Diseño, implementación y control de un prototipo mecatrónico de péndulo invertido móvil

David Leonardo Moyano Quintero y Nicolás Barreto Saavedra

Trabajo de Grado para optar el título de Ingeniero Mecánico

Modalidad Trabajo de Investigación

Directora

Yennifer Yuliana Ríos Diaz

Ingeniera Mecatrónica. MS. PhD.

Codirectora

Diana Katheryn Poveda Rodríguez

Ingeniera Electricista. MSc.

Universidad Industrial de Santander

Facultad de Ingenierías Fisicomecánicas

Escuela de Ingeniería Mecánica

Bucaramanga

2025

Agradecimientos

En primer lugar, quiero expresar mi más profundo agradecimiento a Dios. Ha sido mi refugio en los momentos de incertidumbre y mi guía en cada decisión tomada durante este largo proceso. Su infinita sabiduría, fortaleza y amor han sido fundamentales para llegar hasta aquí.

A mi familia, mi pilar fundamental, les debo todo. Gracias a mis padres por su amor incondicional, sus sacrificios y su apoyo constante en cada etapa de mi formación. A mis hermanos, por sus palabras de aliento y por estar siempre a mi lado, animándome a seguir adelante.

A mis amigos, quienes con su compañía, consejos y confianza ha sido un refugio en los momentos difíciles y una fuente inagotable de alegría y motivación.

A mi directora de proyecto, Yennifer Yuliana Ríos Diaz, le extiendo mi gratitud por su orientación, paciencia y compromiso. Su experiencia y dedicación fueron cruciales para el desarrollo del proyecto. Asimismo, agradezco a mi codirectora, Diana Katheryn Poveda Rodríguez, por su valiosa colaboración, sus sugerencias y el apoyo brindado.

Finalmente, quiero agradecer a mi compañero de tesis, por su trabajo conjunto, su compromiso y por compartir esta experiencia. Juntos enfrentamos retos y celebramos logros, y estoy profundamente agradecido por su compañerismo y esfuerzo.

Nicolas Barreto Saavedra

Agradecimientos

En primer lugar, a Dios por darme salud, sabiduría y sobre todo mucha resiliencia para enfrentar los retos presentados a lo largo del desarrollo de este proyecto y durante mi carrera.

A mis padres, José Moyano y María Quintero, por estar a mi lado en cada paso del camino, por ser siempre mi mano derecha y sobre todo mis guías en este proceso, por no dejarme desfallecer y creer en mí.

A mi hermano Juan Moyano por cada palabra de apoyo y por estar siempre presente en todo momento.

A Juana por el apoyo que me brindo a lo largo de la carrera, por ser mi compañera y estar siempre conmigo, por el apoyo, por cada consejo, por cada palabra y por cada abrazo y cada detalle que me impulsaban a seguir a delante.

A nuestra directora, Yennifer Yuliana Ríos por brindarnos su conocimiento y apoyo durante el desarrollo de este proyecto.

David Leonardo Moyano Quintero

Tabla de Contenido

Pág.

Lista de Figuras

Pág.

Lista de Apéndices

Pág.

Resumen

Título: Desarrollar un Prototipo de Péndulo Invertido Móvil Mediante el Diseño, la Construcción y la Implementación de Diferentes Métodos de Control, Incluyendo la Lógica Difusa, Retroalimentación de Estados, Retroalimentación de Estados con Observador de Luenberger.

Autor: David Leonardo Moyano Quintero y Nicolas Barreto Saavedra

Palabras Clave: Prototipo, péndulo invertido, control automático, mecatrónica, Diseño e implementación.

Descripción: El péndulo invertido móvil es un sistema clásico en ingeniería de control, ampliamente utilizado para estudiar la estabilización de sistemas no lineales y la implementación de estrategias de control avanzadas. Este proyecto tiene como objetivo desarrollar un prototipo funcional de péndulo invertido móvil, implementando y comparando diferentes métodos de control: Lógica difusa, retroalimentación de estados y retroalimentación de estados con observador de Luenberger. El diseño del sistema incluye tanto la construcción mecánica como la implementación electrónica y de software, utilizando Arduino como plataforma de control.

El problema central radica en la inestabilidad inherente del sistema, lo que dificulta mantener el péndulo en posición vertical. Para abordarlo, se propone una metodología que combina simulaciones en MATLAB con pruebas practicas del prototipo. Las simulaciones demostraron que los controladores diseñados son capaces de estabilizar bajo condiciones ideales. Sin embargo, al implementarlos en el prototipo, se observaron discrepancias significativas debido a perturbaciones externas y limitaciones del hardware, lo que afectó la estabilidad a largo plazo.

Los resultados obtenidos evidencian que, aunque el sistema logra estabilidad temporal, es necesario mejorar la robustez de los controladores para garantizar un funcionamiento optimo en condiciones reales. Este trabajo aporta una base sólida para futuras investigaciones de control automático, destacando la importancia de considerar perturbaciones y limitaciones prácticas en el diseño de sistemas de control.

Trabajo de grado. Facultad de Físico mecánicas. Escuela de Ingeniería Mecánica

Abstract

Title: Development of a Mobile Inverted Pendulum Prototype Through Design, Construction, and Implementation of Different Control Methods, Including Fuzzy Logic, State Feedback, and State Feedback with Luenberger Observer.

Authors: David Leonardo Moyano Quintero and Nicolas Barreto Saavedra

Keywords: Prototype, inverted pendulum, automatic control, mechatronics, design and implementation.

Description: The mobile inverted pendulum is a classic system in control engineering, widely used to study the stabilization of nonlinear systems and the implementation of advanced control strategies. This project aims to develop a functional prototype of a mobile inverted pendulum, implementing and comparing different control methods: Fuzzy logic, state feedback, and state feedback with a Luenberger observer. The system design includes mechanical construction as well as electronic and software implementation, using Arduino as the control platform.

The central problem lies in the system's inherent instability, making it challenging to keep the pendulum in an upright position. To address this, a methodology combining MATLAB simulations with practical prototype testing is proposed. The simulations demonstrated that the designed controllers are capable of stabilization under ideal conditions. However, when implementing them in the prototype, significant discrepancies were observed due to external disturbances and hardware limitations, which affected long-term stability.

The results obtained show that, although the system achieves temporary stability, it is necessary to improve the robustness of the controllers to ensure optimal performance under real conditions. This work provides a solid foundation for future research in automatic control, highlighting the importance of considering disturbances and practical limitations in control system design.

Thesis project. Faculty of Physical and Mechanical Sciences. School of Mechanical Engineering.

Introducción

El control del péndulo invertido móvil no solo es un desafío clásico en la teoría de control, sino que tiene aplicaciones prácticas que impactan áreas como la robótica móvil y los vehículos autónomos, con la capacidad de revolucionar sistemas de transporte y automatización. De hecho, investigaciones recientes han demostrado que los avances en el control de sistemas como el péndulo invertido pueden mejorar significativamente la estabilidad y el desempeño de vehículos autónomos, con implicaciones directas en la reducción de accidentes y optimización de rutas (Liu & al., 2019; Pimentel Medina, 2018).

El proyecto estudia un sistema de control basado en un péndulo invertido montado sobre una base móvil (como un carro o plataforma deslizante). El objetivo principal es mantener el péndulo posición vertical equilibrado , mientras la base se desplaza horizontalmente. Este sistema es un desafío clásico en teoría de control debido a dos características clave: (1) su dinámica es altamente no lineal, y (2) es un sistema subactuada, pues el numero de variables a controlar (ángulo del péndulo + posición de la base) supera los grados de libertad de actuación (solo se puede mover la base). Por esta razón, el péndulo invertido móvil se utiliza como banco de pruebas para validar estrategias de control avanzado (Ogata, 2010).

Diversos estudios han explorado métodos para controlar el sistema de péndulo invertido móvil. Por ejemplo, (Apaza Cruz, 2017) diseñó un prototipo funcional de este sistema, utilizando herramientas de código abierto como Arduino y Processing. Implementó técnicas de control PID (Proporcional, Integral y Derivativo) y filtros de Kalman, logrando un equilibrio dinámico

eficiente mediante métodos basados de Lagrange y Taylor. Este enfoque innovador demostró ser una alternativa efectiva frente a metodologías tradicionales.

Por otro lado, Triviño Macias (2020) desarrolló un modelo matemático no lineal del sistema y diseñó dos controladores PID para regular el péndulo y la base móvil. Mediante simulaciones 3D y el método ensayo y error, se optimizaron las ganancias (Kp,Ki,Kd), logrando un comportamiento optimo al excluir la componente integral. Aunque los resultados fueron satisfactorios, el estudio destacó la necesidad de implementar el sistema en un entorno físico para validar su aplicabilidad práctica.

No obstante, existen muchos más tipos de controladores aplicadas en los diferentes sistemas para llevar a cabo un control, entre ellos está el de retroalimentación de estados, en el que Palacios evalúa y compara con un controlador PID. Los resultados indican que los controladores con realimentación de variables de estado tienen un tiempo de establecimiento mucho menor que un PID (Palacios, 2017).

Si bien, los estudios mencionados se han centrado principalmente en aspectos teóricos, lo que limita su aplicabilidad practica en sistemas altamente inestables como el péndulo invertido móvil. Por ende, el propósito de este trabajo es desarrollar un prototipo de péndulo invertido móvil mediante el diseño, la construcción y la implementación de diferentes técnicas de control, incluyendo la lógica difusa, retroalimentación de estados y retroalimentación de estados con observador Luenberger y a su vez, comparar las técnicas de control nombradas anteriormente

1. Planteamiento del problema

1.1 Descripción del problema

El control de sistemas inestables y no lineales, como el péndulo invertido, es un desafío fundamental en ingeniería de control, con aplicaciones prácticas en áreas como la robótica, la industria automotriz y la aeronáutica (Astrom & Murray, 2021). Estos sistemas (péndulo invertido móvil) resultan particularmente complejos de controlar debido a su naturaleza inestable y su alta sensibilidad a perturbaciones externas, como demuestran estudios clásicos en teoría de control (Slotine & Li, 1991; Khalil, 2015) . Por ejemplo, en la industria de vehículos autónomos, la estabilización dinámica es esencial para garantizar la seguridad y eficiencia, pero los métodos de control tradicionales, como el control PID , a menudo no son suficientes para manejar linealidades y perturbaciones presentes en los sistemas (Li, et al., 2019) .

Un caso relevante es el de los segways y vehículos de movilidad personal, que utilizan principios similares al péndulo invertido para mantener el equilibrio. Según Wang (Wang, 2020), estos sistemas enfrentan desafíos significativos en entornos dinámicos, como cambios bruscos en la superficie de la rodadura o variaciones en la carga. Esto ha llevado a la exploración de nuevas técnicas, como la lógica difusa y la retroalimentación de estados, que ofrecen mayor robustez y adaptabilidad (Chen & Liu, 2017).

Por otro lado, a nivel nacional, se evidencia que la mayoría de las instituciones de educación superior en Colombia carecen de herramientas necesarias para el estudio de estos sistemas dinámicos. Según el Ministerio de Educación Nacional (MEN) y el observatorio Colombiano de Ciencia y Tecnología (OCyT), revelan que la infraestructura disponible en la mayoría de las instituciones de educación superior, no alcanza a cubrir los requerimientos básicos

para la investigación y enseñanza de sistemas dinámicos complejos (MEN, 2020; OCyT, 2021). Esta carencia se debe, en gran medida, a los altos costos asociados con la adquisición y fabricación de prototipos avanzados. Estudios recientes, como el de Garcia et al.(2022), destacan que los costos de desarrollo y mantenimiento de los sistemas de control avanzados, como los basados en la logica difusa y observadores de estado, pueden ser prohibitivos para instituciones con recursos limitados.

En este contexto, surge la necesidad de desarrolar un prototipo de pendulo invertido que sea accesible y de bajo costo, pero que permita la implementacion de metodos de control avanzados, como la logica difusa, la retroalimentacion de estados y la retroalimentacion de estados con observador de Luenberger. Este proyecto no solo contribuira en especial a la formacion de estudiantes de ingenieria de la Universidad Industrial de Santander (UIS), si no que tambien servira como una plataforma de investigacion para explorar tecnicas de control innovadoras en sistemas no lineales e inestables.

2. Objetivos

2.1 Objetivo general

Desarrollar un prototipo de péndulo invertido móvil mediante el diseño, la construcción y la implementación de diferentes métodos de control, incluyendo Lógica Difusa, Retroalimentación de Estados y Retroalimentación de Estados con Observador de Luenberger.

2.2 Objetivos específicos

- Diseñar las diferentes partes del sistema mecánico del péndulo invertido móvil con un brazo de 0,28 m que soporta una masa hasta de 0,55 Kg a partir de un modelo construido en un software de Diseño Asistido por Computadora (CAD).
- Sintetizar tres estrategias de control (Lógica difusa, Retroalimentación de estados y Retroalimentación de estados con observador de Luenberger) que permitan regular el ángulo del péndulo y la posición del carro haciendo uso del software Matlab.
- Comparar experimentalmente el desempeño de las tres estrategias de control en términos de tiempo de estabilidad, tiempo de respuesta y robustez ante perturbaciones.

3. Estado del arte

El péndulo invertido es un sistema clásico usado en la teoría de control debido a su naturaleza inestable y no lineal, lo que lo convierte en un banco de pruebas ideal para evaluar estrategias de control en diversas ingenierías como la mecatrónica y robótica. Su estudio ha permitido el desarrollo y perfeccionamiento de técnicas como el control PID, LQR (Linear Quadratic Regulator), lógica difusa y observadores de estado, entre otros. Estas técnicas han demostrado ser eficaces para abordar los desafíos que plantea este sistema, como la inestabilidad intrínseca y la sensibilidad a perturbaciones externas. En este contexto, dos proyectos destacan por sus contribuciones significativas al campo: el diseño de un controlador difuso (Alarcón, 2014) y

el control de un péndulo invertido sobre un carro móvil (Guerra Aranda, 2019). Estos trabajos no solo proporcionan metodologías y conclusiones valiosas, sino que también sirven como base para el desarrollo de un prototipo de péndulo invertido móvil, el cual integra el diseño, construcción e implementación de diferentes métodos de control, incluyendo la lógica difusa, retroalimentación de estados y retroalimentación de estados con observador de Luenberger. A continuación, se presenta una revisión bibliográfica detallada que analiza los enfoques metodológicos, conclusiones y recomendaciones de estos estudios, los cuales guían y justifican el desarrollo del proyecto actual.

El control del péndulo invertido móvil ha sido abordado desde diversas perspectivas, destacándose el uso de técnicas avanzadas como la lógica difusa y los observadores de estado. En primer lugar, el control difuso, basado en la lógica difusa, ha demostrado ser una herramienta poderosa para manejar sistemas no lineales y complejos. A diferencia de los controladores tradicionales, un controlador difuso utiliza reglas lingüísticas y funciones de pertenencia para mapear entradas y generar una acción de control, lo que lo hace especialmente útil en sistemas como el péndulo invertido. Este enfoque fue ampliamente explorado por Alarcón Sánchez (2014), quien diseñó un controlador difuso para estabilizar un péndulo invertido. Su trabajo resaltó la capacidad de esta técnica para manejar perturbaciones y variaciones en las condiciones del sistema, superando en robustez a métodos tradicionales como el control PID. Sin embargo, Alarcón también sugirió que la integración de la lógica difusa con otras técnicas, como los observadores de estado, podría mejorar aún más la precisión en sistemas dinámicos.

Por otro lado, el observador de Luenberger ha sido ampliamente utilizado para estimar los estados de un sistema dinámico cuando no es posible medir directamente todas las variables de estado. Esta herramienta introduce una retroalimentación basada en el error entre las salidas

estimadas y las reales, lo que mejora la precisión del control. Su aplicación en sistemas como el péndulo invertido móvil puede optimizar el desempeño del controlador al proporcionar estimaciones precisas de estados no medidos. Este concepto fue aplicado por García (1993) y ha sido retomado en estudios más recientes, como el de Guerra Aranda (2019), quien abordó el control de un péndulo invertido montado en un carro móvil. En su trabajo, Guerra Aranda implementó controladores clásicos (PID y LQR) y evaluó su desempeño mediante simulaciones y experimentos prácticos. Aunque ambos controladores mostraron un buen desempeño, su efectividad dependió en gran medida de la calibración precisa de los parámetros. El controlador LQR demostró ser más robusto ante perturbaciones en comparación con el PID, aunque ambos requirieron ajustes manuales para mantener la estabilidad. Estas limitaciones resaltan la necesidad de desarrollar herramientas adaptativas que permitan ajustar los parámetros en tiempo real, lo cual será considerado en el proyecto actual.

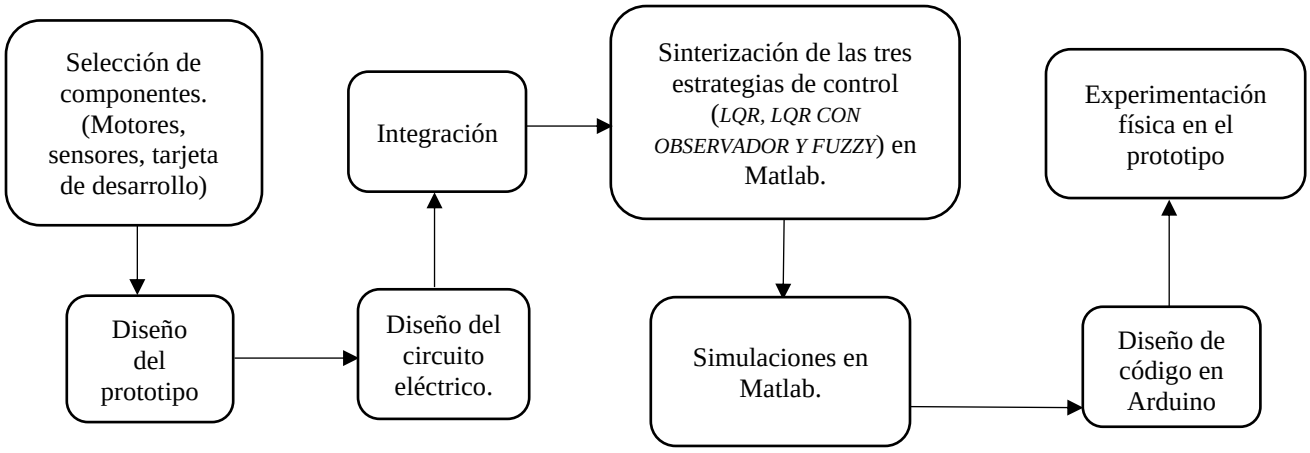
Además de las técnicas de control, es crucial considerar las perturbaciones a las que está sujeto el sistema de péndulo invertido móvil. Estas incluyen fricción, fuerza del viento, vibraciones mecánicas y errores en los sensores, las cuales afectan tanto la estabilidad como la precisión del sistema. Tanto Alarcón Sánchez (2014) como Guerra Aranda (2019) destacan la importancia de diseñar controladores robustos capaces de mitigar estos efectos. Por ejemplo, Alarcón demostró que el controlador difuso es particularmente efectivo para manejar perturbaciones, mientras que Guerra Aranda resaltó la necesidad de ajustes dinámicos en controladores clásicos para mejorar su robustez. Estas consideraciones serán fundamentales en el desarrollo del prototipo, ya que garantizarán su funcionamiento en entornos reales.

La revisión bibliográfica realizada permite identificar las principales contribuciones y brechas en el campo del control del péndulo invertido móvil. Por un lado, el trabajo de Alarcón Sánchez (2014) destaca la efectividad de la lógica difusa para manejar sistemas no lineales y perturbaciones, mientras que el estudio de Guerra Aranda (2019) enfatiza la necesidad de ajustes dinámicos en controladores clásicos como el PID y el LQR. Estas contribuciones justifican el desarrollo de un prototipo que combine la lógica difusa, la retroalimentación de estados y el observador de Luenberger, permitiendo una evaluación comparativa y contribuyendo al avance en el control de sistemas mecatrónicos. Este enfoque híbrido busca superar las limitaciones de los métodos tradicionales, ofreciendo una solución más eficiente y adaptable para aplicaciones prácticas. Además, la consideración de perturbaciones y la implementación de herramientas adaptativas serán clave para garantizar la robustez y precisión del sistema en entornos reales. En resumen, este proyecto no solo se basa en los avances previos, sino que también propone innovaciones que podrían tener un impacto significativo en el campo del control automático.

4. Metodología

Figura 1.

Diagrama de flujo del proceso metodológico



4.1 Diseño del péndulo invertido móvil

4.1.1 Selección de componentes

El sistema de péndulo invertido móvil cuenta con dos partes fundamentales, el carro y el péndulo, en el carro se encuentra el circuito eléctrico, los motores con sus ruedas y el sensor que mide la distancia entre el objeto mas cercano y el carro, por otro lado, en el péndulo se encuentra el sistema que hace que el péndulo solo tenga movimiento rotacional en un plano y el sensor que mide el ángulo del péndulo con respecto a la vertical.

El diseño del carro y de los soportes de los sensores se realizó teniendo en cuenta las dimensiones de estos, por ende el primer paso fue seleccionar los componentes a utilizar, los criterios más relevantes para la selección de los diferentes componentes fueron su disponibilidad y su costo es por eso que se seleccionaron los componentes electrónicos de mejor disponibilidad y asequibilidad para no elevar el costo del prototipo, se escogió un sensor ultrasónico HC- SR04 ya que es uno de los sensores más comunes y funcionales que se encuentran en el mercado, un giroscopio acelerómetro MPU6050 que es de los más completos en el mercado y de bajo costo, se seleccionaron los motores DC N20 ya que cuentan con buena precisión y poseen buen torque lo cual es necesario para el prototipo, para controlar estos motores un controlador DC TB6612FNG para dos motores, de fácil acceso y de bajo costo, en cuanto a la tarjeta de desarrollo se optó por una ESP32S-WIFI ya que es una tarjeta de bajo coste y además posee características como la multitarea que reemplaza la necesidad de usar varias tarjetas de desarrollo para diferentes tareas, para alimentar los motores se optó por utilizar una batería LIPO TURNIGY de 7.4 V ya que al operar con una intensidad de corriente de 1000 mAh la convierte en la más óptima para este caso,

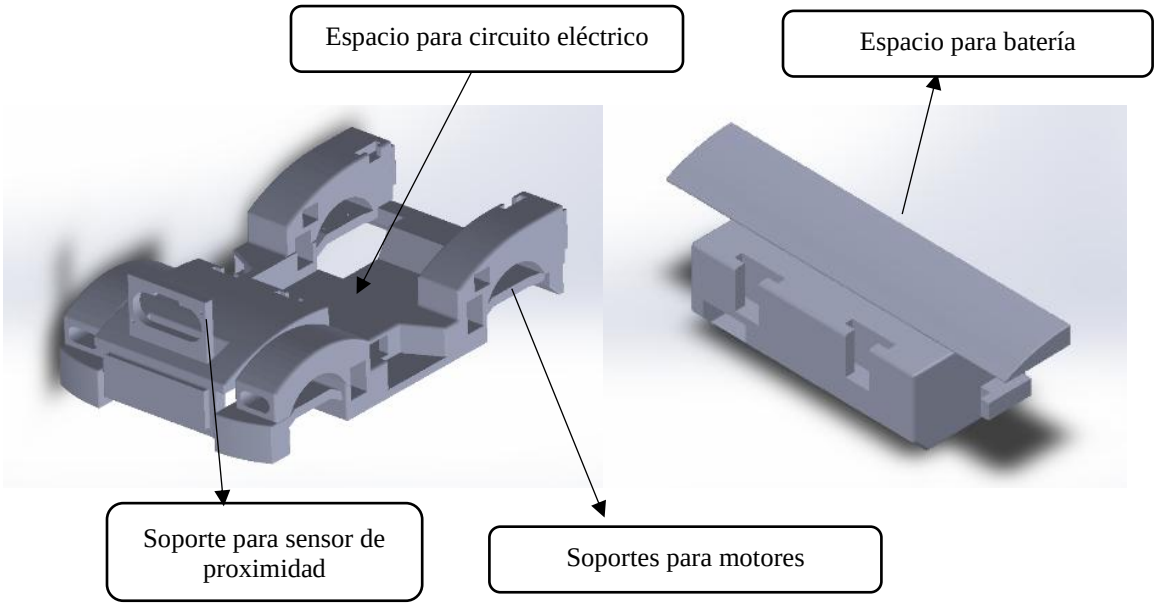
además de esto se usaron bornes de conexión , una placa universal y cable eléctrico. (Ver apéndice B)

4.1.2 Rediseño del carro

Como resultado de una búsqueda bibliográfica, se decidió tomar como base un diseño de un péndulo invertido móvil (Celso & Cutipa, 2020), se modifico y se rediseño en torno a los componentes antes seleccionados.

Figura 2.

Carro y espacio para batería modificados.



En la figura 2a se observa el carro, este cuenta con el espacio para el circuito eléctrico, los soportes para los motores, el soporte para el sensor de proximidad y en la parte posterior del carro se encuentra el espacio en donde encaja la pieza para la batería que tiene la forma de un alerón de automóvil (figura 2b), se realizaron cambios en las dimensiones de los soportes de los motores y

se modificó también el espacio para los componentes del circuito eléctrico haciéndolo más pequeño para optimizar espacio y costos.

Se modifico el espacio requerido para ubicar la batería dentro del alerón teniendo en cuenta las dimensiones de la batería seleccionada.

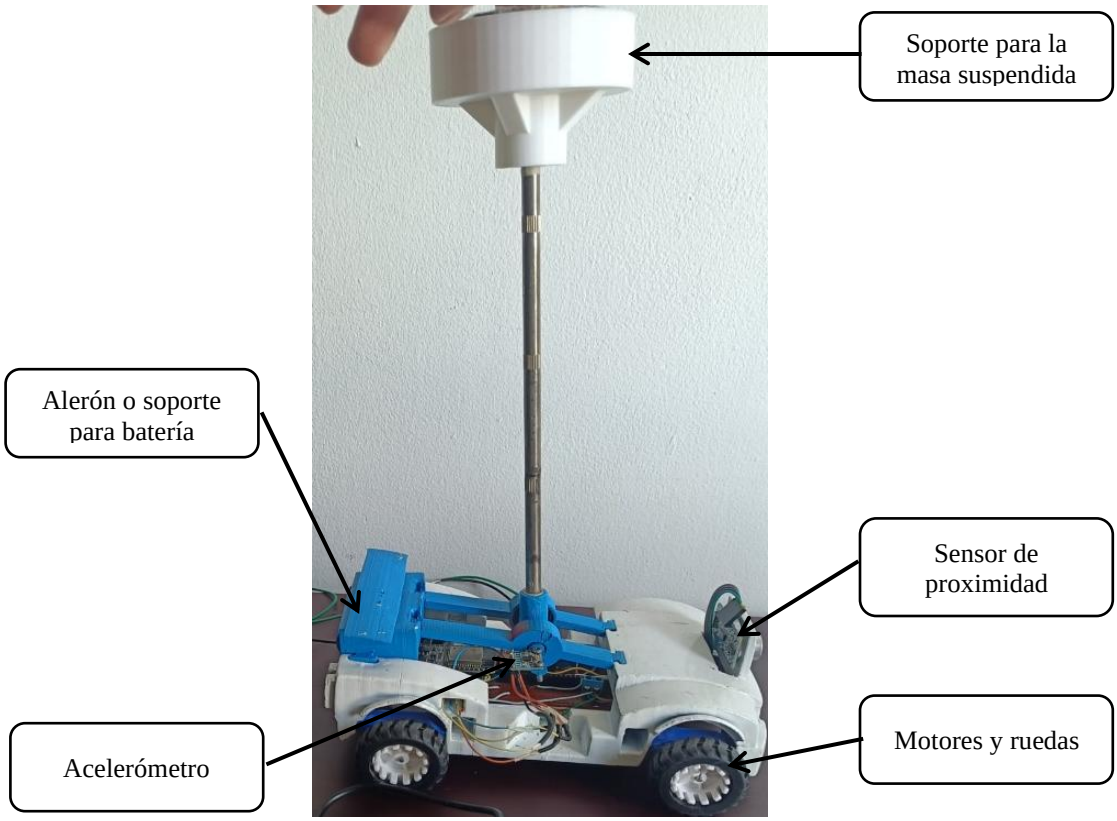
Para el péndulo, se utilizó una varilla de acero con medidas de 27.5 cm de largo y 7 mm de diámetro. Además, para sostener el peso sobre el péndulo, se diseñó un soporte de 8 cm de diámetro que encaja en la parte superior del péndulo.

Este diseño, con todas las modificaciones realizadas, se modeló en el software CAD SOLIDWORKS, como se observa en las imágenes ubicadas en el apéndice A. Finalmente, se fabricó mediante impresión 3D en material PLA (ácido poliláctico).

Al ensamblar el prototipo se generaron perturbaciones que se derivaron de fallas de fabricación, para el caso de las partes que se imprimieron 3D en un 80% presentaron modificaciones en sus dimensiones por lo cual hubo que realizarles procesos como ranurado o lijado, estas imperfecciones en su fabricación fueron las que ocasionaron perturbaciones como el desfase de 1 grado ya que el soporte del sensor MPU6050 no quedo totalmente horizontal por ende al calibrar el sensor el ángulo de este no correspondía exactamente con el del péndulo y se obtuvo una diferencia de casi 1 grado (0.9grados), aunque se tuvo en cuenta las tolerancias con las que trabajaba la impresora 3D en las que se fabricaron las diferentes piezas hubo que realizar varios ranurados para que las piezas encajaran perfectamente, al usar dos rodamientos uno para cada brazo que soporta el péndulo se logró obtener un movimiento muy fluido del péndulo y poca resistencia por fricción.

Figura 3

Prototipo ensamblado



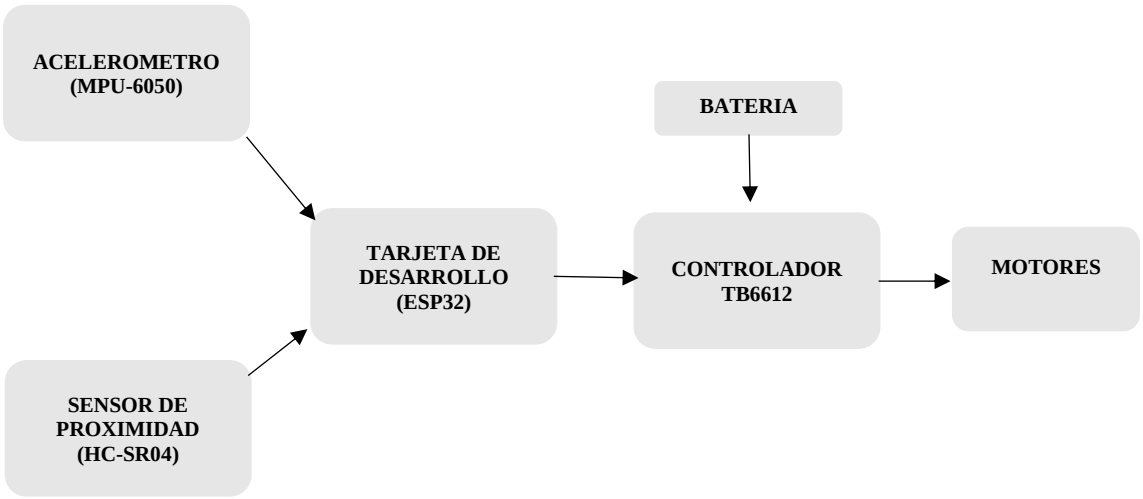
En la figura 3 se observa el prototipo completamente ensamblado con los sensores, los motores, el péndulo, el soporte para la masa suspendida y sus demás componentes.

Las especificaciones finales del prototipo fueron; masa del carro (M) de 0.507 kg, masa suspendida (m) de 0.305 kg, longitud del péndulo de 0.275 m, las dimensiones del carro fueron 0.22 m de largo, 0.13 m de ancho y 0.07 m de alto, la altura final con el péndulo y la masa suspendida fue de 0.36m.

