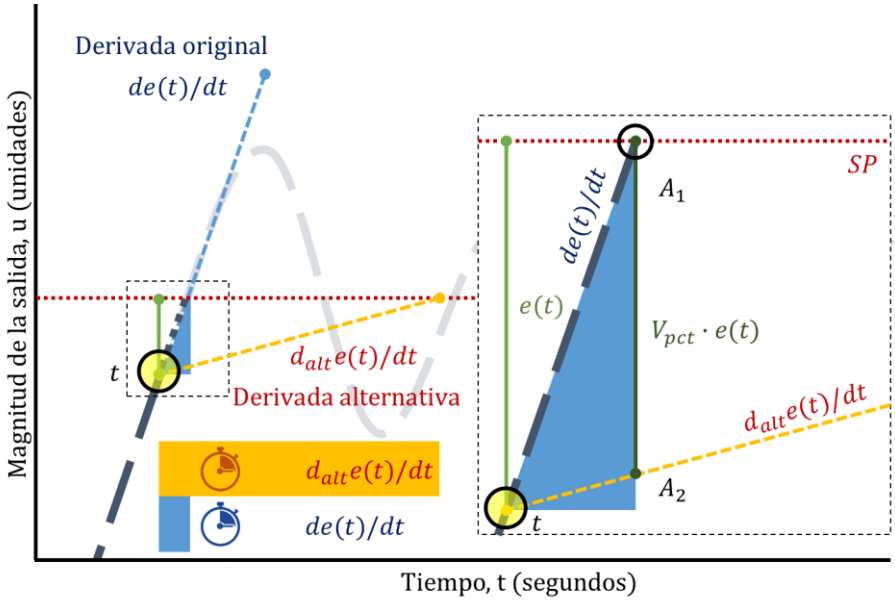
Algoritmo subibaja LS

Dentro de una estrategia de control, la derivada del error $de(t)/dt$ puede entenderse como una estimación futura del comportamiento, en este sentido existe un comportamiento original inherente a la pendiente original m_{orig} ($de(t)/dt$) de una señal $(t, e(t))$. Los algoritmos subibaja consisten en usar el instante actual t como pivote y modificar la pendiente m_{orig} que existe en la estrategia de control (PID control) a partir del error $e(t)$ y su derivada $de(t)/dt$. El resultado da un nuevo valor de pendiente, una nueva pendiente alternativa m_{alt} ($d_{alt}e(t)/dt$), por lo que la perpendicularidad respecto al valor de la pendiente original m_{orig} puede ser manipulada. Al modificar m_{orig} y generar pendientes alternativas m_{alt}, se generan propuestas de comportamientos alternativos que se pueden insertar en la formulación del PID y lograr que los coeficientes del PID se ajusten a dichos comportamientos propuestos. Si se considera que los cambios en el sistema se reflejan en $e(t)$, siendo $e(t)$ y $de(t)/dt$ parámetros dinámicos en el

tiempo, m_{alt} también es dinámica, m_{alt} propone comportamientos según cambia el sistema, al implementar estos algoritmos en el PID, se logra un PID adaptativo ($APID_{subibaja}$) que mejora la respuesta del sistema. En este artículo se muestra la modificación de m_{orig} a partir del error $e(t)$ y su derivada $de(t)/dt$ a través del algoritmo subibaja LS (menos pendiente), el cual procura amortiguar la aproximación hacía el set point SP.

