II Simposio CEA de los GT: Ingeniería de Control - Modelado, Simulación y Optimización - Educación en Automática (2026)

Técnicas de control inteligente aplicadas al robot educativo Lego Mindstorms EV3

Arambarri, Javier[a,*], Hernández, Ignacio[a], Cabanes, Itziar[a], Mancisidor, Aitziber[a], Santos, Matilde[b]

[a]*Departamento de Ingeniería de Sistemas y Automática, Escuela de Ingeniería de Bilbao, Universidad del País Vasco (UPV/EHU), 48013-Bilbao, España.*
[b]*Instituto de Tecnología del Conocimiento, Universidad Complutense de Madrid, 28040-Madrid, España.*

To cite this article: Arambarri, J., Hernández, I., Cabanes, I., Mancisidor, A., Santos, M. 2026. Intelligent control techniques applied to the educational robot Lego Mindstorms EV3. Actas del II Simposio CEA de los GT: Ingeniería de Control - Modelado, Simulación y Optimización - Educación en Automática. Zaragoza, España.

Resumen

Este trabajo presenta una experiencia docente para la enseñanza de conceptos avanzados de automática mediante la experimentación directa con un robot educativo Lego Mindstorms EV3. Se emplean técnicas de control inteligente —control borroso o *fuzzy* y control proporcional optimizado mediante algoritmos genéticos— con el objetivo de corregir desviaciones de trayectoria derivadas de imperfecciones reales del sistema, como diferencias entre motores, rozamientos y perturbaciones externas. La propuesta se fundamenta en el aprendizaje basado en experimentación, permitiendo al estudiante confrontar modelos teóricos con el comportamiento de un sistema físico real, y analizar de forma crítica las limitaciones de la modelización y la sintonización manual de controladores clásicos. La experiencia presentada es fácilmente reproducible y adaptable a asignaturas universitarias de automática y control, constituyendo una alternativa atractiva a la enseñanza basada exclusivamente en simulación.

Palabras clave: robótica educativa, control inteligente, controlador borroso, optimización genética.

Intelligent control techniques applied to the educational robot Lego Mindstorms EV3

Abstract

This paper presents a teaching experience for advanced concepts of automation through direct experimentation with a Lego Mindstorms EV3 educational robot. Intelligent control techniques —fuzzy control and proportional control optimized using genetic algorithms— are used to correct trajectory deviations resulting from real imperfections in the system, such as differences between motors, friction and external disturbances. The proposal is based on experiential learning, allowing students to compare theoretical models with the behavior of a real physical system and critically analyze the limitations of modeling and manually tuning classical controllers. The experience presented is easily reproducible and adaptable to university courses in automation and control, constituting an attractive alternative to teaching based exclusively on simulation.

Keywords: educational robotics, intelligent control, fuzzy controller, genetic optimization.

1. Introducción

La enseñanza de la automática y el control en titulaciones de ingeniería se enfrenta habitualmente a la dificultad de trasladar los conceptos teóricos a situaciones reales. Gran parte de la docencia práctica se apoya en simuladores, lo que limita la exposición del alumnado a las incertidumbres, no linealidades y perturbaciones propias de los sistemas físicos reales. Esta brecha entre teoría y práctica puede dificultar la comprensión profunda de los principios de control y reducir la motivación del estudiantado.

En este contexto, la robótica educativa se ha consolidado como una herramienta especialmente adecuada para la docencia en automática (Zhang and Wan, 2023), al permitir la experimentación directa con sistemas reales de forma segura, accesible y motivadora. Plataformas como Lego Mindstorms EV3 (Lego Education, 2013) ofrecen un entorno ideal para

*Autor para correspondencia: javier.arambarri@ehu.eus

introducir conceptos de modelado, control y optimización, ya que presentan imperfecciones reales —diferencias entre motores, fricciones o errores de medida— que rara vez se consideran en simulación, pero que resultan fundamentales desde el punto de vista formativo.