4.1.3 Diseño del circuito eléctrico

Figura 4

Diagrama de bloques del circuito eléctrico.



En la figura 4 se ilustra el diagrama de bloque del circuito eléctrico donde se resumen las conexiones de los diferentes componentes del prototipo.

El circuito eléctrico se diseñó en torno a la tarjeta de desarrollo ESP32, específicamente el modelo ESP32S-WIFI. Esta tarjeta, al igual que los sensores, se alimenta a través del cable de datos conectado al computador.

La batería solo suministra energía a los motores mediante un controlador para motores de corriente directa (DC), diseñado para recibir señales tipo PWM.

Teniendo una batería de 7.4 V y 1000 mAh, se decidió incorporar un circuito regulador de voltaje antes de alimentar el controlador de los motores con el fin de proteger el circuito de picos de corriente no deseados, este regulador consta de dos capacitores y un transistor.

Al usar un cable de los más flexibles que se encuentran en el mercado se obtuvo una buena respuesta al constante movimiento de estos ya que durante todo el proceso de experimentación ninguno sufrió fallas, los componentes funcionaron de la mejor manera y no hubo ningún problema de conexión.

4.1.4 Integración

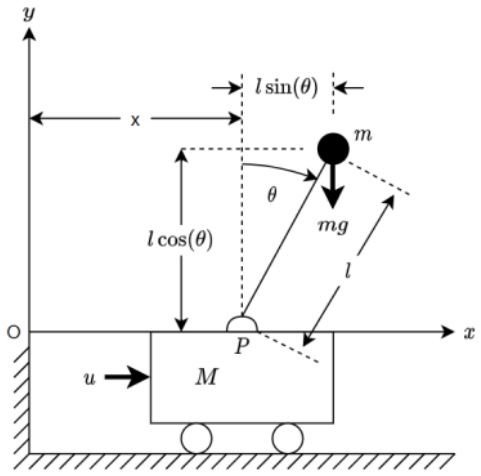
A medida que se añadían elementos al montaje, se comprobaba su correcto funcionamiento mediante códigos simples, con el objetivo de verificar que el circuito y sus conexiones fueran estables. Al momento de leer las variables a través del acelerómetro y giroscopio MPU6050 se hizo uso de su DMP (Digital Motion Processor) el cual permite obtener un dato certero del ángulo del péndulo y también reduce la carga de trabajo al microcontrolador principal (ESP32), junto con la distancia que mide el sensor ultrasónico HC-SR04, se obtienen las dos variables (ángulo del péndulo y posición del móvil) con las cuales se trabajó.

4.2 Controladores

4.2.1 Modelamiento matemático

Figura 5

Sistema del péndulo invertido.



En la figura 5 se muestra el sistema del péndulo invertido donde la masa se concentra en la parte superior de la varilla del péndulo, situando así su centro de gravedad en lo alto de la varilla.

El modelo matemático es una expresión que describe el comportamiento físico del sistema.

El desarrollo de las simulaciones se basó en el modelo matemático lineal en espacio de estados planteado en el libro de ingeniería de control moderna de Ogata (Ogata K. , 2010). Por medio de las ecuaciones 1 y 2, donde la M es la masa del carro, m es la masa suspendida en el péndulo, l es la longitud del péndulo, x la posición del vehículo, θ el ángulo del péndulo con respecto al eje vertical y u es la fuerza:

$$(M+m)\ddot{x} + ml\ddot{\theta} = u \quad \text{(Ec. 1)}$$

$$ml^2\ddot{\theta} + ml\ddot{x} = mgl\theta \quad \text{(Ec. 2)}$$

Las cuales se pueden modificar como:

$$Ml\ddot{\theta} = (M+m)g\theta - u$$

$$M\ddot{x} = u - mg\theta$$

Por lo tanto, definimos las variables de estado como $x1, x2\ , x3, x4$ donde:

$$x1 = \ \theta$$

$$x2 = \ddot{\theta}$$

$$x3 = x$$

$$x4 = \ \dot{x}$$

Teniendo en cuenta que θ y x son las salidas de nuestro sistema entonces:

$$y = \begin{bmatrix} y1 \\ y2 \end{bmatrix} = \begin{bmatrix} \theta \\ x \end{bmatrix} = \begin{bmatrix} x1 \\ x3 \end{bmatrix}$$

Adicionalmente, al tener en cuenta las variables de estado y las ecuaciones resaltadas, se obtiene

$$\dot{x1} = x2$$

$$x2 = \left(\frac{M+\dot{m}}{Ml}\right) gx1 - \frac{1}{Ml}u$$

$$\dot{x3} = x4$$

$$\dot{x4} = -\frac{mg}{M}x1 + \frac{1}{M}u$$

Por último, expresándolo como ecuaciones vectoriales se obtiene

$$\begin{bmatrix} \dot{x1} \\ \dot{x2} \\ \dot{x3} \\ \dot{x4} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ \frac{M+m}{Ml}g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{m}{M}g & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{1}{Ml} \\ 0 \\ \frac{1}{M} \end{bmatrix} u \quad \text{(Ec.3)}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u \quad \text{(Ec.4)}$$

Las ecuaciones 3 y 4, son la representación en el espacio de estados del sistema de péndulo invertido móvil para el proyecto.

4.2.2 Control por Retroalimentación de Estados (LQR)

En este código se implementa un controlador LQR modelado como un sistema de ecuaciones lineales como el propuesto en (Gaona & Niño, 2022). Para ello, se definen las matrices del sistema, obtenidas a partir del modelo matemático representado en espacio de estados en las ecuaciones 3 y 4 que a su vez utiliza las especificaciones físicas del prototipo (Masas, gravedad, longitud).

Este enfoque permite analizar cómo la entrada (fuerza) afecta el comportamiento del sistema, así como determinar qué estados se consideran como salida.

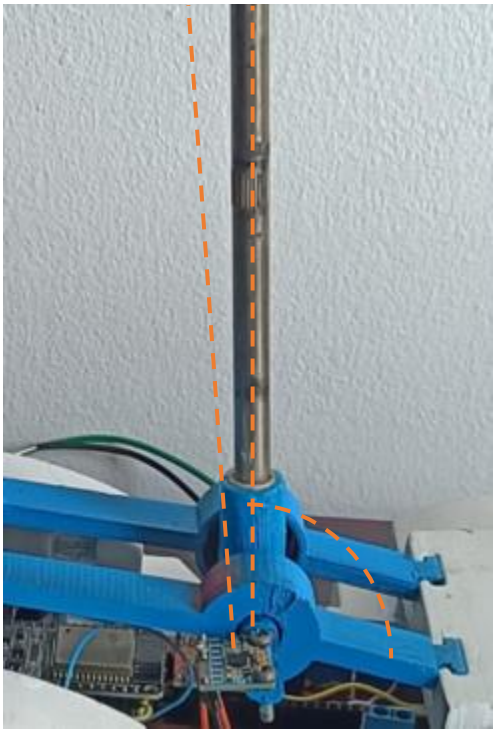
Durante el desarrollo de este código para el controlador LQR (ver código en el apéndice E) se definió que la posición deseada del péndulo respecto a la referencia vertical (x) fuera de 0.2 m.

Aunque la referencia para la posición del ángulo (θ) debería ser cero al momento del montaje y calibración del sensor en el prototipo, se evidenció un desfase de 1 grado respecto al eje

vertical esto debido a que el soporte para el acelerómetro no quedo a 90 grados del soporte de la varilla del péndulo (ver figura 6).

Figura 6

Desfase de la posición del acelerómetro con respecto a la varilla del péndulo.



Por esta razón, en el código del controlador LQR se añadió una referencia adicional de 0.015708 rad como el ángulo deseado, finalmente se simula el sistema cerrado, calculando la fuerza de control en cada paso de tiempo.

En el controlador LQR se tienen en cuenta diferentes variables, Q define la importancia relativa de los estados, R penaliza el esfuerzo de control y se obtienen entonces una ganancia K propia del controlador y dos ganancias en este caso llamadas N_r una para la referencia del ángulo 0.9 grados dada la perturbación nombrada anteriormente y otra para la posición del carro ya que la referencia deseada es 0.2m.

Las condiciones iniciales para las siguientes simulaciones fueron 0.2 rad (11.54 grados) y 0.3 m para la posición del carro, aparte de esto se saturo la señal de salida entre 0 y 5 como referencia al voltaje que se aplica a los motores según la señal PWM.

En la primera simulación se utilizaron valores de ponderación Q más altos para el ángulo y la posición ya que son los dos estados que se medirán en la experimentación con el prototipo físico y un valor bajo para R para así obtener un esfuerzo de control considerable.

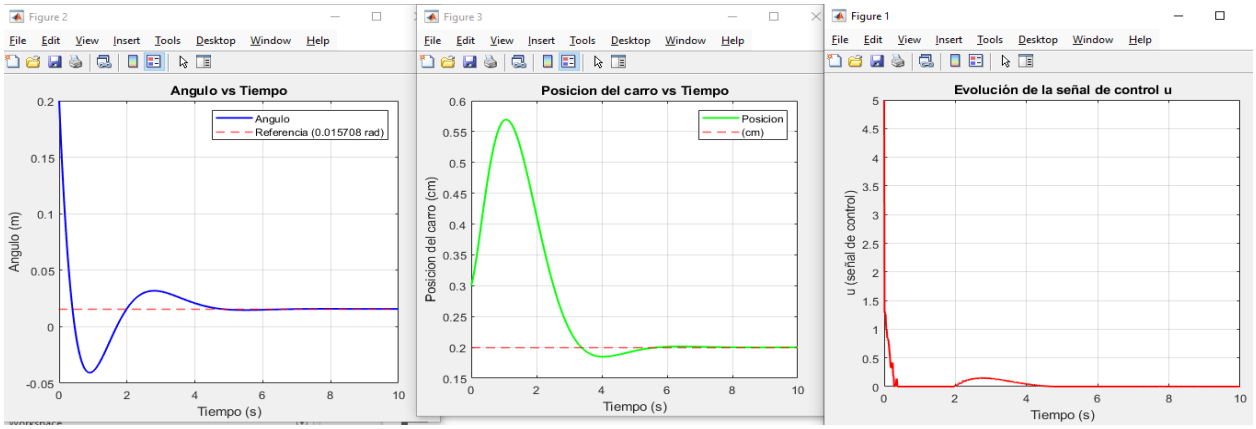
Q = [3000, 500, 3000, 100];

R=0.01

Figura 7.

Primera simulación retroalimentación de estados.

Ángulo vs Tiempo -Posición vs Tiempo -Señal de control (u). vs Tiempo



En la figura 7a, 7b y 7c se observan graficas de ángulo vs tiempo, posición del carro vs tiempo y señal de control (u) vs tiempo, se observa buen control con oscilaciones iniciales amortiguadas, alcanzando los dos objetivos de control.

Para la segunda simulación se incrementa el peso en la matriz de ponderación Q en el ángulo para darle prioridad sobre la posición del carro y dejando el mismo valor para el esfuerzo de control R.

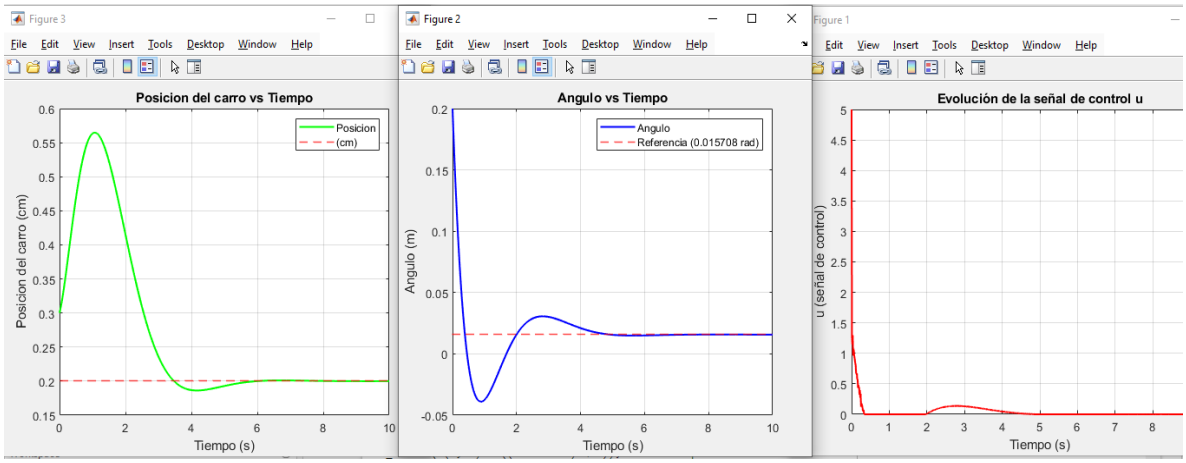
Q = [5000, 500, 3000, 100]

R=0.01

Figura 8.

Segunda simulación retroalimentación de estados.

Ángulo vs Tiempo *Posición vs Tiempo* *Señal de control (u) . vs Tiempo*



En la figura 8 se observan graficas de ángulo vs tiempo, posición del carro vs tiempo y señal de control (u) vs tiempo, a simple vista no se puede concluir demasiado acerca de los resultados de esta segunda simulación por ende se decide realizar una última simulación con una ponderación mayor en la velocidad angular del péndulo y generar un gráfico que junte las tres simulaciones para hacer una comparativa más objetiva.

Los valores para la tercera simulación fueron:

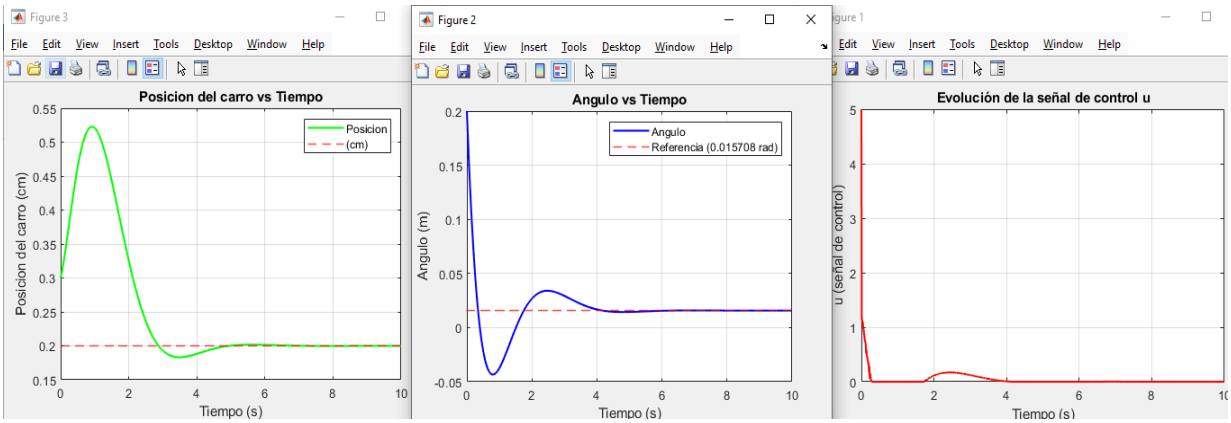
Q = [3000, 1000, 3000, 100]

R=0.01

Figura 9

Tercera simulación retroalimentación de estados.

Ángulo vs Tiempo *Posición vs Tiempo* *Señal de control (u) . vs Tiempo*

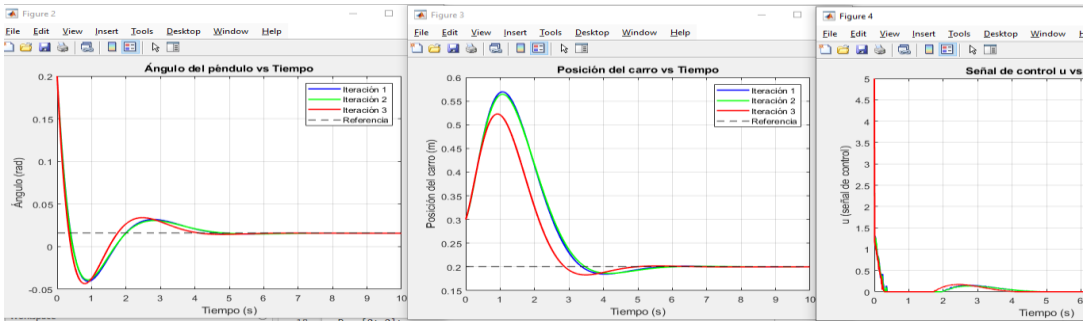


En la figura 9 se observan graficas de ángulo vs tiempo, posición del carro vs tiempo y señal de control (u) vs tiempo.

Figura 10

Simulaciones con retroalimentación de estados en comparación entre las tres simulaciones.

Ángulo vs Tiempo *Posición vs Tiempo* *Señal de control (u) . vs Tiempo*



De la figura 10 se observa que, para la primera simulación, se obtiene un ajuste lento y un mayor sobre impulso, lo que da como resultado un ajuste básico, pero poco eficiente. La segunda simulación no presentó variaciones considerables respecto a la primera. Por otro lado, en la tercera simulación, además de que se obtuvo una oscilación más pequeña en la posición del carro, se logró una respuesta más rápida al ángulo del péndulo, haciendo que el controlador sea más eficiente, respecto al tiempo de estabilización y respuesta se encontraron resultados similares a los referenciados por Guerra Aranda en (Guerra Aranda, 2019).

4.2.3 Controlador por Retroalimentación de Estados (LQR) y Observador Luenberger

Se implementa un controlador LQR con observador de estados basado en retroalimentación lineal, específicamente un observador de tipo Luenberger. En este controlador, además de lo desarrollado en el anterior, se calcula una matriz de ganancias para el observador, y la señal de control (fuerza) se determina utilizando los estados medidos (posición del carro y ángulo del péndulo) y estimando los estados no medidos (velocidad angular del péndulo y velocidad lineal del carro). (ver código en el apéndice F)

Para este controlador se tienen en cuenta además de las variables Q Y R antes nombradas los polos del observador, se obtienen las ganancias K del controlador LQR y además la ganancia L del observador.

Las condiciones iniciales para las siguientes simulaciones fueron 0.2 rad (11.54 grados) y 0.3 m para la posición del carro, aparte de esto se saturo la señal de salida entre -255 y 255.

En la primera simulación se usaron los siguientes valores:

- Matriz Q = [3000, 500, 3000, 100]
- Escalar R=0.01
- Polos del observador: [−10,−11,−12,−13]

En la segunda simulación se hicieron cambios en Q Y R

- Matriz Q = [3000, 300, 500, 500]
- Escalar R=0.1

En la tercera simulación se hicieron cambios en los polos del observador

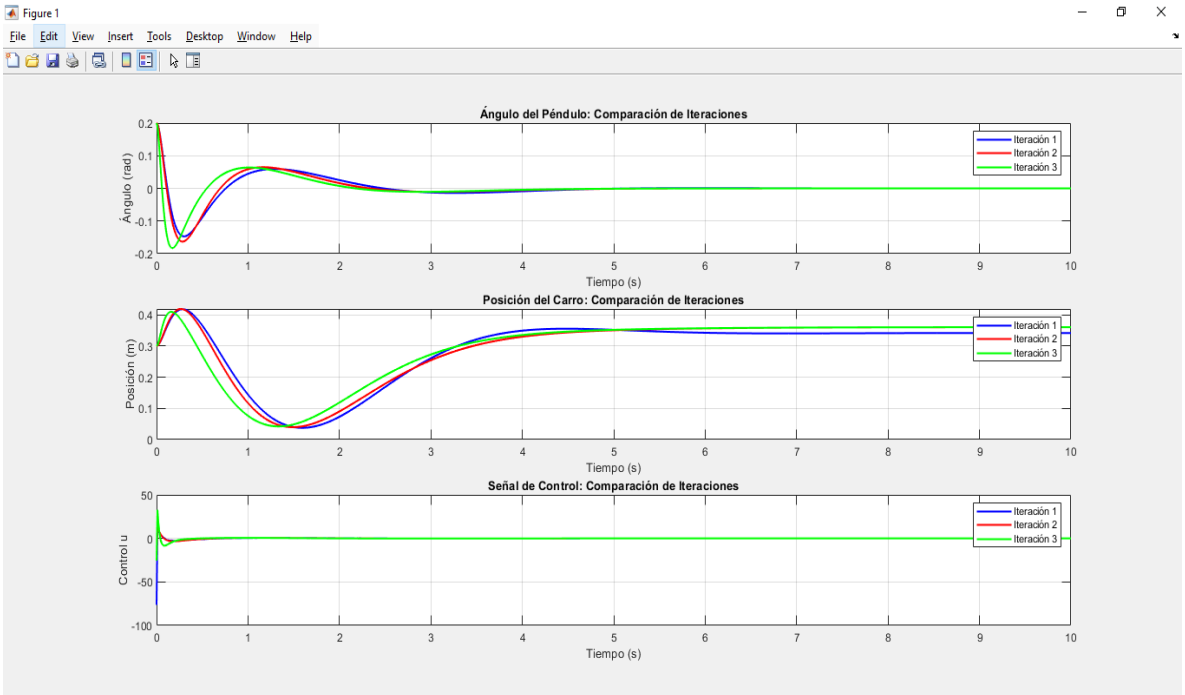
Polos del observador: [−20,−22,−24,−26]

Esta vez en lugar de realizar las simulaciones por aparte se agruparon en un mismo gráfico.

Figura 11

Simulaciones con retroalimentación de estados y observador en comparación entre las tres simulaciones.

Ángulo vs Tiempo - Posición vs Tiempo - Señal de control (u) vs Tiempo.



En la figura 11 se evidencian notables diferencias entre una simulación y otra, para la primera simulación se obtiene un control que es eficaz, para la segunda simulación se obtiene un control más estable y rápido y por último se obtiene de la tercera simulación un control que, aunque genera unas oscilaciones más grandes en el ángulo del péndulo es más rápido que los otros, lo cual es lo óptimo para nuestra aplicación.

Se observa una reducción de más de un segundo en el tiempo de estabilización del péndulo con respecto al controlador basado en LQR, una oscilación más alta para el ángulo del péndulo y una oscilación más baja para la posición del carro, resultados que fueron satisfactorios ya que,

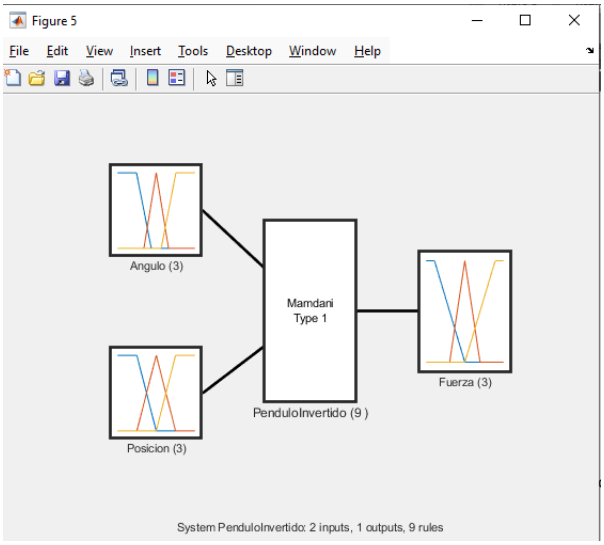
aunque se presentó incremento en la oscilación del ángulo esta es poco significativa ya que es del orden de 0.1 radianes que en comparación con la reducción de tiempo y la oscilación de la posición del carro no resulta ser relevante.

4.2.4 Controlador lógica Fuzzy (Mamdani)

En este código se implementa un sistema difuso basado en lógica Mamdani. En este se definen las entradas (ángulo y posición) y la salida (fuerza). Además, se establecen los conjuntos difusos con sus respectivas reglas como lo propuso Gaona en (Gaona & Niño, 2022) . Finalmente, se generan gráficos que muestran: fuerza vs. tiempo, posición vs. tiempo, y ángulo vs. tiempo. También se visualizan los conjuntos difusos del sistema y la superficie difusa que relaciona las entradas y la salida (ver código en el Apéndice D).

Figura 12

Diagrama lógico difuso



En la figura 12 se observa el diagrama lógico difuso el cual posee dos entradas y una salida.

Entradas:

- Ángulo del péndulo ([-1.57, 1.57] radianes):
- Posición del carro ([0.1, 0.3] metros):

Salida:

- **Fuerza de control ([-255, 255]):**

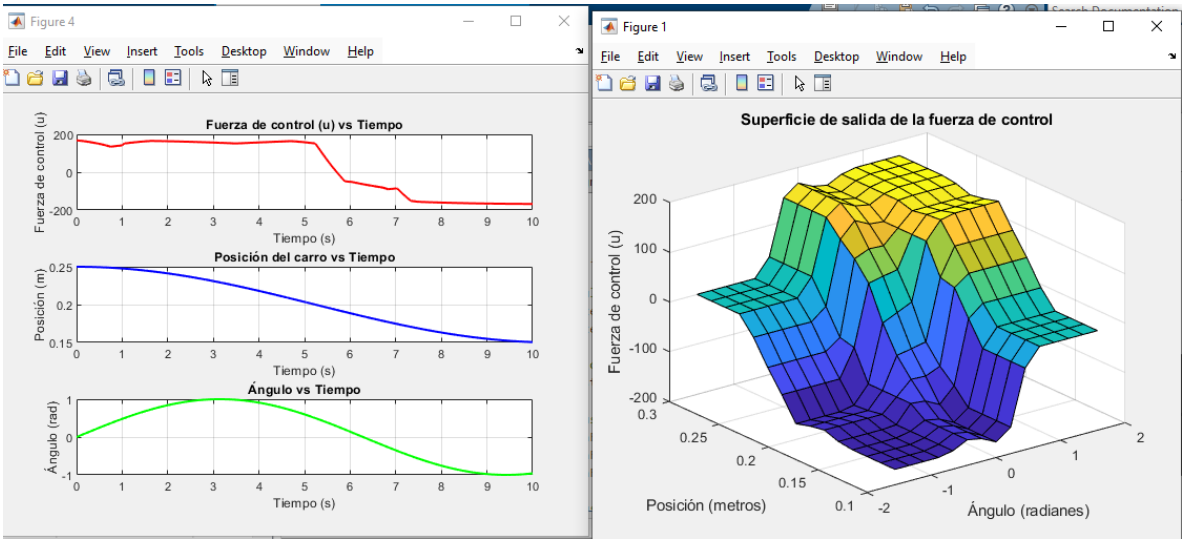
Conjuntos Difusos:

- El ángulo, la posición y la fuerza tienen conjuntos difusos divididos en tres categorías: **Negativa, Cero, Positiva**, con funciones de pertenencia trapezoidales

Figura 13

Resultado grafico en 2D y 3D del control de lógica difusa (primera simulación).

Fuerza de control vs Tiempo - Posición vs Tiempo - Angulo vs Tiempo



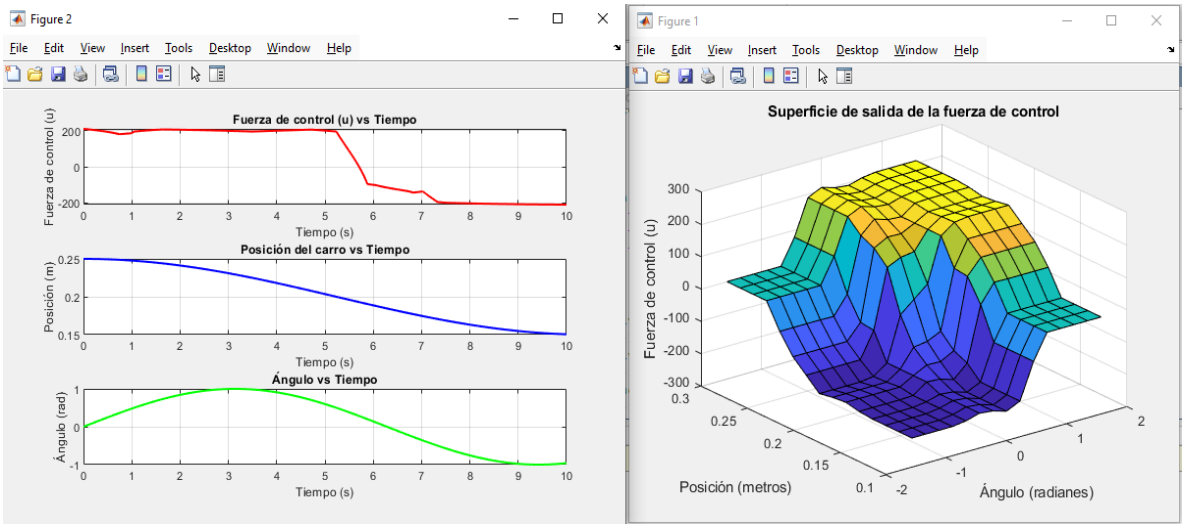
En la figura 13 se observa que el sistema es funcional sin embargo se pueden ajustar parámetros para mejorar la estabilidad y precisión del control.

Se realizaron cambios en el rango de la salida (fuerza de control), de [-255,255] a [-300,300] con el fin de permitir una mayor fuerza si el péndulo tiene desviaciones significativas, además de esto se redujo el rango de la fuerza Cero ([-100, 100] a [-50, 50]) para que las fuerzas pequeñas fueran más precisas y se ajustaron por ende los conjuntos Negativa ([-400 -400 -200 0] a [-400 -400 -200 -50]) y Positiva ([0 -200 -400 -400] a [50 200 400 400]) para adaptarse al nuevo rango.

Figura 14

Resultado grafico en 2D y 3D del control de lógica difusa (segunda simulación).

Fuerza de control vs Tiempo - Posición vs Tiempo - Angulo vs Tiempo



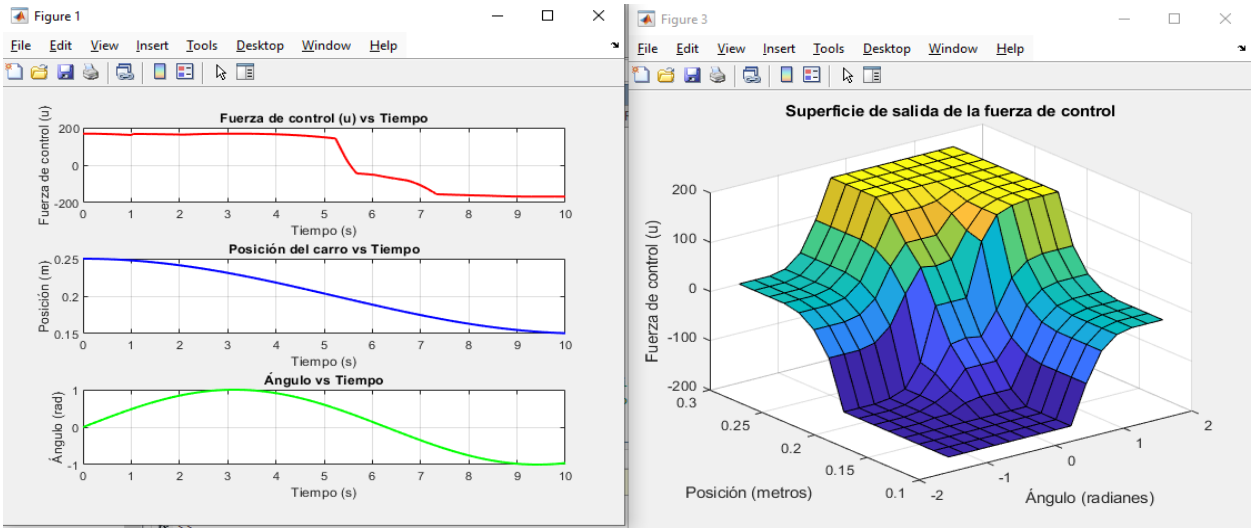
En la figura 14 se observa que el sistema ahora aplica fuerzas más precisas cuando las desviaciones son pequeñas, y tiene un rango ampliado para manejar desviaciones mayores.

Para la tercera iteración se ajustaron las funciones de membresía para que el sistema sea más sensible cerca del equilibrio (Angulo=0), además de esto se redujo el rango de intersección entre los conjuntos “Cero” y “Negativo” /” Positivo”.

