Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Mecatrónica

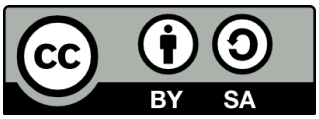
Desarrollo de un sistema de control para el estabilizado de las aletas pectorales de un robot submarino bioinspirado.

Informe de Proyecto de Graduación para optar por el título de

Ingeniero en Mecatrónica con el grado académico de Licenciatura

David José Rodríguez Camacho

Madrid, 28 de julio de 2025



Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.

David José Rodríguez Camacho

Madrid, 28 de julio de 2025

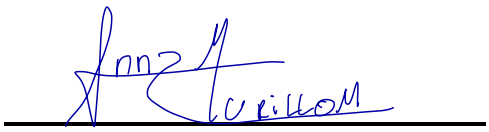
Céd: 2-0803-0701

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

El profesor asesor del presente trabajo final de graduación, indica que el documento presentado por el estudiante cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica del Instituto Tecnológico de Costa Rica para ser defendido ante el jurado evaluador, como requisito final para aprobar el curso Proyecto Final de Graduación y optar así por el título de Ingeniero(a) en Mecatrónica, con el grado académico de Licenciatura.

Estudiante: David José Rodríguez Camacho

Proyecto: Desarrollo de un sistema de control para el estabilizado de las aletas pectorales de un robot submarino bioinspirado.



Ing. Ana María Murillo Morgan

Asesor

Cartago, 5 de agosto del 2025

INSTITUTO TECNOLÓGICO DE COSTA RICA
PROGRAMA DE LICENCIATURA EN INGENIERÍA MECATRÓNICA
PROYECTO FINAL DE GRADUACIÓN
ACTA DE APROBACIÓN

Proyecto final de graduación defendido ante el presente jurado evaluador como requisito para optar por el título de Ingeniero(a) en Mecatrónica con el grado académico de Licenciatura, según lo establecido por el programa de Licenciatura en Ingeniería Mecatrónica, del Instituto Tecnológico de Costa Rica.

Estudiante: David José Rodríguez Camacho

Proyecto: Desarrollo de un sistema de control para el estabilizado de las aletas pectorales de un robot submarino bioinspirado.

Miembros del jurado evaluador

Dra-Ing. Gabriela Ortiz León
Jurado

MSc. Marta Eugenia Vílchez Monge
Jurado

Los miembros de este jurado dan fe de que el presente proyecto final de graduación ha sido aprobado y cumple con las normas establecidas por el programa de Licenciatura en Ingeniería Mecatrónica.

Cartago, 5 de agosto del 2025

Resumen

Este trabajo presenta un sistema de control para las aletas pectorales de un robot submarino bioinspirado que utiliza actuadores de aleación con memoria de forma (SMA). Desarrollado en el CAR UPM-CSIC, el sistema aborda las limitaciones de maniobrabilidad causadas por la latencia térmica de los SMA y la falta de modelos dinámicos precisos. Se implementó un controlador PID mediante el método de sintonización de Ziegler-Nichols, junto con modificaciones en el circuito electrónico para permitir un control proporcional de corriente.

Los resultados experimentales demostraron desplazamientos angulares de hasta 31.6 grados (superando las especificaciones de diseño), un tiempo de subida al 95% de 40 ms, una latencia de comunicación I2C de 178 μs y una fuerte correlación lineal entre el ángulo y la resistencia ($R^2 = 0.94$). El sistema mantuvo la estabilidad durante la operación con múltiples actuadores e incorporó protocolos de seguridad. Aunque no se logró reducir el tiempo de aleteo, los resultados validan el uso de actuadores SMA para el control preciso en robótica submarina cuando se combinan con estrategias de control avanzado.

Como trabajo futuro se propone aumentar el diámetro de los alambres SMA para obtener mayor fuerza e implementar un control directo de corriente para reducir la dependencia térmica. Este desarrollo representa un avance significativo en el campo de los robots submarinos bioinspirados, ofreciendo mejores capacidades de maniobra y control en entornos acuáticos.

Palabras clave: Robot submarino, Aletas pectorales, SMA (Shape Memory Alloy), Control PID, Bioinspirado, Resistencia eléctrica, Tiempo de respuesta.

Abstract

This paper presents a control system for the pectoral fins of a bioinspired underwater robot using shape memory alloy (SMA) actuators. Developed at the CAR UPM-CSIC, the system addresses maneuverability limitations caused by SMA thermal latency and the lack of accurate dynamic models. A PID controller was implemented using the Ziegler-Nichols tuning method, along with modifications to the electronic circuit to enable proportional current control.

Experimental results demonstrated angular displacements of up to 31.6 degrees (exceeding design specifications), a 95% rise time of 40 ms, an I2C communication latency of 178 μs, and a strong linear correlation between angle and resistance ($R^2 = 0.94$). The system maintained stability during multi-actuator operation and incorporated safety protocols. Although flapping time reduction was not achieved, the results validate the use of SMA actuators for precise control in underwater robotics when combined with advanced control strategies.

Future work includes increasing SMA wire diameter to enhance force output and implementing direct current control to reduce thermal dependency. This development represents a significant advancement in the field of bioinspired underwater robots, offering improved maneuverability and control in aquatic environments.

Keywords: Underwater robot, pectoral fins, SMA (Shape Memory Alloy), PID control, bioinspired, electrical resistance, response time.

Dedicatoria:

A mis padres, pilares fundamentales de mi vida:

A mi madre, cuyo amor y sacrificio fueron la luz que guió mis pasos incluso cuando ya no estaba físicamente a mi lado. Aunque el destino no permitió que estuvieras aquí para verme cruzar esta meta, sé que desde el cielo sigues siendo mi mayor inspiración. Cada logro, cada esfuerzo y este título llevan tu nombre, porque sin tu ejemplo de fortaleza y ternura, nada de esto habría sido posible.

A mi padre, por su apoyo incondicional y por creer en mí incluso cuando yo dudaba. Gracias por estar ahí, con tu silencio lleno de sabiduría y tus palabras de aliento.

Este proyecto es el fruto de sus enseñanzas, de los valores que me inculcaron y del amor que sembraron en mí.

Agradecimientos

En primer lugar, quiero expresar mi más sincero agradecimiento al Dr. Claudio Rossi, por brindarme la oportunidad de formar parte de su laboratorio y por su valiosa orientación durante el desarrollo de este proyecto. Su conocimiento, paciencia y dedicación han sido fundamentales para mi crecimiento profesional y académico.

A mis compañeros de laboratorio, Donato Rinaldi, Edoardo Celestino Bernardo Selvaggi y Antoine Titouan Louis Christian Daniel Lemarignier, gracias por hacer de mi estancia en Madrid una experiencia inolvidable. Compartir con ustedes cada día, tanto los desafíos como los momentos de risas y aprendizaje, hizo que este camino fuera mucho más enriquecedor. Su apoyo, colaboración y amistad fueron un pilar clave durante este proceso.

A mis amigos del TEC, María Fernanda Sabatini, Glenn Angulo y Dryan López con quienes tuve la suerte de compartir la mayoría de mis años universitarios. Gracias por las largas noches de estudio, los trabajos interminables y por estar ahí en cada paso, tanto en los éxitos como en los momentos difíciles. Han sido una familia lejos de casa.

Finalmente, pero no menos importante, a mi familia, por ser mi soporte incondicional durante todos estos años. A mis padres, por su amor, sacrificio y por creer en mí incluso en los momentos más complicados. Sin ustedes, este logro no habría sido posible.

Este proyecto no solo representa el cierre de una etapa académica, sino también el fruto del apoyo, la colaboración y el cariño de cada una de las personas que menciono. A todos, ¡mil gracias!

David José Rodríguez Camacho

Madrid, 28 de julio de 2025

Índice general

Índice de figuras

Índice de tablas

Lista de símbolos y abreviaciones

Abreviaciones

CAR	Centro de Automática y Robótica
CSIC	Consejo Superior de Investigaciones Científicas
SMA	Shape Memory Alloy
UPM	Universidad Politécnica de Madrid

Capítulo 1

Introducción

1.1 Entorno del proyecto

El proyecto para optar por el grado académico de licenciatura en Ingeniería Mecatrónica se lleva a cabo en el Centro de Automatización y Robótica. Este es un instituto de investigación creado en colaboración entre el Consejo Superior de Investigaciones Científicas y la Universidad Politécnica de Madrid.
Fundado en noviembre de 2009, su propósito central es llevar a cabo investigaciones en las áreas de Ingeniería de Control, Percepción Artificial y Robótica [1].
El CAR UPM-CSIC está en una posición única para liderar un ambicioso programa de trabajo, situando el estudio de la automatización en el más alto nivel. Además, cuenta con un excelente potencial conceptual e instrumental, basado en la experiencia de sus investigadores y en sus instalaciones. El CAR participa en numerosos proyectos y programas de la Comisión Europea (H2020, FP7 y FP6), del Plan Nacional de España, de programas de las CCAA y en diversas colaboraciones con empresas [2].
En este centro, la investigación se lleva a cabo de manera intregral y multidisciplinaria, contribuyendo al avance del conocimiento para resolver problemas específicos que surgen de distintas áreas de la industria. Estas actividades de investigación están estratégicamente vinculadas a una enseñanza de alta calidad, que incluye el programa de posgrado en Automatización y Robótica de la UPM, tanto a nivel de máster como de doctorado [2].

1.2 Definición del problema

1.2.1 Generalidades

En el CAR se está desarrollando un robot submarino para la exploración y el monitoreo ambiental en entornos de aguas profundas. Este tipo de robot tiene un enorme potencial en la investigación marina, el monitoreo ecológico, la cartografía submarina, y en la detección de cambios ambientales en lugares de difícil acceso. Actualmente, se encuentra en una fase

de pruebas en piscinas de laboratorio, pero su objetivo final es operar en una variedad de cuerpos de agua, desde océanos hasta lagos y ríos. Para ello, deberá ser capaz de enfrentarse a las condiciones extremas que predominan en estos entornos acuáticos, como la alta presión en aguas profundas, la baja visibilidad, y las variaciones drásticas de temperatura y salinidad.
Este robot, bioinspirado en la morfología de un pez, utiliza actuadores basados en alambres de SMA para controlar el movimiento de las aletas pectorales, específicamente en el eje de *pitch* (cabeceo). Este movimiento es esencial para mantener la estabilidad y el control de la trayectoria en entornos submarinos. Sin embargo, uno de los principales desafíos a los que se enfrenta el proyecto es mejorar la capacidad de maniobra del robot. Estos movimientos son cruciales para que el robot se posicione adecuadamente en diferentes escenarios de exploración. Actualmente, el sistema tiene limitaciones en su maniobrabilidad, debido a la falta de un sistema de control avanzado para los actuadores de SMA, lo que reduce su efectividad en el seguimiento de trayectorias complejas y en la capacidad de realizar movimientos finos y controlados.

1.2.2 Justificación

Este proyecto se centra en la falta de un sistema de control que permita mejorar la estabilidad y precisión del movimiento de las aletas pectorales en el eje de pitch, lo que es crucial para la maniobrabilidad del robot en entornos submarinos. En aplicaciones de exploración y monitoreo ambiental, el control preciso del movimiento es esencial para garantizar que el robot pueda seguir trayectorias complejas, adaptarse a variaciones en el entorno y posicionarse de manera precisa para recopilar datos de alta calidad.
El equipo de trabajo en este proyecto incluye entre sus miembros profesores de la Universidad Politécnica de Madrid, quienes, además de su participación en el desarrollo, enfrentan múltiples responsabilidades académicas. Además, el reducido número de miembros en el equipo incrementa la carga de trabajo individual, lo que limita significativamente el tiempo y los recursos disponibles para el diseño y perfeccionamiento de un sistema de control avanzado.
Por otro lado, el uso de aleaciones con memoria de forma (SMA) conlleva un retraso en la respuesta de los actuadores debido a la latencia térmica, lo que complica tanto la etapa de diseño como la sintonización del controlador. Asimismo, el entorno marino, caracterizado por su alta capacidad para provocar corrosión y variaciones drásticas de presión, puede dañar componentes sensibles, comprometiendo la durabilidad y la estabilidad del sistema. Además, emplear materiales con resistencia insuficiente frente a la presión y la corrosión aumenta el riesgo de fallas mecánicas, restringiendo aún más la movilidad del robot. Finalmente, la falta de modelos matemáticos precisos para describir el comportamiento de las SMA bajo condiciones submarinas limita la implementación de algoritmos de control avanzados.
La implementación de un sistema de control avanzado puede superar las limitaciones actuales en la maniobrabilidad del robot submarino. Un sistema de control eficiente permite programar con precisión los movimientos del robot en el espacio. Al incorporar algoritmos de control sofisticados, como el control PID (Proporcional, Integral, Derivativo) o el control

adaptativo, el robot puede ajustar dinámicamente sus movimientos en respuesta a cambios en el entorno, mejorando así su capacidad para seguir trayectorias planificadas y posicionarse con precisión.
Además, un sistema de control bien diseñado integra sensores para proporcionar retroalimentación en tiempo real sobre la posición y el rendimiento de los actuadores. Esta retroalimentación permite al sistema realizar ajustes inmediatos, lo que minimiza las desviaciones y mejora la estabilidad durante la operación en condiciones extremas, como altas presiones y bajas temperaturas.

1.2.3 Diagrama de Ishikawa

En la Figura 1.1 se presenta el diagrama de Ishikawa, una herramienta utilizada para el análisis de Causa-Raíz del problema en cuestión. Este diagrama fue elaborado aplicando la metodología de las 6 M, que incluye los siguientes factores: Mano de Obra, Maquinaria, Materiales, Método, Medio Ambiente y Medida.

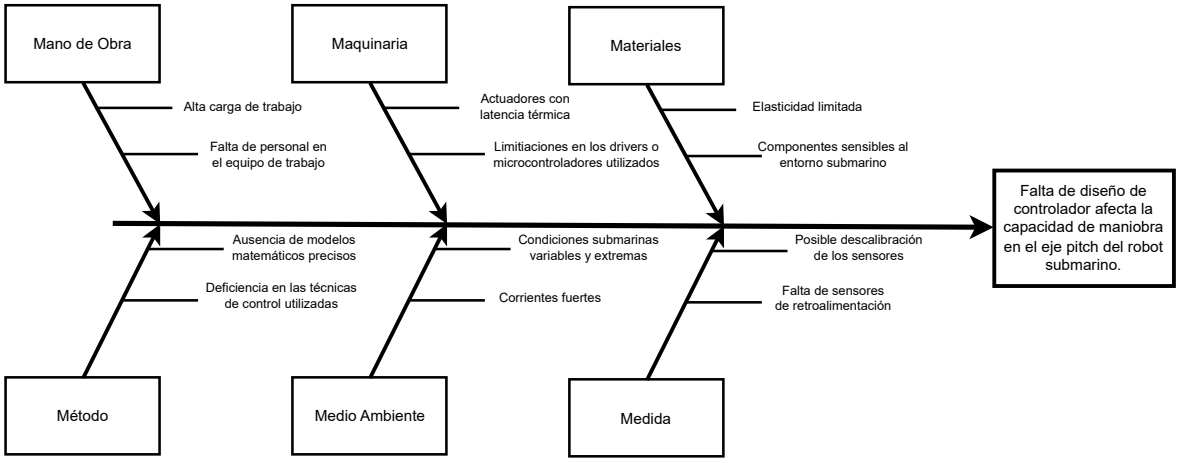


Figura 1.1: Diagrama de Ishikawa del problema planteado.

1.2.4 Síntesis del problema

El robot submarino enfrenta limitaciones significativas en la estabilidad y control del movimiento de sus aletas pectorales en el eje de pitch, lo que restringe su efectividad en la exploración y monitoreo ambiental en entornos acuáticos desafiantes. Estas limitaciones se deben a un método de control deficiente, que no permite movimientos precisos, afectando el seguimiento de trayectorias y el posicionamiento exacto. Esta situación negativa compromete la recolección de datos cruciales y la capacidad del robot para operar eficientemente en condiciones extremas, lo que es esencial para el éxito del proyecto.