Este trabajo presenta una experiencia docente basada en la implementación y comparación de distintos controladores sobre un robot móvil diferencial, combinando técnicas de control clásico e inteligente. Se desarrollan un controlador borroso (*fuzzy*) y un controlador proporcional optimizado mediante algoritmos genéticos, con el objetivo de corregir desviaciones de trayectorias rectilíneas sin depender exclusivamente de modelos ideales ni de sintonizaciones manuales.

Más allá de los resultados técnicos, la propuesta destaca por su valor formativo: permite al alumnado analizar las limitaciones del modelo geométrico, contrastar estimaciones con medidas reales y comprender la influencia de las ganancias, los criterios de rendimiento y las perturbaciones del entorno. Todo ello favorece un aprendizaje activo y crítico alineado con los objetivos de las asignaturas de automática y control.

La implementación de la algoritmia del presente trabajo se encuentra en la rama “sime-2026” del siguiente repositorio:

https://github.com/arambarricalvoj/eduROS2/tree/sime-2026.

Se propone que el alumnado utilice *ROS2 Jazzy Jalisco* (Macenski et al., 2022) como marco de integración, junto con programación en *Python* y *C++*. A tal fin, el controlador borroso se codificará haciendo uso de la librería *fuzzylite* (Rada-Vilela, 2018) mientras que para el algoritmo genético utilizará como referencia Gutiérrez Reina et al. (2020). El esquema de control estará basado en Arambarri Calvo and Arambarri Calvo (2024) y la comunicación con el *hardware* se gestionará mediante el puente *eduROS2* (Arambarri Calvo, 2025).

Este artículo está organizado de la siguiente manera: en la Sección 2 se describe el modelo geométrico del robot diferencial. En la Sección 3 se presenta el controlador borroso diseñado e implementado. En la Sección 4 se expone la optimización desarrollada mediante un algoritmo genético para la sintonización de un controlador proporcional. Finalmente, en las Secciones 5 y 6 se presentan los resultados y las conclusiones de trabajo, respectivamente.

2. Modelo geométrico del robot diferencial

El modelo geométrico del robot diferencial se basa en la relación entre las velocidades de sus dos ruedas y la trayectoria resultante, Figura 1, y ha sido ampliamente estudiado en la literatura (Siegwart et al., 2011).

Cada rueda, de diámetro D, genera una velocidad lineal, $v_{\text{Left}}(t)$ y $v_{\text{Right}}(t)$, Ecuación (1), proporcional a su velocidad angular $\omega_{\text{Left}}(t)$ y $\omega_{\text{Right}}(t)$, mientras que la diferencia entre ambas velocidades lineales determina la velocidad angular del chasis (*yaw rate*, símbolo $\omega(t)$), Ecuación (2), donde L es la distancia entre ruedas, denominada *axle track*. La velocidad lineal total del robot se obtiene como una media de ambas, Ecuación (3).

$$v_{\text{Left}}(t) = \omega_{\text{Left}}(t)\,\frac{D}{2}, \quad v_{\text{Right}}(t) = \omega_{\text{Right}}(t)\,\frac{D}{2} \tag{1}$$

$$\omega(t) = \frac{v_{\text{Left}}(t) - v_{\text{Right}}(t)}{L} \tag{2}$$

$$v(t) = \frac{v_{\text{Left}}(t) + v_{\text{Right}}(t)}{2} \tag{3}$$

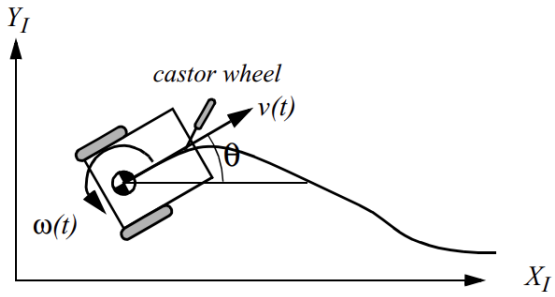

Figura 1: Cinemática diferencial (Siegwart et al., 2011).

La orientación del robot, $\theta(t)$, puede actualizarse de dos formas complementarias: a partir de distancias recorridas, o a partir de velocidades instantáneas. A continuación se resume el modelo geométrico del robot para ambas aproximaciones en la orientación.