Figura 15

Resultado grafico en 2D y 3D del control de lógica difusa (tercera simulación).

Fuerza de control vs Tiempo - Posición vs Tiempo - Angulo vs Tiempo.



En la figura 15 se observa que al hacer más estrechos los nuevos conjuntos difusos en torno a 0 se mejora la precisión en la estabilización cerca del equilibrio, lo que conduce a un control más preciso y estable.

Al obtener resultados como los obtenidos en lógica difusa es difícil compararlos con los encontrados para las anteriores técnicas por lo cual una vez obtenidos los diferentes resultados y comprendido el cómo se comporta cada controlador según como se ajustan sus variables se realizaron varias simulaciones en las cuales además de cambiar las variables antes mencionadas para cada controlador se realizaron simulaciones con y sin la masa suspendida del péndulo con el fin de realizar la experimentación con el prototipo para estos dos casos, se realizaron varias simulaciones con la ayuda del software Matlab las cuales se encuentran en el apéndice I.

4.3 Diseño de código en Arduino

Siguiendo la metodología desarrollada en MATLAB, se decidió programar tres códigos en Arduino para los controladores. Antes de esto, se implementaron varios programas para adaptar el entorno de Arduino a la ESP32. Esto incluyó la descarga de los controladores necesarios y la investigación sobre cómo se programa esta tarjeta de desarrollo, comparándola con placas más comunes como Arduino Uno y otras similares.

Una vez configurado el entorno de trabajo, se descargaron las librerías necesarias para manejar los diferentes sensores y se verificó que funcionaran adecuadamente con la ESP32. También se realizó la calibración del acelerómetro MPU6050 y se comprobó el correcto funcionamiento del sensor HC-SR04 y del controlador de los motores. Al confirmar que todos los elementos del prototipo recibían datos de manera correcta, se desarrolló un código en el que, mediante los dos sensores, se enviaban órdenes a los motores para observar cómo se comportaba el prototipo en conjunto, es decir, el circuito trabajando de forma integrada y no de manera separada.

Posteriormente, se desarrolló el código basado en lógica difusa utilizando la librería Fuzzy.h, junto con las librerías necesarias para los distintos sensores. Es importante destacar que, para los tres códigos, se aprovechó la capacidad multitarea de la ESP32, utilizando su herramienta de multitarea (MULTITASK), lo que permite ejecutar varias tareas de manera paralela y no secuencial. Esto evita cualquier tipo de retraso y permite realizar los cálculos en el menor tiempo posible.

Para el código que utiliza el controlador LQR, se diseñó teniendo en cuenta los cálculos previos realizados en MATLAB. Es decir, las ganancias tanto del controlador como de las referencias deseadas se ingresaron manualmente al código de Arduino. Se programó un

controlador LQR simple para calcular la fuerza, en este caso el voltaje, y enviarla como señal PWM a los motores.

Por último, en el código que utiliza el controlador LQR junto con el observador, se tomó como base el código anterior, añadiendo la parte correspondiente al observador. Por lo tanto, para este código también se deben ingresar manualmente las ganancias tanto del controlador como del observador, y se programó un cálculo simple para determinar la fuerza o voltaje necesario para mantener el péndulo equilibrado y llevar al vehículo a la posición deseada.

5. Presupuesto

A continuación, se presenta una tabla con los respectivos costos del proyecto.

Figura 16

Presupuesto del proyecto.

PRESUPUESTO						
ELEMENTOS	DESCRIPCION	UNIDAD	CANTIDAD	VALOR UNITARIO	VALOR TOTAL	RESPONSABLE
RECURSO HUMANO						
Estudiante	Aplicación de conocimientos	Horas	320	$ 5.000,00	$ 3.200.000,00	Autores
Director del proyecto	Asesoria especializada	Horas	80	$ 18.000,00	$ 1.440.000,00	UIS
Co-directoe del proyecto	Asesoria especializada	Horas	30	$ 18.000,00	$ 540.000,0000	UIS
SOFTWARE						
Matlab/Simulink	Licencia paquete	Año	2	$ 1.112.000,00	$ 2.224.000,00	UIS
Soolidworks Research	Licencia paquete	Año	2	$ 2.020.000,00	$ 4.040.000,00	UIS
Microsoft Office	Licencia paquete	Año	2	$ 220.000,00	$ 440.000,00	UIS
PROTOTIPO						
Bastidor		Cantidad	1	$ 300.000,00	$ 300.000,00	Automatica y Robotica SION
Brazo		Cantidad	1	$ 300.000,00	$ 300.000,00	Automatica y Robotica SION
Microcontrolador	ESP32	Cantidad	2	$ 65.000,00	$ 130.000,00	Automatica y Robotica SION
Bateria	7,4 V/2S 1000 mAh 20 C Lipo	Cantidad	1	$ 80.000,00	$ 80.000,00	Automatica y Robotica SION
Motor	Motor N20 6V 1000RPM	Cantidad	4	$ 96.000,00	$ 384.000,00	Automatica y Robotica SION
Neumaticos		Cantidad	2	$ 40.000,00	$ 80.000,00	Automatica y Robotica SION
Sensor de posicion	HC-SR04	Cantidad	1	$ 7.000,00	$ 7.000,00	Automatica y Robotica SION
Sensor de angulo	MPU-6050	Cantidad	1	$ 11.000,00	$ 11.000,00	Automatica y Robotica SION
Reguladores de voltaje	Condensadores	Cantidad	2	$ 2.500,00	$ 5.000,00	Automatica y Robotica SION
Cables conectores		Cantidad	N/A	$ 50.000,00	$ 50.000,00	Automatica y Robotica SION
Memoria micro SD		Cantidad	1	$ 70.000,00	$ 70.000,00	Automatica y Robotica SION
Modulo micro SD		Cantidad	1	$ 7.000,00	$ 7.000,00	Automatica y Robotica SION
Placa de circuito impreso (PCB)		Cantidad	1	$ 50.000,00	$ 50.000,00	Automatica y Robotica SION
BIBLIOGRAFIA						
Libro	Ingenieria de control Moderna Og	Cantidad	1	$ 500.000,00	$ 500.000,00	UIS
EQUIPOS						
Computador portatil		Cantidad	2	$ 2.500.000,00	$ 5.000.000,00	Autores
Herramientas	Kit de soldadura	Cantidad	1	$ 200.000,00	$ 200.000,00	Autores
Multimetro		Cantidad	2	$ 120.000,00	$ 240.000,00	Autores
Protoboard		Cantidad	2	$ 40.000,00	$ 80.000,00	Autores
ADICIONALES						
Papeleria	Implementos basicos	Cantidad	1	$ 100.000,00	$ 100.000,00	Autores
				Subtotal	$ 19.378.000,00	
				Imprevistos (15%)	$ 2.906.700,00	
				TOTAL	$ 22.284.700,00	

6. Validación de resultados

6.1 Experimentación física en el prototipo

Tras determinar los valores adecuados de ganancias y los datos requeridos para su implementación en el código Arduino (Ver apéndices J, K y L), donde se realizaron múltiples iteraciones. Estas iteraciones se centraron en identificar aquellos parámetros que produjeran cambios significativos en el comportamiento del sistema mecánico. Para el análisis, se seleccionaron dos configuraciones representativas por cada controlador: una sin masa y otra con masa en la parte superior del péndulo. Se priorizaron aquellas iteraciones que lograron una mayor estabilidad en el sistema dinámico.

6.1.1 Pruebas físicas con lógica difusa o código Fuzzy

Figura 17

Angulo vs Tiempo. - Prueba física con lógica difusa con peso.

Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 332

Valores tomados: 332/332 (100% de los valores totales)

Tiempo evaluado: 11 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 8

Tiempo de estabilidad promedio: 2.5 segundos.

En la figura 17 se observan oscilaciones iniciales pronunciadas que disminuyen gradualmente con el tiempo, evidenciando la acción del controlador.

La estabilización del ángulo alrededor del cero indica que el sistema logro mantener el equilibrio del péndulo tras las primeras correcciones.

El controlador difuso es efectivo en la regulación del ángulo del péndulo, ya que logra reducir las oscilaciones iniciales y estabilizarlo en posición vertical. Este comportamiento valida la capacidad del controlador para manejar sistemas no lineales.

Figura 18

Posición vs Tiempo. - Prueba física con lógica difusa y con peso.

Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 332

Valores tomados: 332/332 (100% de los valores totales)

Tiempo evaluado: 11 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones: 8

Tiempo de estabilidad promedio: 2.5 segundos.

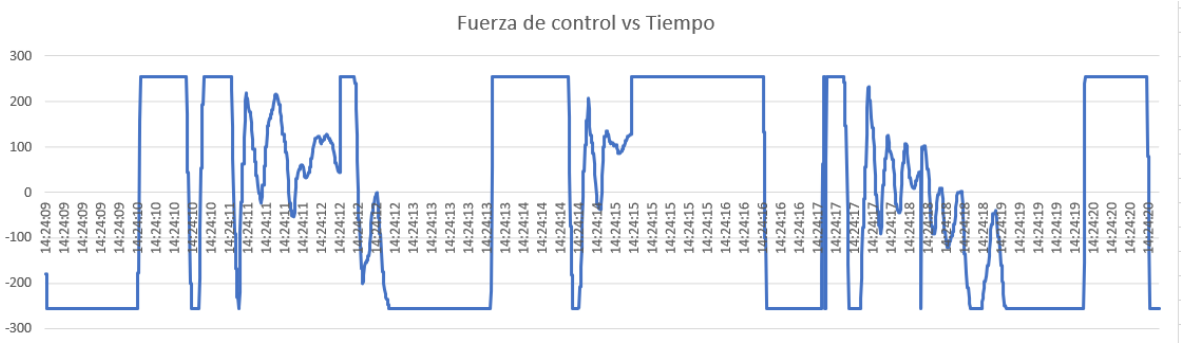
En la figura 18 se observa que el carro experimenta desplazamientos significativos en los primeros momentos debido a las correcciones necesarias para contrarrestar el movimiento inicial del péndulo.

Posteriormente, la posición del carro tiende a estabilizarse en un rango limitado, lo que indica que el sistema se acerca al equilibrio.

Aunque el controlador logra estabilizar la posición del carro, es evidente una inestabilidad residual. Esto podría atribuirse a la necesidad de ajustes adicionales en las reglas difusas o al diseño del sistema mecánico.

Figura 19

Fuerza de control vs tiempo. - Prueba física con lógica difusa.



Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 332

Valores tomados: 332/332 (100% de los valores totales)

Tiempo evaluado: 11 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 8

Tiempo de estabilidad promedio: 2.5 segundos.

En la figura 19 se observa que la fuerza aplicada fluctúa significativamente a lo largo del tiempo, con valores máximos y mínimos que alcanzan aproximadamente $\mp$ 255 unidades, lo cual es satisfactorio debido a que es el valor dado en el código.

Estas fluctuaciones indican las correcciones constantes realizadas por el controlador para estabilizar el sistema frente a perturbaciones o desviaciones.

La alta variabilidad en ciertos intervalos puede deberse a cambios repentinos en el estado del sistema (como oscilaciones del ángulo o desplazamientos del carro).

El controlador difuso responde de manera activa para mantener la estabilidad del péndulo invertido. Sin embargo, el sistema al ser altamente inestable no es suficiente para controlar el sistema real.

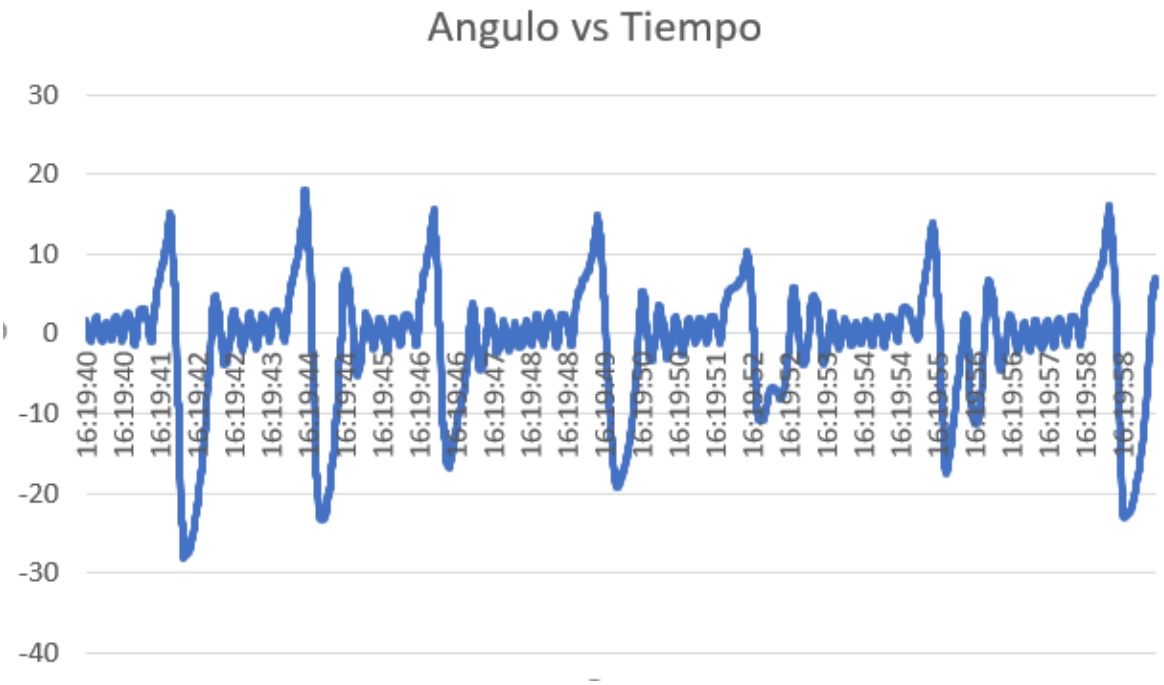
6.2 Pruebas físicas con código retroalimentación de estados (LQR)

6.2.1 Resultados de pruebas físicas con Retroalimentación de estados sin peso

Primera Iteración:

Figura 20

Angulo vs tiempo. - Prueba física Retroalimentación de estados sin peso.



Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 3327

Valores tomados: 3327/3327 (100% de los valores totales)

Tiempo evaluado: 18 segundos

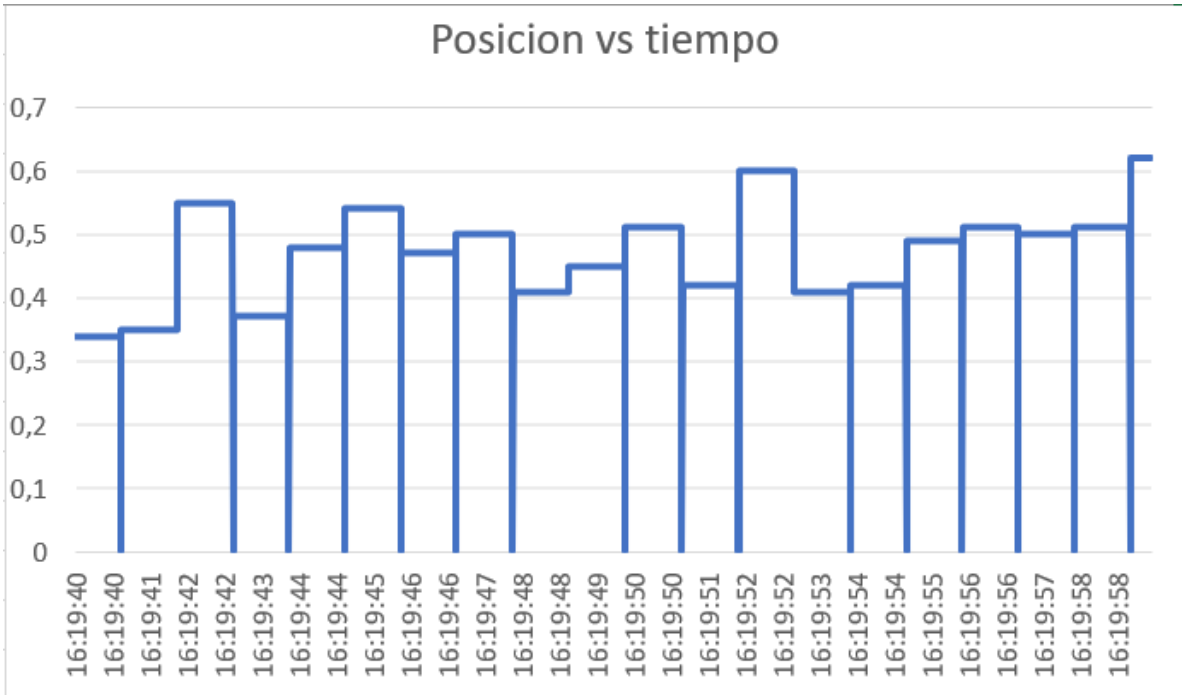
Tiempo de estabilidad máxima: 4.30 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 7

Tiempo de estabilidad promedio: 4 segundos.

Figura 21

Posición vs tiempo. - Prueba física Retroalimentación de estados sin peso.



Nota: Los valores de la Posición están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 3327

Valores tomados: 3327/3327 (100% de los valores totales)

Tiempo evaluado: 18 segundos

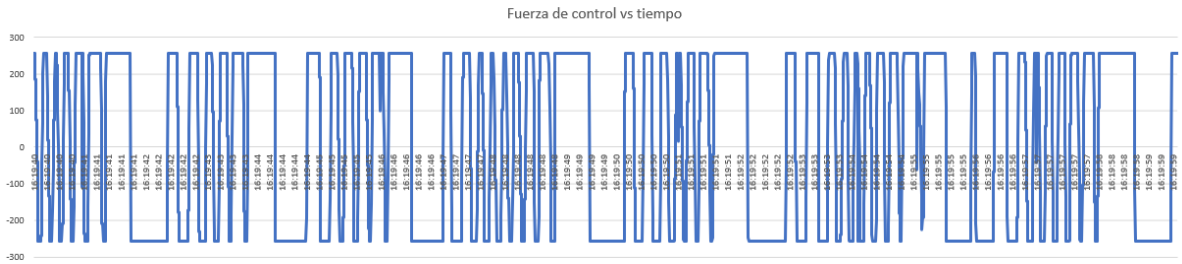
Tiempo de estabilidad máxima: 4 segundos

Numero de interrupciones: 13

Tiempo de estabilidad promedio: 3 segundos.

Figura 22

Fuerza de control vs tiempo. - Prueba física Retroalimentación de estados sin peso.



Nota: Los valores de la Fuerza de control están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 3327

Valores tomados: 3327/3327 (100% de los valores totales)

Tiempo evaluado: 18 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones (oscilación más frecuente): 8

Tiempo de estabilidad promedio: 3 segundos.

Las figuras 20, 21 y 22 corresponden a la iteración más destacada.

Esta iteración se destaca por obtener un tiempo de estabilidad superior en el control del péndulo en comparación con las demás iteraciones, con un total de 3 segundos.

En relación con las otras dos iteraciones, esta presenta un mayor dominio en la gráfica, reflejado en una mayor densidad de valores. Esto es indicativo de una mayor eficiencia del controlador.

El desempeño del controlador en la posición de la base móvil mostró una mayor precisión, siguiendo con mayor eficacia el valor de referencia establecido.

La gráfica correspondiente a la fuerza de control (figura 22) exhibe un mayor número de oscilaciones, lo cual evidencia una mayor inestabilidad en comparación con las iteraciones dos y tres. No obstante, este comportamiento refleja que el sistema priorizó la estabilización del péndulo.

6.2.2 Resultados de pruebas físicas con Retroalimentación de estados con peso

Segunda iteración:

Figura 23

Angulo vs Tiempo. - Resultados prueba física con retroalimentación de estados con peso.

Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 2974

Valores tomados: 2974/2974 (100% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 5 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 4

Tiempo de estabilidad promedio: 3 segundos.

Figura 24

Posición vs Tiempo. - Resultado de retroalimentación de estados con peso.

Nota: Los valores de la Posición están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 2974

Valores tomados: 2974/2974 (100% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 5 segundos

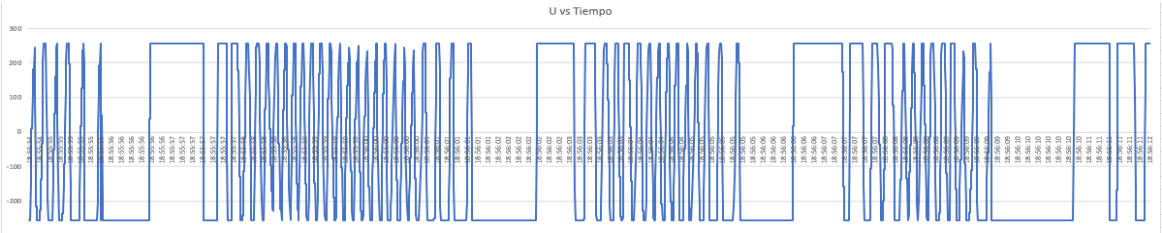
Tiempo de estabilidad máxima: 2 segundos

Numero de interrupciones: 12

Tiempo de estabilidad promedio: 2 segundos.

Figura 25

Fuerza de control vs tiempo. - Resultado retroalimentación de estados con peso.



Nota: Los valores de la Fuerza de control están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 2974

Valores tomados: 2974/2974 (100% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 4 segundos

Numero de interrupciones (oscilación más frecuente): 4

Tiempo de estabilidad promedio: 2.20 segundos

Las figuras 23, 24 y 25 corresponden a la iteración más destacada.

La iteración más destacable fue la segunda debido al mayor control en un tiempo máximo de 5 segundos en el péndulo, por consiguiente; se tuvo lugar un énfasis en el péndulo que en la posición del carro.

De acuerdo con los resultados se observa una mayor estabilidad en la posición, en comparación de la primera y segunda iteración, lo que quiere decir que el controlador mantuvo el valor de referencia en un tiempo significativo.

Hay inferioridad representativa en la cantidad de interrupciones, por lo tanto; es evidente la tasa mínima de oscilaciones con respecto al comportamiento del sistema, sin embargo, al ser un

sistema real no lineal, las perturbaciones son evidentes y de influencia negativa significativa puesto que es un diseño subóptimo (sistema con sobre impulso o alta sensibilidad).

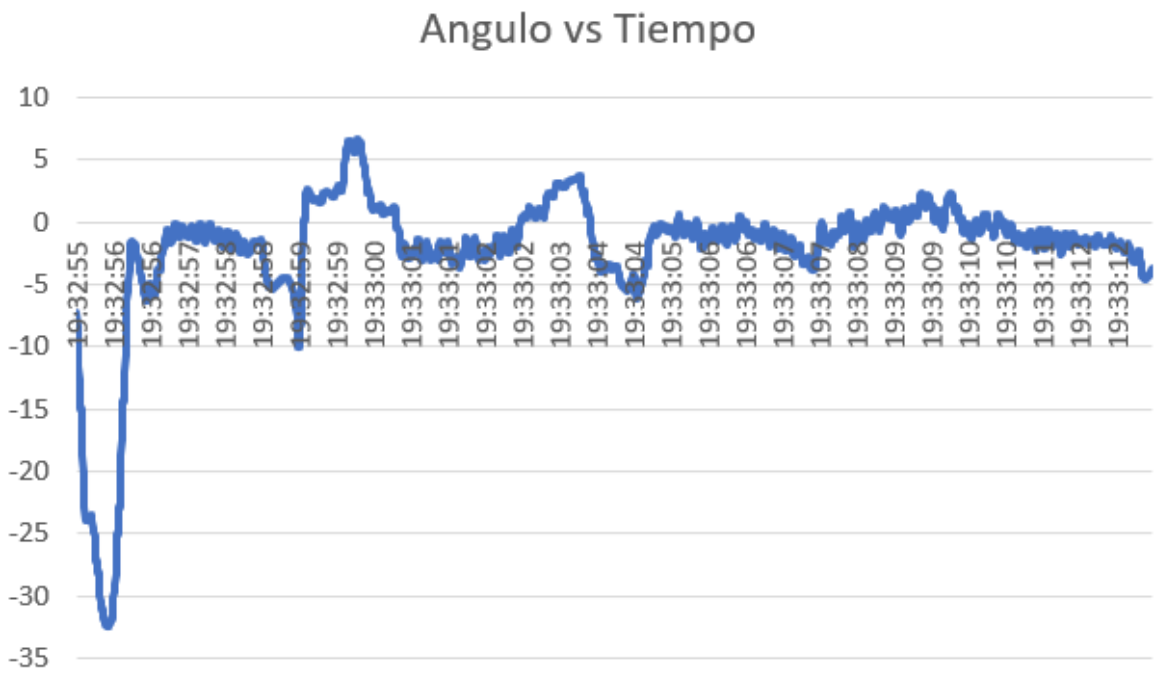
Si bien, que el controlador conlleva mayor estabilidad del sistema, hay un aumento de tiempo de respuesta dado que, el sistema de control tiene que esforzarse para su respectiva estabilidad.

6.3 Pruebas físicas con código lqr y observador

6.3.1 Retroalimentación de estados con observador sin masa

Figura 26

Angulo vs Tiempo. - Retroalimentación de estados con observador sin masa.



Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 7639

Valores tomados: 3039/7639 (39.78% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 7 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 5

Tiempo de estabilidad promedio: 3.20 segundos.

Figura 27

Posición vs Tiempo. - Retroalimentación de estados con observador Luenberger sin peso.

Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 7639

Valores tomados: 3039/7639 (39.78% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 3.30 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 12

Tiempo de estabilidad promedio: 1.10 segundos.

Figura 28

Fuerza de control vs tiempo. - Retroalimentación con observador Luenberger sin peso.

Nota: Los valores de la Fuerza de control están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal), también queda resaltar que para este grafico en el eje horizontal se evaluaron únicamente 6 segundos debido a la alta cantidad de oscilaciones.

Valores totales (Registrados por Arduino): 7639

Valores tomados: 3039/7639 (39.78% de los valores totales)

Tiempo evaluado: 6 segundos

Tiempo de estabilidad máxima: 1 segundo

Numero de interrupciones (oscilación más frecuente): 57

Tiempo de estabilidad promedio: 0.6 segundos

Las figuras 26, 27 y 28 corresponden a la iteración más destacada.

Se destaca debido a que estuvo en mayor tiempo la estabilidad del péndulo con un tiempo de 7 segundos, un valor que abarca significativamente con respecto a las otras iteraciones registradas para esta etapa.

Sin embargo, se posee una mayor cantidad de interrupciones u oscilaciones en la fuerza de control, esto debido a que el sistema posee un controlador más (observador de Luenberger) lo cual genera un mayor esfuerzo de control del sistema.

Otros factores que influyen en el aumento de la fuerza de control en el sistema son: la alta inestabilidad que caracteriza el sistema, factores externos (Vibraciones, viento), variaciones internas como fricción o errores en los actuadores.

No obstante, queda resaltar el aumento del tiempo en que el sistema dura en equilibrio, en comparación del sistema dinámico con solo la retroalimentación de estados.

6.3.2 Retroalimentación de estados con observador Luenberger con peso

Figura 29

Angulo vs Tiempo. - Retroalimentación de estados con observador Luenberger con peso.

Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 4155

Valores tomados: 3055/4155 (73.52% de los valores totales)

Tiempo evaluado: 18 segundos

Tiempo de estabilidad máxima: 4 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 4

Tiempo de estabilidad promedio: 2 segundos.

Figura 30

Posición vs Tiempo. - Retroalimentación de estados con observador Luenberger con peso.

Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 4155

Valores tomados: 3055/4155 (73.52% de los valores totales)

Tiempo evaluado: 18 segundos

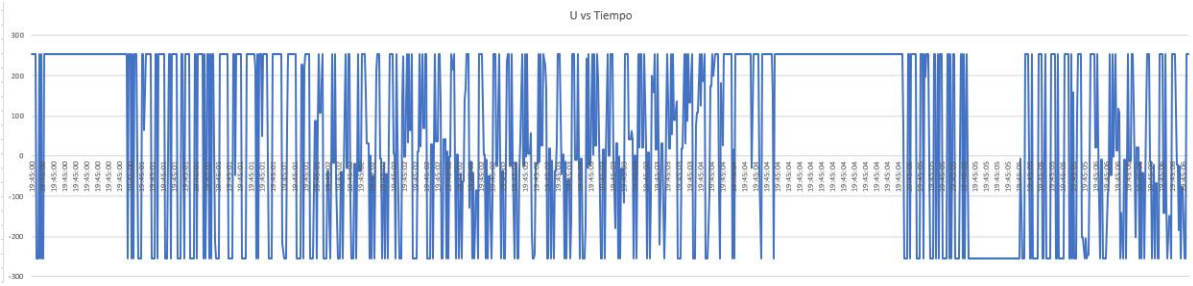
Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 15

Tiempo de estabilidad promedio: 1 segundo.

Figura 31

Fuerza de control vs Tiempo. - Retroalimentación de estados con observador Luenberger con peso.



Nota: Los valores de la Fuerza de control están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal), también queda resaltar que para este grafico en el eje horizontal se evaluaron únicamente 6 segundos debido a la alta cantidad de oscilaciones.

Valores totales (Registrados por Arduino): 4155

Valores tomados: 1055/4155 (25.39% de los valores totales)

Tiempo evaluado: 6 segundos

Tiempo de estabilidad máxima: 0.25 segundos

Numero de interrupciones (oscilación más frecuente): 63

Tiempo de estabilidad promedio: 0.6 segundos

Las figuras 29, 30 y 31 corresponden a la iteración más destacada.

La iteración presento un mayor equilibrio con un tiempo estimado de 4 segundos, se destaca a comparación de las otras dos iteraciones.

Hay una menor fluctuación, por lo que indica que el controlador responde de una mejor manera y precisión.

No obstante, sigue generando valores muy pequeños de tiempo de estabilizarse, por ende, el sistema no es controlable en su totalidad.

Aun cuando el controlador logra alcanzar un mayor tiempo de estabilidad en el sistema en comparación de las otras dos iteraciones es notorio que el vehículo no genera una solidez ya que el tiempo de respuesta no es el adecuado.

Se evidencia que, debido a la masa montada en el sistema dinámico, proporciona una mayor inestabilidad, generando un mayor esfuerzo de control, lo cual es evidente en la gran cantidad de oscilaciones, aun cortando en gran medida los datos totales suministrados por el Arduino.

Los resultados obtenidos de las pruebas físicas con los diferentes controladores muestran que la estabilidad del péndulo invertido mejora con el uso de técnicas avanzadas de control, aunque no logran estabilizar por completo el sistema. La lógica difusa logra estabilizar el ángulo del péndulo, pero presenta una inestabilidad residual en la posición del carro. La retroalimentación de estados (LQR) muestra una mejor capacidad de control en términos de estabilidad y tiempo de respuesta con respecto a la lógica difusa, especialmente sin peso adicional sin embargo se observa una mayor cantidad de oscilaciones en la fuerza de control.

Finalmente, la inclusión del observador de Luenberger mejora la estabilidad del péndulo por mayor tiempo, pero introduce mayores oscilaciones en la fuerza de control debido al esfuerzo adicional requerido para mantener el equilibrio.

En conclusión, aunque el controlador LQR con observador de Luenberger ofrece el mejor desempeño en términos de estabilidad del péndulo, aun se requieren mejoras en el diseño del sistema , leer las variables faltantes directamente y también modelar las físicas del sistema no lineal.

7. Conclusiones

El diseño y construcción del sistema mecánico del péndulo invertido móvil evidencio la influencia de las tolerancias de fabricación en la precisión dimensional, lo que afecto la alineación y el desempeño del sistema. Asimismo, la selección de material para el brazo de 0,28 m y la base del carro influyo en la rigidez estructural y la distribución del peso, repercutiendo en la estabilidad del control. Además, se detectaron ligeras vibraciones no previstas en el modelo teórico desarrollado en el software de Diseño Asistido por Computadora (CAD). Estos resultados destacan la importancia de considerar factores de fabricación y propiedades de los materiales en la transición del diseño virtual a la implementación física.