1.3 Alcance del proyecto

Este proyecto se enfoca en desarrollar una prueba de concepto para un sistema de control avanzado de aletas pectorales en el eje de pitch (cabeceo), utilizando actuadores de aleación con memoria de forma (SMA) en un robot submarino bioinspirado. El objetivo principal es demostrar cómo este sistema puede mejorar la estabilidad y capacidad de respuesta del actuador en condiciones acuáticas controladas.

El alcance abarcará el diseño e implementación de los algoritmos de control diseñados y las pruebas de desempeño en condiciones controladas de laboratorio. Se priorizará el desarrollo del subsistema de control sobre otros aspectos como la navegación autónoma o la integración con otros sistemas del robot. Las pruebas se limitarán a evaluar parámetros básicos como el rango de movimiento angular o velocidad de respuesta, sin considerar aún factores como presión extrema o corrientes marinas reales.

Cabe destacar que este proyecto representa una primera fase exploratoria, por lo que no incluirá optimizaciones avanzadas de materiales ni pruebas en entornos marinos abiertos ni mediciones de fuerza en el eje pitch. Los resultados obtenidos servirán como base para futuras investigaciones que amplíen el desarrollo hacia un sistema robótico completo.

No fue posible realizar las pruebas de fuerza en el eje pitch debido a que el robot sufrió daños en su estructura y actualmente se encuentra desarmado en proceso de rediseño. Tras consultar con el asesor de la empresa, se determinó que demostrar el movimiento del actuador en ángulos controlados era un resultado suficiente para los fines del proyecto.

1.4 Objetivo General

Desarrollar un sistema de control para el movimiento de las aletas pectorales de un robot submarino bioinspirado con forma de pez, que mejore su estabilidad y capacidad de respuesta en condiciones extremas durante la exploración y monitoreo ambiental en entornos acuáticos.

1.5 Objetivos específicos

- Identificar las necesidades del sistema de control, alineando estas con las expectativas del cliente para definir los requisitos técnicos del proyecto.
- Diseñar un sistema de control que optimice la estabilidad y precisión del movimiento de las aletas pectorales en el eje de pitch para operar eficientemente en condiciones extremas.
- Realizar pruebas de concepto y experimentos de validación del sistema de control desarrollado, evaluando su desempeño en diferentes escenarios de operación, para garantizar que cumpla con los requerimientos planteados y su funcionamiento correcto.

1.6 Estructura del documento

Este documento contiene 6 capítulos, los cuáles abarcan las diferentes partes del proyecto.

El Capítulo 2 presenta el marco teórico, el cual incluye todos los conceptos fundamentales y afines al proyecto que resultan esenciales para su comprensión. Dado que el público objetivo posee un conocimiento previo en la materia, el contenido se centra exclusivamente en las tecnologías y conceptos específicos directamente relacionados con el desarrollo del trabajo, evitando redundancias en información básica o de dominio general.

El Capítulo 3 describe la metodología empleada para la ejecución del proyecto. Este proceso inicia con la interacción con el cliente para comprender a fondo el problema y sus causas, seguido de la identificación y priorización de necesidades. Estas se transforman en especificaciones cuantificables, las cuales orientan la generación de propuestas durante la fase de conceptualización. A continuación, se realiza una investigación del estado del arte para identificar soluciones existentes y desarrollar conceptos innovadores, los cuales se evalúan y combinan para crear soluciones candidatas. Estas se someten a pruebas de concepto para verificar su viabilidad, culminando en la selección de un concepto ganador que será desarrollado como propuesta final.

Tras la definición del concepto ganador, en el Capítulo 4 se desarrolla exhaustivamente la implementación de la solución elegida, abarcando tanto el diseño y fabricación de los actuadores bioinspirados como el desarrollo del sistema de control electrónico, incluyendo la caracterización dinámica de los componentes y el diseño del algoritmo de control.

El Capítulo 5 presenta los resultados experimentales obtenidos, evaluando el cumplimiento de las especificaciones técnicas establecidas y analizando el desempeño del sistema en sus diferentes aspectos funcionales, complementado con un análisis económico y de riesgos del desarrollo.

Finalmente, el Capítulo 6 sintetiza los logros obtenidos, contrasta los resultados con los objetivos iniciales, y propone líneas de mejora para futuras iteraciones del diseño, destacando tanto las contribuciones técnicas como las limitaciones encontradas durante el desarrollo.

Capítulo 2

Marco Teórico

2.1 Mecánica del nado

Los peces han desarrollado una amplia variedad de mecanismos de locomoción que les permiten moverse eficientemente en su entorno acuático. Entre estos mecanismos, el uso de las aletas pectorales para la propulsión y el control del movimiento es uno de los más versátiles y complejos. Las aletas pectorales, ubicadas a ambos lados del cuerpo, no solo sirven como superficies de control para maniobras, sino que también pueden generar empuje en diferentes direcciones, incluyendo el eje de *pitch* (movimiento de cabeceo, es decir, hacia arriba y hacia abajo).

2.1.1 Morfología de las aletas pectorales

La morfología de las aletas pectorales en los peces es un aspecto clave que influye en su capacidad para generar fuerzas de propulsión y controlar su movimiento. Las aletas pectorales, ubicadas a ambos lados del cuerpo, varían significativamente en forma, tamaño, orientación y posición según la especie y su entorno.

En los peces más primitivos, como en el Lepisosteus, las aletas pectorales suelen estar ubicadas abajo en el cuerpo en una posición ventral y tienen una base casi horizontal. En contraste, en los peces más evolucionados, como el Lepomis, las aletas pectorales están más verticalmente inclinadas y se ubican en una posición más dorsolateral, cerca del centro de masa del pez [3]. Se puede ver una comparación en el cambio de este tipo de aletas en la Figura 2.1.

La estructura musculoesquelética de las aletas pectorales también juega un papel crucial. Las aletas están compuestas por una serie de radios que se articulan en una base fibrocartilaginosa, lo que permite un amplio rango de movimientos. Los músculos abductores y aductores controlan los movimientos de abducción (hacia afuera) y aducción (hacia adentro) de las aletas, respectivamente, lo que permite a los peces generar fuerzas en diferentes direcciones [3].

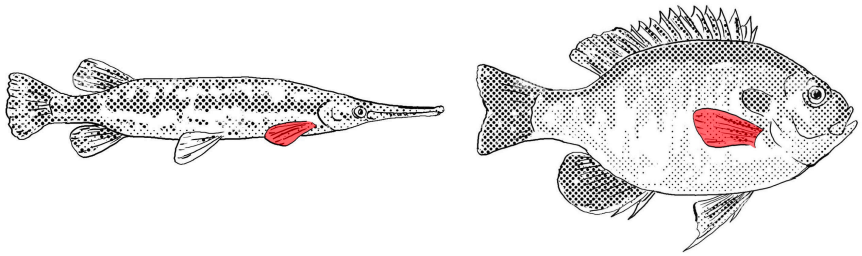


Figura 2.1: Comparación de las aletas pectorales de peces. Izq: Lepisosteus, Der: Lepomis.

2.1.2 Cinemática del movimiento de las aletas pectorales

La cinemática del movimiento de las aletas pectorales se refiere a los patrones de movimiento que estas estructuras realizan para generar fuerzas de propulsión y control. Los peces utilizan principalmente dos tipos de movimientos: oscilatorios y ondulatorios, que varían en amplitud, frecuencia y plano de movimiento.

El *stroke plane* es un concepto clave en la cinemática del movimiento de las aletas pectorales. Se refiere al plano en el que las aletas se mueven durante la locomoción, y su ángulo varía según el tipo de movimiento y la velocidad de nado [3]. Este se puede ver en la Figura 2.2

- Movimiento oscilatorio (*flapping* o *rowing*)
  - *Flapping* (aleteo dorsoventral): En este movimiento, las aletas pectorales se mueven hacia arriba y hacia abajo en un plano vertical. El *stroke plane* en este caso es más vertical, lo que facilita la generación de fuerzas de elevación (*lift*) durante el *downstroke* (movimiento hacia abajo). Durante el *upstroke* (movimiento hacia arriba), las aletas generan una fuerza hacia abajo, lo que permite al pez descender o mantener la estabilidad. Este tipo de movimiento es común en peces que nadan a velocidades más altas y necesitan un control preciso de su posición vertical [3].
  - *Rowing* (remado anteroposterior): En este movimiento, las aletas se mueven hacia adelante y hacia atrás en un plano horizontal. El *stroke plane* en este caso es más horizontal, lo que facilita la generación de fuerzas de empuje (*thrust*) durante el *power stroke* (movimiento hacia atrás). Aunque este movimiento es más común para generar empuje en el eje horizontal, también puede contribuir al control del eje de pitch cuando las aletas se mueven con un ángulo de ataque adecuado [3].
- Movimientos ondulatorios: Algunos peces, como las rayas y los peces globo, utilizan movimientos ondulatorios de las aletas pectorales para generar empuje. En este caso, las aletas realizan ondas que se propagan desde el borde de ataque hacia el borde de salida, generando fuerzas de propulsión continuas [3].

El *stroke plane* no es estático; varía según la velocidad de nado y el tamaño del pez. A medida que aumenta la velocidad de nado, este plano puede volverse más vertical en algunas especies, como el pez sol y el pez loro. Este cambio en el *stroke plane* permite una mayor generación

de fuerzas de elevación y empuje, lo que es esencial para mantener altas velocidades de nado. En peces más grandes, el *stroke plane* puede variar menos con la velocidad, ya que las aletas pectorales tienden a ser proporcionalmente más pequeñas en relación con el cuerpo. Esto puede limitar la capacidad de los peces más grandes para ajustarlo y, por lo tanto, su eficiencia de propulsión a altas velocidades [3].

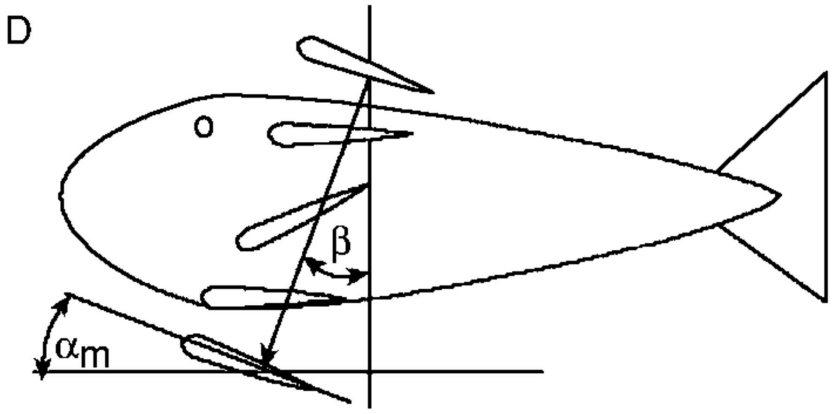


Figura 2.2: Ilustración del *stroke plane* en peces, como se muestra en [3, Fig. 10.7D].

2.1.3 Dinámica de fluidos y generación de empuje

La dinámica de fluidos es fundamental para entender cómo los peces generan empuje y controlan su movimiento mediante el uso de sus aletas pectorales. Cuando las aletas se mueven, interactúan con el agua circundante, creando fuerzas que permiten al pez avanzar, ascender, descender o realizar maniobras. Cuando las aletas pectorales se mueven, ya sea hacia arriba y abajo en un movimiento de *flapping* o hacia adelante y atrás en un movimiento de *rowing*, empujan el agua a su alrededor. Este movimiento genera vórtices, que son remolinos de agua que se forman en los bordes de las aletas, este proceso se ilustra en la Figura 2.3. Estos vórtices son regiones de baja presión que generan una fuerza neta en dirección opuesta al movimiento del agua. Por ejemplo, durante el *downstroke* (movimiento hacia abajo), las aletas empujan el agua hacia abajo, creando un vórtice que genera una fuerza de reacción hacia arriba (*lift*). Esta fuerza ayuda al pez a ascender o mantenerse en una posición estable. Durante el *power stroke* (movimiento hacia atrás), las aletas empujan el agua hacia atrás, creando un vórtice que genera una fuerza de reacción hacia adelante (*thrust*). Esta fuerza impulsa al pez hacia adelante [3].

El ángulo de ataque es el ángulo entre la superficie de la aleta y la dirección del flujo de agua. Este ángulo es crucial para la generación de fuerzas. A bajos ángulos de ataque, el agua fluye suavemente sobre la superficie de la aleta, generando una fuerza de sustentación eficiente. Esto ocurre cuando las aletas se mueven de manera controlada y el flujo de agua sigue la forma de la aleta. A ángulos de ataque más altos, el flujo de agua se separa de la superficie de la aleta, creando un vórtice de borde de ataque. Este vórtice aumenta temporalmente la fuerza de sustentación, pero si el ángulo es demasiado alto, el vórtice se desprende y reduce

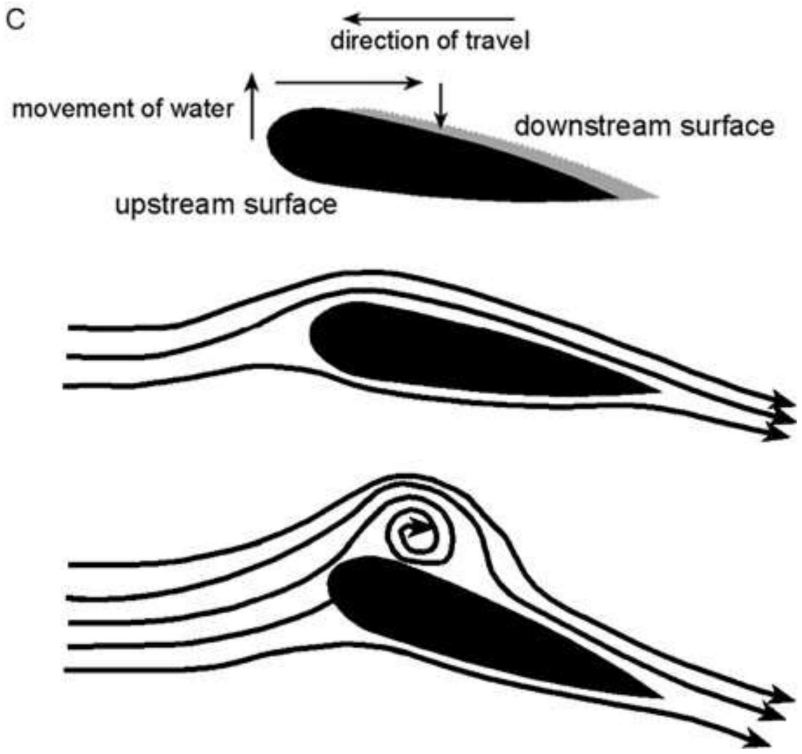


Figura 2.3: Creación de vórtices, ilustrado en [3, Fig. 10.11C].

la eficiencia. Este fenómeno se conoce como *stall* (pérdida de sustentación). Cuando ocurre el *stall*, no solo se reduce la fuerza de elevación, sino que también aumenta la fuerza de arrastre. Esto se debe a que, a ángulos de ataque muy altos, el flujo de agua ya no sigue la superficie de la aleta, sino que se separa completamente, creando una gran turbulencia. Esta turbulencia aumenta la resistencia al movimiento, lo que resulta en un mayor arrastre, en la Figura 2.4 se observa como las fuezas de empuje y arrastre varían con el ángulo de ataque. Por lo tanto, aunque el empuje puede aumentar inicialmente con el ángulo de ataque, alcanza un máximo alrededor de 45° y luego disminuye rápidamente debido al *stall*, mientras que el *drag* continúa aumentando. Este equilibrio entre *lift* y *drag* es crucial para que los peces optimicen su movimiento y minimicen el gasto de energía [3].