- **A partir de distancias recorridas**

Si $d_{\text{Left}}(t)$ y $d_{\text{Right}}(t)$ representan las distancias lineales recorridas por las ruedas izquierda y derecha, respectivamente, el incremento angular del robot en un intervalo de tiempo se calcula como la Ecuación (4). La orientación acumulada se obtiene sumando sucesivamente estos incrementos, como se muestra en la Ecuación (5).

$$\Delta\theta(t) = \frac{d_{\text{Left}}(t) - d_{\text{Right}}(t)}{L} \tag{4}$$

$$\theta(t) = \sum \Delta\theta(t) \tag{5}$$

- **A partir de velocidades instantáneas**

Si se consideran las velocidades lineales de cada rueda $v_{\text{Left}}(t)$ y $v_{\text{Right}}(t)$, la velocidad angular instantánea del robot se expresa como la Ecuación (6), donde $\dot{\theta}(t)$ representa la velocidad de giro del chasis (*yaw rate*). En este caso, la orientación se obtiene integrando la velocidad angular en el tiempo, como se muestra en la Ecuación (7).

$$\dot{\theta}(t) = \frac{v_{\text{Left}}(t) - v_{\text{Right}}(t)}{L} \tag{6}$$

$$\theta(t) = \int \dot{\theta}(t)\,dt \tag{7}$$

De esta forma, el modelo permite trabajar tanto con incrementos discretos de distancia como con velocidades instantáneas, adaptándose a diferentes modos de lectura de los *encoders*. Una vez calculada la orientación, se obtiene el error en la trayectoria rectilínea mediante la Ecuación (8), donde θ_0 es

la orientación inicial del robot en reposo, previo a comenzar el movimiento:

$$\theta(t)_{\text{error}} = \theta(t) - \theta_0 \tag{8}$$

Una vez obtenido el modelo geométrico del robot con dos aproximaciones en el cálculo de la orientación, se expone a continuación la implementación de diferentes estrategias de control avanzado y la comparativa entre ellas. El diámetro de las ruedas del robot utilizado es $D = 62.4$ milímetros (mm) y la separación entre ambas es $L = 110$ milímetros (mm).

3. Control *fuzzy*

El propósito del control *fuzzy* o borroso es manejar la incertidumbre y la imprecisión en sistemas dinámicos (Wang et al., 2025), permitiendo diseñar controladores que no requieren un modelo matemático exacto del proceso. Se basa en la lógica borrosa, introducida por (Zadeh, 1965), que permite representar conocimiento mediante reglas lingüísticas.

Este tipo de control resulta especialmente útil cuando la relación entre las variables de entrada y salida es no lineal o difícil de caracterizar, como ocurre en el caso de los motores Lego Mindstorms EV3, cuyos comportamientos pueden variar por desgaste o diferencias de fabricación. Por tanto, se emplea un controlador borroso para obtener la corrección que se debe aplicar en la velocidad angular del robot (*yaw rate*, $\omega(t)$) de forma que se mantenga una trayectoria recta.

Este controlador se presenta en tres variantes: a) utilizando el modelo geométrico a partir de distancias recorridas, b) a partir de velocidades instantáneas y c) utilizando directamente un sensor de giro. Para sintonizar el controlador a cada una ellas se utilizan ganancias, lo que permite adaptar la salida del controlador sin modificar las funciones de pertenencia.

Diseño del controlador borroso

El controlador se concibe como un esquema genérico, aplicable a distintos casos de uso, por lo que su única variable de entrada es el error expresado en porcentaje. En la implementación propuesta este error de entrada corresponde siempre al error de orientación $\theta(t)_{error}$ y se denota $e_\theta(t)$.

De esta forma, la salida del controlador representa el porcentaje de corrección que debe aplicarse sobre la velocidad angular del robot, $\Delta\omega(t)$. Por ejemplo, si el controlador devuelve una salida del 30 % y la velocidad lineal, de avance o movimiento, deseada es de 25 unidades, la velocidad angular aplicada será el 30 % de dicho valor, es decir, 7.5 unidades.