Se logro sintetizar tres estrategias de control: lógica difusa, retroalimentación de estados (LQR) y retroalimentación de estados con observador de Luenberger, con el objetivo de regular el ángulo del péndulo y la posición del carro mediante el uso del software MATLAB. Cada técnica presento ventajas y desventajas en términos de estabilidad y tiempo de respuesta. La simulación en MATLAB permitió ajustar los parámetros de los controladores antes de su implementación física en Arduino. No obstante, se identificaron diferencias entre la simulación y la realidad debido a perturbaciones como el ruido en los sensores, dinámicas no modeladas, rigidez y distribución del peso del prototipo, así como vibraciones del sistema. Estos resultados destacan la importancia de considerar dichos factores para mejorar la precisión y desempeño del sistema en aplicaciones reales.

A partir de la comparación experimental del desempeño de las tres estrategias de control, se determinó que la retroalimentación de estados con observador de Luenberger proporcionó la mayor estabilidad del sistema, con un menor tiempo de respuesta y un mejor mantenimiento del equilibrio del péndulo sin masa suspendida. Sin embargo, se observó un mayor esfuerzo de control

en los controladores LQR y LQR con observador, debido a las oscilaciones en la gráfica de la fuerza de control vs tiempo. En términos de la estabilidad, la lógica difusa presento el menor desempeño con un tiempo promedio de 2.3 segundos, seguido del control LQR, que alcanzo hasta 4 segundos. El control con observador de Luenberger mejoro la estabilidad en 3 segundos respecto al LQR, aunque mostro limitaciones en la estimación de los estados no medidos. Estos resultados destacan las diferencias de robustez, tiempo de estabilidad y tiempo de respuesta de cada estrategia, proporcionando información clave para la selección del controlador más adecuado según los requerimientos del sistema.

Referencias Bibliográficas

Alarcón, I. (2014). *Diseño y análisis de estabilidad de un controlador difuso para un péndulo invertido*.

Apaza Cruz, J. L. (2017). *Modelado de un péndulo invertido móvil, usando hardware y software libre*. Universidad Nacional del Altiplano.

Åström, K. J., & Murray, R. M. (2021). *Feedback systems: An introduction for scientists and engineers*. Princeton University Press.

Castaños Luna, F. (2003). *Levantamiento y estabilización del péndulo invertido* [Tesis de maestría]. Repositorio Cinvestav.

Celso, O., & Cutipa, J. (2020). A low-cost didactic module for testing advanced control algorithms. *HardwareX*, *8*, e00148. https://doi.org/10.1016/J.OHX.2020.E00148

Chen, Y., & Liu, T. (2017). Advanced control techniques for nonlinear systems: Fuzzy logic and state feedback approaches. *International Journal of Control, Automation, and Systems*, *15*(4), 1234–1245.

Collazos Mamian, I., Erazo Muñoz, E. F., Mora Carabalí, O. R., & Mecatrónico, I. (2006). *Diseño y simulación de un sistema subactuado: Péndulo invertido traslacional configurable* [Trabajo de grado]. Universidad Autónoma de Occidente. http://hdl.handle.net/10614/6347

García, J. (2022). *Controlador de tracción basado en lógica difusa para un robot* [Tesis de pregrado]. Universidad Autónoma de Querétaro.

García, J., Martínez, R., & Sánchez, L. (2022). Cost analysis of advanced control systems in developing countries: Challenges and opportunities. *Journal of Engineering Economics*, *34*(2), 89–102.

García, R. (1993). *Algunos aspectos de la teoría del control de sistemas no lineales*. Universidad de Buenos Aires.

Gaona, J., & Niño, W. (2022). *Diseño de un sistema de control avanzado para un péndulo invertido doble lineal* [Trabajo de grado]. Universidad Autónoma de Bucaramanga.

González Quijano, J., Abderrahim, M., Garrido, S., & Bensalah, C. (2014). Algoritmo de optimización basado en modelos sustitutos de la función objetivo con aplicación a problemas de aprendizaje de control. *ResearchGate*.

Gordillo, F., & Aracil, J. (2010). El péndulo invertido: Un desafío para el control no lineal.

Guerra Aranda, R. (2019). *Control de péndulo invertido sobre un carro móvil*.

Khalil, H. K. (2015). *Nonlinear systems* (3.ª ed.). Prentice Hall.

La República. (2021, 17 de julio). Manufactura y construcción, entre los sectores que más utiliza la automatización. https://www.larepublica.co/especiales/tecnologia-e-industria-2021/manufactura-y-construccion-entre-los-sectores-que-mas-utiliza-la-automatizacion-3202925

Li, Z., Deng, J., Lu, R., Xu, Y., Bai, C., & Liu, C. (2019). Advanced control strategies for nonlinear systems: A comprehensive review. *IEEE Transactions on Industrial Electronics*.

Liu, X., et al. (2019). Application of state feedback control in inverted pendulum systems. *Robotics and Automation Journal*.

MEN (Ministerio de Educación Nacional de Colombia). (2020). *Informe de educación en Colombia: Brechas y desafíos*. https://www.mineducacion.gov.co

OCyT (Observatorio Colombiano de Ciencia y Tecnología). (2021). *Diagnóstico de la infraestructura tecnológica en las universidades colombianas*. https://ocyt.org.co/

Ogata, K. (2010). *Ingeniería de control moderna* (5.ª ed.). PEARSON Educación S.A.

Ogata, K. (2010). *Modern control engineering* (5th ed.). Pearson.

Palacios, R. (2017). State-space feedback controllers and PID controller. *MASKAY, 7*.

Picuino-Pardo, M. (2023, octubre). Controlador PID. https://www.picuino.com/es/control-pid.html

Pimentel Medina, J. (2018). Fundamentals of inverted pendulum control: A case study. *Journal of Control Engineering*.

Pinares-Mamani, O. G. C., & Cutipa-Luque, J. C. (2020). A low-cost didactic module for testing advanced control algorithms. *HardwareX*, *8*. https://doi.org/10.1016/J.OHX.2020.E00148

Romero Rodríguez, G., Sánchez, P., Reyes Cortés, F., Michua, A., Calderón Flores, B., Cid, J., Torres Monsiváis, J. C., & Morales Timal, G. (2009). Modelado, control y simulación de un sistema péndulo invertido sobre base móvil.

Sira, H., Luviano, H., & Cortés, J. (2011). *Control lineal robusto de sistemas no lineales diferencialmente planos*. Universidad Nacional de Colombia.

Slotine, J. J. E., & Li, W. (1991). *Applied nonlinear control*. Prentice Hall.

Triviño Macías, L. (2020). *Modelado, simulación y control de un péndulo* [Tesis de maestría]. Universitat Autònoma de Barcelona.

Wang, J. (2020). Dynamic stability analysis of personal mobility vehicles using inverted pendulum models. *Journal of Mechanical Engineering*, *56*(3), 45–58.

 Triviño Macias, L. (2020). *Modelado, simulación y control de un péndulo*. Barcelona: Universitat Autònoma de Barcelona.

 Wang, J. (2020). *Dynamic Stability Analysis of Personal Mobility Vehicles Using Inverted Pendulum Models*. Journal of Mechanical Engineering, 56(3), 45–58.

APÉNDICES

Apéndice A. Diseño y construcción de las partes del carro por modelado CAD (SolidWorks).

Figura A1. *Chasis modificado para impresión*

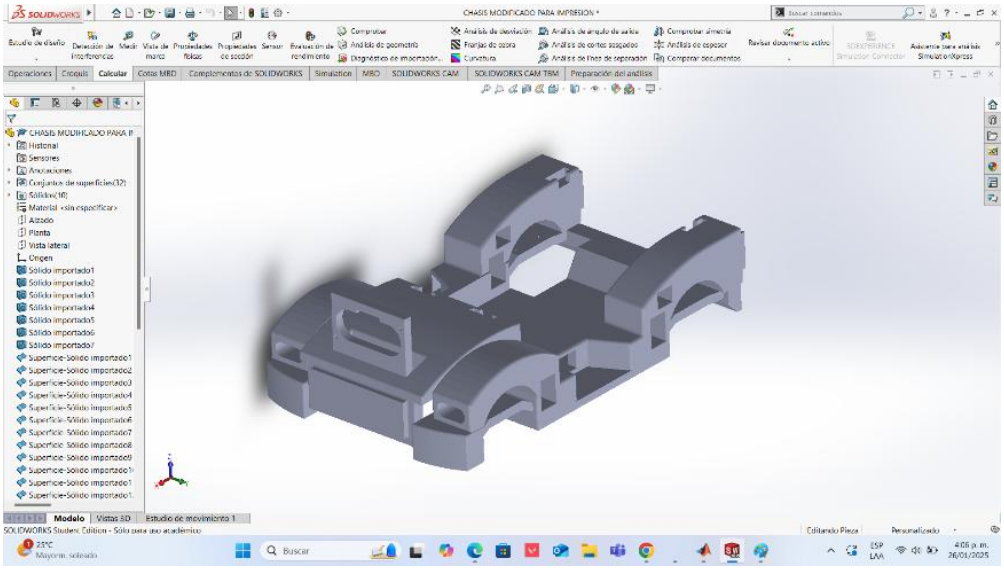


Figura A2. *Brazo derecho*

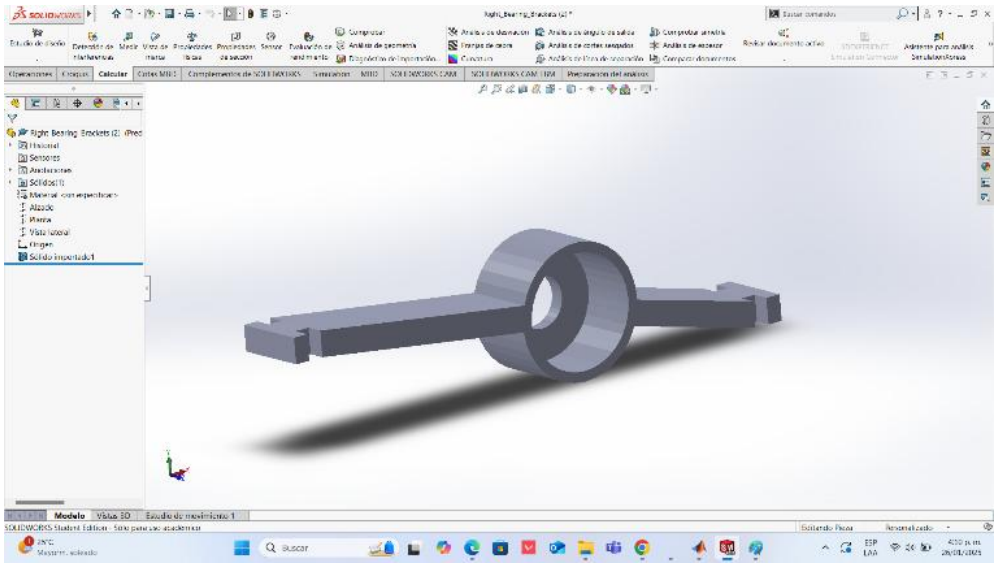


Figura A3. *Eje*

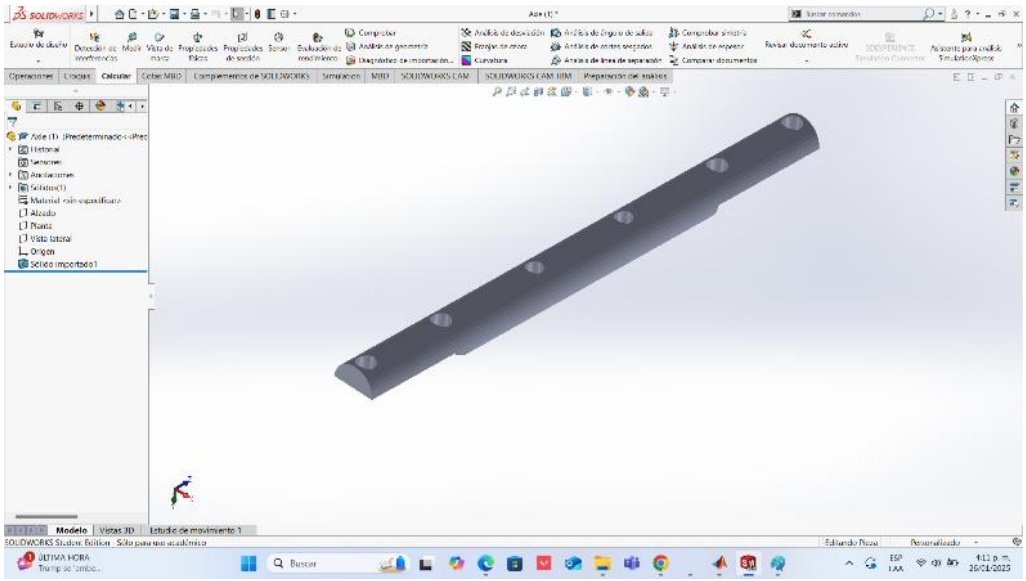


Figura A4. *Brazo izquierdo*

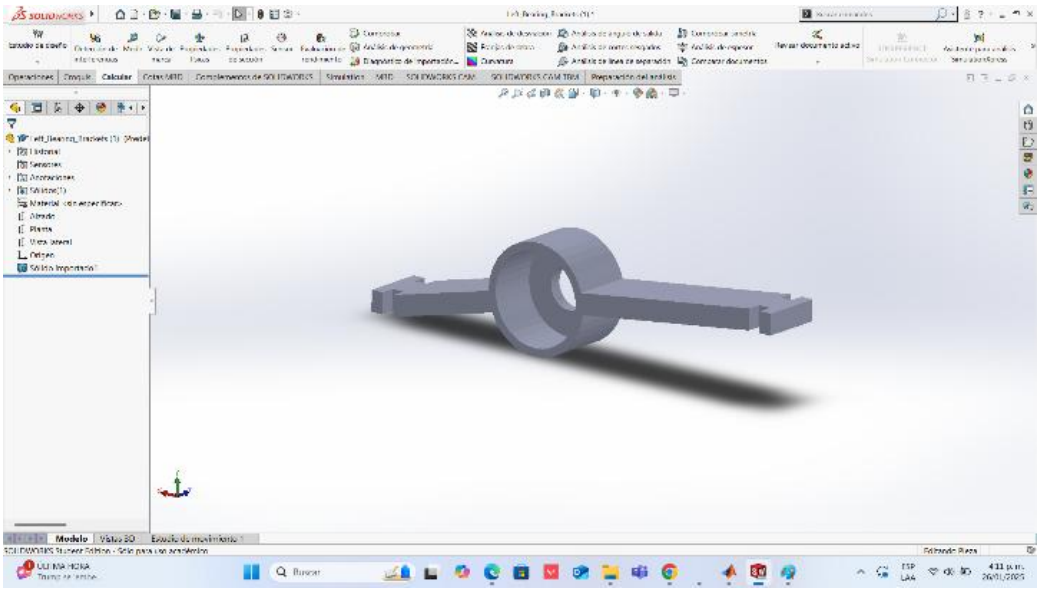


Figura A5. Alerón puesto de la batería

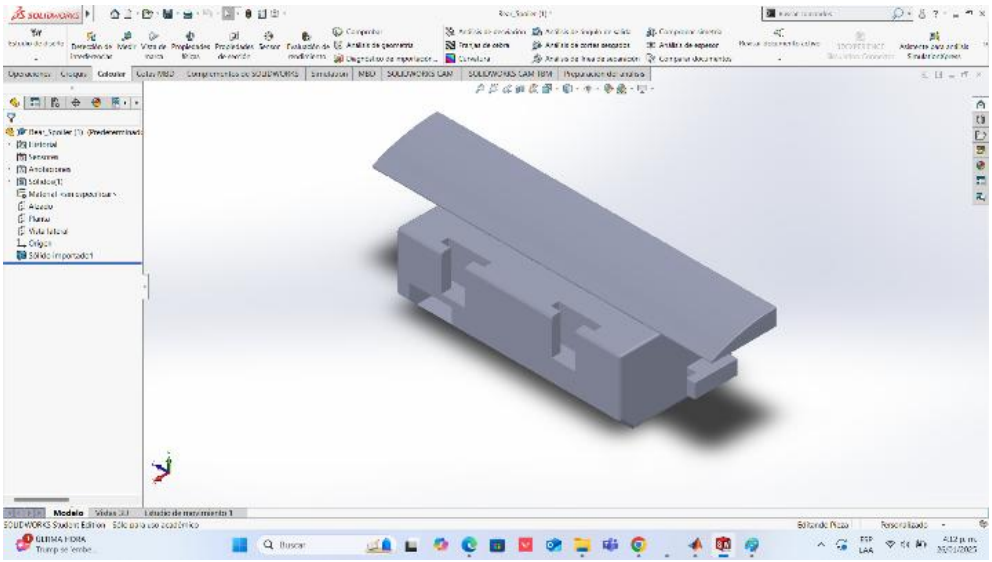


Figura A6. Tubo en T

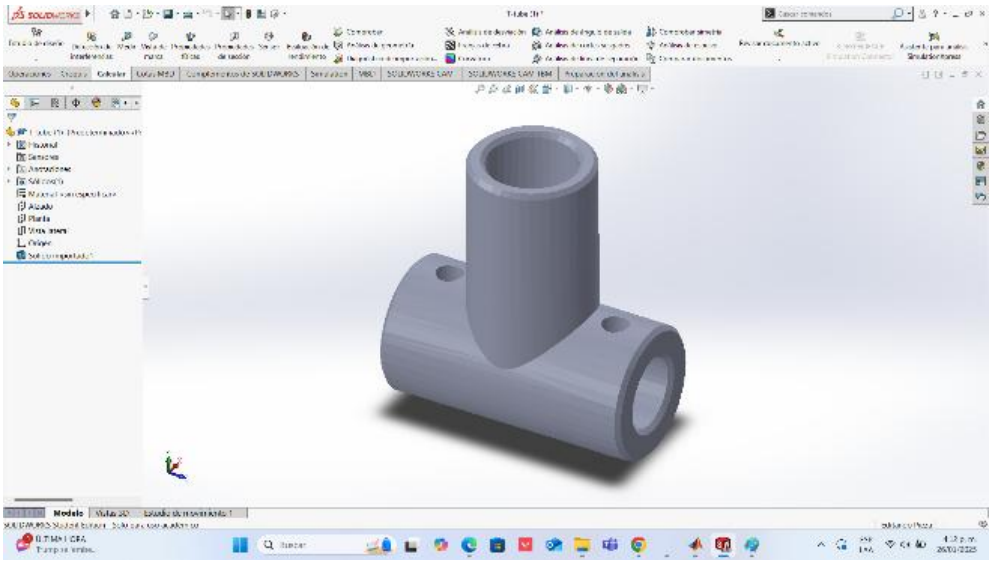
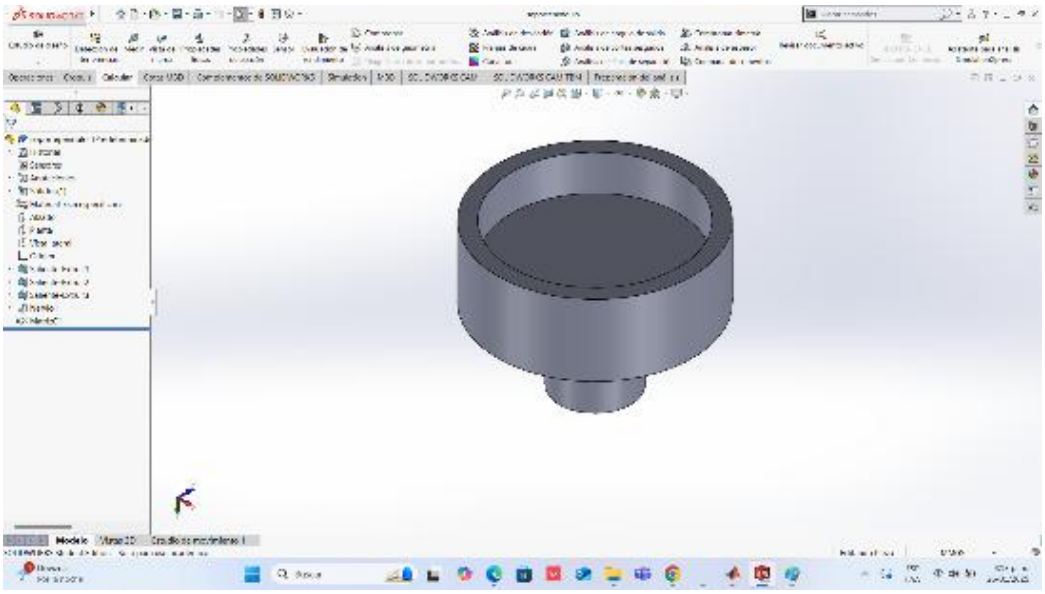


Figura A7. Soporte péndulo



Apéndice B. Diseño del circuito electrónico

Figura B1. Batería

Figura B2. ESP32, controlador de motores y circuito regulador de voltaje.

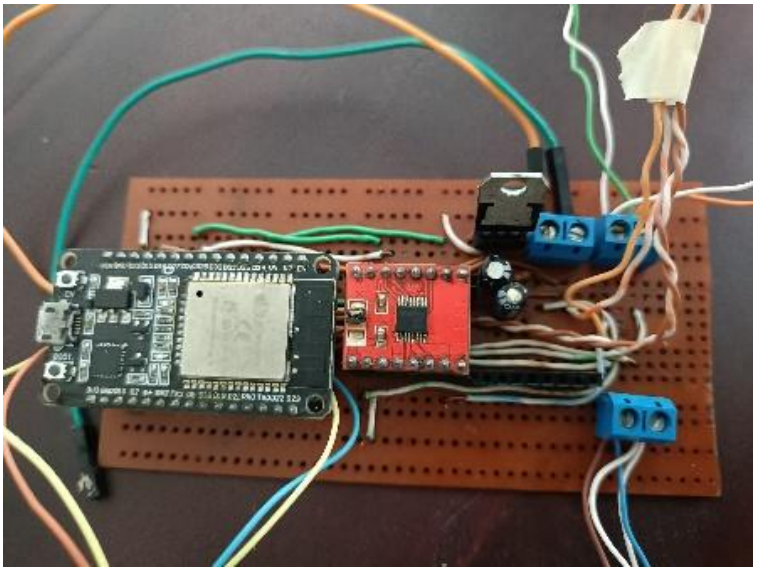
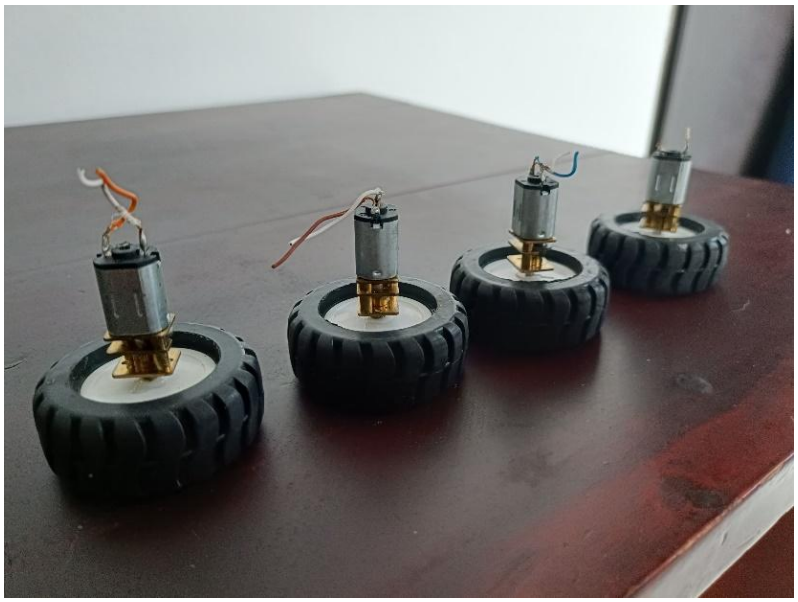


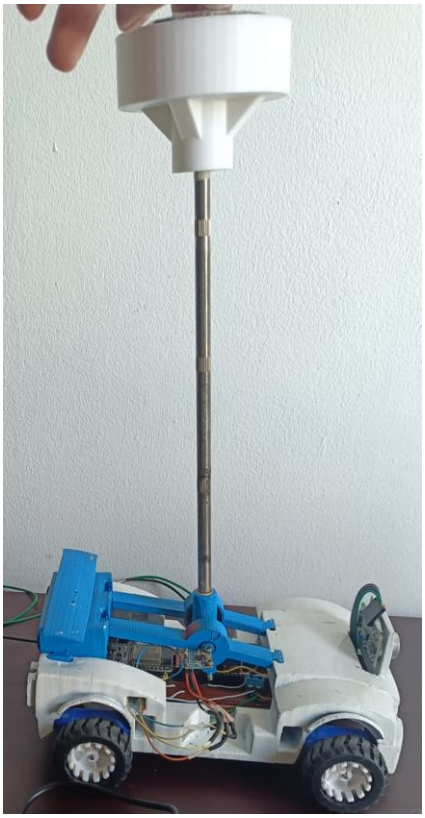
Figura B3. Sensor MPU6050

Figura B4. Motores DC N20 acoplados a las ruedas



Apéndice C. Prototipo del sistema de péndulo invertido móvil ensamblado.

Figura C1. Sistema ensamblado completamente.



Apéndice D. Código del Sistema de Control de Lógica Difusa (Fuzzy Logic)-MATLAB

```
CODIGO FUZZY
clear; clc;
% Crear un sistema difuso
sistemaFuzzy = mamfis ('Name', 'PenduloInvertido');
% Definir las entradas: Ángulo (en radianes) y Posición (en metros)
sistemaFuzzy = addInput (sistemaFuzzy, [-1.57
1.57], 'Name', 'Angulo');
sistemaFuzzy = addInput (sistemaFuzzy, [0.05
0.4], 'Name', 'Posicion');
% Definir la salida: Fuerza de control (u)
sistemaFuzzy = addOutput (sistemaFuzzy, [-400
400], 'Name', 'Fuerza');
% Definir los conjuntos difusos para el ángulo
sistemaFuzzy = addMF(sistemaFuzzy, 'Angulo', 'trapmf', [-1.57 -
1.57 -0.8 -0.2], 'Name', 'Negativo');
sistemaFuzzy = addMF(sistemaFuzzy, 'Angulo', 'trapmf', [-0.5 -0.2
0.2 0.5], 'Name', 'Cero');
```

```
    sistemaFuzzy = addMF(sistemaFuzzy, 'Angulo', 'trapmf', [0.2 0.8
1.57 1.57], 'Name', 'Positivo');
    % Definir los conjuntos difusos para la posición
    sistemaFuzzy = addMF(sistemaFuzzy, 'Posicion', 'trapmf', [-0.05 -
0.05 0.25 0.275], 'Name', 'Negativa');
    sistemaFuzzy = addMF(sistemaFuzzy, 'Posicion', 'trapmf', [0.25
0.275 0.37 0.4], 'Name', 'Cero');
    sistemaFuzzy = addMF(sistemaFuzzy, 'Posicion', 'trapmf', [0.375
0.4 0.6 0.6], 'Name', 'Positiva');
    % Definir los conjuntos difusos para la fuerza de control (u)
    sistemaFuzzy = addMF(sistemaFuzzy, 'Fuerza', 'trapmf', [-400 -400
-200 0], 'Name', 'Negativa');
    sistemaFuzzy = addMF(sistemaFuzzy, 'Fuerza', 'trapmf', [-170 -35
35 170], 'Name', 'Cero');
    sistemaFuzzy = addMF(sistemaFuzzy, 'Fuerza', 'trapmf', [0 200 400
400], 'Name', 'Positiva');
    % Definir nuevas reglas difusas
    ruleList = [...
    1 1 1 1 1; % Ángulo Negativo y Posición Negativa -> Fuerza Negativa
    1 2 1 1 1; % Ángulo Negativo y Posición Cero -> Fuerza Negativa
    1 3 1 1 1; % Ángulo Negativo y Posición Positiva -> Fuerza Negativa
    2 1 3 1 1; % Ángulo Cero y Posición Negativa -> Fuerza Positiva
    2 2 2 1 1; % Ángulo Cero y Posición Cero -> Fuerza Cero
    2 3 1 1 1; % Ángulo Cero y Posición Positiva -> Fuerza Negativa
    3 1 3 1 1; % Ángulo Positivo y Posición Negativa -> Fuerza Positivo
    3 2 3 1 1; % Ángulo Positivo y Posición Cero -> Fuerza Positiva
    3 3 3 1 1; % Ángulo Positivo y Posición Positiva -> Fuerza Positiva
    ];
    sistemaFuzzy = addRule(sistemaFuzzy, ruleList);
    % Simulación del sistema difuso en función del tiempo
    ts = 0.01; % Paso de tiempo
    t = 0:ts:10; % Vector de tiempo
    angulo = sin(0.5 * t); % Ángulo del carro (oscilación senoidal)
    posicion = 0.2 + 0.05 * cos(0.3 * t); % Posición del carro
(oscilación pequeña)
    % Inicializar vectores para resultados
    fuerza_control = zeros(size(t));
    for i = 1:length(t)
    inputs = [angulo(i), posicion(i)];
    fuerza_control(i) = evalfis(sistemaFuzzy, inputs);
    end
    % Graficar los resultados
    figure;
    subplot(3, 1, 1);
    plot(t, fuerza_control, 'r', 'LineWidth', 1.5);
    title('Fuerza de control (u) vs Tiempo');
```

```
xlabel('Tiempo (s)');
ylabel('Fuerza de control (u)');
grid on;
subplot(3, 1, 2);
plot(t, posicion, 'b', 'LineWidth', 1.5);
title('Posición del carro vs Tiempo');
xlabel('Tiempo (s)');
ylabel('Posición (m)');
grid on;
subplot(3, 1, 3);
plot(t, angulo, 'g', 'LineWidth', 1.5);
title('Ángulo vs Tiempo');
xlabel('Tiempo (s)');
ylabel('Ángulo (rad)');
grid on;
% Evaluar el sistema difuso para un caso específico
inputs = [0 0.2]; % Ejemplo: ángulo = 0.5 rad, posición = 0.2 m
output = evalfis(sistemaFuzzy, inputs);
% Mostrar el resultado
disp(['Fuerza de control (u): ', num2str(output)]);
% Visualizar los conjuntos difusos
figure;
plotfis(sistemaFuzzy);
% Graficar la superficie de salida
figure;
gensurf(sistemaFuzzy);
title('Superficie de salida de la fuerza de control');
xlabel('Ángulo (radianes)');
ylabel('Posición (metros)');
zlabel('Fuerza de control (u)');
```

Apéndice E. Código del Sistema de Control por Retroalimentacion de Estados - MATLAB

CÓDIGO RETROALIMENTACION DE ESTADOS

```
clc;
% Matrices del sistema
A = [0, 1, 0, 0;
53.9587, 0, 0, 0;
0, 0, 0, 1;
-5.0287, 0, 0, 0];

B = [0;
-6.1115;
0;
1.6807];
```

```
      C = [1, 0, 0, 0; % angulo
      0, 0, 1, 0]; % posicion del carro

      D = [0;
      0];

      % Referencias deseadas
      x_ref = [0.015708; 0; 0.2; 0]; % 20 centimetros

      % Matriz Q: Ponderación de los estados
      Q = diag([3000, 3000, 500, 100]); % Mayor peso en el angulo y la
posicion

      % Matriz R: Ponderación del esfuerzo de control
      R = 0.01;

      % Cálculo de la ganancia K usando LQR
      K = lqr(A, B, Q, R);

      disp('Ganancia de retroalimentación K:');
      disp(K);

      % Cálculo de N_r para cada salida (posición y ángulo) usando
división izquierda
      N_r1 = -(C(2, :) * ((A - B * K) \ B)); % Para la posición del carro
      N_r2 = -(C(1, :) * ((A - B * K) \ B)); % Para el ángulo del péndulo

      disp('Ganancia N_r para la posición del carro (20 cm):');
      disp(N_r1);
      disp('Ganancia N_r para el ángulo del péndulo (3.6 grados):');
      disp(N_r2);

      % Valores iniciales
      x0 = [0.2; 0; 0.3; 0]; % [ángulo; velocidad angular;posición;
velocidad]

      % Referencias deseadas
      x_ref = [0.015708; 0; 0.2; 0]; % 20 cm y 3.6 grados

      % Tiempo de simulación
      tspan = 0:0.01:10; % De 0 a 10 segundos, con paso de 0.01 s

      % Sistema cerrado con retroalimentación de estados
      A_cl = A - B * K; % Matriz del sistema cerrado

      % Simulación del sistema
```

```
    [t, x] = ode45(@(t, x) A_cl * (x - x_ref), tspan, x0);

    % Calcular u en cada paso
    u_values = zeros(length(t), 1); % Preasignar espacio para la señal
de control
    for i = 1:length(t)
    u_values(i) = -K * (x(i, :)' - x_ref); % Calcular u(t)
    end

    % Saturar u entre 0 y 5 (opcional, si tienes límites físicos)
    u_values = max(0, min(5, u_values));

    % Imprimir los valores de u
    disp('Valores de u durante la simulación:');
    disp(table(t,                          u_values, 'VariableNames',
{'Tiempo', 'Control_u'}));

    % Graficar la señal de control u
    figure;
    plot(t, u_values, 'r', 'LineWidth', 1.5);
    xlabel('Tiempo (s)');
    ylabel('u (señal de control)');
    title('Evolución de la señal de control u');
    grid on;

    % Gráfica del angulo del pendulo
    figure;
    plot(t, x(:, 1), 'b', 'LineWidth', 1.5);
    hold on;
    yline(0.015708, '--r', 'LineWidth', 1.2); % Línea de referencia
para 0.020944 rad
    xlabel('Tiempo (s)');
    ylabel('Angulo (m)');
    title('Angulo vs Tiempo');
    legend('Angulo', 'Referencia (0.015708 rad)');
    grid on;

    % Gráfica de la posicion del carro
    figure;
    plot(t, x(:, 3), 'g', 'LineWidth', 1.5);
    hold on;
    yline(0.2, '--r', 'LineWidth', 1.2); % Línea de referencia para
20cm grados
    xlabel('Tiempo (s)');
    ylabel('Posicion del carro (cm)');
    title('Posicion del carro vs Tiempo');
```

```
    legend('Posicion', '(cm)');
    grid on;
```

Apéndice F.