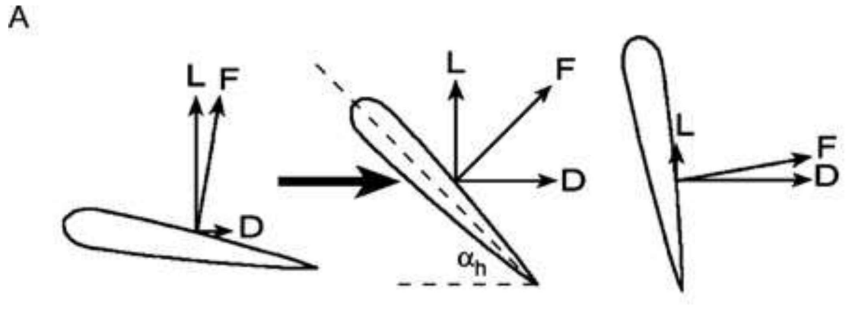


Figura 2.4: Fuerza resultante F con su componenetes de *lift* (L), *drag* (D) y ángulo de ataque α_n menor, igual y mayor a 45^{o}, ilustrado en [3, Fig. 10.11A].

Además de las fuerzas generadas por los vórtices, los peces también aprovechan la reacción

de aceleración para generar empuje. Este fenómeno ocurre cuando las aletas se mueven rápidamente, acelerando el agua circundante. Según la tercera ley de Newton (acción y reacción), el agua ejerce una fuerza en dirección opuesta al movimiento de las aletas. Por ejemplo, cuando las aletas se mueven rápidamente hacia abajo, aceleran el agua hacia abajo, lo que genera una fuerza de reacción hacia arriba. De manera similar, cuando las aletas se mueven rápidamente hacia atrás, aceleran el agua hacia atrás, lo que genera una fuerza de reacción hacia adelante. Este mecanismo es particularmente importante en movimientos rápidos y maniobras, donde las aletas pueden generar fuerzas significativas en un corto período de tiempo [3].

2.1.4 Contribución de las aletas pectorales al movimiento de cabeceo

El movimiento en pitch (cabeceo) en peces está determinado fundamentalmente por la capacidad de las aletas pectorales para regular con precisión las fuerzas verticales y los momentos de torsión. En especies como la trucha arcoíris (*Oncorhynchus mykiss*), la posición ventral de las aletas hace que la línea de acción de la fuerza de frenado pase por debajo del centro de masa, generando un momento de cabeceo significativo que requiere compensación por otras aletas. En contraste, en peces como el pez luna (Lepomis macrochirus), la ubicación dorsolateral de las aletas —cercana al centro de masa— permite que el vector de fuerza de frenado se distribuya en componentes horizontal y vertical, neutralizando este momento desestabilizador [4], [5].

El control neuromuscular específico es crítico para esta función [6], [7]. Músculos como el *arrector ventralis* y el *adductor superficialis* no solo regulan la abducción/aducción, sino que también modulan la desaceleración durante transiciones de movimiento y ajustan el borde de ataque de la aleta, influyendo directamente en la generación de fuerzas verticales [8]. La flexibilidad de los radios y la capacidad de modular activamente la rigidez permiten controlar la magnitud y dirección de las fuerzas, mientras que la curvatura de los radios define la orientación del empuje. Peces altamente maniobrables, como *Embiotoca lateralis*, exhiben un control motor fino que les permite ajustes instantáneos durante la brazada, esencial para correcciones rápidas en pitch. Sin embargo, el desfase temporal entre señales neurales y respuesta mecánica es un factor limitante en este sistema de control [8], [9], [10], [11],[12], [13].

Además de la propulsión, las aletas pectorales contribuyen a la estabilización postural, amortiguando perturbaciones en pitch y otros ejes. Su capacidad para minimizar oscilaciones transversales durante el avance y evitar fases de arrastre negativo en la inversión de la brazada resulta en un empuje estable y continuo. La rotación de la base de la aleta y la modulación del ángulo de ataque optimizan la generación de fuerzas, incluso en morfologías menos especializadas.

En conjunto, las aletas pectorales funcionan como un sistema de control activo, análogo a superficies aerodinámicas, permitiendo ajustes precisos de la orientación vertical. Mediante la modulación de su forma, rigidez y trayectoria, generan fuerzas direccionales que facilitan ascenso, descenso y frenado sin comprometer la estabilidad, asegurando que el vector de fuerza resultante pase por el centro de masa para evitar movimientos no deseados en pitch.

2.2 Actuadores bioinspirados basados en aleaciones con memoria de forma

2.2.1 Aleaciones con memoria de forma

Las Aleaciones con Memoria de Forma (SMA) tienen la capacidad de "recordar" su forma original gracias a un fenómeno conocido como efecto de memoria de forma. Este efecto es la base para su uso en aplicaciones de actuación. Existen dos tipos principales de efecto de memoria de forma:

- Efecto de memoria de forma unidireccional (*One-way*): En la fase de martensita, la aleación puede deformarse fácilmente aplicando una fuerza externa. Una vez que se retira la fuerza, la aleación mantiene la deformación de manera permanente. Sin embargo, al calentarse, el material recupera su forma original. El enfriamiento posterior no cambia la forma a menos que se aplique una fuerza externa nuevamente.

- Efecto de memoria de forma bidireccional (*Two-way*): Además del efecto unidireccional, en este caso el material cambia de forma tanto al calentarse como al enfriarse, sin necesidad de aplicar una fuerza externa. Para lograr este comportamiento, la SMA generalmente debe ser "entrenada" mediante ciclos repetidos de deformación y recuperación.

El efecto de memoria de forma ocurre debido a un cambio en la estructura interna del material, llamado transformación de fase, entre dos estados cristalinos: la martensita, que es estable a temperaturas bajas, y la austenita, que domina a temperaturas más elevadas. En la fase de martensita, el material es más flexible y puede deformarse con facilidad, pero al calentarse, la martensita deformada se convierte en austenita, recuperando así su forma inicial. Este proceso de transformación no sigue un comportamiento lineal y muestra una histéresis térmica significativa, lo que significa que las temperaturas a las que ocurre la transformación durante el calentamiento y el enfriamiento no son las mismas [14].

Para este proyecto, es fundamental destacar que, siguiendo los requisitos del cliente, solo se debe controlar una región lineal del efecto de contracción de los alambres de SMA durante el calentamiento. Esta decisión se basa en que, dentro de esta región, la relación entre la temperatura (o la corriente eléctrica aplicada) y la contracción del alambre (o la resistencia eléctrica) es altamente predecible y fácil de gestionar, esto se puede observar en la Figura

2.5. Al restringir el funcionamiento a esta zona lineal, se minimizan las complicaciones derivadas de la no linealidad y la histéresis térmica inherentes a los SMA, simplificando significativamente el diseño y la implementación del sistema de control. Este enfoque no solo optimiza la eficiencia del sistema, sino que también garantiza un comportamiento más confiable y repetible en aplicaciones prácticas.

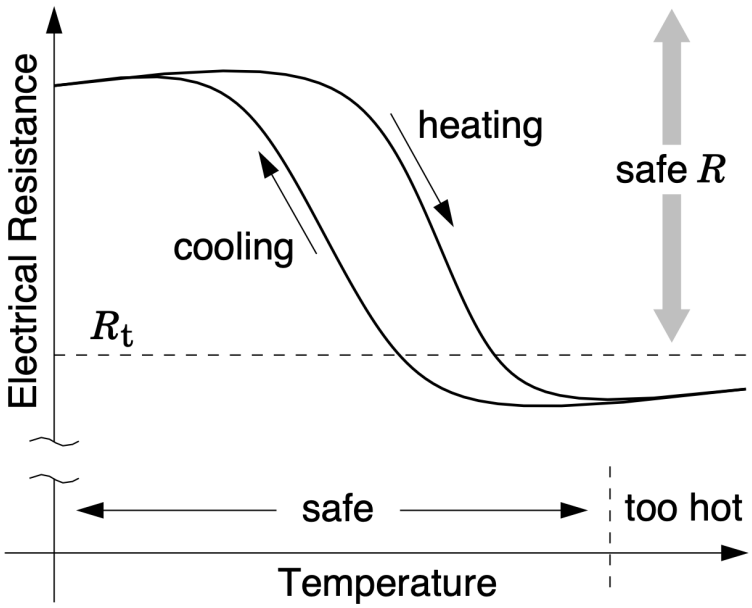


Figura 2.5: Resistencia eléctrica del NiTiNOL vs temperatura. Tomada de [4, Fig. 9].

2.2.2 Actuadores de SMA

Construcción

Los actuadores de SMA están compuestos por alambres de Nitinol, una aleación de níquel-titanio conocida por sus propiedades de memoria de forma. Estos alambres, con un diámetro de 150 micrómetros, están incrustados en una estructura de silicona (PlatSil® FS-20), un material flexible y resistente al agua que actúa como la base de las aletas. La silicona no solo proporciona la flexibilidad necesaria para el movimiento, sino que también asegura que las aletas sean completamente impermeables, una característica esencial para aplicaciones submarinas [15].

Dentro de cada aleta, se disponen dos alambres de SMA en una configuración antagonista. Esto significa que un alambre se activa para doblar la aleta en una dirección, mientras que el otro permanece inactivo. Cuando se invierte el proceso, el segundo alambre se activa para doblar la aleta en la dirección opuesta. Para maximizar la eficiencia del movimiento, los alambres se disponen en forma de *zigzag*, lo que permite que cada segmento del alambre contribuya al desplazamiento total de la aleta. Además, para evitar que los alambres se deslicen dentro de la silicona durante la contracción, se utilizan soportes impresos en 3D con

una forma específica que mantiene los alambres en su posición correcta [15]. La construcción de estos actuadores se ilustra en la Figura 2.6

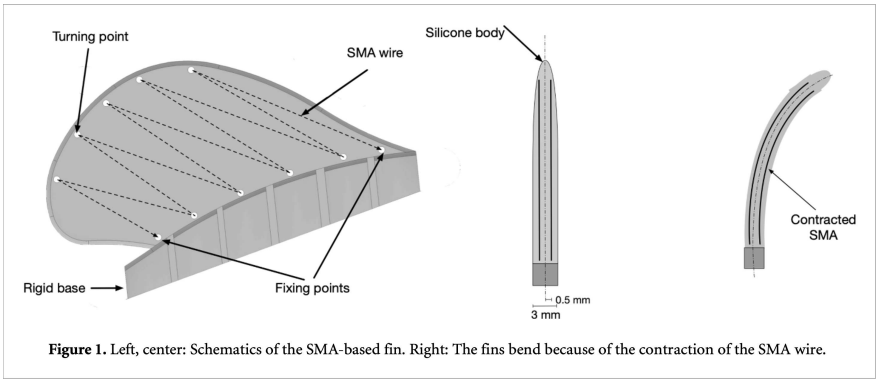


Figura 2.6: Esquemáticos de los actuadores, como se ilustran en [5, Fig. 1].

Una de las principales ventajas de este diseño es su simplicidad mecánica. Al integrar los actuadores de SMA directamente en la estructura de las aletas, se elimina la necesidad de mecanismos externos como motores o cables, lo que resulta en un diseño ligero y compacto. Además, el uso de silicona como material base asegura que las aletas sean completamente impermeables, lo que es esencial para aplicaciones submarinas

Principio de funcionamiento

El principio de funcionamiento de los actuadores de aleaciones con memoria de forma (SMA) se basa en la transformación de fase entre dos estructuras cristalinas: martensita y austenita, como se explicó anteriormente. Este fenómeno permite que los alambres de SMA cambien de forma de manera controlada en respuesta a variaciones de temperatura.

En el contexto de las aletas robóticas, cuando se aplica una corriente eléctrica a uno de los alambres de SMA, este se calienta debido al efecto Joule. El efecto Joule es un fenómeno físico por el cual, cuando una corriente eléctrica pasa a través de un material conductor (en este caso, el alambre de SMA), se genera calor como resultado de la resistencia eléctrica del material [16]. Este calor provoca un aumento de temperatura en el alambre, lo que desencadena la transformación de la fase martensita (estable a bajas temperaturas) a la fase austenita (estable a altas temperaturas). Durante esta transición, el alambre se contrae, generando una fuerza que dobla la aleta en una dirección específica.

Una vez que se retira la corriente, el alambre se enfría, ya sea por conducción térmica o convección en el medio circundante (en este caso, el agua), lo que provoca que el material vuelva a la fase martensita. Durante este enfriamiento, el alambre recupera su forma original, permitiendo que la aleta regrese a su posición inicial. Este ciclo de calentamiento y enfriamiento se repite de manera controlada, permitiendo generar movimientos precisos y repetibles en las aletas.

2.3 Control de SMA

El control de sistemas de aleaciones con memoria de forma es un área de investigación activa debido a las propiedades únicas de estos materiales, como su alta relación fuerza-peso y su capacidad para generar grandes desplazamientos. Sin embargo, las SMA presentan desafíos significativos en su control debido a su comportamiento no lineal, histéresis y baja eficiencia energética. A continuación, se describen los principales enfoques utilizados para el control de sistemas SMA.

2.3.1 Control lineal

Los controladores lineales, como el controlador proporcional-integral-derivativo (PID), son los primeros en ser considerados debido a su simplicidad y amplia aplicación en sistemas de control. Este tipo de controlador combina tres acciones de control: proporcional (P), integral (I) y derivativa (D), para ajustar la señal de control en función del error entre la entrada de referencia y la salida medida.

El controlador PID se puede describir con la Ecuación 2.1 [17].

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{t} \tag{2.1}$$

Dónde:

- *u(t)* es una señal de control.
- *e(t)* es el error en el tiempo, el cual es la diferencia entre una entrada de referencia y la medición actual.
- K_p, K_i y K_d son las ganancias de las partes proporcional, integral y derivativa, respectivamente.

Este tipo de controladores han sido usados para el control de alambres de SMA en varias ocasiones, algunos ejemplos de estos son: Majima et al. [18] combinaron un controlador PID con un lazo de alimentación directa basado en un modelo estático de histéresis de Preisach para alcanzar el estado deseado en un sistema SMA. También, Singh et al. [19] implementaron un controlador PID con ganancias seleccionadas mediante el método de Ziegler-Nichols para el seguimiento de una pala de rotor.

2.3.2 Control no lineal

Dado el comportamiento no lineal de las SMA, los controladores no lineales han demostrado ser más efectivos para manejar la histéresis y otras no linealidades. Estos utilizan técnicas de control no lineal para manejar las no linealidades inherentes de los SMA, como la histéresis y la dependencia de la temperatura. Algunas de las técnicas mencionadas incluyen el control de estructura variable (VSC, por sus siglas en inglés) y el control basado en modelos [20].

El control no lineal puede ser más complejo de implementar, pero puede ofrecer un mejor rendimiento en aplicaciones donde las no linealidades son significativas. A pesar de los numerosos estudios realizados sobre el control de actuadores basados en aleaciones con memoria de forma (SMA), los logros alcanzados han enfrentado limitaciones significativas en términos de velocidad y precisión. Estos actuadores presentan comportamientos no lineales, como histéresis que se asemeja al juego mecánico y efectos de saturación, lo que complica su control preciso. Además, los SMA son comúnmente percibidos como actuadores lentos debido a su respuesta térmica, ya que el tiempo necesario para calentar y enfriar el material restringe su capacidad para responder rápidamente en aplicaciones que demandan movimientos dinámicos y ágiles [20].

Capítulo 3

Metodología

La metodología es un elemento fundamental en la ejecución de cualquier proyecto, puesto que establece un esquema organizado para enfrentar los retos y asegurar la coherencia entre las metas planteadas. En este apartado, se expodrá un enfoque metodológico para la creación de un sistema de control para actuadores basados en alambres SMA.

En los siguientes segmentos, se detallarán cada una de las etapas de la metodología, incluyendo las técnicas de recopilación de información, el análisis de necesidades, la definición de métricas y valores objetivo, la creación de conceptos conceptuales, así como la elección de la concepto final. Este procedimiento busca brindar una perspectiva integral del trabajo realizado y garantizar claridad en las decisiones adoptadas durante la evolución del proyecto.