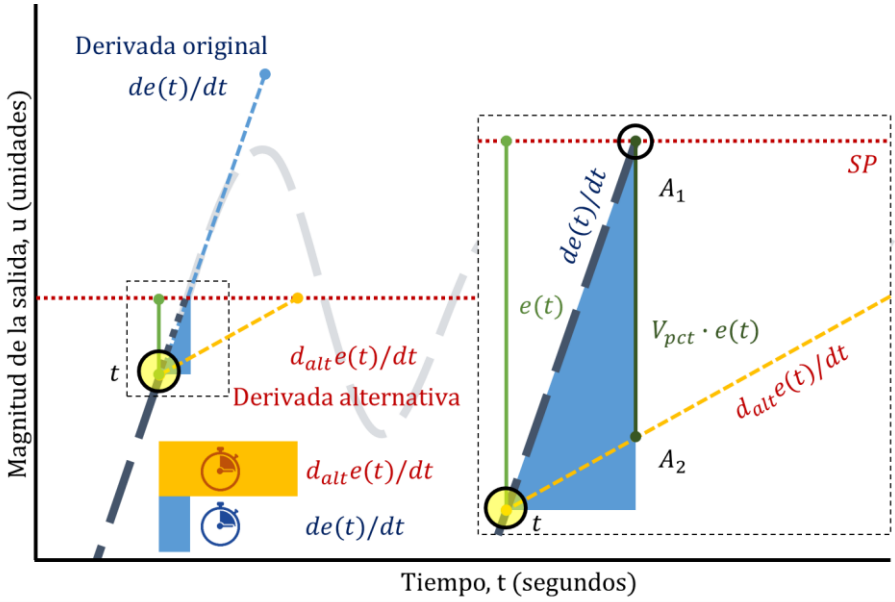
Para generar una pendiente alternativa m_{alt} con el algoritmo subibaja LS (ver Figura 1), considérese t y m_{orig} para generar un punto A_1 en la intersección de m_{orig} y SP. Con A_1 se genera un punto A_2 definido por $V_{pct} \cdot e(t)$, siendo V_{pct} una magnitud porcentual (entre 0 y 1) aplicada sobre el eje y, por lo tanto $V_{pct} \cdot e(t)$ es una variación porcentual premeditada de la magnitud de $e(t)$ sobre el eje y. A partir de A_2 es posible generar una nueva m_{alt} menor que m_{orig}. La Figura 1 muestra m_{alt}, cuando $V_{pct} \cdot e(t) = 0.9 \cdot e(t)$. La Figura 2 muestra m_{alt}, cuando $V_{pct} \cdot e(t) = 0.8 \cdot e(t)$.

Figura 1. *Esquemático del algoritmo subibaja LS cuando* $V_{pct} \cdot e(t) = 0.9 \cdot e(t)$.



 ISSN: 2594-1410

Figura 2. *Esquemático del algoritmo subibaja LS cuando* $V_{pct} \cdot e(t) = 0.8 \cdot e(t)$.



El algoritmo subibaja LS es aplicable para cualquier t convergente hacia SP, no se aplica a señales divergentes de SP. La suavidad del comportamiento alternativo que propone m_{alt} con el algoritmo subibaja LS en cuanto a la convergencia hacia SP se manipula a partir de V_{pct}.

El cálculo de m_{alt} para el algoritmo subibaja LS se realiza con la ecuación (1).

$$m_{alt} = m_{orig}\left(\frac{SP + e(t)\left(V_{pct} - 1\right)}{SP - e(t)}\right) \tag{1}$$

La programación correspondiente a la ecuación (1) considera la siguiente lógica:

If
Condition: $e(t) > 0$ & $m_{orig} < 0$ %positive error
Action: $m_{orig}\left(SP + e(t)\left(V_{pct} - 1\right)/SP - e(t)\right)$
Output: m_{alt}
Else If
Condition: $e(t) < 0$ & $m_{orig} > 0$ %negative error
Action: $m_{orig}\left(SP + e(t)\left(V_{pct} - 1\right)/SP - e(t)\right)$
Output: m_{alt}
Else
Action: m_{orig}
Output: m_{orig}

Controlador adaptativo $APID_{subibaja}$

Para incorporar el comportamiento propuesto por m_{alt} al controlador PID, es requerido calcular las ganancias $k_{P_{alt}}$, $k_{P_{Adapt}}$, $k_{I_{Adapt}}$ y $k_{D_{Adapt}}$.

$$k_{P_{alt}} = \Lambda \cdot k_P \frac{m_{alt}}{m_{orig} + \varepsilon} \tag{2}$$

$$k_{P_{Adapt}} = k_P + k_{P_{alt}} \tag{3}$$

$$k_{I_{Adapt}} = \frac{k_{P_{Adapt}}}{\tau_I} \int_0^t e(t)dt \tag{4}$$

$$k_{D_{Adapt}} = k_{P_{Adapt}} \cdot \tau_D \frac{de(t)}{dt} \tag{5}$$

Las ganancias $k_{P_{Adapt}}$, $k_{I_{Adapt}}$ y $k_{D_{Adapt}}$ se incorporan a la formulación del PID para lograr un controlador adaptativo.