Código del Sistema de Control por Retroalimentacion de Estados con Observador Luenberger – MATLAB

```
    CODIGO RETROALIMENTACION DE ESTADOS Y OBSERVADOR LUENBERGER
    % Definición de las matrices del sistema


    A = [0, 1, 0, 0;
    53.9587, 0, 0, 0;
    0, 0, 0, 1;
    -5.0287, 0, 0, 0];

    B = [0;
    -6.1115;
    0;
    1.6807];

    C = [1, 0, 0, 0; % Medición del ángulo
    0, 0, 1, 0]; % Medición de la posición del carro
```

```
    D = [0;
    0];

    % Diseño del controlador LQR
    Q = diag([3000, 3000, 500, 100]); % Ponderación de los estados
    R = 0.01; % Ponderación del esfuerzo de control
    K = lqr(A, B, Q, R);

    disp('Ganancia de retroalimentación K:');
    disp(K);

    % Diseño del observador (ganancia L)
    poles_observador = [-10, -11, -12, -13]; % Polos deseados para el
observador
    L = place(A', C', poles_observador)'; % Ganancia del observador

    disp('Ganancia del observador L:');
    disp(L);

    % Valores iniciales
    x0 = [0.2; 0; 0.3; 0]; % Estado inicial real
    x_hat0 = [0; 0; 0; 0]; % Estado inicial estimado del observador

    % Referencia deseada
    x_ref = [0.015708; 0; 0.2; 0]; % 20 cm y 3.6 grados

    % Simulación del sistema con observador
    tspan = 0:0.01:10;
    [t, x] = ode45(@(t, x) sistema_con_observador(t, x, A, B, C, K, L,
x_ref), tspan, [x0; x_hat0]);

    % Extraer estados reales y estimados de la simulación
    x_real = x(:, 1:4);
    x_est = x(:, 5:8);

    % Calcular la señal de control u para toda la simulación
    u = zeros(length(t), 1);
    for i = 1:length(t)
    x_est_i = x_est(i, :)';
    u(i) = max(-255, min(255, -K * (x_est_i - x_ref))); % Aplicar
saturación
    end

    % Graficar resultados
    figure;
```

```
    subplot(3, 1, 1);
    plot(t, x_real(:, 1), 'b', 'LineWidth', 1.5); hold on;
    plot(t, x_est(:, 1), 'r--', 'LineWidth', 1.5);
    xlabel('Tiempo (s)');
    ylabel('Ángulo (rad)');
    title('Ángulo del Péndulo: Real vs Estimado');
    legend('Ángulo Real', 'Ángulo Estimado');
    grid on;

    subplot(3, 1, 2);
    plot(t, x_real(:, 3), 'g', 'LineWidth', 1.5); hold on;
    plot(t, x_est(:, 3), 'm--', 'LineWidth', 1.5);
    xlabel('Tiempo (s)');
    ylabel('Posición del Carro (m)');
    title('Posición del Carro: Real vs Estimado');
    legend('Posición Real', 'Posición Estimada');
    grid on;

    subplot(3, 1, 3);
    plot(t, u, 'k', 'LineWidth', 1.5);
    xlabel('Tiempo (s)');
    ylabel('Señal de Control u');
    title('Señal de Control (Saturada)');
    grid on;

    % Función que define el sistema con observador
    function dx = sistema_con_observador(t, x, A, B, C, K, L, x_ref)
    % Separar los estados reales y estimados
    x_real = x(1:4);
    x_est = x(5:8);
    % Calcular la señal de control usando los estados estimados con
saturación
    u = max(-255, min(255, -K * (x_est - x_ref))); % Saturar u entre -
255 y 255
    % Dinámica del sistema real
    dx_real = A * x_real + B * u;
    % Dinámica del observador
    dx_est = A * x_est + B * u + L * (C * x_real - C * x_est);
    % Combinar ambas dinámicas
    dx = [dx_real; dx_est];
    end
```

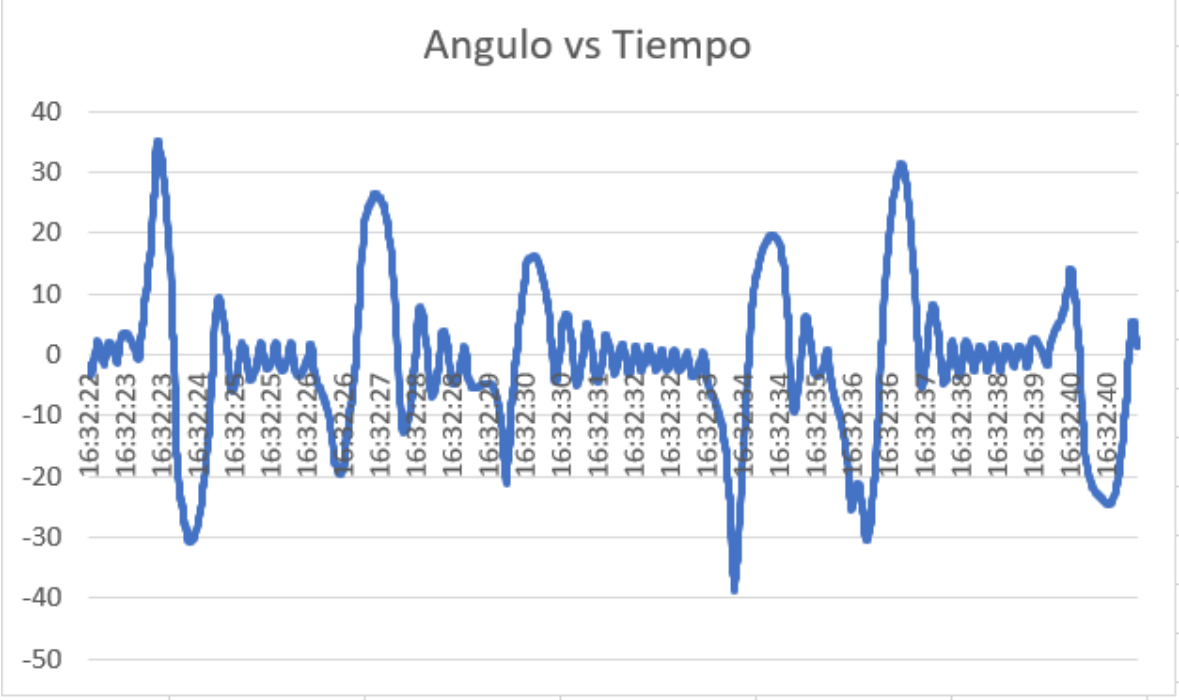
Apéndice G.Resultados experimentales por medio de Arduino

En esta sección cabe resaltar que se tomaron las iteraciones descartadas en la práctica experimental. Sin embargo, previamente fueron seleccionadas debido que el sistema evidencio cambios significativos, con respecto a la gran variedad de combinaciones realizadas con anterioridad.

Retroalimentación de estados (sin peso)

Segunda iteración:

Figura G1. Angulo vs Tiempo



Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 3736

Valores tomados: 3236/3736 (86.61% de los valores totales)

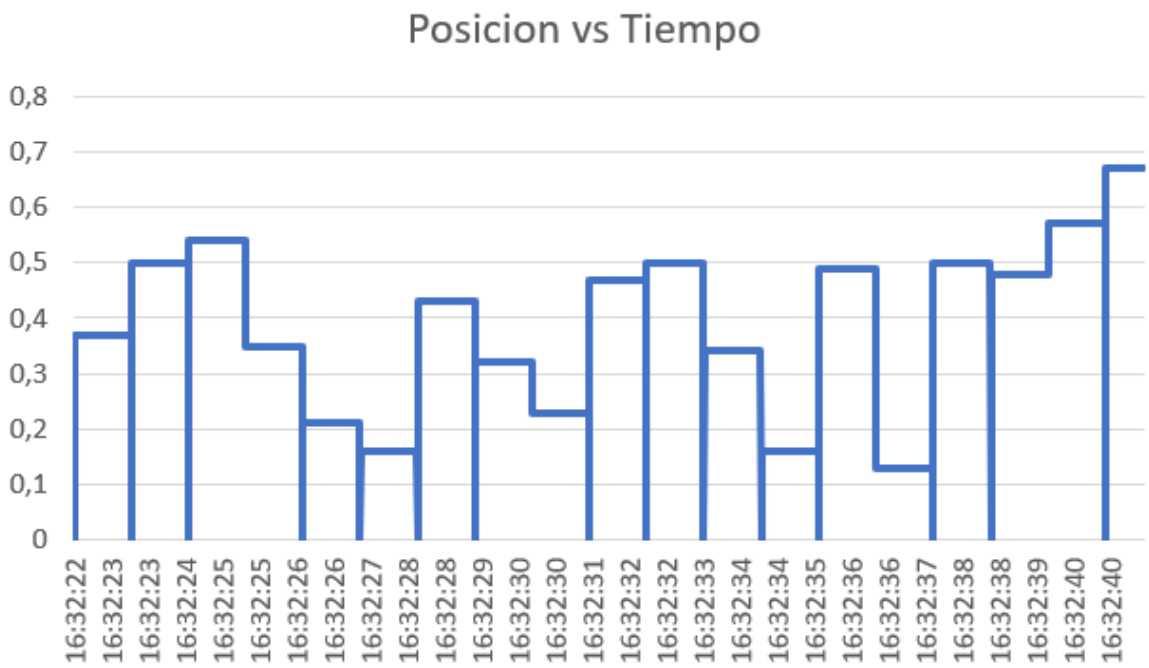
Tiempo evaluado: 18 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 6

Tiempo de estabilidad promedio: 3 segundos.

Figura G2. Posición vs Tiempo.



Nota: Los valores de la Posición están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 3736

Valores tomados: 3236/3736 (86.61% de los valores totales)

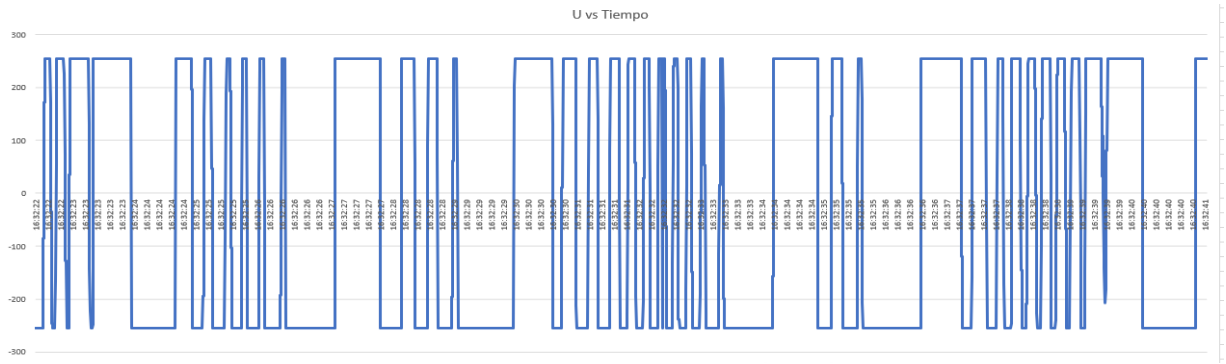
Tiempo evaluado: 18 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones: 16

Tiempo de estabilidad promedio: 2 segundos.

Figura G3. Fuerza de control vs tiempo.



Nota: Los valores de la Fuerza de control están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 3736

Valores tomados: 3236/3736 (86.61% de los valores totales)

Tiempo evaluado: 18 segundos

Tiempo de estabilidad máxima: 4 segundos

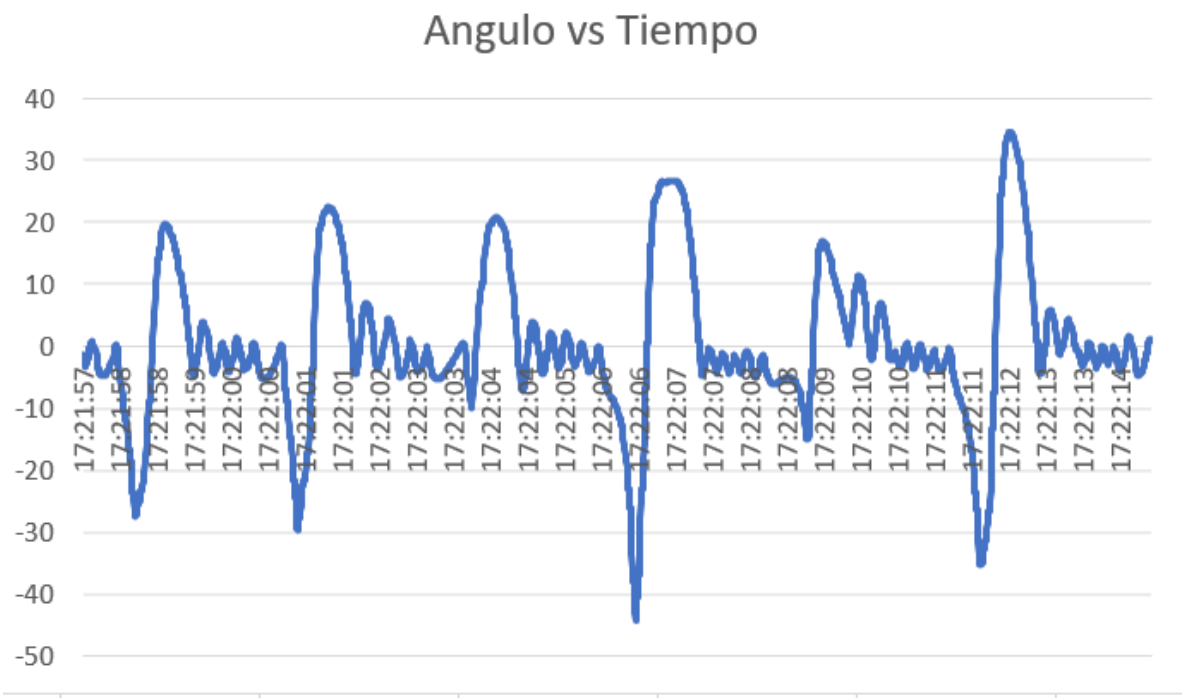
Numero de interrupciones (oscilación más frecuente): 6

Tiempo de estabilidad promedio: 3.30 segundos.

Retro de estados (Sin peso):

Tercera iteración:

Figura G4. Ángulo vs Tiempo



Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 2975

Valores tomados: 2975/2975 (100% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 6

Tiempo de estabilidad promedio: 2.30 segundos.

Figura G5. Posición vs Tiempo

Nota: Los valores de la Posición están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 2975

Valores tomados: 2975/2975 (100% de los valores totales)

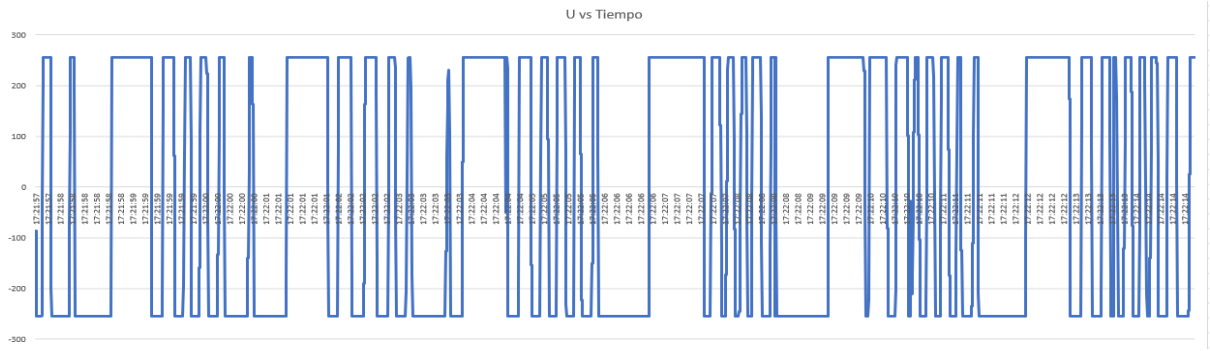
Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones: 14

Tiempo de estabilidad promedio: 2 segundos.

Figura G6. Fuerza de control vs tiempo.



Nota: Los valores de la Fuerza de control están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 2975

Valores tomados: 2975/2975 (100% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 2.40 segundos

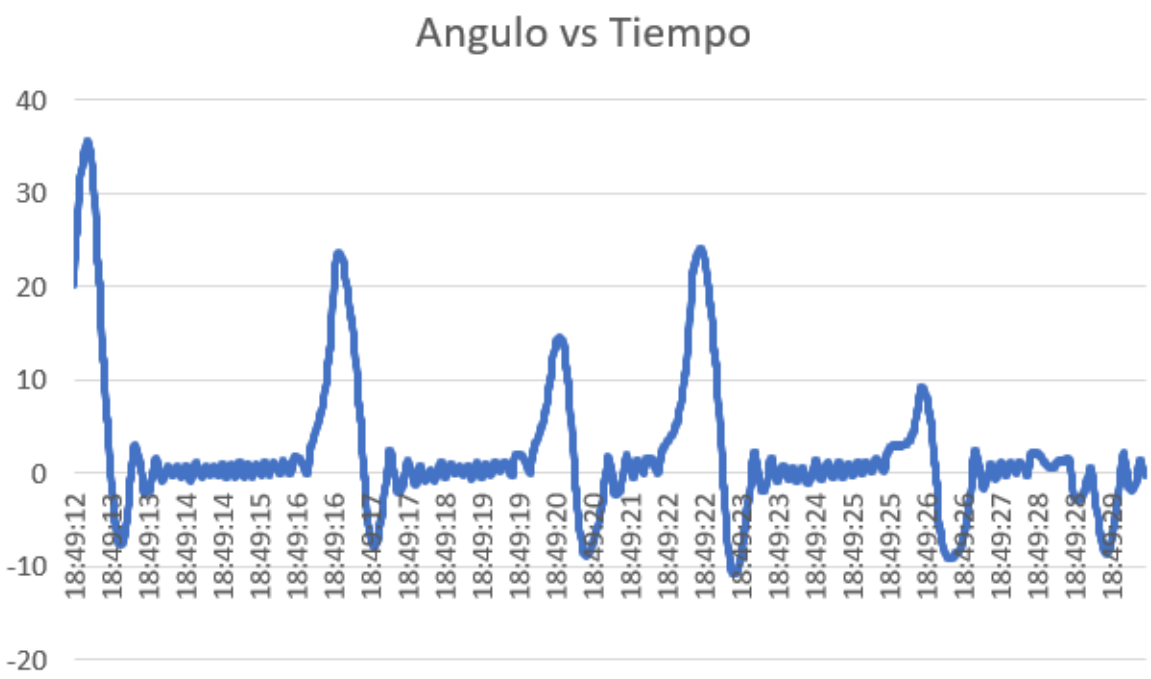
Numero de interrupciones (oscilación más frecuente): 7

Tiempo de estabilidad promedio: 2.30 segundos

Retroalimentación de estados (Con peso en el péndulo):

Primera iteración:

Figura G7. Ángulo vs Tiempo



Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 3808

Valores tomados: 3008/3808 (78.99% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 5

Tiempo de estabilidad promedio: 3 segundos.

Figura G8. Posición vs tiempo

Nota: Los valores de la Posición están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 3808

Valores tomados: 3008/3808 (78.99% de los valores totales)

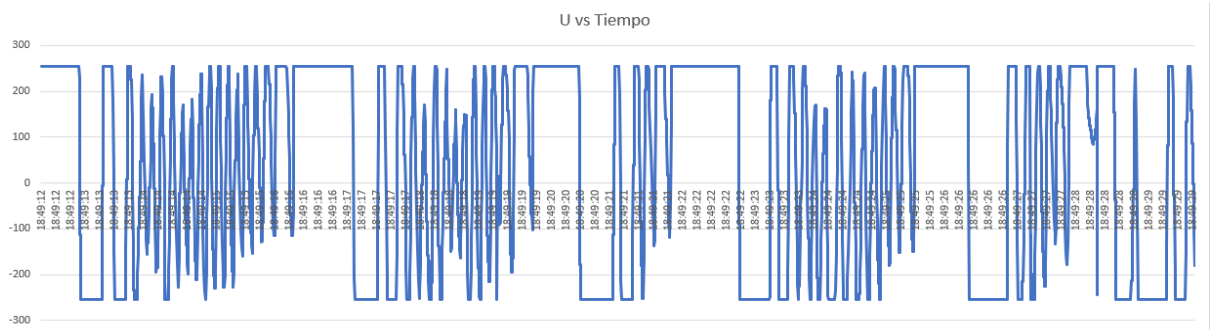
Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones: 12

Tiempo de estabilidad promedio: 2 segundos.

Figura G9. Fuerza de control vs tiempo.



Nota: Los valores de la Fuerza de control están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores Totales (Registrados por Arduino): 3808

Valores tomados: 3008/3808 (78.99% de los valores totales)

Tiempo evaluado: 17 segundos

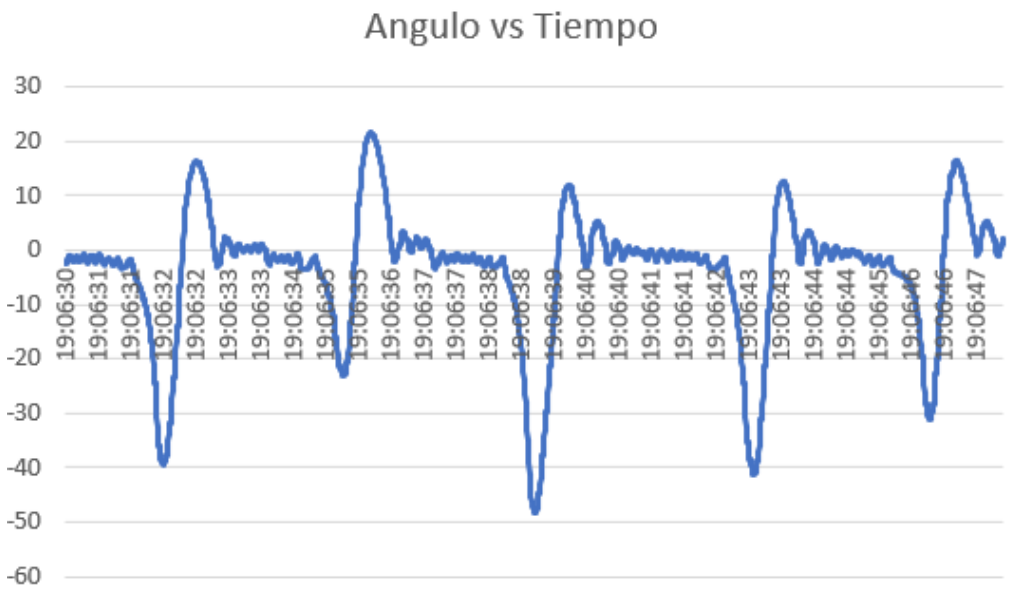
Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones (oscilación más frecuente): 5

Tiempo de estabilidad promedio: 2.30 segundos

Tercera iteración:

Figura G10. Ángulo vs Tiempo



Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 3599

Valores tomados: 2999/3599 (83.32% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 5

Tiempo de estabilidad promedio: 2.20 segundos.

Figura G11. Posición vs tiempo.

Nota: Los valores de la Posición están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 3599

Valores tomados: 2999/3599 (83.32% de los valores totales)

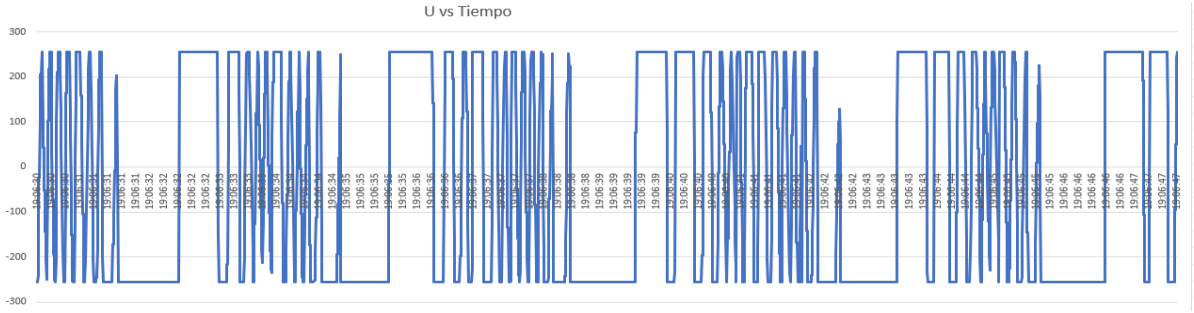
Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones: 16

Tiempo de estabilidad promedio: 1 segundo.

Figura G12. Fuerza de control vs tiempo.



Nota: Los valores de la Fuerza de control están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 3599

Valores tomados: 2999/3599 (83.32% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 3 segundos

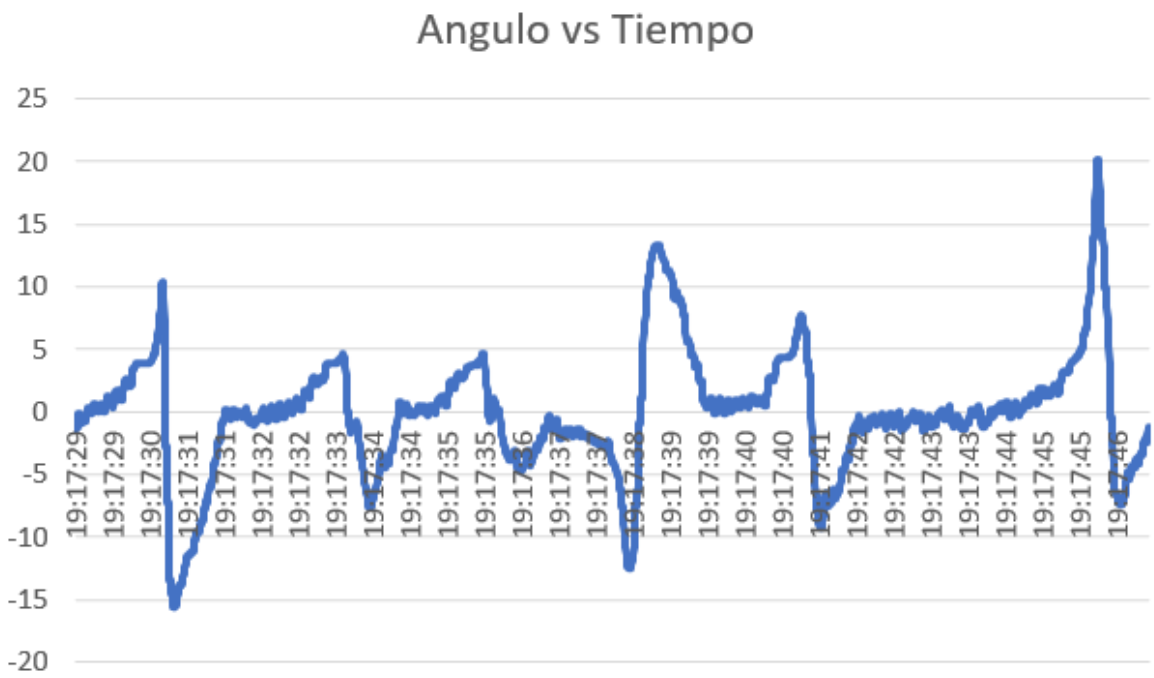
Numero de interrupciones (oscilación más frecuente): 5

Tiempo de estabilidad promedio: 2.10 segundos

Retroalimentación de estados con observador Luenberger (Sin peso)

Primera iteración:

Figura G13. Ángulo vs Tiempo.



Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 4658

Valores tomados: 3058/4658 (65.65% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 5.10 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 6

Tiempo de estabilidad promedio: 2.20 segundos.

Figura G14. Posición vs tiempo.

Nota: Los valores de la Posición están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 4658

Valores tomados: 3058/4658 (65.65% de los valores totales)

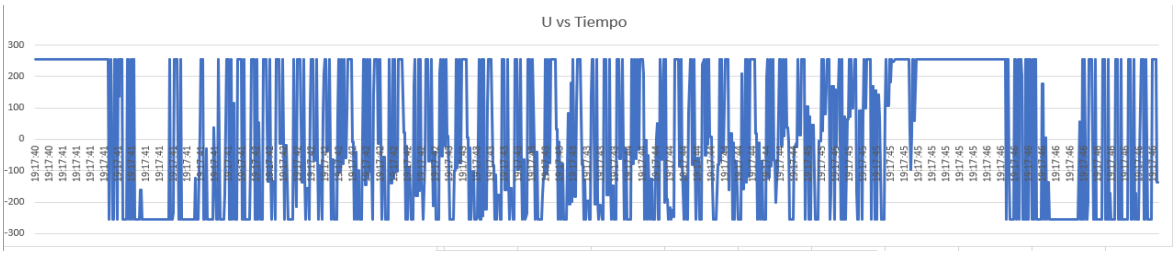
Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 2.20 segundos

Numero de interrupciones: 16

Tiempo de estabilidad promedio: 1.30 segundo

Figura G15. Fuerza de control vs tiempo.



Nota: Los valores de la Fuerza de control están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal), también queda resaltar que para este grafico en el eje horizontal se evaluaron únicamente 6 segundos debido a la alta cantidad de oscilaciones.

Valores totales (Registrados por Arduino): 4658

Valores tomados: 1058/4658 (22.71% de los valores totales)

Tiempo evaluado: 6 segundos

Tiempo de estabilidad máxima: 1 segundo

Numero de interrupciones (oscilación más frecuente): 48

Tiempo de estabilidad promedio: 0.5 segundos

Segunda iteración

Figura G16. Ángulo vs Tiempo.

Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 3123

Valores tomados: 3123/3123 (100% de los valores totales)

Tiempo evaluado: 18 segundos

Tiempo de estabilidad máxima: 5.20 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 5

Tiempo de estabilidad promedio: 3 segundos.

Figura G17. Posición vs tiempo.