Al inciar esta etapa ya se tenía en el laboratorio definido el método/circuito de medición y el microcontrolador a usar, por lo tanto se centró solo en las características que debía tener el programa de implementación de PID.

3.1 Pasos de la metodología

La metodología utilzada para el desarrollo de este proyecto corresponde a la propuesta por K.T. Ulrich y S.D. Eppinger en su libro Diseño y Desarrollo de Productos [21].Esta metodología es ideal para este proyecto, gracias a su enfoque estructurado y probado. Este marco garantiza un proceso sistemático, desde la definición de requisitos hasta la validación final, lo que es crucial dada la complejidad del proyecto. Dicho proyecto combina desafíos térmicos, mecánicos y de control, además de las exigentes condiciones del entorno subacuático. La metodología permite gestionar estos retos mediante análisis de necesidades y benchmarking. Asegurando que el sistema cumpla con los requerimientos prácticos de operación submarina, sino que también integra herramientas complementarias, como modelado en MATLAB y pruebas en tanques de agua, para optimizar el comportamiento de los SMA. De esta forma, el marco de Ulrich y Eppinger no solo organiza el desarrollo, sino que reduce incertidumbres en un proyecto donde la fiabilidad es un aspecto crítico.

Ulrich y Eppinger se basa en la resolución o abordaje de una serie de bloques, que se abor-

dan de manera serial. Este proceso tiene una naturaleza iterativa, por lo tanto si después de reflexionar al final de cada fase se volverá a esta etapa a aplicar las correcciones correspondientes.

Las etapas correspondientes de esta metodología son las siguentes [21].

- Identificación de necesidades: Esta primera etapa metodológica consiste en recopilar, estructurar y priorizar sistemáticamente todos los requisitos que debe cumplir la solución. A través de entrevistas y cuestionarios con el cliente, se transforman sus expectativas en necesidades técnicas claras y cuantificables, asignando prioridades según su importancia. El resultado es una matriz de requerimientos validada que servirá como base objetiva para guiar el desarrollo del proyecto, asegurando que cada decisión de diseño responda a las necesidades reales del cliente desde las fases iniciales.

- Establecimiento de especificaciones objetivo: Esta segunda etapa consiste en traducir las necesidades del cliente en parámetros técnicos medibles que servirán como criterios de evaluación para las soluciones propuestas. Para cada requerimiento identificado previamente, se establecen métricas cuantitativas con valores objetivo (ideal) y mínimos aceptables, fundamentados en investigación técnica y benchmarking. Estas especificaciones funcionan como puente entre las expectativas del cliente y los desarrollos ingenieriles, permitiendo evaluar objetivamente el cumplimiento de cada necesidad durante todo el ciclo de diseño. La metodología exige que cada métrica sea trazable a necesidades específicas y que sus valores umbrales se justifiquen mediante datos técnicos relevantes.

- Generación de conceptos: Esta tercera etapa tiene como objetivo desarrollar múltiples alternativas de diseño que aborden integralmente las necesidades del cliente. El proceso inicia modelando el sistema como una "caja negra" para identificar claramente sus entradas, salidas y funciones principales, que luego se descomponen en subfunciones específicas. Para cada subfunción se realiza una búsqueda sistemática de conceptos, combinando investigación de fuentes especializadas (patentes, literatura técnica) con técnicas creativas internas, lo que permite generar soluciones híbridas e innovadoras. El resultado es un portafolio de conceptos diversos que serán evaluados en etapas posteriores, asegurando que se explore un espectro amplio de posibilidades técnicas antes de seleccionar la solución óptima.

- Selección de candidatos: Esta cuarta etapa metodológica evalua sistemáticamente todas las alternativas generadas para identificar la solución más viable. El proceso utiliza una matriz de selección ponderada que compara cada concepto contra los criterios técnicos establecidos y las necesidades priorizadas del cliente. Se inicia con una fase de filtrado donde se descartan opciones no factibles, seguida de una evaluación detallada que analiza fortalezas, debilidades y posibles combinaciones entre conceptos. Como resultado se selecciona el diseño óptimo (o combinación de diseños) que mejor equilibra desempeño técnico, viabilidad y cumplimiento de requerimientos, el cual avanzará a las etapas de desarrollo e implementación. La metodología garantiza una decisión

objetiva mediante el uso de métricas cuantificables y un concepto de referencia para comparaciones consistentes.

- Validación y pruebas de concepto: Para esta última etapa, se da la validación del concepto ganador seleccionado. Se realizan pruebas de concepto para validar que se alcancen los valores objetivo mediante la definición de variables de muestra y factores de influencia para cada prueba y se detalla de que manera estas se deben realizar. Además, se analizan los resultados para verificar la validez de la solución propuesta. Esto se puede encontrar en el capítulo 5.

3.1.1 Identificación de necesidades

Inicialmente, se recolectaron datos del cliente utilizando métodos como entrevistas, correos electrónicos, documentos internos y conversaciones. De cada interacción se extrajeron frases importantes que se pudieran identificar como una necesidad y se procesaron tomando en cuenta lo siguiente [22]:

- Formular las necesidades en función de las capacidades que debe tener el producto, no de las soluciones técnicas específicas para implementarlas.
- Expresar la información con la misma precisión con que fue expresada por el cliente, sin realizar afirmaciones de detalle que no están presentes en la definición de necesidades.
- Utilizar enunciados de forma positiva.
- Emplee enunciados en forma afirmativa, utilizando negaciones únicamente cuando sea estrictamente necesario.
- Emplee enunciados en forma afirmativa, utilizando negaciones únicamente cuando sea estrictamente necesario.
- Defina cada necesidad como una característica medible del producto.
- Evitar las palabras debe y debería.

Una vez identificadas y documentadas todas las necesidades del cliente, el siguiente paso crítico es asignarles un nivel de importancia relativa utilizando una escala numérica del 1 al 5, como se muestra en la Tabla 3.1. Este proceso de priorización no solo establece una jerarquía clara entre los requerimientos, sino que también sirve como mecanismo de validación para confirmar que se han interpretado correctamente las expectativas del cliente.

La escala de importancia se estructura en cinco niveles: los valores 5 (imprescindible) y 4-3 (deseable) identifican funciones críticas o valoradas, mientras que el 2 (indiferente) y 1 (indeseable) ayudan a descartar elementos superfluos o contraproducentes. Para implementar este sistema, se utilizó un formulario estandarizado que incluía definiciones precisas de cada nivel, el listado completo de necesidades y espacios para asignar prioridades. Este ejercicio

no solo formalizó los criterios de evaluación, sino que también generó retroalimentación valiosa del cliente, permitiendo ajustar el alcance del proyecto y resolver ambigüedades antes de avanzar a etapas de diseño.

Tabla 3.1: Escala de priorización de necesidades.

Importancia	**Significado**
1	La función es indeseable.
2	La función no es importante, pero me gustaría tenerla.
3	Sería bueno tener esta función, pero no es necesaria.
4	La función es altamente deseable, pero consideraría una solución sin ella.
5	La función es de importancia crítica.

Las necesidades identificadas con su importancia respectiva se pueden observar en la Tabla 3.2. Es importante recalcar que ninguna necesidad tuvo una puntuación menor a tres; esto indica que se identificaron las necesidades del cliente adecuadamente.

Tabla 3.2: Necesidades identificadas y su importancia respectiva.

Núm	**Necesidades**	**Importancia**
1	El SD permite ángulos de al menos 28º.	4
2	El SD responde en décimas de segundo.	4
3	El SD ajusta movimientos usando retroalimentación.	3
4	El SD limita la corriente para proteger los SMA.	3
5	El SD prioriza la rapidez de respuesta.	4
6	El SD evita movimientos fuera de rango.	4
7	El SD detecta fallos básicos en actuadores.	4
8	EL SD está optimizado para 2 actuadores.	4
9	El SD usa protocolos de comunicación estándar.	4
10	El SD correlaciona resistencia eléctrica del SMA con posición angular.	4

3.1.2 Establecimiento de especificaciones objetivo

En esta fase, se transformaron las necesidades del cliente en parámetros técnicos medibles, estableciendo métricas claras que permitieran evaluar objetivamente el cumplimiento de cada requerimiento. Cada métrica se asoció directamente con una o varias necesidades identificadas previamente, asegurando una cobertura completa de todas las expectativas del cliente. En algunos casos, una misma métrica pudo aplicarse a múltiples necesidades cuando estas compartían características evaluables similares.

Para cada métrica definida, se establecieron dos valores de referencia: un valor ideal que representaba el óptimo deseado y un valor marginal que marcaba el límite mínimo aceptable. La amplitud de estos rangos varió según la importancia relativa de cada necesidad, siendo más estrictos para los requerimientos críticos y más flexibles para los secundarios.

El conjunto completo de métricas definidas se encuentra documentado en la Tabla 3.3. Como mecanismo de verificación, se empleó una matriz de correlación necesidades-métricas, Tabla 3.4, que garantiza la cobertura integral de todos los requerimientos. En esta última tabla el nombre de las necesidades empiezan con la letra "N" y su número corresondiente y las métricas empiezan con la letra "M".

Tabla 3.3: Métricas establecidas.

Ítem	Necesidad	Métrica
1	1, 6	Ángulo máximo alcanzado.
2	2	Tiempo de respuesta.
3	3, 5	Error angular aceptable.
4	4	Cambio de resistencia.
5	7	Detección de fallos (circuito abierto).
6	8	Funcionamiento estable con 2 actuadores simultáneos.
7	9	Comunicación funcional a 400 kHz.
8	10	Coeficiente R^2 en curva de calibración.

Tabla 3.4: Correspondencia entre métricas y necesidades de la Tabla 3.3.

Ítem	M1	M2	M3	M4	M5	M6	M7	M8
N1	X							
N2		X						
N3			X					
N4				X				
N5			X					
N6	X							
N7					X			
N8						X		
N9							X	
N10								X

Para establecer los valores ideales y marginales de cada métrica, se siguió un proceso que combina dos fuentes de información. Por un lado, se consideraron los requerimientos específicos expresados directamente por el cliente. Por otro lado, para la mayoría de las métricas se recurrió a fuentes externas, las cuales se cruzaron con las expectativas del cliente.

Este enfoque permitió definir rangos realistas y técnicamente fundamentados. Los valores ideales representan el desempeño óptimo que se desea alcanzar, mientras que los valores marginales establecen el límite mínimo aceptable para considerar que se cumple con la necesidad. La determinación de estos rangos tuvo en cuenta tanto la importancia relativa de cada necesidad como la viabilidad técnica y económica de su implementación. Estos valores se pueden ver en la Tabla 3.5.

Tabla 3.5: Valores marginales e ideales para las métricas de la Tabla 3.3

Métrica	Unidades	V. Marginal	V. Ideal
1	Deg	≥26	≥ 28
2	ms	≤500	≤ 300
3	% Error	≤15	≤10
4	Ω	14	15
5	Binaria	Sí	Sí
6	Binaria	Sí	Sí
7	ms	≤50	≤30
8	R^2	≥ 0.7	≥ 0.8

Métrica 1

Los valores de la primera métrica, *"Ángulo máximo alcanzado"*, se obtuvieron a partir de un manual de fórmulas para la contracción de un alambre de aleación con memoria de forma (SMA) adherido a una tira de policarbonato, elaborado por Claudio Rossi y William Coral [23].

En dicho manual se especifica que, para un alambre simple, el ángulo de flexión es de aproximadamente 28°. Tras consultar con el cliente, se estableció que el valor ideal debía ser mayor o igual a este valor, mientras que el valor marginal aceptable sería superior a 26°.

Métrica 2

Para la métrica *"Tiempo de respuesta"*, el estudio [24] establece que el tiempo de respuesta total de un actuador SMA puede variar desde décimas de segundo (para actuadores muy finos con enfriamiento forzado y calentamiento rápido) hasta varios segundos (para actuadores más grandes o bajo convección libre).

El factor que más influye en esta variación es la potencia eléctrica, debido principalmente a que:

- La fase de calentamiento puede ser extremadamente corta, ya que la transformación puede iniciarse en cintas delgadas de Ni-Ti en tan solo 0.005 segundos con un impulso de calentamiento potente.
- El aumento de corriente DC (corriente continua) reduce drásticamente el tiempo de calentamiento.

Considerando estos factores y el requerimiento del cliente de que el actuador responda en "décimas de segundo", se establecieron los siguientes valores objetivo:

- Valor ideal: menor a 300 ms
- Valor marginal: menor a 500 ms

Métrica 3

El *"Error angular aceptable"* representa la discrepancia entre la posición real y la posición esperada del actuador. Aunque la aplicación no requiere alta precisión, el cliente especificó como requisito mantener este error controlado a menos de 15 por ciento:

- Valor marginal: $< 15\%$
- Valor ideal: $< 10\%$

Métrica 4

Para garantizar la seguridad del actuador SMA y prevenir sobrecalentamientos, se implementa una métrica que monitorea el cambio de resistencia del alambre de Ni-Ti en lugar de establecer límites fijos de corriente. Este enfoque resulta más efectivo porque la corriente requerida para alcanzar la temperatura de transformación depende críticamente de condiciones ambientales, particularmente de la temperatura del medio circundante.

La base física de esta métrica radica en el comportamiento termoresistivo característico de los alambres SMA. Cuando el material se calienta por encima de su temperatura de transformación, experimenta una contracción del 3% al 4% de su longitud inicial, acompañada de un aumento proporcional en su diámetro transversal debido a la conservación de volumen. [25].

Hacerlo de esta manera presenta varias ventajas, como:

- Elimina la necesidad de calibración previa para diferentes condiciones ambientales. Por ejemplo: si el agua está más fría se necesita más corriente para calentar el actuador.
- Previene daños por sobrecalentamiento.
- Optimiza el consumo energético al suministrar sólo la potencia necesaria.
- Permite operación segura en un rango amplio de temperaturas.

Para determinar estos valores hay que hacerlo experimentalmente, para esto se realizaron mediciones con un actuador de aleta pectoral. Se aplicó una corriente de aproximadamente 700 mA, suficiente para alcanzar el máximo desplazamiento del actuador. El proceso se repitió 10 veces, registrando los valores de resistencia tanto en posición de reposo como en máxima deformación. Para cada repetición, se calcularon las diferencias de resistencia mediante la Ecuación 3.1. Los resultados se pueden ver en la tabla 3.6, el intervalo de confianza se calculó para una confianza del 95%

$$\%\Delta R = \frac{R_{\max} - R_{\min}}{R_{\max}} * 100 \tag{3.1}$$

Tabla 3.6: Datos del experimento para calcular el porcentaje de cambio de la resistencia del SMA del actuador de aleta pectoral.

N	$R_{\mathbf{max}}$ **(± 0.072 Ω)**	$R_{\mathbf{min}}$ **(± 0.434 Ω)**	**ΔR (± 0.510 %)**
1	17.195	15.880	-7.647
2	17.129	15.769	-7.940
3	17.075	15.130	-11.391
4	16.983	14.854	-12.536
5	16.877	14.710	-12.840
6	16.727	14.696	-12.142
7	16.705	14.689	-12.068
8	16.713	14.684	-12.140
9	16.639	14.672	-11.822
10	16.639	14.640	-12.013
Promedio	16.868	14.992	-11.254
Intervalo de confianza	[16.823,16.912]	[14.723,15.261]	[-11.570,-10.937]

Los resultados experimentales (Tabla 3.6) muestran una resistencia mínima promedio de 15.00 Ω. Mediante un análisis estadístico con un nivel de confianza del 95 %, se determinó que el valor real se encuentra en el intervalo 14.68 Ω - 15.30 Ω. Este intervalo fue calculado considerando la distribución t-Student para la muestra experimental, lo que proporciona una estimación robusta del rango de valores esperados para este parámetro.

Este valor puede considerarse como un valor ideal, ya que garantiza un rango de movimiento total sin comprometer el actuador. Sin embargo, si se requiere un límite mínimo menor, se puede tomar como valor marginal 14 Ω sin comprometer significativamente la integridad del dispositivo.