Se definen funciones de pertenencia triangulares solapadas, tal y como se muestra en las Figuras 2 y 3, de modo que cualquier valor dentro del rango de entrada activa al menos un conjunto difuso y produce una salida.

Las reglas lingüísticas definidas para la fase de inferencia se presentan en la Tabla 1, donde $e_\theta(t)$ es el error en porcentaje de la orientación, esto es, la entrada definida, y $\Delta\omega(t)$ el porcentaje de corrección que debe aplicarse sobre la velocidad angular del robot, es decir, la salida definida. Para combinar las reglas activadas en una única salida, $\Delta\omega(t)$, el operador de agregación empleado es el máximo, que selecciona el mayor grado de pertenencia aportado por las reglas (Kozielski et al., 2024).

Tabla 1: Reglas lingüísticas del controlador borroso.

Condición en $e_\theta(t)$	**Acción en $\Delta\omega(t)$**
muy bajo	negativo fuerte
bajo	negativo suave
medio	nulo
alto	positivo suave
muy alto	positivo fuerte

Finalmente, la *defuzzificación* se realiza mediante el método del centroide (Ross, 2010) con 200 muestras, con el objetivo de conseguir suficiente resolución que asegure transiciones suaves entre las acciones de control.

El resumen de los parámetros principales para configurar el controlador se presenta en la Tabla 2 y los conjuntos borrosos junto con sus rangos en las Tablas 3 y 4, donde a y c son los vértices de la base del triángulo.

Tabla 2: Parámetros del controlador borroso.

Parámetro	**Valor o rango**
Entrada: $e_\theta(t)$ en %	[-100, 100]
Salida: corrección vel. angular en %, $\Delta\omega(t)$	[-100, 100]
Defuzzificación	Centroide
Operador de agregación	Máximo
Ganancia variante velocidad	1.0
Ganancia variante distancias	0.4
Ganancia variante sensor de giro	2.5

Tabla 3: Funciones de pertenencia definidas para la variable de entrada $e_\theta(t)$.

Conjunto	**Rango (a,b,c)**
muy bajo	(-100, -80, -40)
bajo	(-60, -30, 0)
medio	(-20, 0, 20)
alto	(0, 30, 60)
muy alto	(40, 80, 100)

Tabla 4: Funciones de pertenencia definidas para la variable de salida $\Delta\omega(t)$.

Conjunto	**Rango (a,b,c)**
negativo fuerte	(-100, -100, -50)
negativo suave	(-60, -30, 0)
nulo	(-20, 0, 20)
positivo suave	(0, 30, 60)
positivo fuerte	(50, 100, 100)

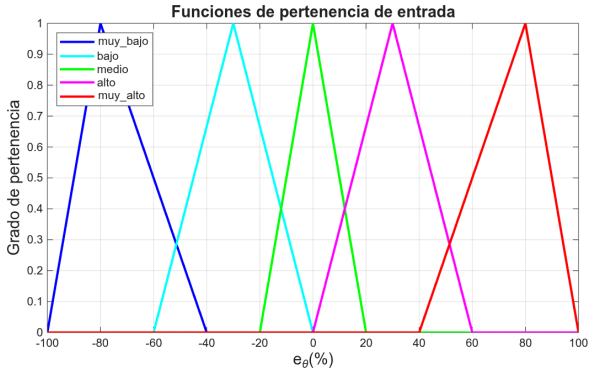

Figura 2: Funciones de pertenencia de entrada del controlador borroso.

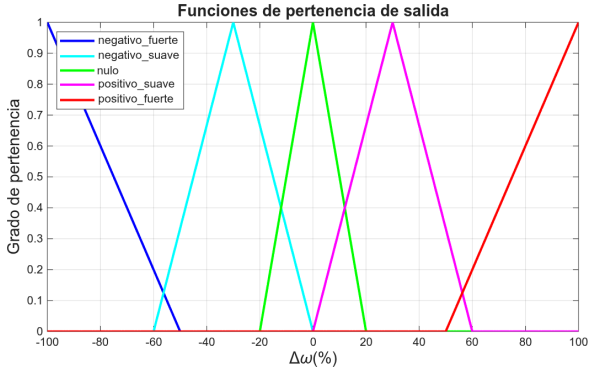

Figura 3: Funciones de pertenencia de salida del controlador borroso.