$$P_{Adapt} = k_{P_{Adapt}} \cdot e(t) \tag{6}$$

$$I_{Adapt} = k_{I_{Adapt}} \int_0^t e(t)dt \tag{7}$$

$$D_{Adapt} = k_{D_{Adapt}} \frac{de(t)}{dt} \tag{8}$$

$$APID_{subibaja} = P_{Adapt} + I_{Adapt} + D_{Adapt} \tag{9}$$

El cálculo de $k_{P_{alt}}$, $k_{P_{Adapt}}$, $k_{I_{Adapt}}$ y $k_{D_{Adapt}}$ se realiza con las siguientes funciones:

`Function:` $k_{P_{alt}} = k_P \cdot (m_{alt}/(m_{orig} + \varepsilon))$
`Function:` $k_{P_{Adapt}} = k_P + (k_{P_{alt}} \cdot \Lambda_P)$
`Function:` $k_{I_{Adapt}} = k_I + (k_{P_{alt}}/\tau_I) \cdot \Lambda_I)$
`Function:` $k_{D_{Adapt}} = k_D + (k_{P_{alt}} \cdot \tau_D) \cdot \Lambda_D)$

Robot auto-balanceado

El robot auto-balanceado es un robot comercial de la marca ELEGOO denominado como Tumblller V1.1 (*ELEGOO Tumblller Self-Balancing Robot Car V1.1/V1.0 Tutorial*, 2020), el cual incluye dos motores GA37-520 DC, un driver dual TB6612FNG, un módulo GY-521 en donde se monta una MPU6050 que incorpora un giroscopio y un acelerómetro con los que se logra obtener el ángulo de inclinación, y por último, también incluye una tarjeta de expansión en donde se monta un Arduino Nano como tarjeta de control. Si bien el robot incluye más elementos como sensores infrarrojos y un sensor ultrasónico, estos no competen dentro de la experimentación realizada. La información relativa a este robot puede ser encontrada en el sitio web del fabricante.

Programación de la tarjeta Arduino Nano

Para la obtención de m_{alt}, la programación realizada se expresa a continuación:

```
if((kalmanfilter_angle - angle_zero) > 0 && speed_control_output
< 0)
{LS = speed_control_output * (0 + ((kalmanfilter_angle -
angle_zero) * (Vpct - 1))) / (0 - (kalmanfilter_angle -
angle_zero) + Epsilon);}
if((kalmanfilter_angle - angle_zero) < 0 && speed_control_output
> 0)
{LS = speed_control_output * (0 + ((kalmanfilter_angle -
angle_zero) * (Vpct - 1))) / (0 - (kalmanfilter_angle -
angle_zero) + Epsilon);}
else
{LS=speed_control_output;}
```

Siendo m_{orig} = speed_control_output, $e(t)$ = kalmanfilter_angle - angle_zero,SP = 0° y m_{alt} = LS. Epsilon se implementó para evadir indeterminación, debido a que SP se fija en 0°.

Para la obtención de las ganancias adaptativas y subsecuentemente del controlador $APID_{subibaja}$, la programación realizada se expresa a continuación:

```
kpALTLS=kp_balance * (LS / (speed_control_output + Epsilon));

kpADPTLS = kp_balance + (kpALTLS * 0.055);
kiADPTLS = ki_balance + (0 * (kpALTLS / Ti));
kdADPTLS = kd_balance + (0.85 * (Td * kpALTLS));

double balance_control_output = ((kpADPTLS)) *
(kalmanfilter_angle - angle_zero)) + ((kdADPTLS)) *
(kalmanfilter.Gyro_x - angular_velocity_zero));
```

Siendo $k_{P\,alt}$= kpALTLS, $k_{P\,Adapt}$ = kpADPTLS, $k_{I\,Adapt}$ = kiADPTLS, $k_{D\,Adapt}$ = kdADPTLS, $e(t)$ = kalmanfilter_angle - angle_zero, $de(t)/dt$ = kalmanfilter.Gyro_x - angular_velocity_zero, Λ_P = 0.055, Λ_I = 0 y Λ_D = 0.85.