Métrica 5

La métrica de detección de fallos se mantendrá como variable binaria, según lo acordado con el cliente. Esta decisión se basa en que la implementación mediante software es técnicamente viable y se considera que funcionará correctamente en la mayoría de los casos. Además, esta funcionalidad se ha concebido como una característica adicional al sistema de control principal.

Métrica 6

Al igual que con la métrica anterior, se ha adoptado el enfoque binario para esta funcionalidad. El sistema debe garantizar el control simultáneo de al menos dos actuadores como requisito fundamental.

Métrica 7

En sistemas de control que requieren lectura de sensores, una menor latencia mejora significativamente la efectividad de la acción resultante [26]. Para cumplir con requisitos de tiempo real, la literatura establece que la latencia bidireccional debe ser inferior a 50 ms [27]. Sin embargo, dado que nuestro sistema debe garantizar una respuesta total menor a 300 ms, por lo tanto la latencia de comunicación debe ser al menos menor a esto, se proponen los siguientes umbrales de diseño:

- **Valor marginal**: 50 ms (límite superior para comunicación en tiempo real)
- **Valor ideal**: 30 ms (margen adicional para robustecer el sistema)

Estos valores aseguran que, incluso en condiciones adversas, el sistema mantenga su capacidad de respuesta dentro de los requisitos operativos.

Métrica 8

Para la métrica del coeficiente de determinación R^2 en la curva de calibración, se establecieron los siguientes umbrales:

- Valor marginal: $R^2 \geq 0.7$
- Valor ideal: $R^2 \geq 0.8$

Estos umbrales se seleccionaron por ser considerados aceptables en el ámbito de las ciencias exactas [28, 29].

3.1.3 Generación de conceptos

La generación de conceptos debe precederse de una identificación exhaustiva de todas las funciones del sistema mediante una división funcional. Este proceso implica descomponer el sistema en bloques progresivamente más pequeños y específicos hasta alcanzar un nivel de detalle adecuado para proponer conceptos individuales. Esta parte inicia con un diagrama de primer nivel que establece únicamente las entradas y salidas principales, proporcionando así una base clara para el desarrollo posterior de subsistemas específicos. Este enfoque sistemático garantiza la cobertura completa de los requisitos funcionales antes de la etapa conceptual.

Para este proyecto se desarrolló únicamente un bloque funcional principal, correspondiente al sistema de control, dado que constituye el único subsistema complejo a diseñar. Este bloque gestiona específicamente el control de la fase de austenita en alambres de aleación con memoria de forma (SMA). Como se observa en la Figura 3.1, su diagrama de primer nivel recibe señales de los sensores (integrados en los alambres SMA) y genera como salidas: (1) datos procesados, (2) movimiento de actuadores, y (3) señales de control.

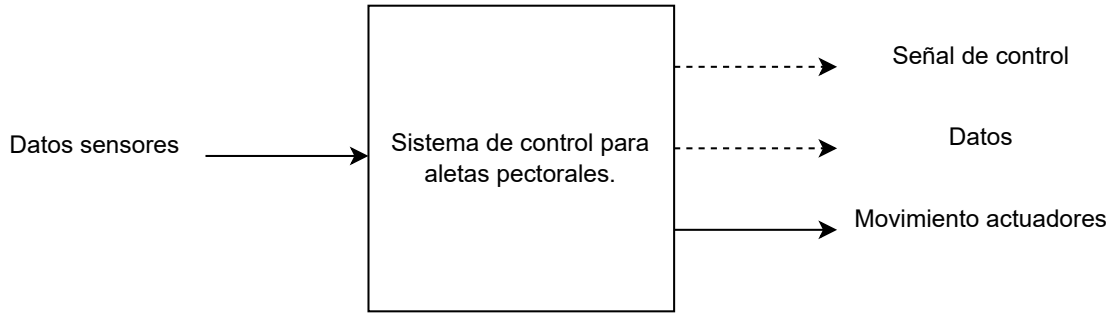


Figura 3.1: Bloque funcional de primer nivel.

El diagrama de segundo nivel, Figura 3.2, que descompone las funciones del sistema de control, presenta una estructura simplificada basada en el flujo básico de un sistema de control convencional. Esta descomposición considera cuatro aspectos fundamentales: primero, la generación de valores de referencia, que aborda el que posiciones el controlador debe seguir; segundo, la seguridad del actuador, enfocado en mantener el buen funcionamiento del actuador y que no se dañe; y tercero, la estrategia de control, donde se evalúan alternativas lineales y no lineales para la implementación final; y cuarto, como mostrar datos, que toma en cuenta la manera en la que se presentan los datos al usuario.

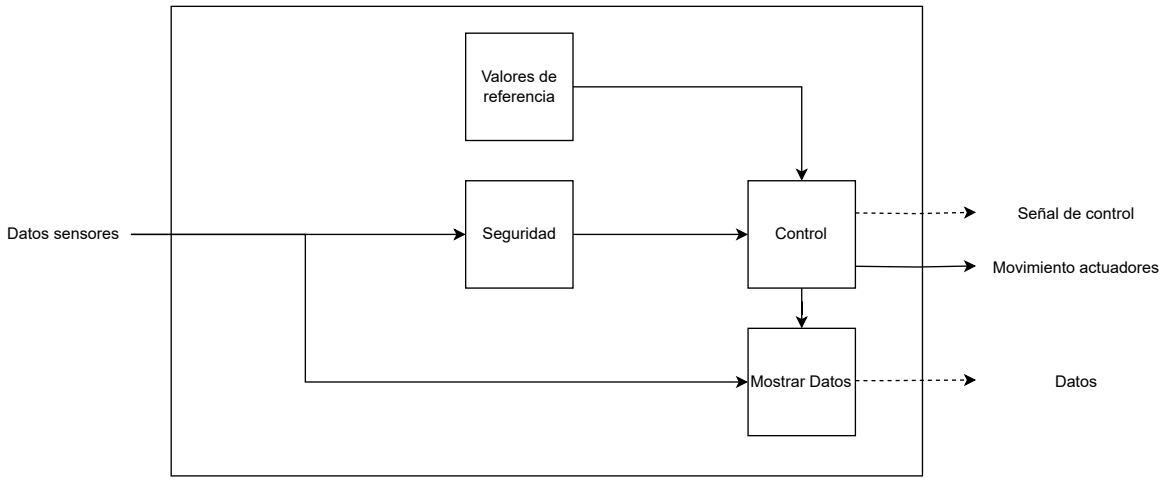


Figura 3.2: Bloque funcional de segundo nivel.

Con todas las funciones identificadas y desglosadas, se generó una lista de conceptos potenciales para cada subsistema. El proceso comenzó con una búsqueda interna mediante lluvia de ideas, que produjo los conceptos iniciales, seguida de una investigación externa para complementar las posibles soluciones no consideradas en la primera etapa. Las combinaciones conceptuales resultantes, ilustradas en la Figura 3.3. Un análisis detallado de sus características y viabilidad se desarrollará posteriormente durante la etapa de selección del concepto ganador.

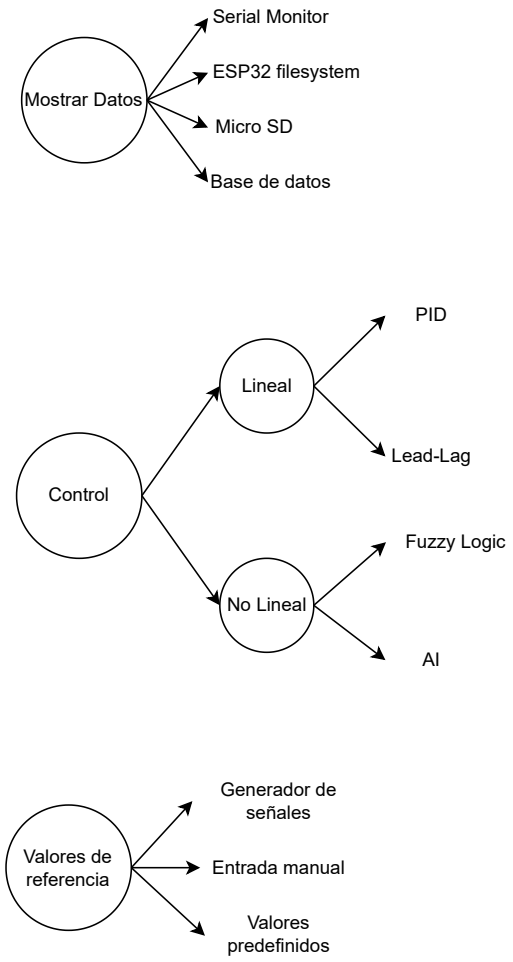


Figura 3.3: Combinaciones conceptuales generadas.

Para la función de mostrar los datos, el análisis inicial mostró preferencia por métodos como mostrar los datos de manera serial en la computadora o guardarlos en un archivo .txt en una tarjeta microSD. Sin embargo, una investigación más profunda reveló otros protocolos como guardar los datos en una base de datos en la nube o en la misma memoria flash del microcontrolador [30]. La selección final incluyó estos cuatro métodos.

La decisión sobre cómo generar los datos de referencia para el controlador se tomó mediante una lluvia de ideas. Se definieron tres opciones principales: que el usuario introduzca los datos manualmente, integrar un generador de señales directamente en el programa o utilizar archivos con valores predefinidos que puedan cargarse cuando sea necesario.

La metodología de diseño del controlador combinó enfoques clásicos (PID y Lead-Lag) con técnicas no lineales investigadas adicionalmente, particularmente aquellas relacionadas con inteligencia artificial. Se consideraron dos alternativas avanzadas: control mediante redes neuronales artificiales y sistemas de lógica difusa (fuzzy logic), seleccionadas por su capacidad

para manejar no linealidades. Este enfoque híbrido permite comparar soluciones establecidas con métodos adaptativos modernos.

No se evaluaron opciones adicionales para la función de seguridad del actuador, ya que esta dependerá directamente del diseño específico del controlador. La investigación relacionada con este tipo de actuadores en particular proviene del trabajo desarrollado en el laboratorio, y no se cuenta con muchas alternativas documentadas. Además, ya existe una métrica establecida que está estrechamente vinculada a esta función. Por ello, se decidió limitar el valor mínimo de resistencia del actuador a los niveles de estado estable. En caso de detectar un valor inferior al límite, el sistema se apagará automáticamente o reducirá temporalmente la corriente suministrada al actuador hasta que se registre una medición por encima del umbral permitido.

3.1.4 Selección de concepto

El proceso de selección de conceptos se implementó mediante un enfoque estructurado en dos etapas secuenciales. La primera etapa consistió en un filtrado preliminar donde cada concepto candidato fue evaluado comparativamente contra un diseño de referencia preestablecido. Esta evaluación determinó cuantitativamente si el concepto representaba una mejora o retroceso en relación con los criterios técnicos definidos, utilizando un sistema de puntuación que sumaba algebraicamente los impactos positivos y negativos identificados. Los conceptos que no superaron un umbral mínimo de mejora fueron descartados en esta fase inicial.

La segunda etapa empleó una matriz de evaluación más rigurosa que incorporaba pesos específicos para cada criterio de diseño. Estos pesos, normalizados para sumar el 100% del valor total, reflejaban la importancia relativa de cada requerimiento del sistema. Cada concepto sobreviviente al filtrado inicial fue puntuado en una escala ordinal del 1 al 5, donde la unidad indicaba un desempeño inferior al concepto de referencia, el 3 representaba equivalencia funcional, y el 5 denotaba una mejora significativa. Las puntuaciones intermedias (2 y 4) calificaban variaciones menores respecto a la referencia.

El cálculo del desempeño global difirió entre etapas: el filtrado inicial utilizó una suma simple de valores, mientras que la evaluación final aplicó una suma ponderada que consideraba tanto las puntuaciones como los pesos asignados. Los criterios de evaluación se derivaron metódicamente de los requisitos del sistema, priorizando aquellos aspectos más sensibles a las decisiones conceptuales. Este enfoque sistemático permitió una comparación objetiva y reproducible entre alternativas de diseño, garantizando que la selección final optimizara el cumplimiento de las necesidades técnicas identificadas.

El proceso inicia con la generación de combinaciones conceptuales, como se muestra en la Tabla 3.7, mediante la integración selectiva de los conceptos previamente desarrollados. Dada la complejidad que implicaría evaluar todas las posibles combinaciones, se implementó un criterio de selección basado en juicio propio que conservó únicamente las alternativas más interesantes, descartando aquellas con menor viabilidad técnica.

Todos los conceptos recibieron identificadores alfabéticos únicos para su trazabilidad. El

Concepto D emerge como la opción más equilibrada para servir como concepto de referencia en el diseño del sistema de control. Su principal fortaleza radica en combinar simplicidad operativa, escalabilidad en el manejo de datos y confiabilidad en el control, sin introducir complejidades prematuras. Esta combinación de atributos lo posiciona como punto de comparación ideal.

Tabla 3.7: Conceptos generados para el sistema de control.

Conceptos Generados	
A	Valores Manuales + Serial + Fuzzy Logic
B	Valores Pred. + SD + Lead-Lag
C	Generador + Serial + PID
D	**Valores Manuales + Base de datos + Lead-Lag**
E	Valores Pred. + SD + AI
F	Generador + ESP32 flash + Lead-Lag

Los conceptos se evaluaron mediante los siguientes criterios comparativos:

1. **Facilidad de implementación**: Los sistemas de control lineal, como los compensadores *Lead-Lag* o PID, requieren una implementación algorítmica más sencilla en comparación con técnicas no lineales (lógica difusa o redes neuronales). Esta simplicidad se extiende también a la visualización de datos: imprimir valores por comunicación serial demanda menos desarrollo que configurar una base de datos o gestionar archivos en una tarjeta SD. La reducción de capas técnicas acelera la fase de prototipado y minimiza errores iniciales.

2. **Costo y recursos**: El diseño debe equilibrar el tiempo de desarrollo y los componentes físicos. Por ejemplo, sintonizar un controlador PID implica ajustar parámetros empíricos, mientras que entrenar un modelo de IA requiere recopilar datos, seleccionar arquitecturas y validar resultados, incrementando el costo temporal. En hardware, soluciones como el almacenamiento en la memoria flash del ESP32-S3 eliminan la necesidad de tarjetas SD (que además requerirían adaptadores por la falta de lector integrado), reduciendo costos materiales y puntos de fallo.

3. **Facilidad de actualización**: Los algoritmos lineales permiten modificaciones incrementales (como ajustar ganancias o polos/ceros) sin rediseñar la estructura completa del controlador. En contraste, sistemas basados en IA pueden necesitar reentrenamiento ante cambios en los actuadores, y protocolos como bases de datos exigen actualizar esquemas de almacenamiento. Esta flexibilidad es crítica en etapas iterativas del proyecto.

4. **Escalabilidad**: Un sistema pensado para controlar múltiples actuadores SMA debe evitar cuellos de botella computacionales. Los controles lineales, al tener menor carga de procesamiento, facilitan su replicación en paralelo. Además, protocolos como el envío de datos por serial o HTTP (en lugar de SD) permiten integrar más dispositivos sin depender de hardware adicional.

5. **Robustez y fiabilidad**: La tolerancia a fallos se optimiza al minimizar componentes externos (ej.: evitar tarjetas SD, propensas a corrupción) y priorizar redundancia (como backups en la nube). Técnicas lineales, al estar matemáticamente bien caracterizadas, ofrecen estabilidad predecible frente a perturbaciones, mientras que sistemas no lineales pueden introducir comportamientos caóticos no anticipados.

6. **Capacidad de simulación**: Modelar matemáticamente sistemas lineales (con herramientas como MATLAB o Simulink) permite predecir respuestas ante distintas entradas y validar el diseño antes de su implementación física. Por el contrario, sistemas no lineales (como los basados en IA) exigen simulaciones numéricas intensivas y sus resultados son menos interpretables, ralentizando la fase de análisis.

El filtrado de conceptos se puede ver en la Tabla 3.8.