4. Control basado en algoritmos genéticos

El propósito de los algoritmos genéticos es optimizar parámetros de control mediante técnicas inspiradas en la evolución natural (Waysi et al., 2025), evitando la necesidad de una sintonización manual. En este trabajo se emplea un algoritmo genético (*GA*) para ajustar el parámetro K_p de un controlador proporcional encargado de corregir las desviaciones de trayectoria utilizando el sensor de giro. En este tipo de controlador la corrección aplicada sobre la velocidad angular del robot, $\Delta\omega(t)$ se calcula como la Ecuación (9),

$$\Delta\omega(t) = K_p \cdot e_\theta(t), \tag{9}$$

donde $e_\theta(t)$ es el error de orientación. En este caso no se plantea utilizar el modelo geométrico en ninguna de sus versiones ya que como se observa en los resultados del controlador borroso, Sección 5.1, el controlador que utiliza el sensor de giro presenta un considerable mejor desempeño.

El proceso evolutivo comienza con una población inicial de valores candidatos de K_p, que se evalúan en función de su desempeño en mantener la trayectoria recta del robot. Del conocimiento de *eduROS2* (Arambarri Calvo, 2025) y la dinámica del robot, se puede acotar el rango de valores posibles de la población, en este caso, [2.0, 9.0], ya que con ese mínimo el robot tarda en corregir y con ese máximo genera demasiadas oscilaciones.

Después, se aplican los operadores de *selección*, *cruce* y *mutación* para generar nuevas poblaciones, favoreciendo aquellos individuos con mejor rendimiento. De esta forma, el algoritmo converge hacia un valor óptimo de K_p que mejora la estabilidad del controlador proporcional. Cada individuo se evalúa por episodios de 4 segundos y cada generación se configura con una población de 6 individuos, estableciéndose un máximo de 5 generaciones.

A continuación se detalla el diseño de las operaciones propuestas.

Función de aptitud

Evalúa el rendimiento de cada individuo y es única para cada aplicación. Se ha diseñado para evaluar las oscilaciones (`osc`) que genera el individuo en estudio y el tiempo de recuperación (`TR`) necesario por el robot. Las oscilaciones se obtienen como la variación media entre salidas consecutivas del controlador, y el tiempo de recuperación como el número de muestras fuera del rango de error permitido, ±2°. La función de aptitud diseñada se calcula como:

$$fitness = w_{osc} \cdot \texttt{osc} + w_{TR} \cdot \texttt{TR}, \tag{10}$$

donde $w_{osc} = 0.2$ y $w_{TR} = 0.5$ son los pesos asignados a cada criterio. Como $w_{TR} > w_{osc}$, se le da mayor importancia al tiempo de recuperación. Los individuos con menor valor de *fitness* son considerados más aptos.

Selección

Tras evaluar todos los individuos de una generación, estos se ordenan en función de su idoneidad y se conservan los dos de menor valor, ya que son los dos más aptos.

Cruce

A partir de los individuos seleccionados se generan nuevos candidatos combinando los valores de K_p de los dos progenitores. El cruce implementado es de tipo interpolación (Herrera et al., 2004), donde el nuevo individuo (descendiente) se obtiene como una combinación lineal de los progenitores:

$$K_p^{desc.} = \beta \cdot K_p^{progen.\,1} + (1-\beta) \cdot K_p^{progen.\,2} \tag{11}$$

con $\beta \in [0, 1]$ elegido arbitrariamente en cada cruce. Este operador permite explorar nuevos valores en el rango delimitado por sus progenitores, favoreciendo la convergencia hacia regiones prometedoras del espacio de búsqueda.

Mutación

Para mantener diversidad en la población y evitar la convergencia prematura, se aplica un operador de mutación que introduce una pequeña variación aleatoria sobre el valor de K_p del individuo generado. La mutación se implementa añadiendo ruido uniforme (Corne and Lones, 2025) en el rango [-0.5, 0.5], con una probabilidad de aplicación del 10 %. De esta forma se asegura que el algoritmo pueda explorar nuevas soluciones fuera de las combinaciones lineales de los ascendientes, evitando estancamientos en óptimos locales.