Para su contraste, se muestra la programación original del PID, el cual omite de igual manera el termino integral (*ELEGOO Tumbller Self-Balancing Robot Car V1.1/V1.0 Tutorial*, 2020).

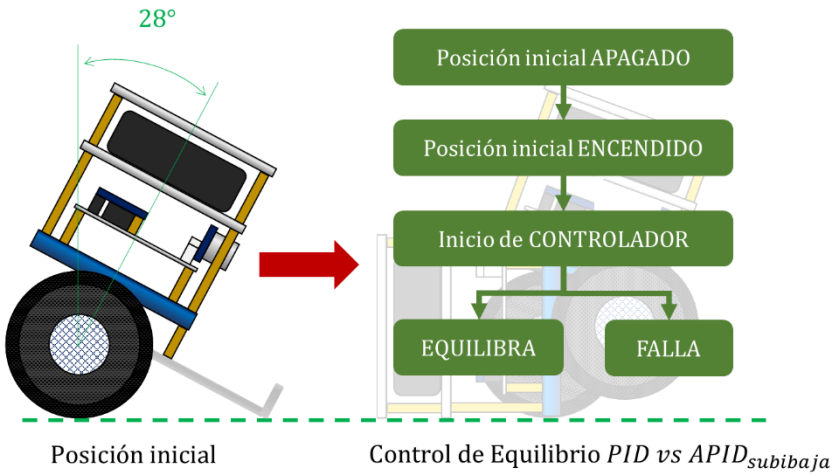
```
double balance_control_output = (kp_balance *
(kalmanfilter_angle - angle_zero)) + (kd_balance *
(kalmanfilter.Gyro_x - angular_velocity_zero));
```

Propuesta experimental

El planteamiento consiste en identificar una sintonización de PID en la cual el robot fuese inestable, en otras palabras, que no se pudiera equilibrar. La intención de esta selección es hacer evidente el aporte del $APID_{subibaja}$, ya que como lo expresa su formulación, las ganancias adaptativas se conforman de la suma de las ganancias originales y ganancias alternativas, que cambian de magnitud en relación al error $e(t)$ y a la derivada del error $\frac{de(t)}{dt}$.

El robot ELEGOO Tumbller tiene una posición de arranque en reposo, en la cual, el robot se inclina hacia el frente, hasta que una extensión del robot hace contacto con el piso, manteniendo al robot con una inclinación de 28° aproximadamente, cuando el robot se enciende en esta posición, el controlador busca lograr la estabilidad a 0°.

Figura 3. *Secuencia experimental*



La experimentación consiste en cargar el PID y el $APID_{subibaja}$, ambos con las mismas ganancias, realizando 10 iteraciones para cada caso. Cabe mencionar que el fabricante, omite el término integral en su sintonización, por lo que se decidió conservar esta condición. Si bien no se optó por emplear la sintonización original (*ELEGOO Tumbller Self-Balancing Robot Car V1.1/V1.0 Tutorial*, 2020), esto fue en gran medida por buscar hacer evidente la condición de adaptabilidad. La Tabla 1 muestra la sintonización original del fabricante y la sintonización realizada para llevar al PID convencional a una condición de inestabilidad que sirviese como punto de partida para probar el controlador propuesto.

 ISSN: 2594-1410

Tabla 1.

Parámetros de los controladores

Sintonización ***PID*** original			Sintonización ***PID*** inestable		
	PID			PID	$APID_{subibaja}$
k_P	55		k_P	27	27
k_I	0		k_I	0	0
k_D	0.75		k_D	0.75	0.75
τ_I	0		τ_I	0	0
τ_D	0.01363		τ_D	0.02777	0.02777
			V_{pct}		0.01
			Λ_P		0.055
			Λ_I		0
			Λ_D		0.85

Resultados

La Tabla 2 muestra la ejecución de las iteraciones mencionadas. Entiéndase por inestable que el robot se cae, y por estable que el robot se equilibra.