Tabla 3.8: Matriz de filtrado de conceptos

Filtrado de conceptos						
Criterios de selección	A	B	C	**D**	F	G
Facilidad de implementación	-	0	0	0	-	-
Costo y recursos	-	-	0	0	-	0
Facilidad de actualización	-	0	+	0	-	0
Escalabilidad	-	-	+	0	-	0
Robustez y fiabilidad	-	-	0	0	-	0
Capacidad de simulación	-	0	0	0	-	0
Suma +	0	0	2	0	0	0
Suma 0	0	3	4	6	0	5
Suma -	6	3	0	0	6	1
Evaluación neta	-6	-3	2	0	-6	-1
Lugar	5	4	1	2	5	3
¿Continuar?	NO	NO	SÍ	SÍ	NO	NO

Aunque el análisis identificó claramente un concepto ganador, se optó por avanzarlo a la siguiente fase junto al concepto de referencia sin combinarlo con el concepto G debido a su notable similitud. Esta exploración metódica sirvió para corroborar la robustez de la solución óptima seleccionada. El desarrollo completo de este proceso de evaluación se detalla en la Tabla 3.9. En esta tabla se puede ver que aún así el Concepto C sale ganador, por lo que se desarrollará.

Tabla 3.9: Matriz de evaluación de conceptos

Evaluación de conceptos					
		Conceptos			
		C		**D**	
Criterios de selección	**Peso**	**Calificación**	**Evaluación Ponderada**	**Calificación**	**Evaluación Ponderada**
Facilidad de implementación	0.20	3	0.60	**3**	0.60
Costo y recursos	0.25	**3**	0.75	2	0.50
Facilidad de actualización	0.20	**3**	0.60	3	0.60
Escalabilidad	0.15	4	0.60	**3**	0.45
Robustez y fiabilidad	0.10	3	0.30	**3**	0.30
Capacidad de simulación	0.10	3	0.30	**3**	0.30
	Puntos	3.15		2.75	
	Lugar	1		2	
¿Desarrollar?		**SÍ**		NO	

Capítulo 4

Diseño del sistema de control para actuadores bioinspirados de aletas pectorales

El solución presente desarrolla un sistema de control adaptable para las aletas pectorales del robot bioinspirado del CAR-UPM/CSIC, con capacidad de modificación para su aplicación en otros sistemas de aletas. Esta solución aborda dos necesidades críticas de la investigación: (1) flexibilidad arquitectónica para distintos actuadores, y (2) optimización de recursos al evitar el desarrollo de múltiples controles específicos. Como resultado, se reduce sustancialmente el tiempo de implementación para nuevas configuraciones, permitiendo enfocar los esfuerzos en la mejora de los demás sistemas del robot.

Este capítulo se estructura en cuatro secciones principales:

La Sección 4.1 detalla el proceso de fabricación de los actuadores, incluyendo los diseños de moldes y pines empleados en su manufactura.

La Sección 4.2 describe las modificaciones realizadas en el sistema electrónico de medición y control, junto con los principios de funcionamiento del subsistema resultante.

La Sección 4.3 presenta la metodología de diseño del sistema de control, abordando los criterios técnicos y las decisiones de implementación.

Finalmente, la Sección 4.4 documenta la implementación completa del sistema, integrando todos los subsistemas desarrollados.

4.1 Fabricación de los actuadores

Los actuadores se fabricaron según la metodología reportada en [15], integrando alambres SMA en una estructura biomimética de aleta pectoral basada en *Micropterus salmoides*. El mecanismo de actuación aprovecha la contracción térmica inducida eléctricamente en los alambres SMA, posicionados a 0.5 mm del centro estructural para producir flexión asimétrica.

La configuración en zig-zag de los alambres replica la disposición muscular observada en la especie de referencia, optimizando la transferencia de fuerza a la esctrutura de la aleta.

Los actuadores emplean alambres SMA de aleación Ni-Ti (Dynalloy Muscle Wires) con diámetro de 150μmm y temperatura de activación de 70 ^{o}C. El encapsulado utiliza silicona PlatSil Gel-25 (Polytec Development Corp) caracterizada por dureza Shore A25, tiempo de trabajo de 5 minutos y tiempo de curado de 60 minutos. Los moldes y pines se fabricaron mediante impresión 3D de resina usando una impresora ANYCUBIC Photon Mono X UV.

El proceso de fabricación de los actuadores se realiza mediante el siguiente procedimiento detallado:

1. Preparación de pines de posicionamiento: Se insertan pines en el molde diseñado, los cuales cumplen una función crítica: fijar la posición exacta del alambre de SMA dentro de la estructura de silicona. Estos pines presentan un diseño específico con puntos de fractura controlada tanto en su base como en su cabeza, permitiendo que durante el desmolde solo permanezca incrustada la sección superior del pin dentro del actuador terminado, manteniendo así el alambre SMA en su posición designada.

2. Instalación y tensado del alambre SMA: El alambre de aleación con memoria de forma (SMA) se enrolla cuidadosamente alrededor de los pines siguiendo el patrón en zig-zag predefinido. Durante esta operación:

- Se aplica una pequeña cantidad de adhesivo instantáneo en los puntos de contacto entre el alambre y los pines para su fijación.
- Se verifica meticulosamente la tensión del alambre, asegurando que:
  - No presente holguras o curvaturas no deseadas
  - Mantenga una trayectoria completamente recta entre puntos de anclaje
  - Conserve una distancia constante y uniforme respecto a la base del molde en toda su extensión

3. Conexión eléctrica: Los extremos libres del alambre SMA se unen mecánica y eléctricamente a los cables de alimentación mediante refuerzo con terminales crimpados.

4. Colado de silicona: Se prepara y vierte el compuesto de silicona PlatSil Gel-25 en el molde, asegurando que:

- Cubra completamente el ensamble de alambres y pines
- No queden bolsas de aire atrapadas
- Se respete el tiempo de trabajo de 5 minutos antes del inicio del gelificado.

5. Curado y desmolde: El conjunto se deja curar, mínimo por un par de horas o idealmente 24 horas. Posteriormente:

- Se fracturan controladamente los pines en sus puntos débiles designados
- Se extrae cuidadosamente el actuador del molde
- Se retira cualquier exceso de silicona

Este proceso completo, se encuentra documentado visualmente en el Anexo A, donde pueden observarse cada una de las etapas.

4.1.1 Diseño de los moldes

El diseño de los moldes se desarrolló a partir de un modelo existente creado por William Coral, incorporando las siguientes modificaciones clave:

- **Escalado dimensional**: Aumento del 20% en todas las dimensiones del molde original
- **Reubicación de guías**: Desplazamiento de 5 mm en la posición de los agujeros para pines
- **Adición de características**: Incorporación de nuevos agujeros en la base para alojar dos alambres SMA independientes

Estos cambios permitieron implementar un actuador con dos grados de libertad (DoF), donde los segmentos superior e inferior pueden moverse independientemente gracias a la configuración dual de alambres SMA.

Proceso de evolución del diseño:

Primer enfoque (molde cerrado): Se implementó inicialmente un molde bipartido (Figura 4.1), similar a un molde de inyección convencional. Sin embargo, este diseño presentó problemas debido a:

- Alta viscosidad del silicon PlatSil Gel-25
- Flujo inadecuado del material dentro de la cavidad del molde
- Formación de defectos por aire atrapado

Figura 4.1: Concepto de molde cerrado.

Segundo enfoque (molde abierto): La solución adoptada fue un molde abierto (Figura 4.3) que resolvió los problemas anteriores mediante:

- Mayor facilidad para el vertido manual de la silicona
- Incorporación de una pared base con orificios para el paso de cables
- Mejor control visual del proceso de llenado

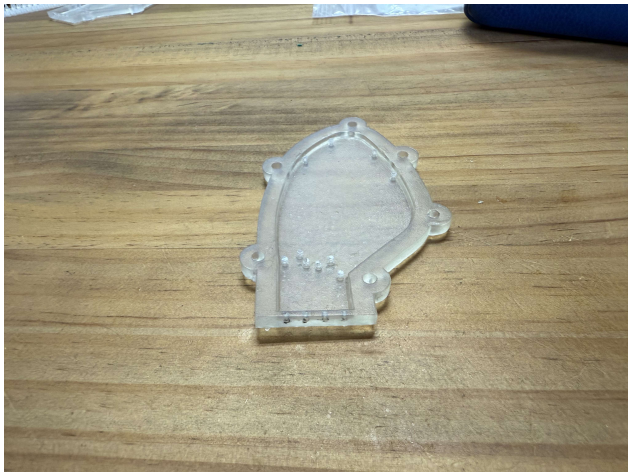


Figura 4.2: Concepto de molde abierto.

Mejora final: Para optimizar la independencia de movimiento entre las mitades del actuador, se diseñó una tapa complementaria (Figura **??**) que:

- Se fija mediante tornillos y tuercas
- Reduce a la mitad (1.5mm) el grosor en la sección central
- Permite control preciso del espesor en la zona de articulación

Figura 4.3: Concepto de molde final.

Esta evolución en el diseño de moldes permitió fabricar actuadores funcionales con movimiento independiente en sus dos secciones, cumpliendo con los requerimientos de múltiples grados de libertad.

4.1.2 Diseño de los pines

Los pines impresos en resina presentan una geometría especializada: una base que encaja en los agujeros del molde, y una sección con diámetro reducido que forma el punto de fractura. La cabeza comienza a 1.5 mm del nivel del centro del actuador, incorporando una muesca perimetral para enrollado del SMA y una zona inferior de 0.5 mm para posicionamiento preciso. Este diseño permite la ruptura controlada durante el desmolde, conservando solo la sección funcional en el actuador final (Figs. 4.4, 4.5).

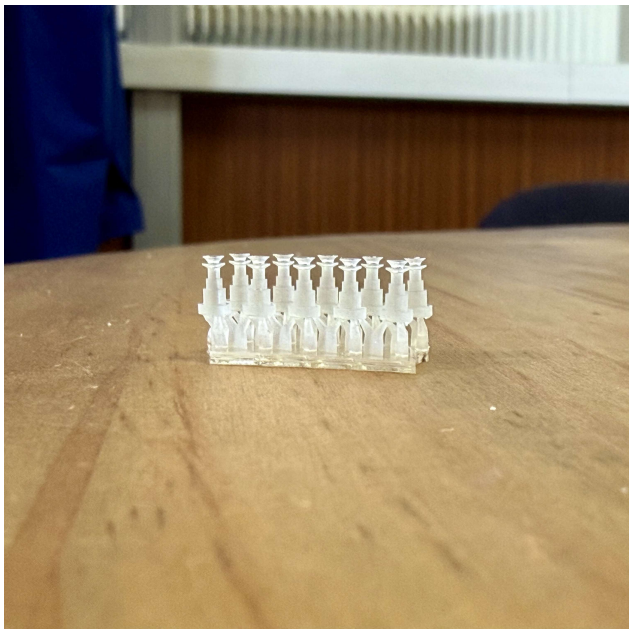


Figura 4.4: Pines manufacturados.

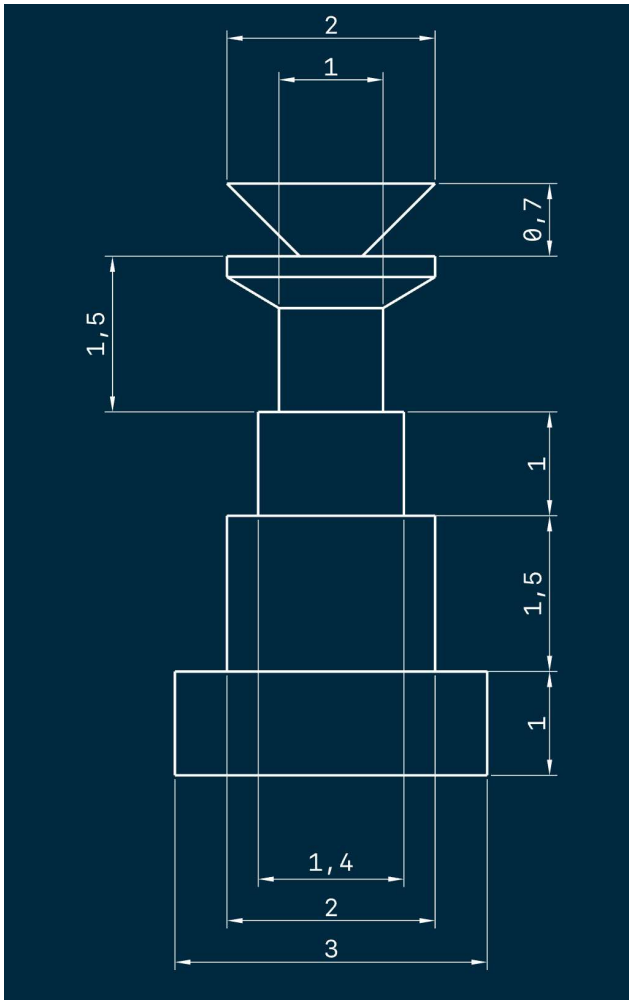


Figura 4.5: Medidas de los pines.

4.2 Electrónica de medición y control

La electrónica de medición y control se basó en un diseño previo de William Coral, utilizando un microcontrolador ESP32-S3 conectado a un convertidor digital-analógico (DAC) MCP4728 de 12 bits con cuatro canales independientes, donde cada canal controla un alambre SMA distinto. El circuito consta de cuatro secciones idénticas e independientes que cumplen dos funciones principales: regular la corriente suministrada a cada SMA en función de la tensión proporcionada por el DAC correspondiente, y realizar mediciones indirectas de la resistencia dinámica del alambre. Sin embargo, estas secciones presentaron un problema crítico en su funcionamiento: independientemente del valor de tensión que el DAC entregara, los circuitos siempre suministraban la máxima potencia disponible a los SMAs, lo que hacía imposible cualquier forma de control preciso sobre los actuadores.

4.2.1 Medición y control

El circuito original (Figura 4.6) opera mediante el siguiente principio: el DAC suministra un voltaje a la base del transistor bipolar NPN BC817, controlando así la corriente que fluye desde su colector al emisor. Esta corriente regula la tensión en el *gate* del MOSFET - a mayor conducción del BC817, menor tensión en el *gate*. Dado que se emplea un MOSFET de canal P (PMOS), la reducción de tensión en el *gate* incrementa la corriente entre *drain* y *source*, permitiendo el control preciso de la corriente a través del SMA, que equivale a la corriente *drain-source* del MOSFET.

Para la medición indirecta de la resistencia dinámica del actuador SMA, el circuito implementado adquiere simultáneamente dos señales de tensión:

- V_1: Corresponde a la caída de tensión en la resistencia shunt R_5
- V_3: Representa la tensión de salida del divisor de tensión formado por R_6 y R_7

Dada la mínima caída de tensión generada en R_5 (del orden de milivoltios), el diseño incorpora un amplificador instrumental INA333 para acondicionar la señal V_1. Este componente, configurado con una resistencia de ganancia $R_G = 4.12\,\text{k}\Omega$, proporciona una amplificación precisa según la relación [31]:

$$G = 1 + \frac{100\,\text{k}\Omega}{R_G} \approx 25.27$$

Esta ganancia de aproximadamente 25 veces permite elevar la señal a un rango medible con mayor precisión.