5. Análisis de los resultados

Los experimentos muestran que los controladores propuestos logran mantener la trayectoria rectilínea en presencia de diferencias entre motores y perturbaciones externas. Las señales de error se representan en grados (º) respecto del tiempo.

5.1. *Control fuzzy*

La Figura 4 muestra la evolución de la señal de error para los controladores borrosos implementados en las tres aproximaciones presentadas: a) el basado en el modelo geométrico de velocidades (en color rojo), b) el basado en las posiciones de los motores (en color verde) y c) el que emplea directamente el sensor de giro (en color azul). Se aprecia claramente cómo la elección de la ganancia influye de manera decisiva en la estabilidad de la respuesta. En particular, el controlador que utiliza las posiciones de los motores presenta una oscilación significativamente mayor, lo que evidencia la necesidad de ajustar cuidadosamente la ganancia para evitar sobrecompensaciones. Aun así, la no linealidad de los motores hace que el modelo geométrico genere un error de orientación incluso cuando no existe un error real.

En contraste, el controlador con sensor de giro y una ganancia de 2.5 ofrece la mejor respuesta, ya que utiliza información que refleja con mayor precisión el estado real del robot en cada instante. Esta señal apenas oscila y mantiene una trayectoria estable, corrigiendo las desviaciones de forma rápida y eficaz.

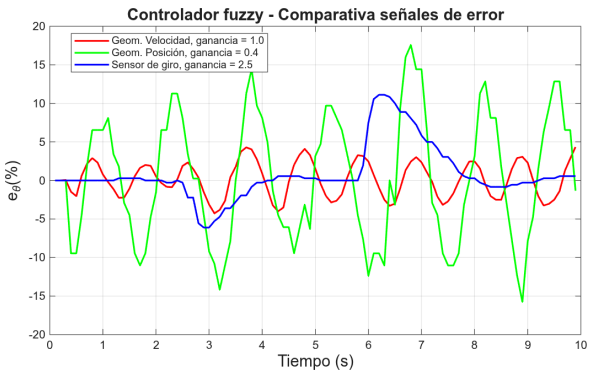

Figura 4: Comparativa de señales de error del controlador *fuzzy* para las 3 aproximaciones: a) en rojo, b) en verde, c) en azul.

5.2. *Algoritmos genéticos*

La Tabla 5 muestra la evolución de la población en distintas generaciones, donde se aprecia cómo los valores de K_p tienden progresivamente hacia regiones más estables y con menor *fitness*. Este comportamiento refleja la capacidad del algoritmo para explorar el espacio de búsqueda y converger hacia parámetros que mejoran la respuesta del controlador. En particular, se observa que valores intermedios de K_p ofrecen un equilibrio adecuado entre rapidez de corrección y ausencia de oscilaciones excesivas, estando en $K_p = 5.31$ el óptimo logrado.

En la Figura 5 se representa la señal del error respecto al tiempo, utilizando los valores poblacionales límite 2.0 y 9.0 junto con el valor óptimo obtenido 5.31. Se observa que este último (curva verde) constituye efectivamente una solución adecuada, ya que proporciona una respuesta relativamente rápida con menor nivel de oscilación. En cambio, la señal correspondiente a $K_p = 2.0$ (curva roja) muestra una corrección demasiado lenta, mientras que la de $K_p = 9.0$ (curva azul) presenta un exceso de oscilaciones, generando picos más pronunciados y elevados que los observados en la señal verde correspondiente a $K_p = 5.31$.

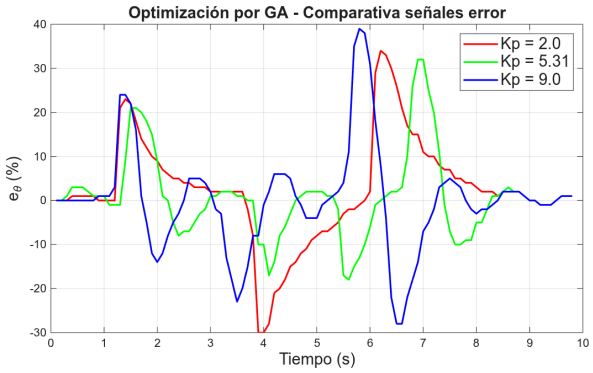

Figura 5: Comparativa de señales de error con K_p optimizado por un algoritmo genético (*GA*).