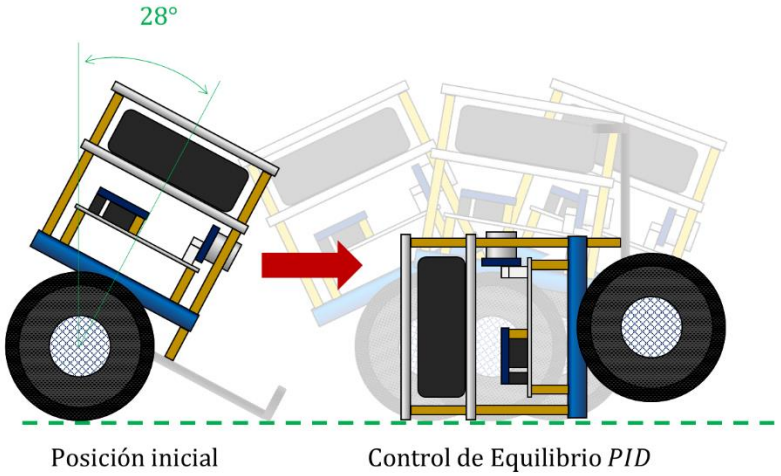
Tabla 2.

Prueba de equilibrio, desde posición de reposo a 28°

Iteración	***PID***	$APID_{subibaja}$
1	Inestable	Estable
2	Inestable	Estable
3	Inestable	Estable
4	Inestable	Estable
5	Inestable	Estable
6	Inestable	Estable
7	Inestable	Estable
8	Inestable	Estable
9	Inestable	Estable
10	Inestable	Estable

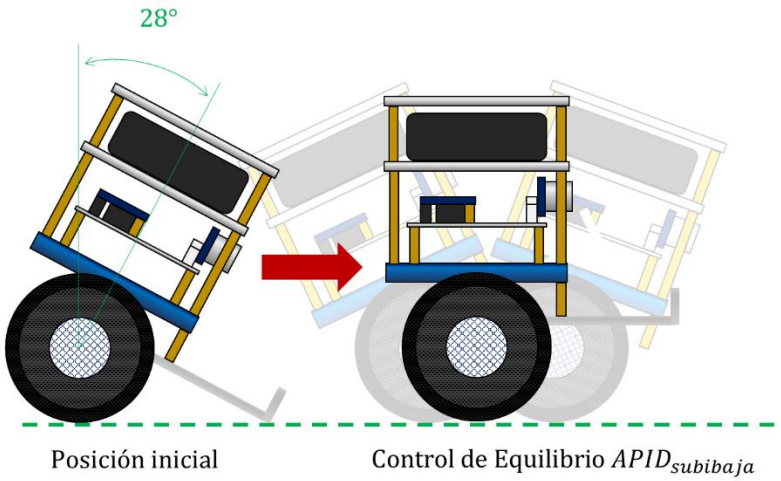
Después de 10 iteraciones de encendido desde la posición de 28° hacia el frente, en 10 de 10 ocasiones (que si bien, era algo previsto), el controlador *PID* no logró la estabilidad, haciendo que el robot cayera hacía atrás. Al encender el robot, el controlador en toda ocasión movía el robot hacia adelante, incrementando la velocidad progresivamente, logrando que el robot redujera el ángulo tendiendo hacia 0°, sin embargo, el impulso de corrección generaba una inercia, en la que el ángulo de 0° era sobrepasado, yendo hacia valores negativos, que ciertamente el robot intentaba corregir yendo hacia atrás, aunque, la inercia resultó en todos los casos mayor que la acción correctiva del controlador y el robot terminaba cayendo de espaldas (ver Figura 4).

Figura 4. *Esquemático representativo de la acción del controlador PID*



En el caso del controlador $APID_{subibaja}$, en las 10 iteraciones consiguió la estabilidad, la secuencia se ejecutó de igual manera. Al encender el robot, este se movía hacia adelante, logrando reducir el ángulo tendiendo hacia los 0°, aun así, la inercia de la corrección, provocaba que el SP se sobrepasara, yendo hacia valores negativos, sin embargo, el controlador era capaz de corregir adecuadamente introduciendo una acción de control de mayor magnitud sin provocar mayores oscilaciones en la posteridad y logrando el equilibrio del robot (ver Figura 5).