El cálculo de la resistencia dinámica del actuador se realiza mediante un análisis de circuito que relaciona las tensiones medidas V_1 y V_3 con los valores de las resistencias R_5, R_6 y R_7. La relación matemática resultante permite determinar el valor de la resistencia del actuador a partir de estas mediciones eléctricas:

- $$I_{\text{SMA}} = \frac{V1}{R5} \tag{4.1}$$

- $$V_{\text{SMA}} = (\frac{V3}{R7} * R6 + V3) - V1 \tag{4.2}$$

- $$R_{\text{SMA}} = \frac{V_{\text{SMA}}}{I_{\text{SMA}}} \tag{4.3}$$

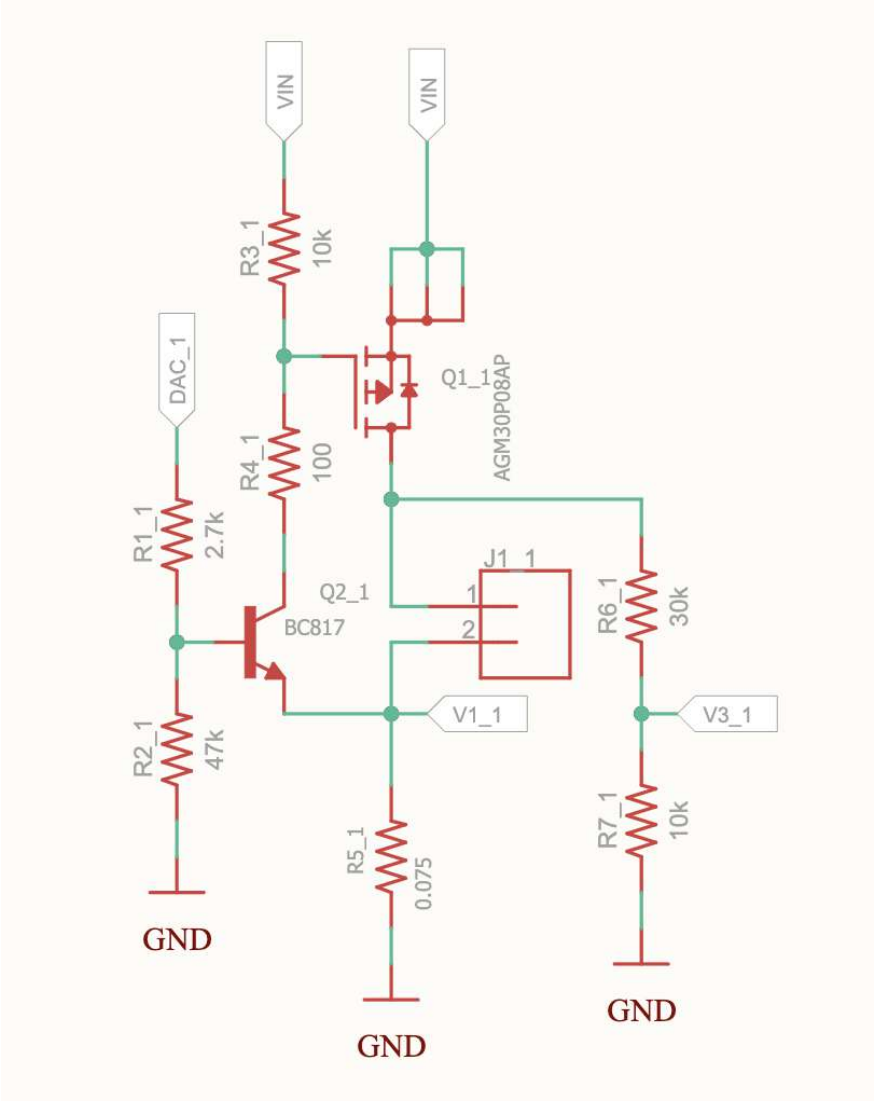


Figura 4.6: Circuito original diseñado por William Coral.

El circuito presenta una limitación fundamental debido al uso de una resistencia de emisor extremadamente baja (75 mΩ) en el BC817, implementada como shunt para medir la corriente del SMA. Esta configuración genera una caída de tensión colector-emisor insuficiente, forzando al transistor bipolar a operar exclusivamente en saturación. Como consecuencia,

el MOSFET de canal P solo recibe dos estados discretos de tensión de gate: máxima (cero corriente base-colector) o mínima (máxima corriente base-colector), resultando en un comportamiento binario donde la corriente drain-source solo puede ser cero o la máxima disponible. Para lograr un control proporcional efectivo de la corriente a través del SMA, es esencial modificar el circuito permitiendo que el BC817 opere en su región activa, donde pequeñas variaciones en la tensión del DAC produzcan cambios graduales en la corriente del MOSFET.

Para verificar que el transistor operaba efectivamente en la región de saturación, se midió la tensión entre el colector y el emisor (V_CE) utilizando un multímetro, obteniéndose un valor de 43 mV. Este resultado se comparó con el valor típico de (V_CE) en saturación especificado en la hoja de datos del transistor BC817 (700 mV) [32].

La tensión medida (43 mV) resultó significativamente menor que el valor de referencia del fabricante, lo que indica que el transistor se encuentra en un estado de saturación más pronunciado que el especificado. Este comportamiento, junto con el observado en el circuito, confirma que el dispositivo está operando en su región de saturación.

4.2.2 Solución al problema de control de la corriente

Para solucionar este problema se pensó en aumentar la resistencia de 75 mΩ a una más grande para que la caída de tensión entre colector y emisor sea mucho mayor y el transistor pueda estar en su zona activa. Para esto se eligieron varios valores de resistencias comerciales mayores o iguales a 1 Ω, siendo 1 Ω, 2.2 Ω, 2.7 Ω, 3.3 Ω y 3.9 Ω. Y con ayuda de una simulación de LTSpice se confirma si están funcionando de manera correcta, Figura 4.7.

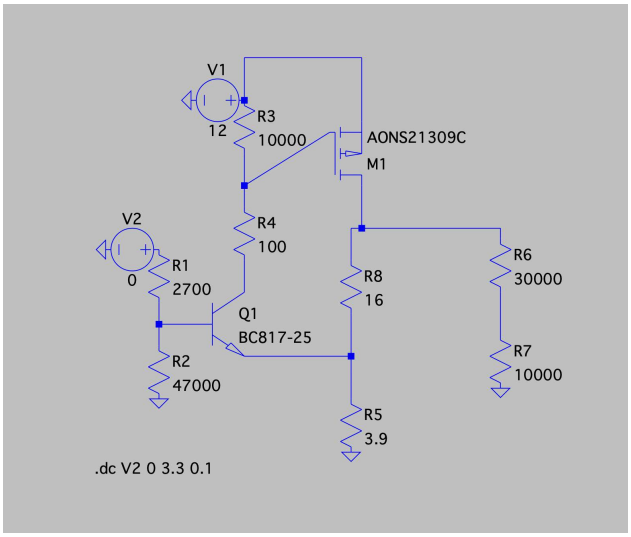


Figura 4.7: Simulación en LTSpice para circuito de medición y control.

En esta simulación, el SMA se modeló como una resistencia de 16 Ω, valor que corresponde a la resistencia nominal medida del alambre utilizado. En los resultados de la simulación (Figs. 4.8, 4.9, 4.10, 4.11, 4.12) se observan dos gráficas:

- La gráfica inferior muestra el barrido del DAC desde 0 V hasta 3.3 V
- La gráfica superior representa la corriente que circula por el SMA durante este barrido

Se comprueba que la corriente aumenta conforme se incrementa el valor del DAC. Sin embargo, la resistencia R_5 afecta dos parámetros clave:

1. El **voltaje de entrada** al que la corriente alcanza su valor máximo
2. El **valor máximo de corriente** entregada

Como ejemplo comparativo:

- Para $R_5 = 1\,\Omega$:
  - Rango de voltaje: 0.6 V a 1.4 V
  - Corriente máxima: $> 700\,\text{mA}$
- Para $R_5 = 3.3\,\Omega$:
  - Rango de voltaje: 0.6 V a 2.8 V
  - Corriente máxima: $\approx 630\,\text{mA}$

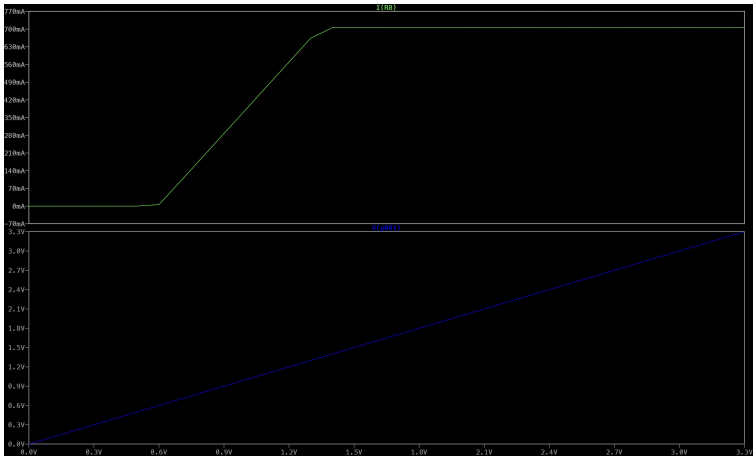


Figura 4.8: Resultado de simulación LTSpice para R5=1 Ω.

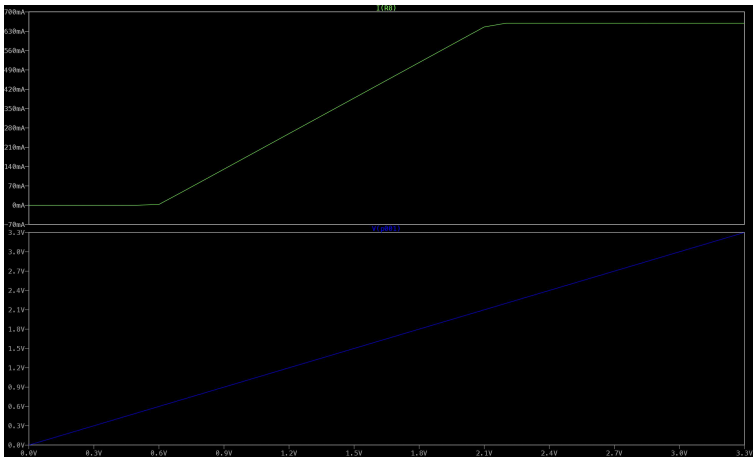


Figura 4.9: Resultado de simulación LTSpice para R5=2.2 Ω.

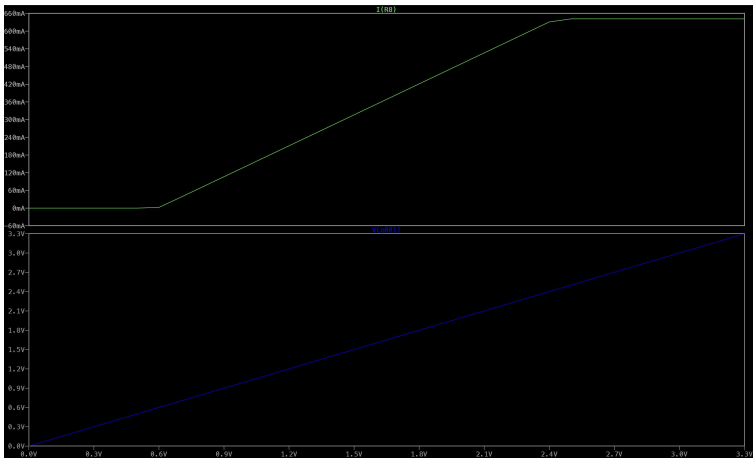


Figura 4.10: Resultado de simulación LTSpice para R5=2.7 Ω.

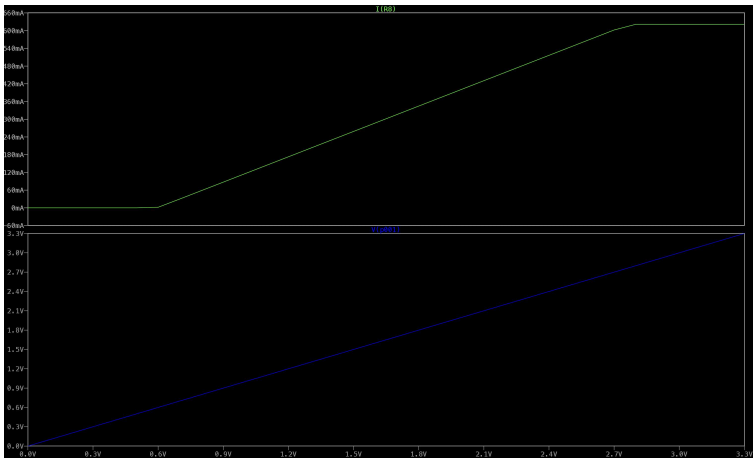


Figura 4.11: Resultado de simulación LTSpice para R5=3.3 Ω.

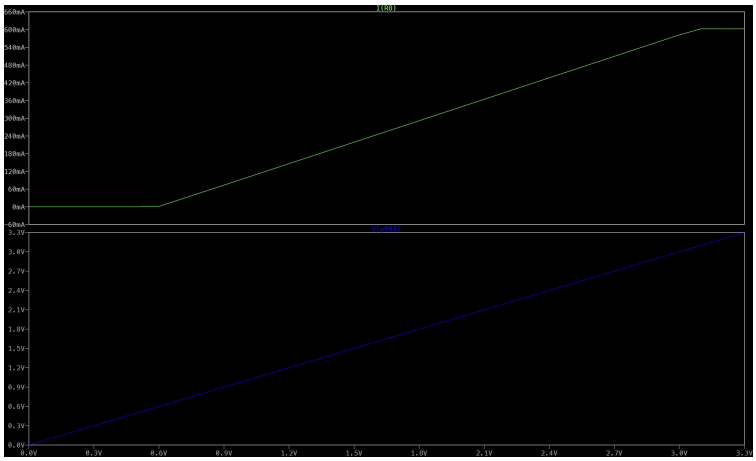


Figura 4.12: Resultado de simulación LTSpice para R5=3.9 Ω.

Para resolver este problema, se optó por utilizar la resistencia de 3.3 Ω como solución óptima. Esta elección se fundamenta en dos aspectos clave: por un lado, proporciona un rango de control suficientemente amplio en el DAC donde la corriente puede variar de manera efectiva; por otro, ofrece valores de corriente máxima dentro de márgenes aceptables para la aplicación. Cabe destacar que, en caso de requerir mayores valores de corriente, el sistema permite incrementar fácilmente la tensión de la fuente de alimentación externa sin necesidad de modificar otros componentes.

La alternativa de 3.9 Ω fue descartada tras un análisis detallado. Si bien esta resistencia presentaba una ventana de control aún más extensa, su implementación planteaba varios inconvenientes prácticos. En primer lugar, no se disponía de este valor particular en el laboratorio al momento del desarrollo. Más importante aún, al operar cerca del límite máximo del DAC (3.3 V), el sistema podría experimentar una respuesta más lenta del control, lo que resultaría especialmente crítico cuando el SMA requiera cambios rápidos de corriente para movimientos bruscos.

La solución adoptada con la resistencia de 3.3 Ω ofrece un compromiso técnico ideal, proporcionando una ventana de control de 2.2 V que garantiza suficiente margen para un control preciso y eficiente, sin comprometer la capacidad de respuesta del sistema en condiciones de operación exigentes.

Sin embargo, este cambio provocó que el valor de V_1 aumentara significativamente, lo que saturaba permanentemente al amplificador INA333, cuyo rango de entrada está limitado a ±0.1 V según su hoja de datos [31]. Esta saturación hacía imposible realizar mediciones.

Dado que las placas PCB ya estaban fabricadas y no era posible eliminar físicamente el componente, se implementó una solución alternativa: se removió la resistencia R_G, lo que teóricamente hace infinita la relación en la ecuación 4.2.1 y reduce la ganancia a 1. Esta modificación efectivamente inhabilita la función de amplificación del INA333, permitiendo así medir directamente el valor real de la tensión V_1 sin saturación.

En la Figura 4.13 se puede ver el circuito antes y después de los cambios en las porciones correspondientes a los canales A y C del DAC.