Nota: Los valores de la Posición están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 3123

Valores tomados: 3123/3123 (100% de los valores totales)

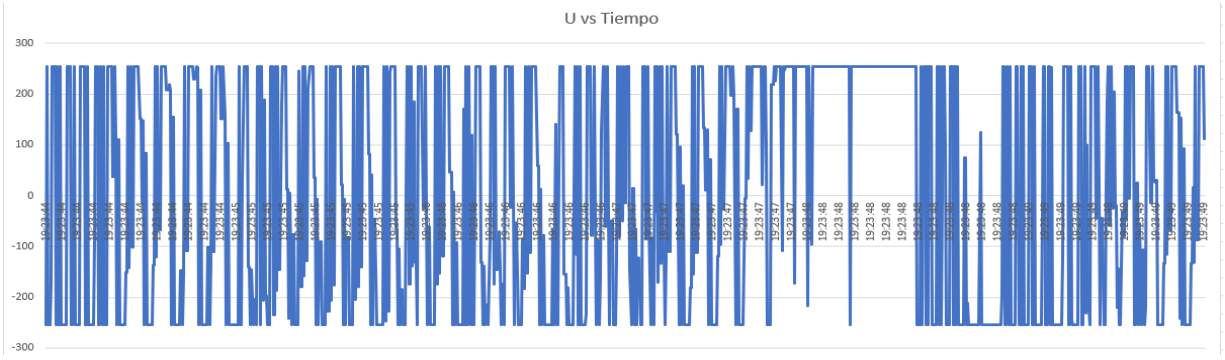
Tiempo evaluado: 18 segundos

Tiempo de estabilidad máxima: 2.20 segundos

Numero de interrupciones: 16

Tiempo de estabilidad promedio: 1.30 segundo

Figura G18. Fuerza de control vs tiempo.



Nota: Los valores de la Fuerza de control están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal), también queda resaltar que para este grafico en el eje horizontal se evaluaron únicamente 5 segundos debido a la alta cantidad de oscilaciones.

Valores totales (Registrados por Arduino): 3123

Valores tomados: 1023/3123 (32.75% de los valores totales)

Tiempo evaluado: 6 segundos

Tiempo de estabilidad máxima: 1 segundo

Numero de interrupciones (oscilación más frecuente): 52

Tiempo de estabilidad promedio: 0.4 segundos

Apéndice H. Retroalimentación de estados con observador Luenberger (con peso en el péndulo)

Nota: No se toma en cuenta el análisis del análisis con Fuzzy ya que en el contenido de este trabajo se tomó en cuenta solo un diseño de dicho controlador.

Segunda iteración

Figura H1. Ángulo vs Tiempo

Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 4700

Valores tomados: 3000/4700 (63.82% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 5

Tiempo de estabilidad promedio: 2.30 segundos.

Figura H2. Posición vs Tiempo.

Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 4700

Valores tomados: 3000/4700 (63.82% de los valores totales)

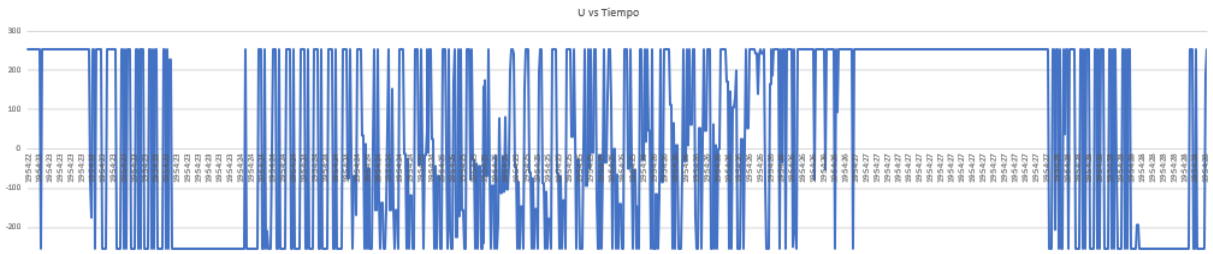
Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 3 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 15

Tiempo de estabilidad promedio: 1 segundo

Figura H3. Fuerza de control vs tiempo.



Nota: Los valores de la Fuerza de control están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal), también queda resaltar que para este grafico en el eje horizontal se evaluaron únicamente 6 segundos debido a la alta cantidad de oscilaciones.

Valores totales (Registrados por Arduino): 4700

Valores tomados: 1000/4700 (21.27% de los valores totales)

Tiempo evaluado: 6 segundos

Tiempo de estabilidad máxima: 0.25 segundos

Numero de interrupciones (oscilación más frecuente): 46

Tiempo de estabilidad promedio: 0.6 segundos

Tercera iteración

Figura H4. Ángulo vs Tiempo.

Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 8149

Valores tomados: 3000/8149 (36.81% de los valores totales)

Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 3.30 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 5

Tiempo de estabilidad promedio: 2.40 segundos.

Figura H5. Posición vs Tiempo.

Nota: Los valores del Angulo están en grados (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal).

Valores totales (Registrados por Arduino): 8149

Valores tomados: 3000/8149 (36.81% de los valores totales)

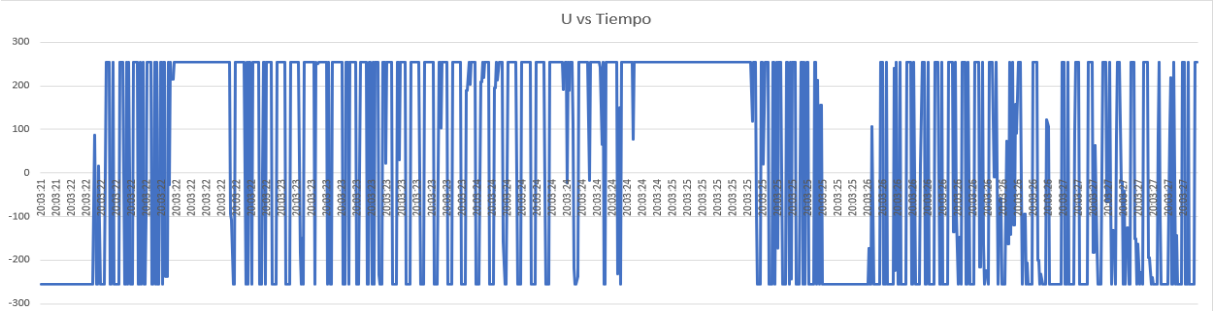
Tiempo evaluado: 17 segundos

Tiempo de estabilidad máxima: 1 segundos

Numero de interrupciones (ver picos y valles de la gráfica): 16

Tiempo de estabilidad promedio: 1 segundo

Figura H6. Fuerza de control vs tiempo.



Nota: Los valores de la Fuerza de control están en metros (Eje vertical) y el tiempo está en formato de 24 horas (Eje horizontal), también queda resaltar que para este grafico en el eje horizontal se evaluaron únicamente 6 segundos debido a la alta cantidad de oscilaciones.

Valores totales (Registrados por Arduino): 8149

Valores tomados: 1000/8149 (21.27% de los valores totales)

Tiempo evaluado: 6 segundos

Tiempo de estabilidad máxima: 0.54 segundos

Numero de interrupciones (oscilación más frecuente): 42

Tiempo de estabilidad promedio: 0.36 segundos

Apéndice I. Simulaciones con Controlador LQR y LQR con Observador Luenberger (Con peso y sin peso).

Análisis del Control de Retroalimentación de Estados con Peso

(masa contrapeso= 0.305 Kg)

A continuación, se procede a realizar las simulaciones de la estrategia de control por retroalimentación de estados, en el sistema de péndulo invertido móvil, con un peso suspendido en la parte superior del péndulo de masa de 305 gramos, donde se realiza una serie de iteraciones por medio del software MATLAB. En este trabajo se toman en cuenta tres 5 iteraciones en las cuales fueron los más significativos, en el comportamiento del sistema.

- **Primera iteración:**

Valores iniciales:

- Matriz Q=diag([3000,3000,100,100]):
- R= 0.01

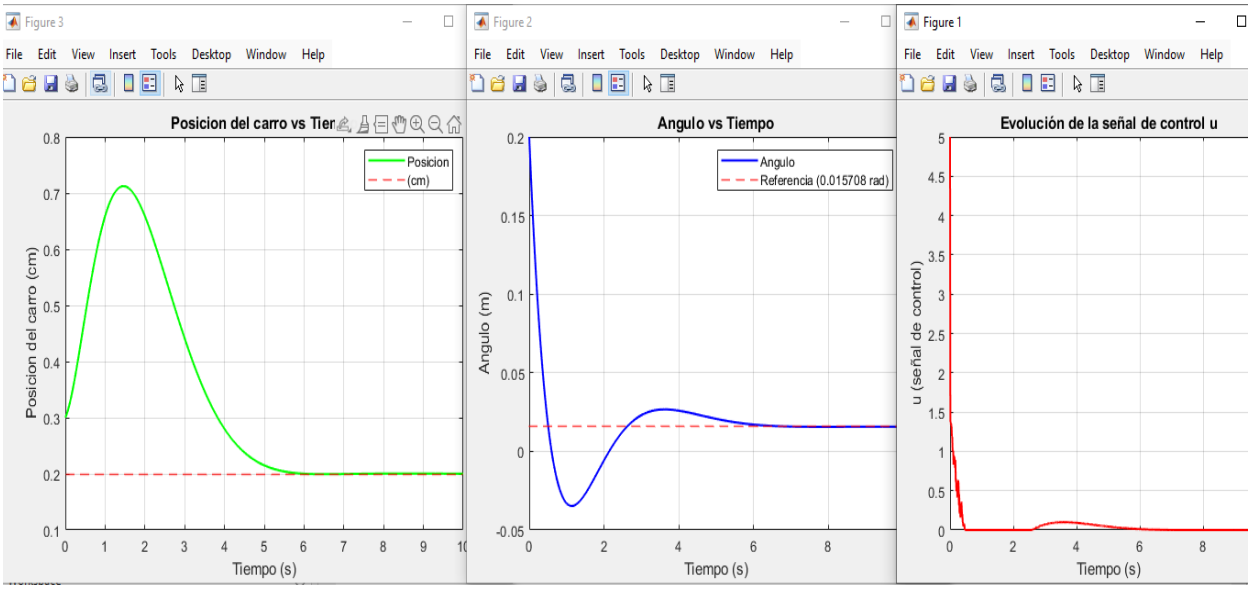
Resultados:

- Ganancias:

Figura I1. *Ganancias Nr del controlador. - Resultados de Matlab.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -1.6484   -0.6062   -0.1000   -0.2084

Ganancia N_r para la posición del carro (20 cm):
   -0.0100

Ganancia N_r para el ángulo del péndulo (3.6 grados):
   9.2183e-18
```

Figura I2. *Posición vs Tiempo – Angulo vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados con peso.*



❖ **Segunda iteración:**

Valores iniciales:

- Matriz Q=diag([3000,3000,500,500]):
- R= 0.01

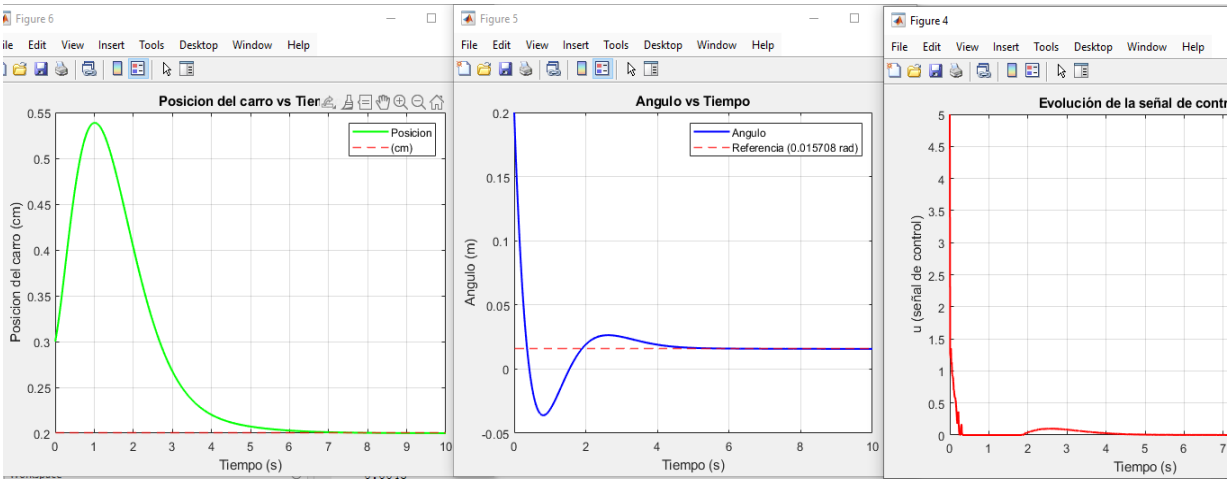
Resultados:

- Ganancias:

Figura I3. *Ganancias Nr del controlador. - Resultados de Matlab. - segunda iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.2621   -0.6593   -0.2236   -0.3908

Ganancia N_r para la posición del carro (20 cm):
   -0.0045

Ganancia N_r para el ángulo del péndulo (3.6 grados):
  -2.2533e-17
```

Figura I4. *Posición vs Tiempo – Angulo vs Tiempo – Fuerza de control vs Tiempo - Retroalimentación de estados con peso (Segunda iteración)*



❖ **Tercera iteración:**

Valores iniciales:

- Matriz Q=diag([3000,2000,3000,100]):
- R= 0.01

Resultados:

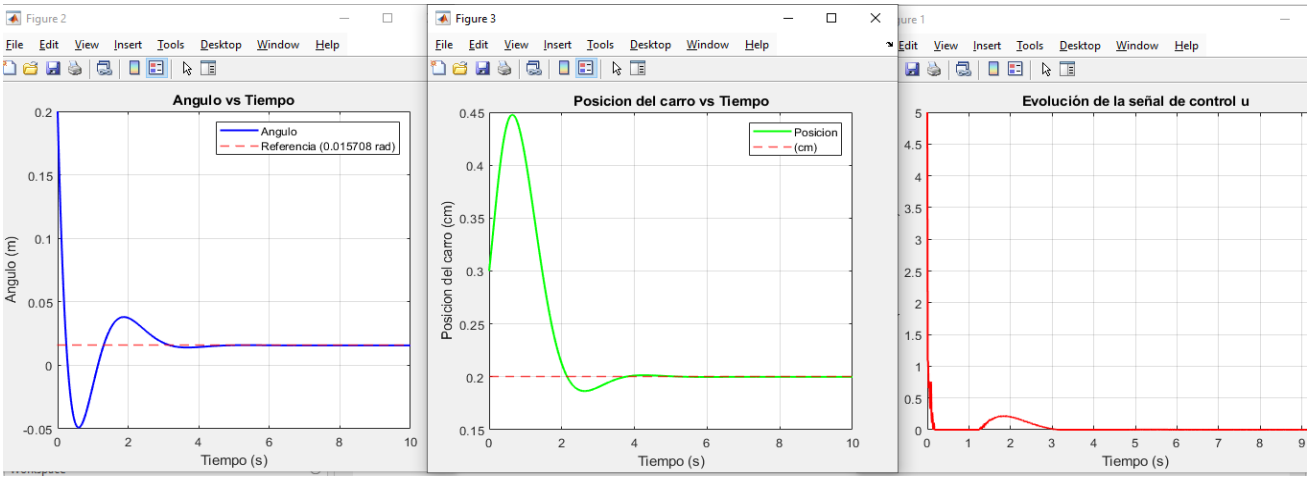
- Ganancias:

Figura I5. *Ganancias Nr del controlador. - Resultados de Matlab. - Tercera iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.3972   -0.5935   -0.5477   -0.5260

Ganancia N_r para la posición del carro (20 cm):
   -0.0018

Ganancia N_r para el ángulo del péndulo (3.6 grados):
  -1.5029e-17
```

- Graficas:

Figura I6. *Posición vs Tiempo – Angulo vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados con peso (Tercera iteración).*



❖ **Cuarta iteración:**

Valores iniciales:

- Matriz Q=diag([2000,2000,3000,1000]):
- R= 0.01

Resultados:

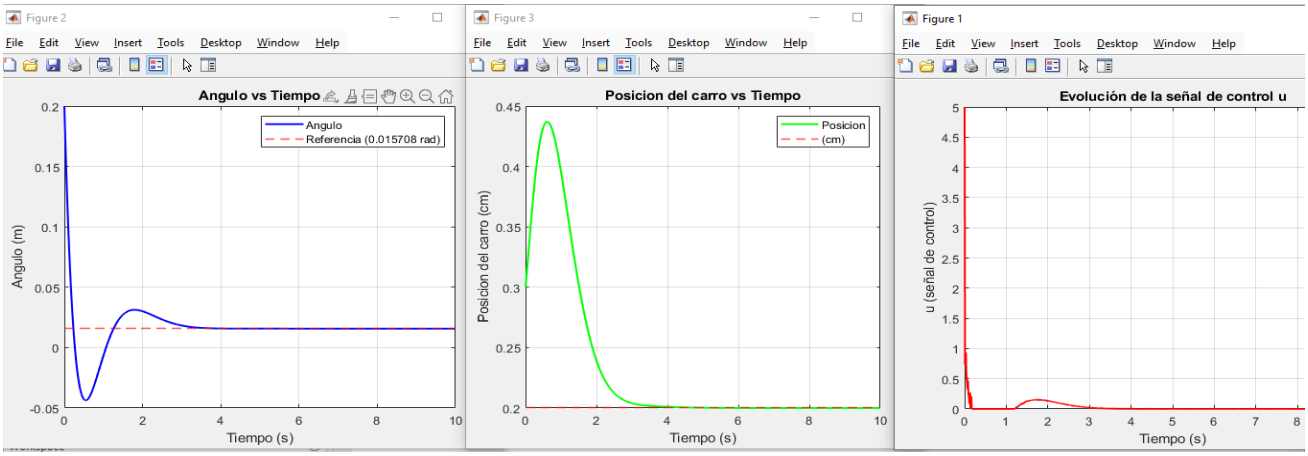
- Ganancias:

Figura I7. *Ganancias Nr del controlador. - Resultados de Matlab. - Cuarta iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.6940   -0.6304   -0.5477   -0.6323

Ganancia N_r para la posición del carro (20 cm):
   -0.0018

Ganancia N_r para el ángulo del péndulo (3.6 grados):
   4.0492e-18
```

- Graficas:

Figura I8. *Posición vs Tiempo – Angulo vs Tiempo – Fuerza de control vs Tiempo – Retroalimentacion de estados con peso (Cuarta iteracion).*



❖ **Quinta iteración:**

Valores iniciales:

- Matriz Q=diag([3000,3000,3000,500]):
- R= 0.01

Resultados:

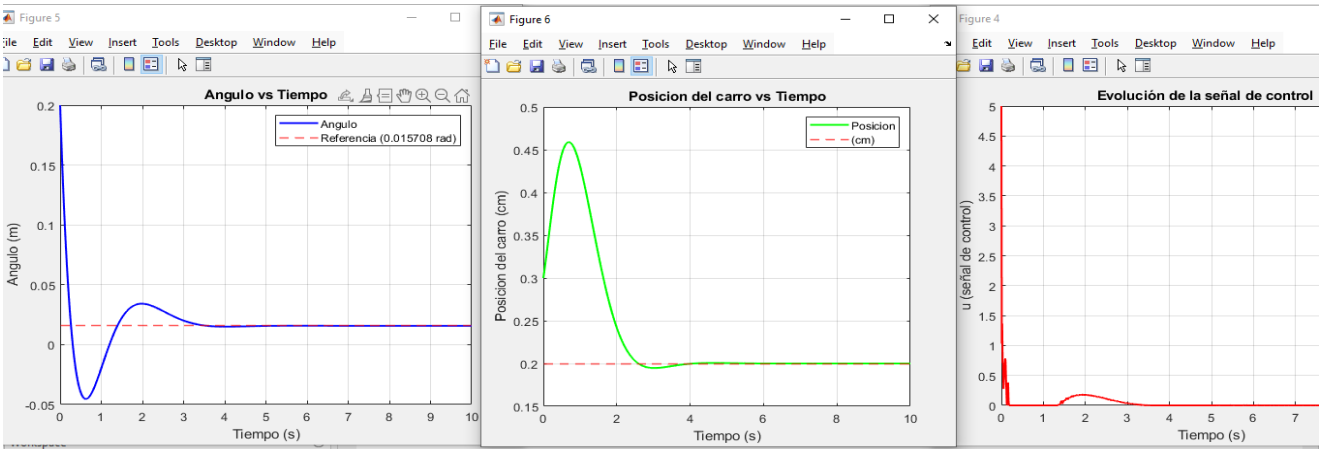
- Ganancias:

Figura I9. *Ganancias Nr del controlador. - Resultados de Matlab. - Quinta iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.8304   -0.7181   -0.5477   -0.6042

Ganancia N_r para la posición del carro (20 cm):
   -0.0018

Ganancia N_r para el ángulo del péndulo (3.6 grados):
  -1.6924e-18
```

- Graficas:

Figura I10. *Posición vs Tiempo – Angulo vs Tiempo – Fuerza de control vs Tiempo – Retroalimentacion de estados con peso (Quinta iteración).*



ANÁLISIS DEL CONTROL DE RETROALIMENTACION DE ESTADOS SIN PESO

(masa contrapeso= 0 Kg)

❖ **Primera iteración:**

Valores iniciales:

- Matriz Q=diag([3000,3000,100,100]):
- R= 0.01

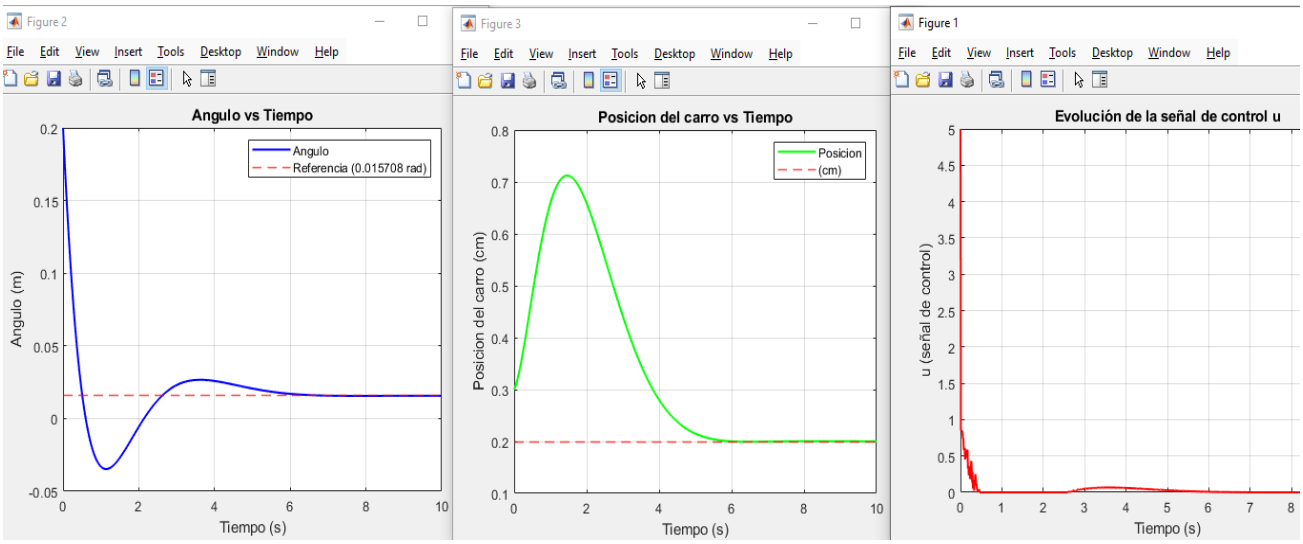
Resultados:

- Ganancias:

Figura I11. *Ganancias Nr del controlador Retroalimentación de estados. sin peso - Resultados de Matlab. - Primera iteración*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -1.6454   -0.6062   -0.1000   -0.2084

Ganancia N_r para la posición del carro (20 cm):
   -0.0100

Ganancia N_r para el ángulo del péndulo (3.6 grados):
   1.5867e-17
```

Figura I12. *Posición vs Tiempo – Angulo vs Tiempo – Fuerza de control vs Tiempo – Retroalimentacion de estados con peso (Primera iteración).*



❖ **Segunda iteración:**

Valores iniciales:

- Matriz Q=diag([3000,3000,500,500]):
- R= 0.01

Resultados:

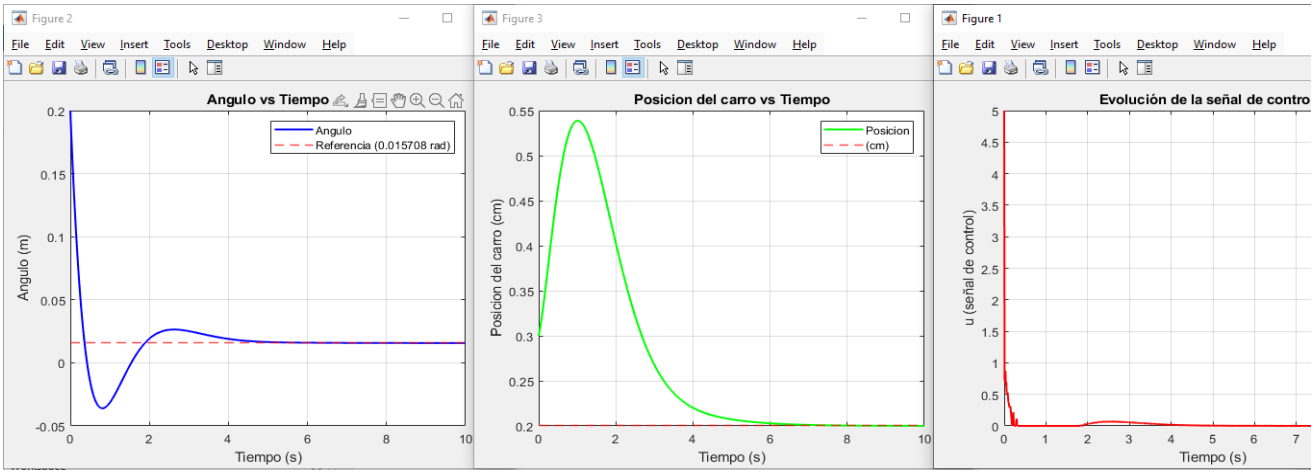
- Ganancias:

Figura I13. *Ganancias Nr del controlador Retroalimentación de estados. sin peso - Resultados de Matlab. - Segunda iteracion*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.2591   -0.6593   -0.2236   -0.3908

Ganancia N_r para la posición del carro (20 cm):
   -0.0045

Ganancia N_r para el ángulo del péndulo (3.6 grados):
  -2.2122e-17
```

- Graficas:

Figura I14. *Posición vs Tiempo – Angulo vs Tiempo – Fuerza de control vs Tiempo – Retroalimentacion de estados con peso (Segunda iteración)*



❖ **Tercera iteración:**

Valores iniciales:

- Matriz Q=diag([3000,2000,3000,100]):
- R= 0.01

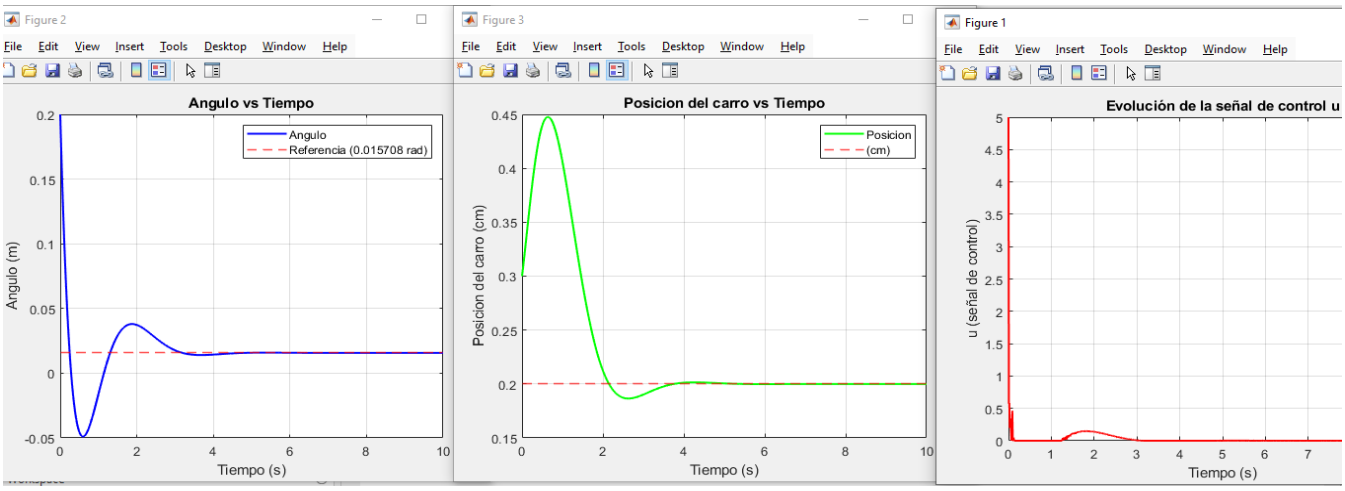
Resultados:

- Ganancias:

Figura I15. *Ganancias Nr del controlador Retroalimentación de estados. sin peso - Resultados de Matlab. - (Tercera iteración)*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.3942   -0.5935   -0.5477   -0.5260

Ganancia N_r para la posición del carro (20 cm):
   -0.0018

Ganancia N_r para el ángulo del péndulo (3.6 grados):
  -7.9711e-18
```

Figura I16. *Posición vs Tiempo – Angulo vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados con peso tercera iteración)*



❖ **Cuarta iteración:**

Valores iniciales:

- Matriz Q=diag([2000,2000,3000,1000]):
- R= 0.01

Resultados:

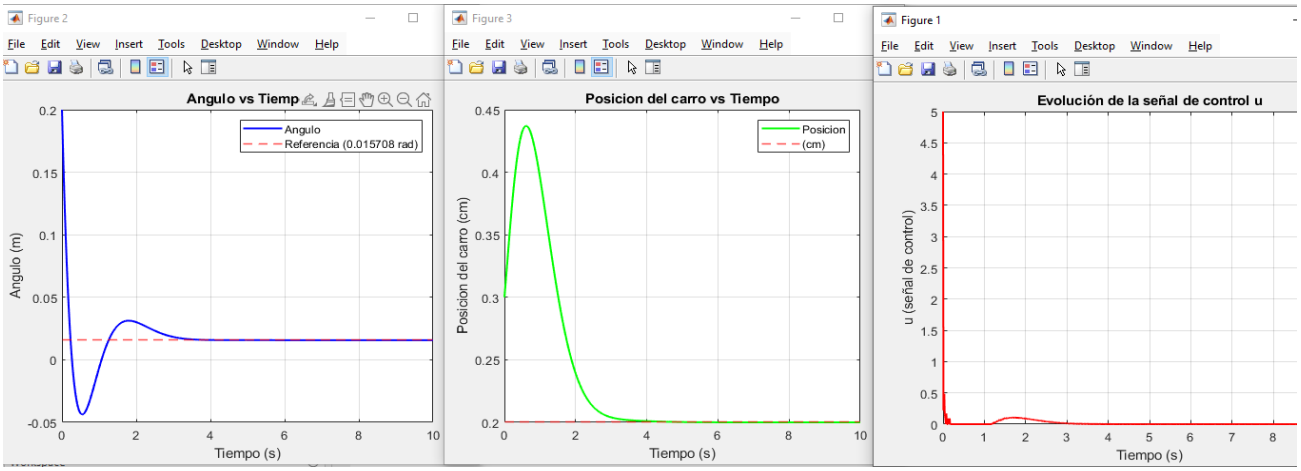
- Ganancias:

Figura I17. *Ganancias Nr del controlador Retroalimentación de estados. sin peso - Resultados de Matlab. - cuarta iteración*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.6910   -0.6304   -0.5477   -0.6323