Figura 5. *Esquemático representativo de la acción del controlador* $APID_{subibaja}$



Discusión

La intención de este trabajo, adicional a realizar una primera implementación física del algoritmo en una planta física, era también para ofrecer una versión de resultados cualitativos del uso del controlador $APID_{subibaja}$, en comparativa con el controlador PID.

 ISSN: 2594-1410

En un trabajo previo, se abordó extensamente un análisis cuantitativo de la respuesta transitoria y en estado estable de ambos controladores, resultando el $APID_{subibaja}$ a nivel simulación, ser superior que el controlador PID en todas las métricas abordadas.

La elección para este tipo de planta se basó en que los sistemas tipo péndulo invertido, hacen relativamente muy evidente si el controlador funciona o no. Si bien el PID con una sintonización adecuada es enteramente capaz de lograr la estabilidad del sistema propuesto. La intención de llevarlo a la inestabilidad fue principalmente para mostrar que el controlador $APID_{subibaja}$ puede compensar y adaptar la acción de control, cuando el PID no.

Otro punto, para la elección de esta planta, está basado en que es un robot comercial, cuyos componentes resultaran idénticos para todo aquello que adquiera el equipo. Haciendo altamente repetibles los resultados de este trabajo.

Futuramente, se considera ejecutar más pruebas (en plantas basadas en el péndulo invertido) que evidencien, como en este caso, claramente de manera cualitativa el aporte de un controlador adaptativo (u otro) en complemento con un extenso análisis cuantitativo. Actualmente, el equipo desarrollador trabaja en el establecimiento de otras pruebas cualitativas como la adición estratégica de masa para alterar el centro de gravedad del dispositivo.

Referencias

Ang, K. H., Chong, G., & Li, Y. (2005). PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, *13*(4), 559-576. IEEE Transactions on Control Systems Technology. https://doi.org/10.1109/TCST.2005.847331

Åström, K. J., & Hägglund, T. (2001). The future of PID control. *PID Control*, *9*(11), 1163-1175. https://doi.org/10.1016/S0967-0661(01)00062-4

de Zarzà, I., de Curtò, J., Roig, G., & Calafate, C. T. (2023). LLM Adaptive PID Control for B5G Truck Platooning Systems. *Sensors*, *23*(13), Article 13. https://doi.org/10.3390/s23135899

Díaz-Rodríguez, I. D., Han, S., & Bhattacharyya, S. P. (2019). *Analytical Design of PID Controllers*. Springer International Publishing. https://doi.org/10.1007/978-3-030-18228-1

ELEGOO Tumbller Self-Balancing Robot Car V1.1/V1.0 Tutorial. (2020, octubre 21). ELEGOO Official. https://www.elegoo.com/blogs/arduino-projects/elegoo-tumbller-self-balancing-robot-car-tutorial

Johnson, M. A., Moradi, M. H., & Crowe, J. (2005). *PID control: New identification and design methods*. Springer New York.

Kebbati, Y., Ait-Oufroukh, N., Vigneron, V., Ichalal, D., & Gruyer, D. (2021). Optimized self-adaptive PID speed control for autonomous vehicles. *2021 26th International Conference on Automation and Computing (ICAC)*, 1-6. https://doi.org/10.23919/ICAC50006.2021.9594131

Manzo Hernández, J. P., Solano Vargas, J. C., Barrera Fernández, J. M., Moreno Orduña, D., & Sánchez Saquín, C. H. (2024). A novel adaptive PID controller with new seesaw algorithms using alternative derivatives. *Systems Science & Control Engineering*, *12*(1), 2363625. https://doi.org/10.1080/21642583.2024.2363625

Noordin, A., Mohd Basri, M. A., & Mohamed, Z. (2023). Real-Time Implementation of an Adaptive PID Controller for the Quadrotor MAV Embedded Flight Control System. *Aerospace*, *10*(1), Article 1. https://doi.org/10.3390/aerospace10010059

Swarnkar, P., Jain, S. K., & Nema, R. K. (2014). Adaptive Control Schemes for Improving the Control System Dynamics: A Review. *IETE Technical Review*, *31*(1), 17-33. https://doi.org/10.1080/02564602.2014.890838