Tabla 5: Evolución del algoritmo genético en las distintas generaciones.

Gen.	Indiv.	K_p	Fitness
0	0	8.1319	10.7280
0	1	6.6784	8.9099
0	2	4.0937	12.3187
0	3	5.4566	11.2959
0	4	5.7875	11.3842
0	5	5.0053	14.5987
1	0	6.6784	12.0384
1	1	8.1319	9.8214
1	2	5.2523	10.1088
1	3	5.3148	7.3046
1	4	5.5053	11.2158
1	5	7.8817	11.3902
2	0	5.3148	6.7274
2	1	8.1319	9.1407
2	2	5.3319	8.6478
2	3	5.4473	7.4172
2	4	5.2687	9.8123
2	5	7.6908	13.5509
3	0	8.1319	10.1570
3	1	5.3148	9.1667
3	2	7.9424	13.0775
3	3	5.2710	11.1285
...			

5.3. *Comparativa entre las diferentes estrategias*

La Figura 4 para el controlador borroso y la Figura 5 para $Kp = 5.31$, ambos para la aproximación c), es decir, utilizando el sensor de giro, muestran los resultados de los mejores controladores obtenidos. A simple vista, el controlador *fuzzy*

parece responder mejor a las perturbaciones, pero no se puede confirmar con dichos gráficos porque el controlador proporcional tradicional está haciendo frente a perturbaciones mayores.

Por tanto, en la Figura 6 se presenta la comparativa del controlador borroso con sensor de giro y el controlador proporcional con sensor de giro sintonizado con algoritmos genéticos, ambos sometidos a perturbaciones similares. Las señales de error se han desfasado para comparar con mayor claridad, y aunque el controlador borroso, señal azul, se enfrenta a perturbaciones algo mayores, muestra la misma estabilidad que el controlador proporcional, siendo este un claro indicador de la robustez del controlador basado en lógica difusa desarrollado.

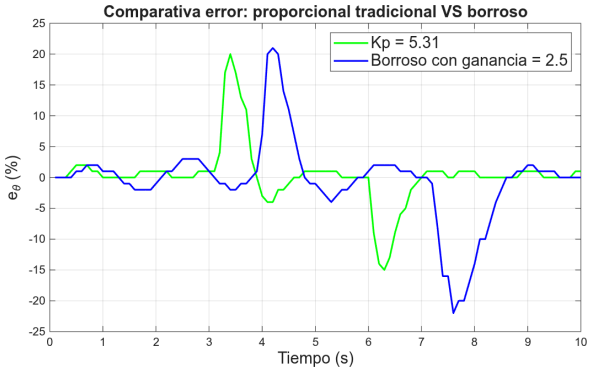

Figura 6: Comparativa de los controladores propocional tradicional y borroso.

6. Conclusiones

El trabajo presentado demuestra la viabilidad y el valor didáctico de emplear un robot educativo de bajo coste como plataforma para la enseñanza práctica de técnicas avanzadas de control en el ámbito universitario. La experimentación directa con el Lego Mindstorms EV3 permite al alumnado enfrentarse a problemas reales de control, caracterizados por incertidumbres, perturbaciones y limitaciones de los modelos teóricos, aspectos difíciles de reproducir mediante simulación exclusivamente.

Desde un punto de vista educativo, los resultados ponen de manifiesto la utilidad de combinar enfoques de control clásico e inteligente en un mismo entorno experimental. El controlador borroso facilita la introducción de conceptos de control no lineal y razonamiento basado en reglas lingüísticas, mientras que la optimización mediante algoritmos genéticos permite abordar de forma sistemática la sintonización de parámetros y reflexionar sobre la definición de criterios de rendimiento. Esta combinación favorece una comprensión más profunda de los compromisos inherentes al diseño de sistemas de control.