Ganancia N_r para la posición del carro (20 cm):
   -0.0018

Ganancia N_r para el ángulo del péndulo (3.6 grados):
  -1.2448e-18
```

- Graficas:

Figura I18. *Posición vs Tiempo – Angulo vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados con peso. (cuarta iteración)*



❖ **Quinta iteración:**

Valores iniciales:

- Matriz Q=diag([3000,3000,3000,5000]):
- R= 0.01

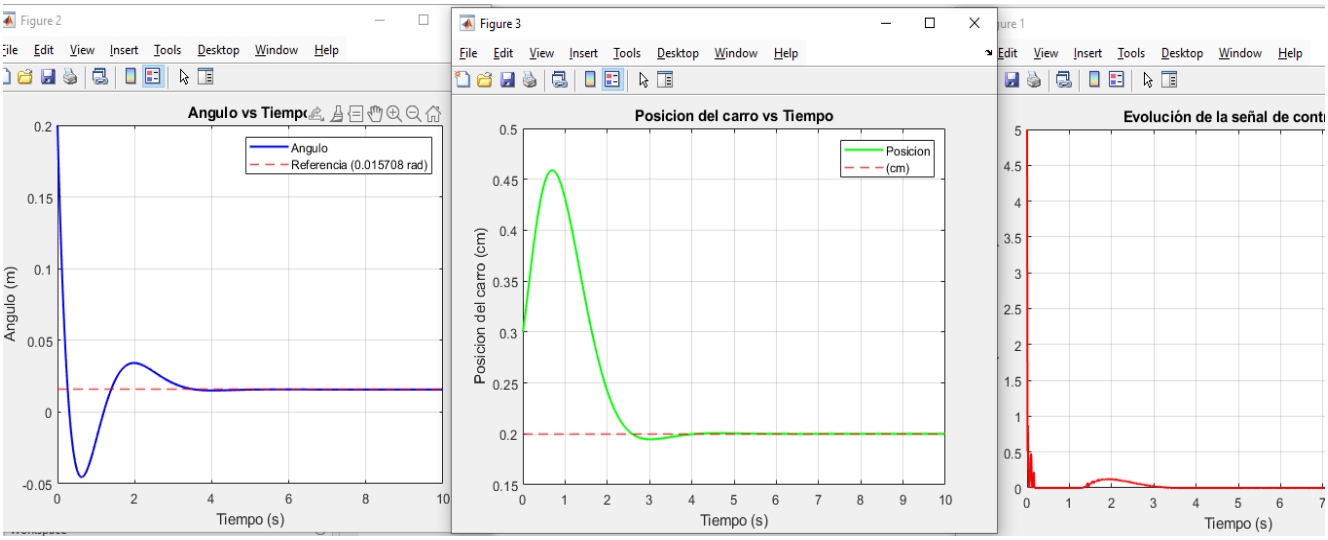
Resultados:

Figura I19. *Ganancias Nr del controlador Retroalimentación de estados. sin peso - Resultados de Matlab. - Quinta iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.8274   -0.7181   -0.5477   -0.6042

Ganancia N_r para la posición del carro (20 cm):
   -0.0018

Ganancia N_r para el ángulo del péndulo (3.6 grados):
   6.4824e-18
```

- Graficas:

Figura I20. *Posición vs Tiempo – Angulo vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados con peso. (Quinta iteración)*



ANÁLISIS DEL CONTROL DE RETROALIMENTACION DE ESTADOS CON OBSERVADOR DE LUENBERGER CON PESO

(masa contrapeso= 0.305 Kg)

- **Primera iteración:**

Valores iniciales:

- Matriz Q=diag([3000,3000,100,100]):
- R= 0.01

Resultados:

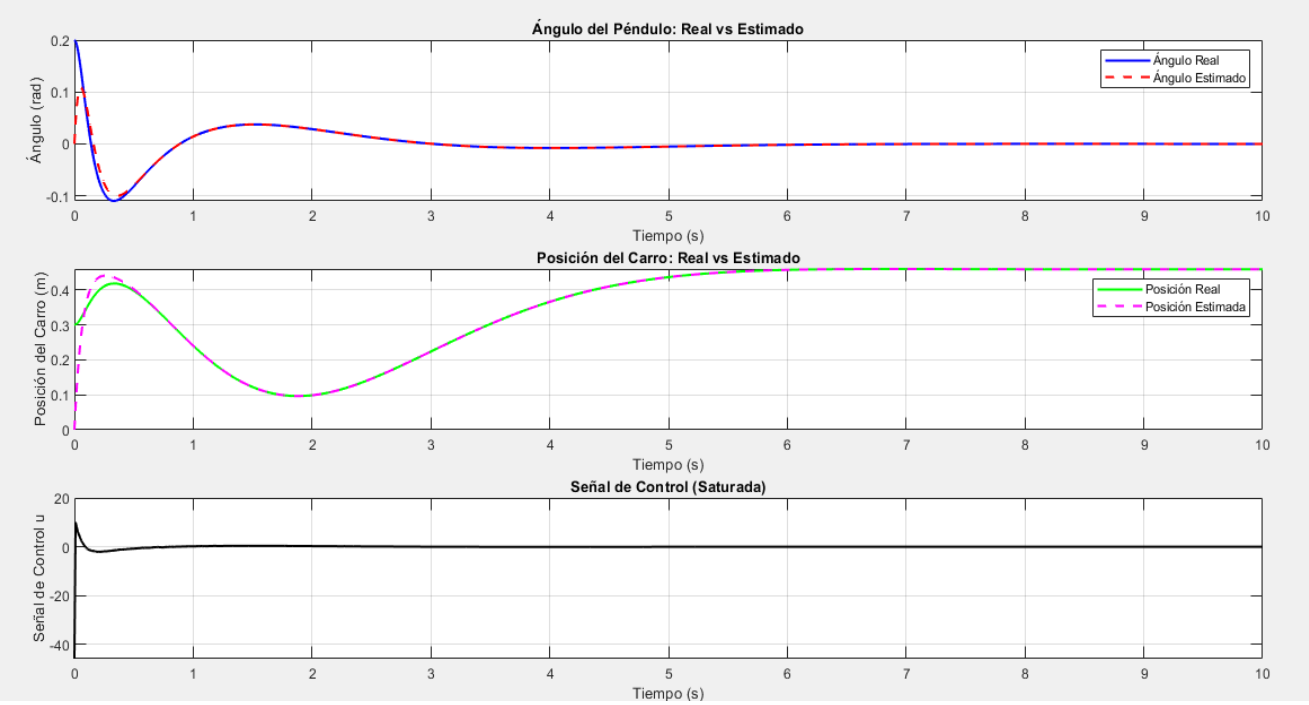
- Ganancias:

Figura I21. *Ganancias Nr del controlador Retroalimentación de estados. con observador de Luenberger con peso - Resultados de Matlab. - Primera iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -1.6484   -0.6062   -0.1000   -0.2084

Ganancia del observador L:
   22.9338    1.0388
  184.7386   11.9770
    0.9570   23.0662
    5.9544  132.2187
```

- Graficas:

Figura I22. *Ángulo vs Tiempo -Posición vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados con peso. (Primera iteración)*



- **Segunda iteración:**

Valores iniciales:

- Matriz Q=diag([3000,3000,500,500]):
- R= 0.01

Resultados:

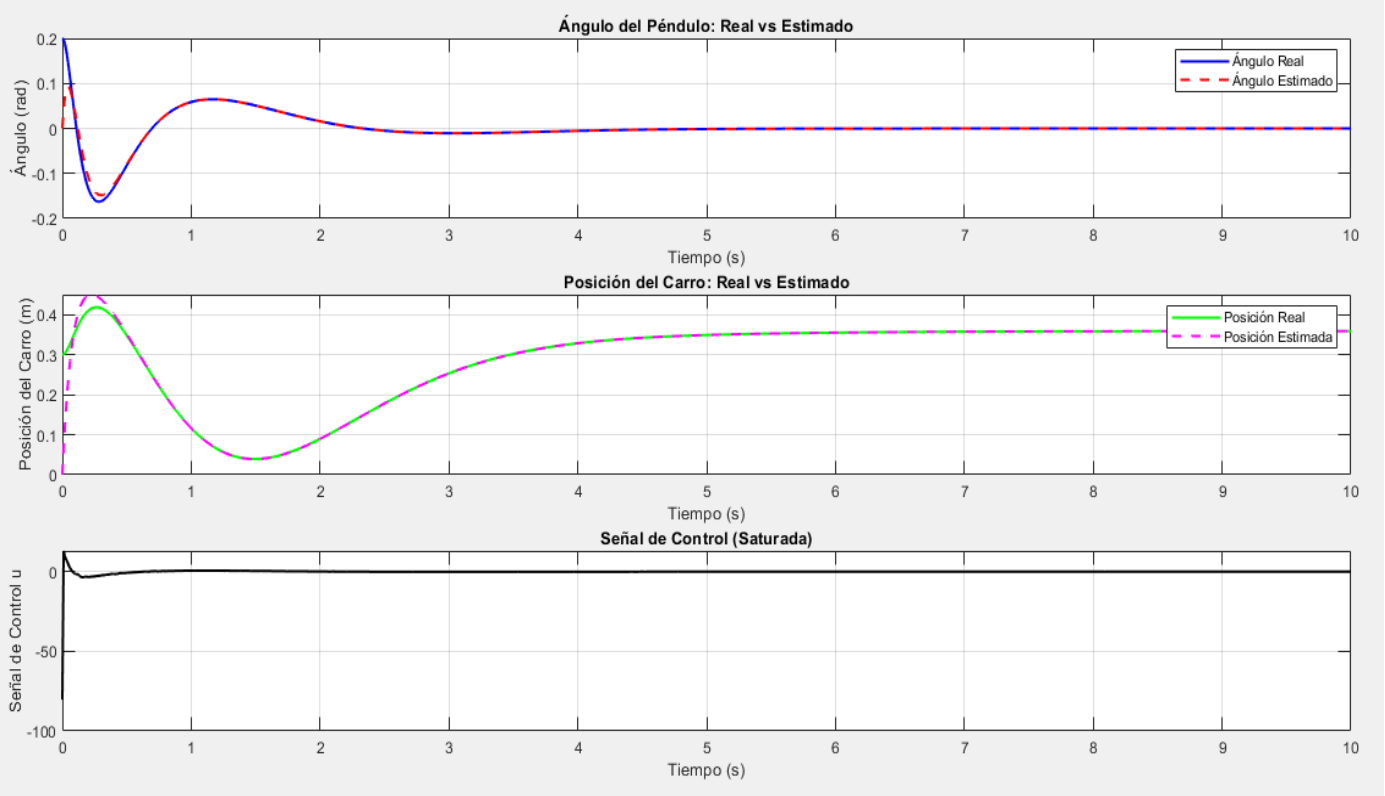
- Ganancias:

Figura I23. *Ganancias Nr del controlador Retroalimentación de estados. con observador de Luenberger con peso - Resultados de Matlab. - Segunda iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.2621   -0.6593   -0.2236   -0.3908

Ganancia del observador L:
   22.9338    1.0388
  184.7386   11.9770
    0.9570   23.0662
    5.9544  132.2187
```

- Graficas:

Figura I24. *Ángulo vs Tiempo -Posición vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados con peso. (Segunda iteración)*



❖ **Tercera iteración:**

Valores iniciales:

- Matriz Q=diag([3000,2000,3000,100]):
- R= 0.01

Resultados:

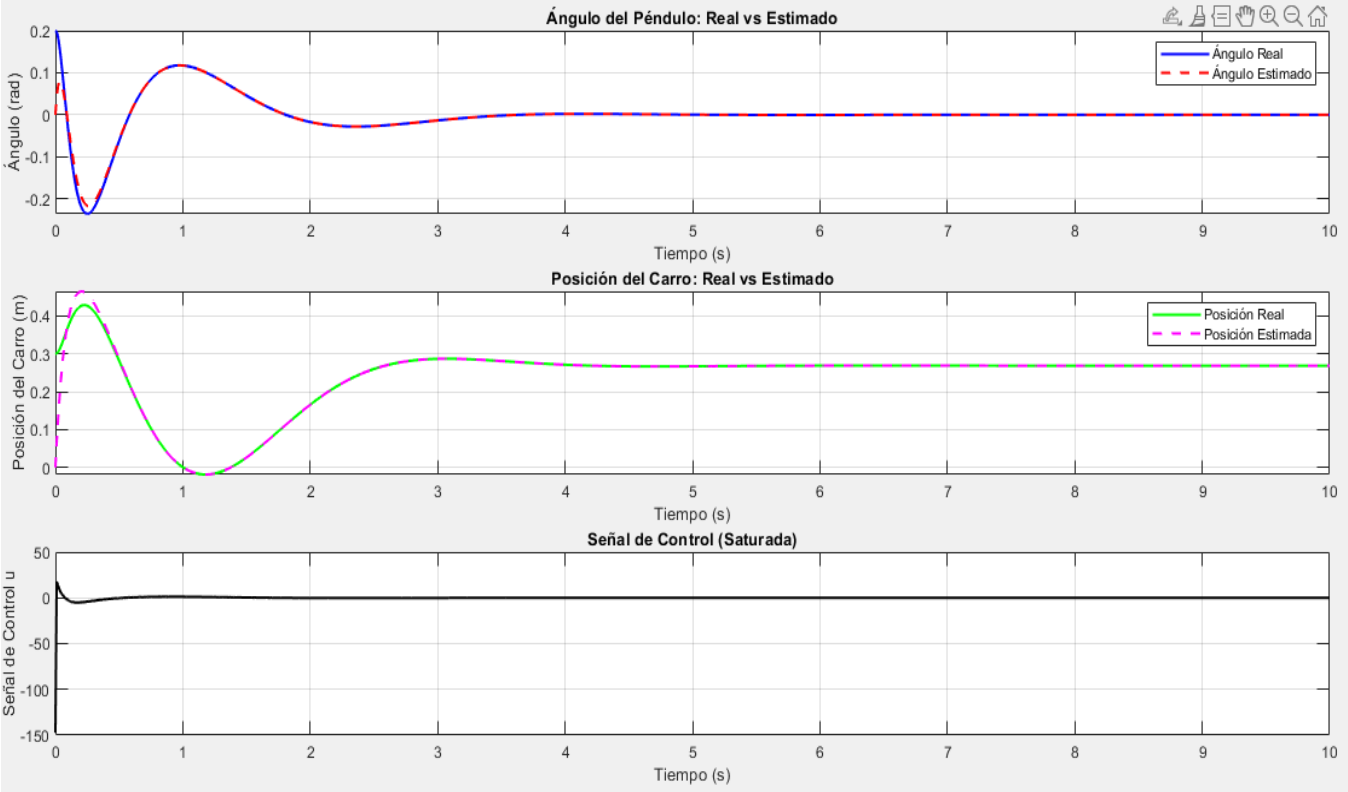
- Ganancias:

Figura I25. *Ganancias Nr del controlador Retroalimentación de estados. con observador de Luenberger con peso - Resultados de Matlab. - Tercera iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.3972   -0.5935   -0.5477   -0.5260

Ganancia del observador L:
   22.9338    1.0388
  184.7386   11.9770
    0.9570   23.0662
    5.9544  132.2187
```

- Graficas:

Figura I26. *Ángulo vs Tiempo -Posición vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados con peso. (Tercera iteración)*



❖ **Cuarta iteración:**

Valores iniciales:

- Matriz Q=diag([2000,2000,3000,1000]):
- R= 0.01

Resultados:

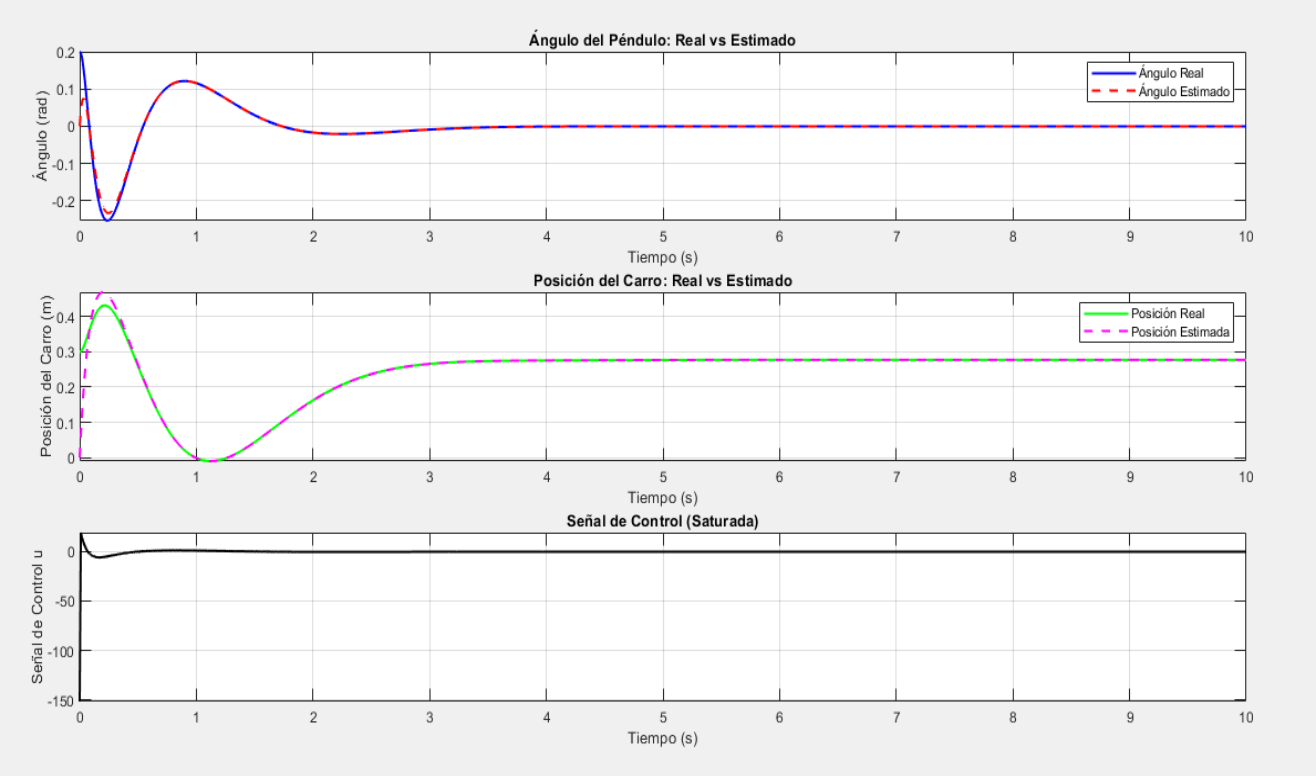
- Ganancias:

Figura I27. *Ganancias Nr del controlador Retroalimentación de estados. con observador de Luenberger con peso - Resultados de Matlab. - Cuarta iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.6940   -0.6304   -0.5477   -0.6323

Ganancia del observador L:
   22.9338    1.0388
  184.7386   11.9770
    0.9570   23.0662
    5.9544  132.2187
```

- Graficas:

Figura I28. *Ángulo vs Tiempo -Posición vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados con peso. (Cuarta iteración)*



- **Quinta iteración:**

  Valores iniciales:

- Matriz Q=diag([3000,3000,3000,500]):
- R= 0.01

  Resultados:

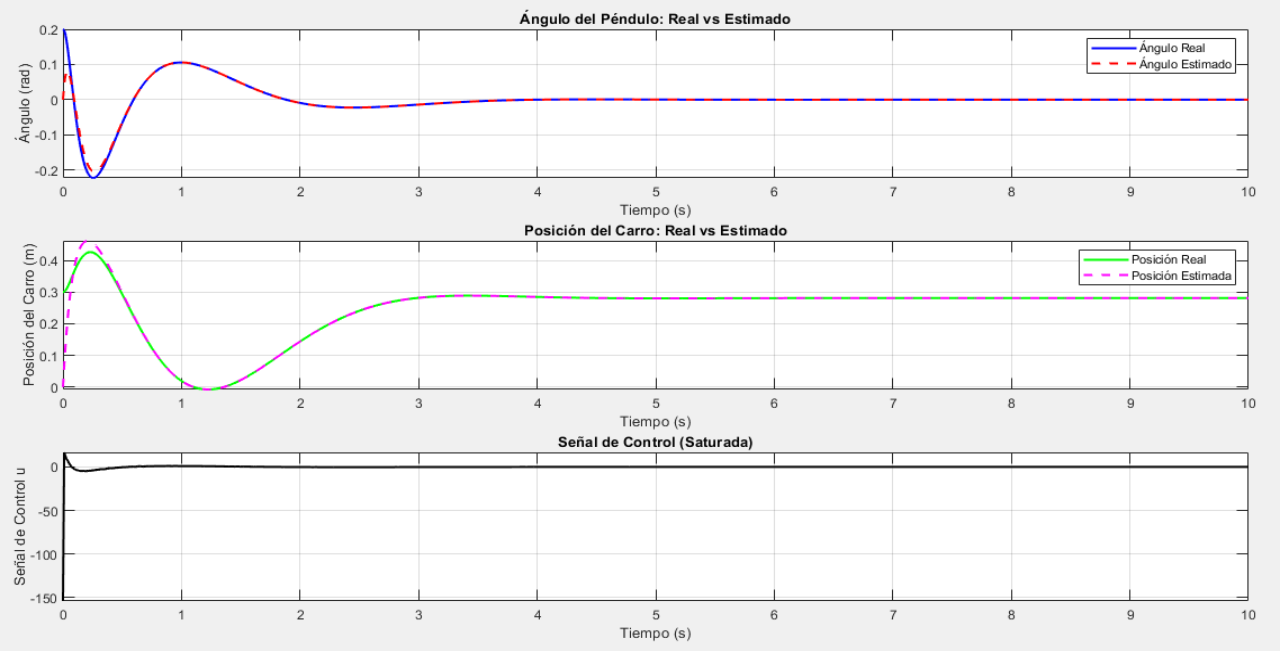
- Ganancias:

Figura I29. *Ganancias Nr del controlador Retroalimentación de estados. con observador de Luenberger con peso - Resultados de Matlab. - Quinta iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.8304   -0.7181   -0.5477   -0.6042

Ganancia del observador L:
   22.9338    1.0388
  184.7386   11.9770
    0.9570   23.0662
    5.9544  132.2187
```

- Graficas:

Figura I30. *Ángulo vs Tiempo -Posición vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados con peso. (Quinta iteración)*



ANÁLISIS DEL CONTROL DE RETROALIMENTACION DE ESTADOS CON OBSERVADOR DE LUENBERGER SIN PESO

(masa contrapeso= 0 Kg)

- **Primera iteración:**

Valores iniciales:

- Matriz Q=diag([3000,3000,100,100]):
- R= 0.01

Resultados:

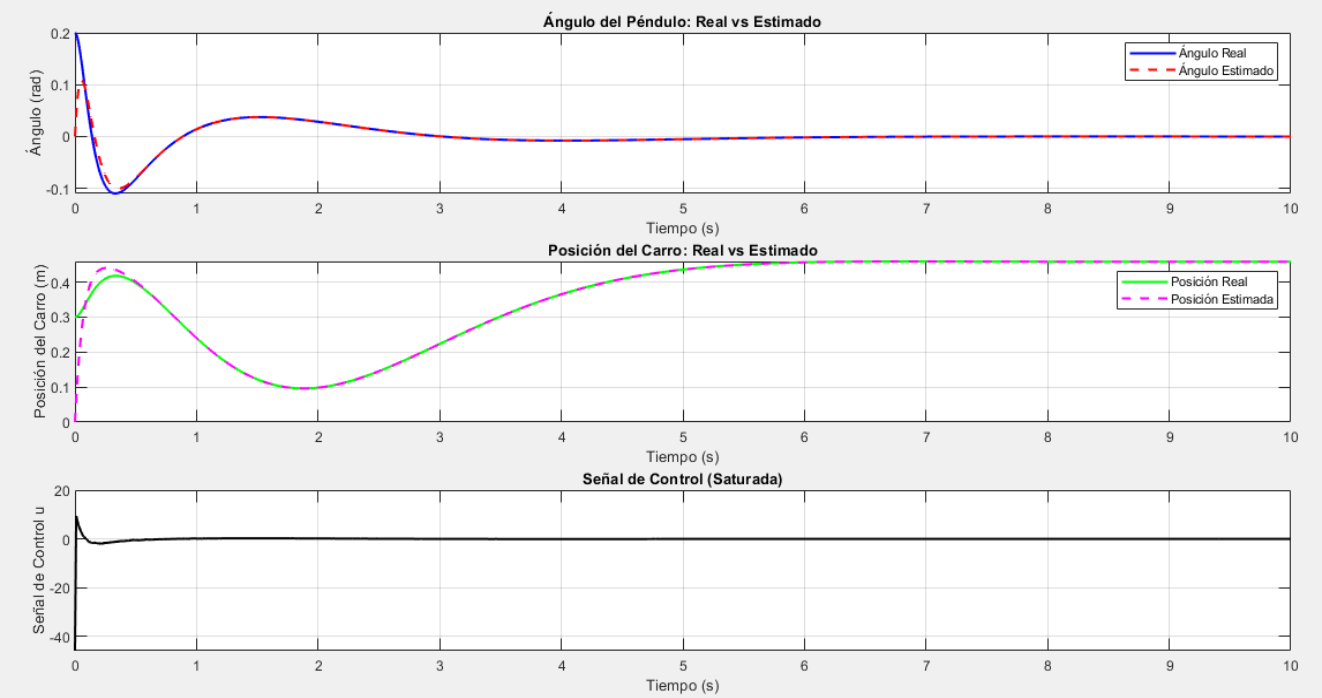
- Ganancias:

Figura I31. *Ganancias Nr del controlador Retroalimentación de estados. con observador de Luenberger sin peso - Resultados de Matlab. - Primera iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -1.6454   -0.6062   -0.1000   -0.2084

Ganancia del observador L:
   22.9338    1.0388
  166.4526   11.9770
    0.9570   23.0662
   10.9830  132.2187
```

- Graficas:

Figura I32. *Ángulo vs Tiempo -Posición vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados sin peso. (Primera iteración)*



❖ **Segunda iteración:**

Valores iniciales:

- Matriz Q=diag([3000,3000,500,500]):

- R= 0.01

Resultados:

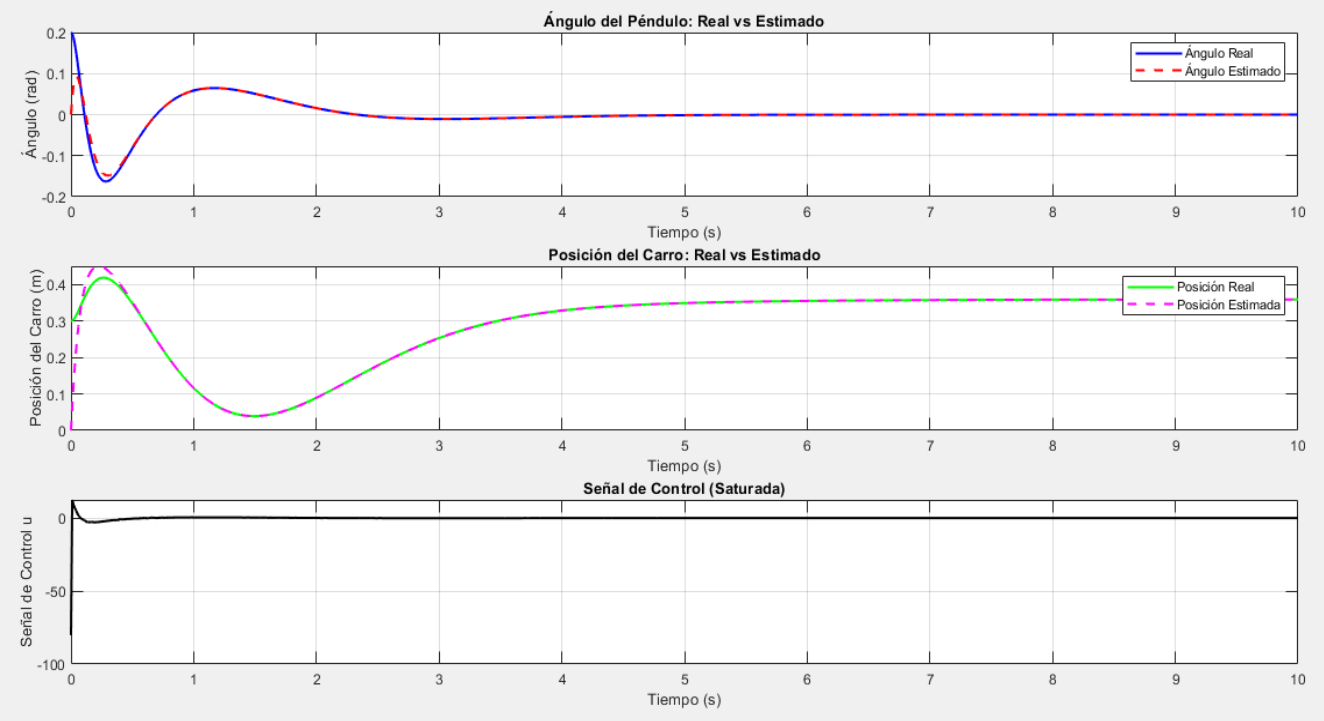
- Ganancias:

Figura I33. *Ganancias Nr del controlador Retroalimentación de estados. con observador de Luenberger sin peso - Resultados de Matlab. - Segunda iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.2591   -0.6593   -0.2236   -0.3908

Ganancia del observador L:
   22.9338    1.0388
  166.4526   11.9770
    0.9570   23.0662
   10.9830  132.2187
```

- Graficas:

Figura I34. *Ángulo vs Tiempo -Posición vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados sin peso. (Segunda iteración).*



- **Tercera iteración:**

  Valores iniciales:

- Matriz Q=diag([3000,2000,3000,100]):
- R= 0.01

  Resultados:

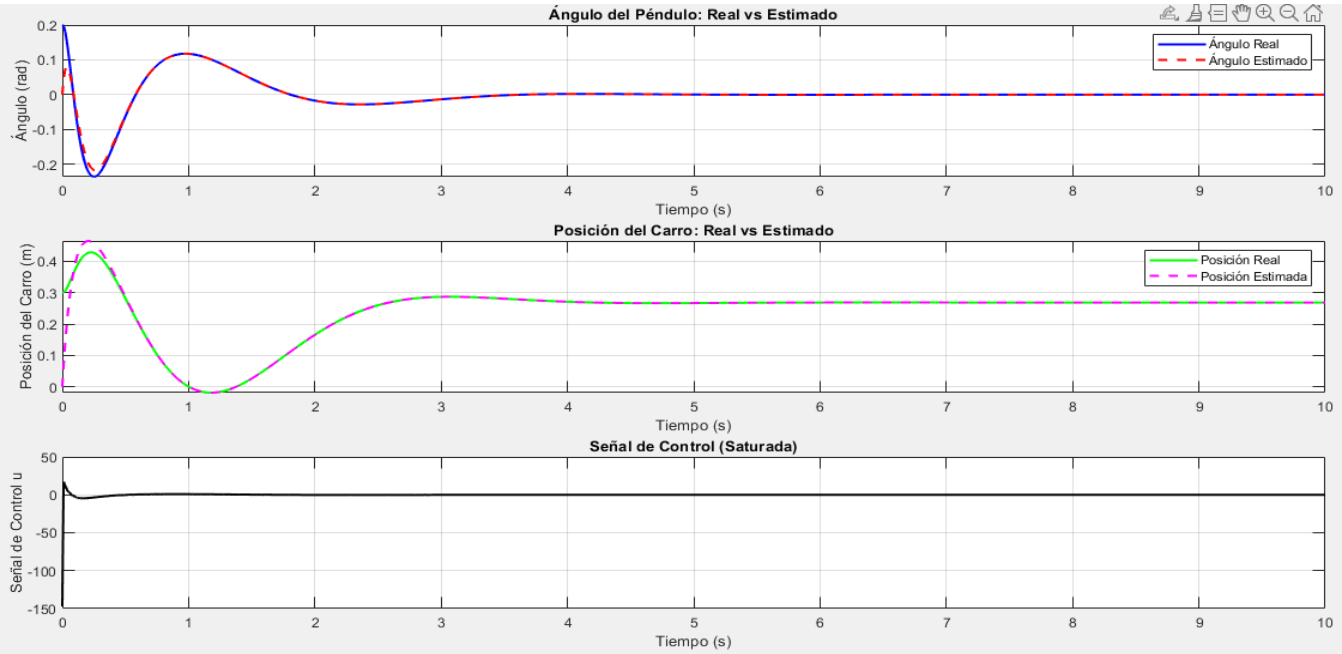
- Ganancias:

Figura I35. *Ganancias Nr del controlador Retroalimentación de estados. con observador de Luenberger sin peso - Resultados de Matlab. - Tercera iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.3942   -0.5935   -0.5477   -0.5260

Ganancia del observador L:
   22.9338    1.0388
  166.4526   11.9770
    0.9570   23.0662
   10.9830  132.2187
```