Asimismo, la comparación entre estimaciones basadas en modelos geométricos y medidas directas obtenidas a partir del sensor de giro resulta especialmente enriquecedora desde el punto de vista formativo, al evidenciar las limitaciones de la modelización ideal y la importancia de la instrumentación en sistemas reales. El uso de ganancias externas en los controladores borrosos permite, además, reutilizar el diseño del motor de inferencia en distintos contextos docentes, simplificando su integración en prácticas de laboratorio.

En conjunto, la experiencia presentada contribuye a reducir la brecha entre teoría y práctica en la enseñanza de la automática, incrementa la motivación del alumnado y facilita la adquisición de competencias prácticas en control y robótica. Por todo ello, se considera que este enfoque constituye una alternativa sólida y replicable para la docencia en asignaturas de automática y control, complementando de manera eficaz el uso tradicional de herramientas de simulación.

Las líneas futuras del artículo se centran en tres direcciones principales: la simulación mediante herramientas como Gazebo para generar perturbaciones homogéneas y controladas, sintonización de un controlador PID mediante algoritmos genéticos y desarrollo de controladores neuronales, así como la exploración de enfoques basados en aprendizaje por refuerzo.

Agradecimientos

Este trabajo ha sido financiado por el Gobierno Vasco: proyectos Ref. SENDOA KK-2025/00102 y Ref. AURRERA KK-2024/0024.

Referencias

Arambarri Calvo, J., 2025. eduros2: robótica educativa lego con ros2. Repositorio en GitHub.
URL: `https://github.com/arambarricalvoj/eduROS2/tree/ROSConEs25`

Arambarri Calvo, J., Arambarri Calvo, J., 2024. Explorando la Robótica: Guía para First Lego League y World Robot Olympiad. Amazon KDP, España.
URL: `https://www.amazon.es/dp/B0DJDJM1J8`

Corne, D., Lones, M. A., 2025. Evolutionary algorithms. In: Handbook of Heuristics. Springer Nature Switzerland, Cham, pp. 125–146.

Gutiérrez Reina, D., Tapia Córdoba, A., Rodríguez del Nozal, Á., 2020. Algoritmos Genéticos con Python: Un enfoque práctico para resolver problemas de ingeniería. Marcombo, Barcelona.

Herrera, F., Lozano, M., Sánchez, A. M., 2004. Operadores de cruce con múltiples descendientes para algoritmos genéticos con codificación real: Estudio experimental. In: Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados (MAEB).

Kozielski, M., Prokopowicz, P., Mikołajewski, D., 2024. Aggregators used in fuzzy control—a review. Electronics 13 (16), 3251.
DOI: `10.3390/electronics13163251`

Lego Education, 2013. Lego Mindstorms EV3. Robotics kit and programming platform.

Macenski, S., Foote, T., Gerkey, B., Lalancette, C., Woodall, W., 2022. Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics 7 (66), eabm6074.
DOI: `10.1126/scirobotics.abm6074`

Rada-Vilela, J., 2018. The fuzzylite libraries for fuzzy logic control.
URL: `https://fuzzylite.com`

Ross, T. J., 2010. Fuzzy Logic with Engineering Applications, 3rd Edition. Wiley.

Siegwart, R., Nourbakhsh, I., Scaramuzza, D., 2011. Introduction to Autonomous Mobile Robots. MIT Press.

Wang, C., Xu, X., Kao, Y., Xia, H., 2025. Analysis and Design for Fuzzy Systems. Springer.

Waysi, D., Ahmed, B. T., Ibrahim, I. M., 2025. Optimization by nature: A review of genetic algorithm techniques. The Indonesian Journal of Computer Science 14 (1).

Zadeh, L., 1965. Fuzzy logic. Fuzzy Sets and Systems, 100–106.

Zhang, M., Wan, Y., 2023. Improving learning experiences using lego mindstorms ev3 robots in control systems course. International Journal of Electrical Engineering & Education 60 (4), 329–351.
DOI: `10.1177/00207209211055343`