- Graficas:

Figura I36. *Ángulo vs Tiempo -Posición vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados sin peso. (tercera iteración).*



❖ **Cuarta iteración:**

Valores iniciales:

- Matriz Q=diag([2000,2000,3000,1000]):
- R= 0.01

Resultados:

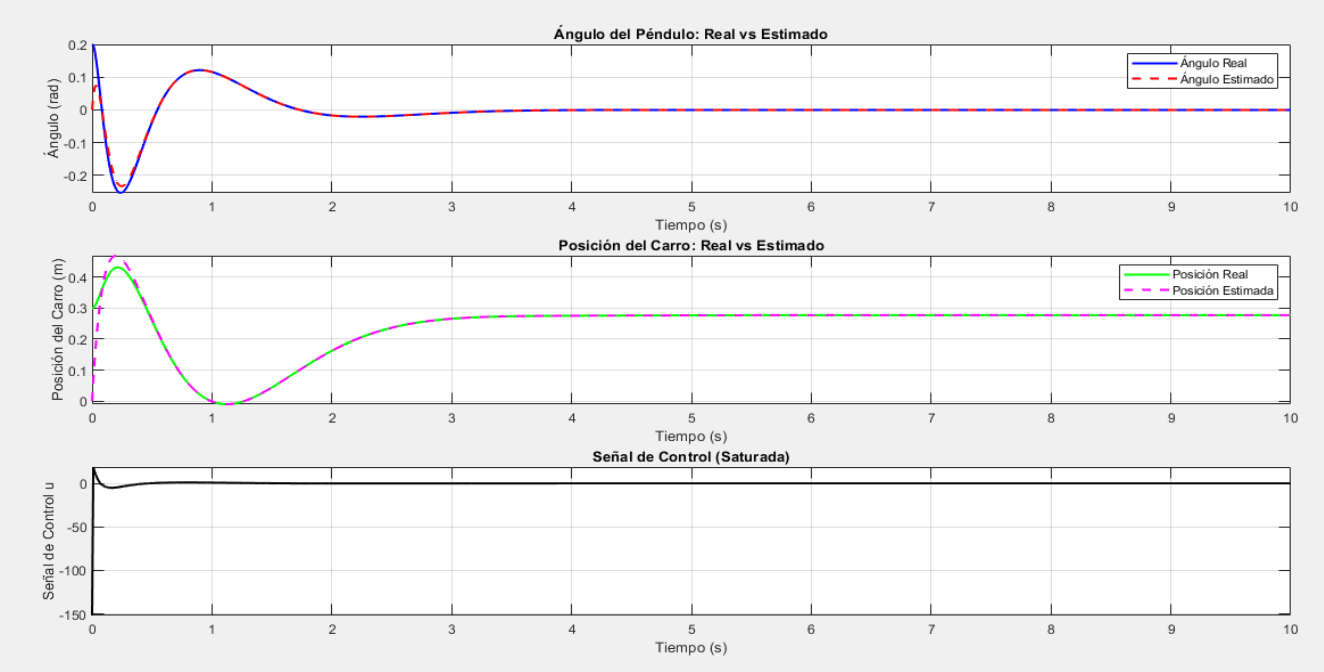
- Ganancias:

Figura I37. *Ganancias Nr del controlador Retroalimentación de estados. con observador de Luenberger sin peso - Resultados de Matlab. - Cuarta iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.6910   -0.6304   -0.5477   -0.6323

Ganancia del observador L:
   22.9338    1.0388
  166.4526   11.9770
    0.9570   23.0662
   10.9830  132.2187
```

- Graficas:

Figura I38. *Ángulo vs Tiempo -Posición vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados sin peso. (Cuarta iteración).*



❖ **Quinta iteración:**

Valores iniciales:

- Matriz Q=diag([3000,3000,3000,500]):
- R= 0.01

Resultados:

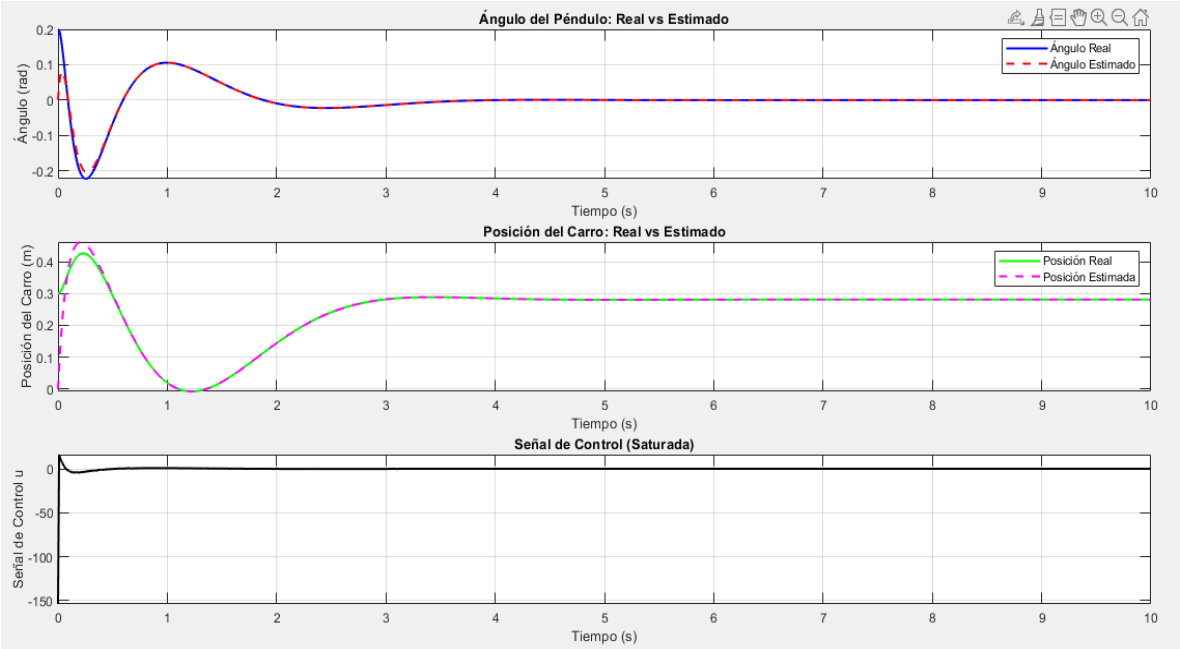
- Ganancias:

Figura I39. *Ganancias Nr del controlador Retroalimentación de estados. con observador de Luenberger sin peso - Resultados de Matlab. - Quinta iteración.*

```
Ganancia de retroalimentación K:
   1.0e+03 *

   -2.8274   -0.7181   -0.5477   -0.6042

Ganancia del observador L:
   22.9338    1.0388
  166.4526   11.9770
    0.9570   23.0662
   10.9830  132.2187
```

- Graficas:

Figura I40. *Ángulo vs Tiempo -Posición vs Tiempo – Fuerza de control vs Tiempo – Retroalimentación de estados sin peso. (Quinta iteración).*



Apéndice J. Código Retroalimentacion de estados ç(LQR) - Arduino

LQR

```
#include <I2Cdev.h>
#include <Wire.h>
#include "MPU6050_6Axis_MotionApps20.h"

// Instancia MPU6050
MPU6050 mpu;

// Variables MPU6050
bool dmpReady = false;
uint8_t mpuIntStatus, devStatus;
uint16_t packetSize;
uint16_t fifoCount;
uint8_t fifoBuffer[64];
Quaternion q;
VectorFloat gravity;
float ypr[3];  // yaw, pitch, roll
float xangle;  // Ángulo en grados (pitch)

const int TRIG_PIN = 5;
const int ECHO_PIN = 18;
```

```
float distanceMeters;  // Distancia medida por ultrasonido


// Variables Motores
int motoresderechos_IN1 = 27;
int motoresderechos_IN2 = 14;
int motoresizquierdos_IN1 = 19;
int motoresizquierdos_IN2 = 23;
int stby = 33;
int pwma = 32;
int pwmb = 13;
const int frecuencia = 40000;
const int canal = 0;
const int canal1 = 1;
const int resolucion = 8;
int dutycicle = 255;


// Ganancias del controlador LQR
float K[4] = { -2.3972e03, -0.5935e03, -0.5477e03, -0.5260e03 };


float N_r1 = -0.0018;      //ganaciacarro
float N_r2 = -1.5029e-17;  //ganaciapendulo


// Referencias deseadas
float refAngulo = 0.015708;  // 1.6 grados en radianes
```

```
float refPosicion = 0.2;     // 20 cm


TaskHandle_t Tarea0;


void setup() {
 Serial.begin(115200);
 Wire.begin();


 // Inicializar MPU6050
 Serial.println("Inicializando MPU6050...");
 mpu.initialize();
 devStatus = mpu.dmpInitialize();
 if (devStatus == 0) {
  mpu.setDMPEnabled(true);
  dmpReady = true;
  packetSize = mpu.dmpGetFIFOPacketSize();
  Serial.println("DMP activado correctamente.");
 } else {
  Serial.print("Error al inicializar DMP. Código: ");
  Serial.println(devStatus);
 }


 // Configurar pines de motores
 pinMode(motoresderechos_IN1, OUTPUT);
```

```
  pinMode(motoresderechos_IN2, OUTPUT);
  pinMode(motoresizquierdos_IN1, OUTPUT);
  pinMode(motoresizquierdos_IN2, OUTPUT);
  pinMode(stby, OUTPUT);
  pinMode(pwma, OUTPUT);
  pinMode(pwmb, OUTPUT);

  ledcSetup(canal, frecuencia, resolucion);
  ledcAttachPin(pwma, canal);
  ledcSetup(canal1, frecuencia, resolucion);
  ledcAttachPin(pwmb, canal1);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  // Crear tarea secundaria
  xTaskCreatePinnedToCore(loop0, "TAREA_0", 1000, NULL, 1, &Tarea0, 1);
}

void loop() {
  if (!dmpReady) return;

  // Leer ángulo del MPU6050
  fifoCount = mpu.getFIFOCount();
```

```
if (fifoCount == 1024) {
  mpu.resetFIFO();
  Serial.println("FIFO overflow! Reiniciando...");
  return;
}
if (fifoCount >= packetSize) {
  mpu.getFIFOBytes(fifoBuffer, packetSize);
  mpu.dmpGetQuaternion(&q, fifoBuffer);
  mpu.dmpGetGravity(&gravity, &q);
  mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
  xangle = ypr[1];  // Pitch en radianes
}

// Estado del sistema (ángulo y posición)
float x[4] = { xangle, 0, distanceMeters, 0 };

// Calcular el error
float error[4] = {
  x[0] - refAngulo,
  x[1],
  x[2] - refPosicion,
  x[3]
};
```

```
float factorEscala = 4;                    // Escalar la señal de control
float derivativo = 0.1 * (x[0] - refAngulo);  // Componente derivativa

// Calcular la señal de control u
float u = 0;
for (int i = 0; i < 4; i++) {
  u -= K[i] * error[i];
}
u += N_r1 * refPosicion + N_r2 * refAngulo;
u = u * factorEscala + derivativo;  // Escalado y derivativo

// Saturar la señal entre -255 y 255
u = constrain(u, -255, 255);

// Control de motores basado en la señal de control
if (u > 0) {
  moverAtras(abs(u));
} else {
  moverAdelante(abs(u));
}

Serial.print("Ángulo: ");
Serial.print(x[0] * (180 / 3.1416));  // Convertir a grados
Serial.print("°, Posición: ");
```

```
  Serial.print(x[2]);
  Serial.print("m, Control: ");
  Serial.println(u);
}

void loop0(void* parameter) {
  for (;;) {
    // Medir distancia
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    long duration = pulseIn(ECHO_PIN, HIGH);
    distanceMeters = (duration * 0.000343) / 2;  // Calcula la distancia

    vTaskDelay(2 / portTICK_PERIOD_MS);  // Delay para tareas FreeRTOS
  }
}

// Función para mover motores hacia adelante
void moverAdelante(int velocidad) {
  digitalWrite(stby, HIGH);
  ledcWrite(canal, velocidad);
```

```
  ledcWrite(canal1, velocidad);
  digitalWrite(motoresderechos_IN1, LOW);
  digitalWrite(motoresderechos_IN2, HIGH);
  digitalWrite(motoresizquierdos_IN1, LOW);
  digitalWrite(motoresizquierdos_IN2, HIGH);
}

// Función para mover motores hacia atrás
void moverAtras(int velocidad) {
  digitalWrite(stby, HIGH);
  ledcWrite(canal, velocidad);
  ledcWrite(canal1, velocidad);
  digitalWrite(motoresderechos_IN1, HIGH);
  digitalWrite(motoresderechos_IN2, LOW);
  digitalWrite(motoresizquierdos_IN1, HIGH);
  digitalWrite(motoresizquierdos_IN2, LOW);
}
```

Apéndice K. Retroalimentacion de estados con observador Luenberger - Arduino

LQR Y OBSERVADOR

```
#include <I2Cdev.h>

#include <Wire.h>

#include "MPU6050_6Axis_MotionApps20.h"


// Instancia MPU6050

MPU6050 mpu;


// Variables MPU6050

bool dmpReady = false;

uint8_t mpuIntStatus, devStatus;

uint16_t packetSize;

uint16_t fifoCount;

uint8_t fifoBuffer[64];

Quaternion q;

VectorFloat gravity;

float ypr[3];  // yaw, pitch, roll

float xangle;  // Ángulo en radianes (pitch)


// Variables de derivadas numéricas

float prevXangle = 0;

float prevDistance = 0;

unsigned long prevTime = 0;
```

```
float xangleDot = 0;

float distanceDot = 0;


const int TRIG_PIN = 5;

const int ECHO_PIN = 18;

float distanceMeters;  // Distancia medida por ultrasonido


// Variables Motores

int motoresderechos_IN1 = 27;

int motoresderechos_IN2 = 14;

int motoresizquierdos_IN1 = 19;

int motoresizquierdos_IN2 = 23;

int stby = 33;

int pwma = 32;

int pwmb = 13;

const int frecuencia = 40000;

const int canal = 0;

const int canal1 = 1;

const int resolucion = 8;

int dutycicle = 255;


// Ganancias del controlador LQR

float K[4] = {-2.4411e03,   -0.6769e03,   -0.3162e03,   -0.4548e03};
```

```
// Matriz de ganancia del observador L (4x2) calculada en MATLAB
float L[4][2] = {
  {22.9338, 1.0388},
  {184.7385, 11.9770},
  {0.9570, 23.0662},
  {5.9543, 132.2187}
};

// Referencias deseadas
float refAngulo = 0.015708;  // 3.6 grados en radianes
float refPosicion = 0.2;     // 20 cm

TaskHandle_t Tarea0;

void setup() {
  Serial.begin(115200);
  Wire.begin();

  // Inicializar MPU6050
  mpu.initialize();
  devStatus = mpu.dmpInitialize();
  if (devStatus == 0) {
    mpu.setDMPEnabled(true);
```

```
  dmpReady = true;
  packetSize = mpu.dmpGetFIFOPacketSize();
 } else {
  Serial.print("Error al inicializar DMP. Código: ");
  Serial.println(devStatus);
 }

 // Configurar pines de motores
 pinMode(motoresderechos_IN1, OUTPUT);
 pinMode(motoresderechos_IN2, OUTPUT);
 pinMode(motoresizquierdos_IN1, OUTPUT);
 pinMode(motoresizquierdos_IN2, OUTPUT);
 pinMode(stby, OUTPUT);
 pinMode(pwma, OUTPUT);
 pinMode(pwmb, OUTPUT);

 ledcSetup(canal, frecuencia, resolucion);
 ledcAttachPin(pwma, canal);
 ledcSetup(canal1, frecuencia, resolucion);
 ledcAttachPin(pwmb, canal1);

 pinMode(TRIG_PIN, OUTPUT);
 pinMode(ECHO_PIN, INPUT);
```

```
  // Crear tarea secundaria para medir la distancia
  xTaskCreatePinnedToCore(loop0, "TAREA_0", 1000, NULL, 1, &Tarea0, 1);
}

void loop() {
  if (!dmpReady) return;

  // Leer ángulo del MPU6050
  fifoCount = mpu.getFIFOCount();
  if (fifoCount >= packetSize) {
    mpu.getFIFOBytes(fifoBuffer, packetSize);
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
    xangle = ypr[1];  // Pitch en radianes
  }

  // Calcular derivadas numéricas
  unsigned long currentTime = millis();
  float deltaTime = (currentTime - prevTime) / 1000.0;

  if (deltaTime > 0) {
    xangleDot = (xangle - prevXangle) / deltaTime;
    distanceDot = (distanceMeters - prevDistance) / deltaTime;
```

```
}

// Actualizar valores anteriores
prevXangle = xangle;
prevDistance = distanceMeters;
prevTime = currentTime;

// Estado del sistema (ángulo, velocidad angular, posición, velocidad)
float x[4] = {xangle, xangleDot, distanceMeters, distanceDot};

// Salida medida
float y[2] = {xangle, distanceMeters};

// Corrección del observador: L * (y - C * x)
float correction[4] = {0};
for (int i = 0; i < 4; i++) {
  for (int j = 0; j < 2; j++) {
    correction[i] += L[i][j] * (y[j] - x[j * 2]);
  }
}

// Corregir los estados usando la corrección calculada
for (int i = 0; i < 4; i++) {
  x[i] += correction[i];
```

```
}

// Calcular el error respecto a las referencias
float error[4] = {
  x[0] - refAngulo,
  x[1],
  x[2] - refPosicion,
  x[3]
};

// Calcular la señal de control u
float u = 0;
for (int i = 0; i < 4; i++) {
  u -= K[i] * error[i];
}

// Escalar y saturar la señal de control
u = constrain(u * 8, -255, 255);

// Control de motores basado en la señal de control
if (u > 0) {
  moverAtras(abs(u));
} else {
  moverAdelante(abs(u));
```

```
  }

  Serial.print("Ángulo: ");
  Serial.print(x[0] * (180 / PI));
  Serial.print("°, Posición: ");
  Serial.print(x[2]);
  Serial.print("m, Control: ");
  Serial.println(u);
}

void loop0(void* parameter) {
  for (;;) {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
    long duration = pulseIn(ECHO_PIN, HIGH);
    distanceMeters = (duration * 0.000343) / 2;

    vTaskDelay(2 / portTICK_PERIOD_MS);
  }
}
// Función para mover motores hacia adelante
```

```
void moverAdelante(int velocidad) {
  digitalWrite(stby, HIGH);
  ledcWrite(canal, velocidad);
  ledcWrite(canal1, velocidad);
  digitalWrite(motoresderechos_IN1, LOW);
  digitalWrite(motoresderechos_IN2, HIGH);
  digitalWrite(motoresizquierdos_IN1, LOW);
  digitalWrite(motoresizquierdos_IN2, HIGH);
}

// Función para mover motores hacia atrás
void moverAtras(int velocidad) {
  digitalWrite(stby, HIGH);
  ledcWrite(canal, velocidad);
  ledcWrite(canal1, velocidad);
  digitalWrite(motoresderechos_IN1, HIGH);
  digitalWrite(motoresderechos_IN2, LOW);
  digitalWrite(motoresizquierdos_IN1, HIGH);
  digitalWrite(motoresizquierdos_IN2, LOW);
}
```

Apéndice L. Código Lógica difusa (Fuzzy Logic) -Arduino

Fuzzy

```
#include <I2Cdev.h>

#include <Wire.h>

#include "MPU6050_6Axis_MotionApps20.h"

#include <Fuzzy.h>  // Librería eFLL (asegúrate de instalarla correctamente)


// Instancia MPU6050

MPU6050 mpu;


// Variables MPU6050

bool dmpReady = false;

uint8_t mpuIntStatus, devStatus;

uint16_t packetSize;

uint16_t fifoCount;

uint8_t fifoBuffer[64];

Quaternion q;

VectorFloat gravity;

float ypr[3];  // yaw, pitch, roll

float xangle;  // Ángulo en grados (pitch)


const int TRIG_PIN = 5;

const int ECHO_PIN = 18;
```

```
float distanceMeters;  // Distancia medida por ultrasonido


// Variables Motores
int motoresderechos_IN1 = 27;
int motoresderechos_IN2 = 14;
int motoresizquierdos_IN1 = 19;
int motoresizquierdos_IN2 = 23;
int stby = 33;
int pwma = 32;
int pwmb = 13;
const int frecuencia = 40000;
const int canal = 0;
const int canal1 = 1;
const int resolucion = 8;
int dutycicle = 255;
TaskHandle_t Tarea0;


// Declarar el sistema difuso
Fuzzy* fuzzy = new Fuzzy();


// Variables difusas
FuzzyInput* inputAngulo;
FuzzyInput* inputPosicion;
FuzzyOutput* outputFuerza;
```

```
void setup() {
  Serial.begin(115200);
  Wire.begin();

  // Inicializar MPU6050
  Serial.println("Inicializando MPU6050...");
  mpu.initialize();
  devStatus = mpu.dmpInitialize();
  if (devStatus == 0) {
    mpu.setDMPEnabled(true);
    dmpReady = true;
    packetSize = mpu.dmpGetFIFOPacketSize();
    Serial.println("DMP activado correctamente.");
  } else {
    Serial.print("Error al inicializar DMP. Código: ");
    Serial.println(devStatus);
  }

  // Configurar pines de motores
  pinMode(motoresderechos_IN1, OUTPUT);
  pinMode(motoresderechos_IN2, OUTPUT);
  pinMode(motoresizquierdos_IN1, OUTPUT);
  pinMode(motoresizquierdos_IN2, OUTPUT);
```

```
  pinMode(stby, OUTPUT);
  pinMode(pwma, OUTPUT);
  pinMode(pwmb, OUTPUT);

  ledcSetup(canal, frecuencia, resolucion);
  ledcAttachPin(pwma, canal);
  ledcSetup(canal1, frecuencia, resolucion);
  ledcAttachPin(pwmb, canal1);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  xTaskCreatePinnedToCore(loop0, "TAREA_0", 1000, NULL, 1, &Tarea0, 1);

  // Configuración del sistema difuso
  setupFuzzy();
}

void loop() {
  if (!dmpReady) return;

  // Leer ángulo del MPU6050
  fifoCount = mpu.getFIFOCount();
  if (fifoCount >= packetSize) {
```

```
  mpu.getFIFOBytes(fifoBuffer, packetSize);
  mpu.dmpGetQuaternion(&q, fifoBuffer);
  mpu.dmpGetGravity(&gravity, &q);
  mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
  xangle = ypr[1];  // Pitch en radianes
}

// Leer la distancia
float posicion = distanceMeters;

// Establecer valores para las entradas difusas
fuzzy->setInput(1, xangle);
fuzzy->setInput(2, posicion);

// Ejecutar el sistema difuso
fuzzy->fuzzify();

// Obtener la salida difusa (fuerza de control)
float fuerza = fuzzy->defuzzify(1);

// Control de motores basado en la señal de control
if (posicion < 0.18) {
  moverAtras(abs(fuerza));
}
```

```
  if (posicion > 0.22) {
    moverAdelante(abs(fuerza));
  }

  // Mostrar el resultado
  Serial.print("Ángulo: ");
  Serial.print(xangle * (180 / PI));
  Serial.print("°, Posición: ");
  Serial.print(posicion);
  Serial.print(" m, Fuerza de control: ");
  Serial.println(fuerza);

  delay(10);
}

void setupFuzzy() {
  // Definir entradas difusas
  inputAngulo = new FuzzyInput(1);
  FuzzySet* anguloNegativo = new FuzzySet(-1.57, -1.57, -0.8, -0.2);
  FuzzySet* anguloCero = new FuzzySet(-0.5, -0.2, 0.2, 0.5);
  FuzzySet* anguloPositivo = new FuzzySet(0.2, 0.8, 1.57, 1.57);

  inputAngulo->addFuzzySet(anguloNegativo);
  inputAngulo->addFuzzySet(anguloCero);
```

```
inputAngulo->addFuzzySet(anguloPositivo);
fuzzy->addFuzzyInput(inputAngulo);

inputPosicion = new FuzzyInput(2);
FuzzySet* posicionNegativa = new FuzzySet(-0.05,-0.05, 0.25, 0.275);
FuzzySet* posicionCero = new FuzzySet(0.25, 0.275, 0.375, 0.4);
FuzzySet* posicionPositiva = new FuzzySet(0.375, 0.4, 0.6, 0.6);

inputPosicion->addFuzzySet(posicionNegativa);
inputPosicion->addFuzzySet(posicionCero);
inputPosicion->addFuzzySet(posicionPositiva);
fuzzy->addFuzzyInput(inputPosicion);

// Definir salida difusa
outputFuerza = new FuzzyOutput(1);
FuzzySet* fuerzaNegativa = new FuzzySet(-250, -250, -200, 0);
FuzzySet* fuerzaCero = new FuzzySet(-170, -35, 35, 100);
FuzzySet* fuerzaPositiva = new FuzzySet(0, 200, 250, 250);

outputFuerza->addFuzzySet(fuerzaNegativa);
outputFuerza->addFuzzySet(fuerzaCero);
outputFuerza->addFuzzySet(fuerzaPositiva);
fuzzy->addFuzzyOutput(outputFuerza);
```

```
// Reglas difusas actualizadas
// 1. Ángulo Negativo y Posición Negativa -> Fuerza Negativa
FuzzyRuleAntecedent* rule1Antecedent = new FuzzyRuleAntecedent();
rule1Antecedent->joinWithAND(anguloNegativo, posicionNegativa);
FuzzyRuleConsequent* rule1Consequent = new FuzzyRuleConsequent();
rule1Consequent->addOutput(fuerzaNegativa);
fuzzy->addFuzzyRule(new FuzzyRule(1, rule1Antecedent, rule1Consequent));

// 2. Ángulo Negativo y Posición Cero -> Fuerza Negativa
FuzzyRuleAntecedent* rule2Antecedent = new FuzzyRuleAntecedent();
rule2Antecedent->joinWithAND(anguloNegativo, posicionCero);
FuzzyRuleConsequent* rule2Consequent = new FuzzyRuleConsequent();
rule2Consequent->addOutput(fuerzaNegativa);
fuzzy->addFuzzyRule(new FuzzyRule(2, rule2Antecedent, rule2Consequent));

// 3. Ángulo Negativo y Posición Positiva -> Fuerza Negativa
FuzzyRuleAntecedent* rule3Antecedent = new FuzzyRuleAntecedent();
rule3Antecedent->joinWithAND(anguloNegativo, posicionPositiva);
FuzzyRuleConsequent* rule3Consequent = new FuzzyRuleConsequent();
rule3Consequent->addOutput(fuerzaNegativa);
fuzzy->addFuzzyRule(new FuzzyRule(3, rule3Antecedent, rule3Consequent));

// 4. Ángulo Cero y Posición Negativa -> Fuerza Positiva
FuzzyRuleAntecedent* rule4Antecedent = new FuzzyRuleAntecedent();
```

```
rule4Antecedent->joinWithAND(anguloCero, posicionNegativa);
FuzzyRuleConsequent* rule4Consequent = new FuzzyRuleConsequent();
rule4Consequent->addOutput(fuerzaPositiva);
fuzzy->addFuzzyRule(new FuzzyRule(4, rule4Antecedent, rule4Consequent));

// 5. Ángulo Cero y Posición Cero -> Fuerza Cero
FuzzyRuleAntecedent* rule5Antecedent = new FuzzyRuleAntecedent();
rule5Antecedent->joinWithAND(anguloCero, posicionCero);
FuzzyRuleConsequent* rule5Consequent = new FuzzyRuleConsequent();
rule5Consequent->addOutput(fuerzaCero);
fuzzy->addFuzzyRule(new FuzzyRule(5, rule5Antecedent, rule5Consequent));

// 6. Ángulo Cero y Posición Positiva -> Fuerza Negativa
FuzzyRuleAntecedent* rule6Antecedent = new FuzzyRuleAntecedent();
rule6Antecedent->joinWithAND(anguloCero, posicionPositiva);
FuzzyRuleConsequent* rule6Consequent = new FuzzyRuleConsequent();
rule6Consequent->addOutput(fuerzaNegativa);
fuzzy->addFuzzyRule(new FuzzyRule(6, rule6Antecedent, rule6Consequent));

// 7. Ángulo Positivo y Posición Negativa -> Fuerza Positiva
FuzzyRuleAntecedent* rule7Antecedent = new FuzzyRuleAntecedent();
rule7Antecedent->joinWithAND(anguloPositivo, posicionNegativa);
FuzzyRuleConsequent* rule7Consequent = new FuzzyRuleConsequent();
rule7Consequent->addOutput(fuerzaPositiva);
```

```
  fuzzy->addFuzzyRule(new FuzzyRule(7, rule7Antecedent, rule7Consequent));

  // 8. Ángulo Positivo y Posición Cero -> Fuerza Positiva
  FuzzyRuleAntecedent* rule8Antecedent = new FuzzyRuleAntecedent();
  rule8Antecedent->joinWithAND(anguloPositivo, posicionCero);
  FuzzyRuleConsequent* rule8Consequent = new FuzzyRuleConsequent();
  rule8Consequent->addOutput(fuerzaPositiva);
  fuzzy->addFuzzyRule(new FuzzyRule(8, rule8Antecedent, rule8Consequent));

  // 9. Ángulo Positivo y Posición Positiva -> Fuerza Positiva
  FuzzyRuleAntecedent* rule9Antecedent = new FuzzyRuleAntecedent();
  rule9Antecedent->joinWithAND(anguloPositivo, posicionPositiva);
  FuzzyRuleConsequent* rule9Consequent = new FuzzyRuleConsequent();
  rule9Consequent->addOutput(fuerzaPositiva);
  fuzzy->addFuzzyRule(new FuzzyRule(9, rule9Antecedent, rule9Consequent));
}

void loop0(void* parameter) {
  for (;;) {
    digitalWrite(TRIG_PIN, LOW);
    delayMicroseconds(2);
    digitalWrite(TRIG_PIN, HIGH);
    delayMicroseconds(10);
    digitalWrite(TRIG_PIN, LOW);
```

```
    long duration = pulseIn(ECHO_PIN, HIGH);
    distanceMeters = (duration * 0.000343) / 2;  // Calcula la distancia

    vTaskDelay(2 / portTICK_PERIOD_MS);
  }
}

// Función para mover motores hacia adelante
void moverAdelante(int velocidad) {
  digitalWrite(stby, HIGH);
  ledcWrite(canal, velocidad);
  ledcWrite(canal1, velocidad);
  digitalWrite(motoresderechos_IN1, LOW);
  digitalWrite(motoresderechos_IN2, HIGH);
  digitalWrite(motoresizquierdos_IN1, LOW);
  digitalWrite(motoresizquierdos_IN2, HIGH);
}

// Función para mover motores hacia atrás
void moverAtras(int velocidad) {
  digitalWrite(stby, HIGH);
  ledcWrite(canal, velocidad);
  ledcWrite(canal1, velocidad);
  digitalWrite(motoresderechos_IN1, HIGH);
```

```
  digitalWrite(motoresderechos_IN2, LOW);
  digitalWrite(motoresizquierdos_IN1, HIGH);
  digitalWrite(motoresizquierdos_IN2, LOW);
}
```