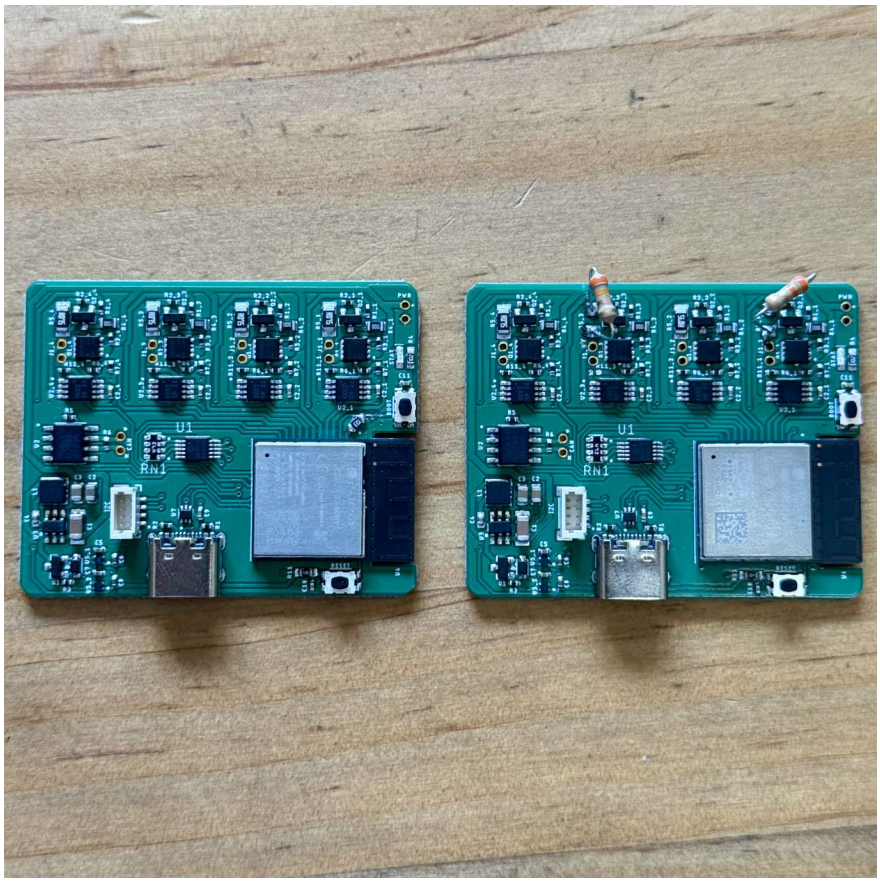


Figura 4.13: Circuito de medición y control. Izq: Original Der: Cambios realizados en los canales A y C.

4.3 Diseño del sistema de control

En el ámbito del diseño de sistemas de control, contar con un modelo matemático preciso de la planta a controlar constituye un aspecto de vital importancia, especialmente cuando se persiguen objetivos de alto rendimiento. La disponibilidad de dicho modelo permite realizar pruebas y validaciones mediante simulaciones computacionales, evitando así la costosa y arriesgada experimentación directa sobre el sistema físico real.

Esta aproximación basada en modelos se vuelve particularmente valiosa cuando se requieren estrategias de control avanzadas para aplicaciones complejas. Un modelo preciso no solo guía la selección del esquema de control más adecuado, sino que además proporciona un marco sistemático para el ajuste óptimo de sus parámetros.

Los modelos existentes para actuadores de aleaciones con memoria de forma (SMA) se han basado predominantemente en enfoques fenomenológicos. Estos modelos buscan representar la compleja naturaleza termo-eléctrica de los materiales SMA, haciendo especial énfasis en capturar sus características de histéresis. Sin embargo, dichos modelos presentan varias limitaciones prácticas. Los estudios tradicionales se han centrado en describir el comportamiento de gran señal de los SMA, el cual exhibe fuertes no linealidades, histéresis pronunciada y baja repetibilidad. Estos modelos suelen incorporar variables internas del material -como la temperatura o la fracción de fase martensítica- que, aunque teóricamente relevantes, resultan poco prácticas para fines de control. La medición directa de estos parámetros resulta técnicamente compleja y, en muchos casos, inviable en aplicaciones reales [14].

4.3.1 Caracterización

Para controlar actuadores de aleación con memoria de forma, es clave desarrollar un modelo dinámico que vincule la entrada de control con la salida medida. Dado que estos actuadores normalmente se activan mediante calentamiento eléctrico resistivo, resulta más práctico usar la potencia aplicada como señal de control en lugar de la temperatura, mientras que la respuesta del sistema suele evaluarse mediante la fuerza generada o el desplazamiento alcanzado.

Para caracterizar el comportamiento del actuador, se analizó la respuesta de la resistencia ante pulsos cortos (de 0.5 a 1.5 segundos) con distintos niveles de corriente o tensión (según los valores del DAC), con unos 4 segundos en apagado para que el actuador tuviera tiempo para enfriarse por completo. El estudio se realizó en dos aleaciones con memoria de forma (SMA) distintas, aplicando pulsos aleatorios a cada una con valores del DAC entre 3300 y 4000 durante dos minutos. Cabe destacar que al emplear un DAC de 12 bits, el rango máximo posible es de 4095 niveles, lo que permite una resolución precisa en la generación de señales. Los resultados de este experimento se pueden ver en las Figuras 4.14 y 4.15.

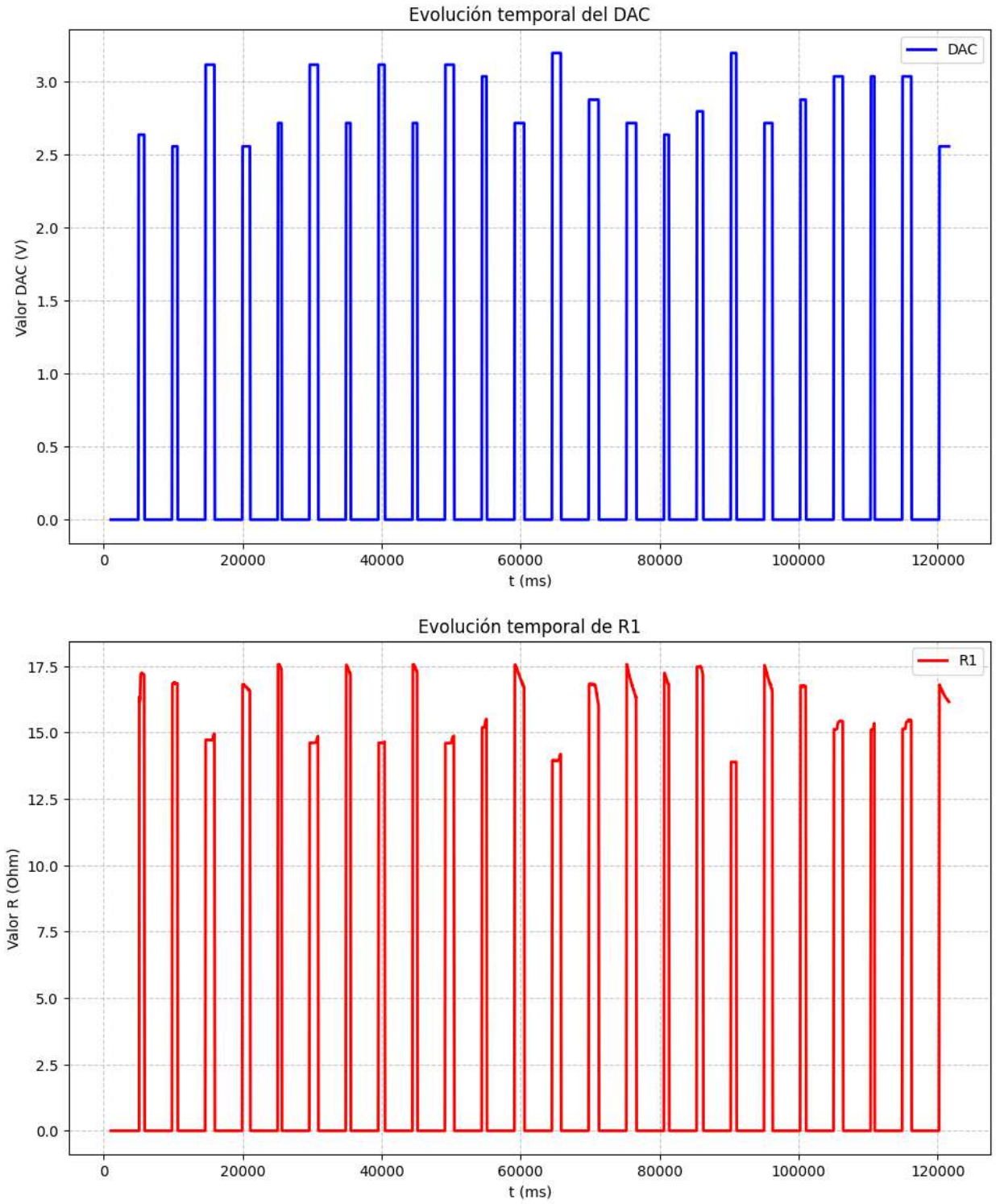


Figura 4.14: Gráficas de entrada (Tensión DAC) y salida (Resistencia SMA) del SMA 1.

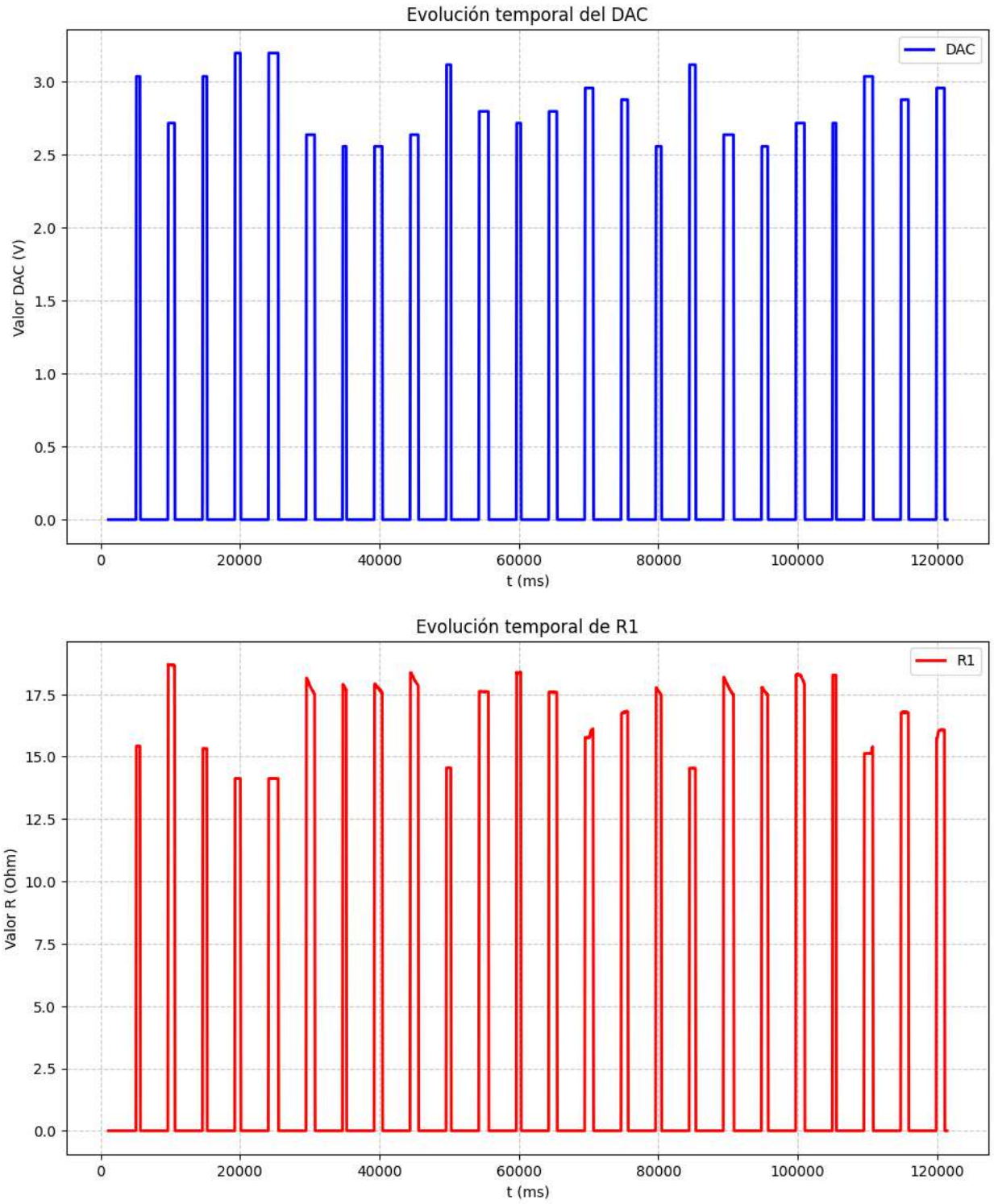


Figura 4.15: Gráficas de entrada (Tensión DAC) y salida (Resistencia SMA) del SMA 2.

Ambos actuadores SMA mostraron respuestas idénticas, presentando variaciones medibles en la resistencia en la mayoría de los casos. Sin embargo, en los valores extremos del rango de operación, tanto para tensiones muy altas como muy bajas, no se observaron cambios significativos, ya que el material alcanzaba estados de saturación térmica o insuficiente activación. También se observa que la señal tiene muy poco ruido.

Se seleccionó el primer conjunto de datos y se procesó mediante la *System Identification Tool* de MATLAB para obtener una función de transferencia continua. Para el análisis, se utilizó aproximadamente el 70% de los datos para estimación y el 30%restante para validación. Se generaron múltiples modelos de diferente complejidad:

- tf1 = Un polo y cero ceros
- tf2 = Un polo y un cero
- tf3 = Dos polos y cero ceros
- tf4 = Dos polos y un cero

Los resultados del ajuste de cada una de estas se presentan en la Figura 4.16

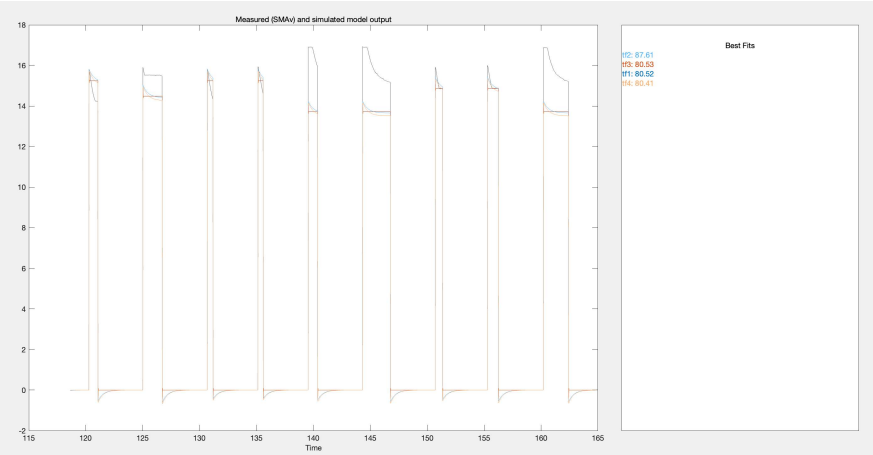


Figura 4.16: Resultados de ajuste de las diferentes funciones de transferencia generadas por el *System Identification Tool* de MATLAB.

Los valores de ajuste obtenidos para las distintas funciones de transferencia fueron aceptables en general, siendo el modelo tf2 -compuesto por un polo y un cero- el que presentó el mejor desempeño con un 87.61% de ajuste. Si bien este porcentaje resulta satisfactorio para la aplicación, un examen más detallado de la Figura 4.17 revela que, aunque el modelo reproduce adecuadamente la tendencia principal de la respuesta, no logra capturar con fidelidad las variaciones más sutiles de la señal. Esta limitación podría representar un inconveniente en aplicaciones que demanden alta precisión, pero dado que en este caso particular no se requiere un control extremadamente exacto de la posición, el modelo tf2 resulta perfectamente viable para su implementación. El balance entre simplicidad del modelo (un polo y un cero) y su capacidad para representar la dinámica fundamental del sistema lo convierten en una opción práctica y suficientemente robusta para el control del actuador, a pesar de no reproducir los detalles más finos del comportamiento. La función de transferencia tf2 se describe con la ecuación 4.4.

$$G(s) = \frac{4.955s + 10.37}{s + 2.183} \tag{4.4}$$

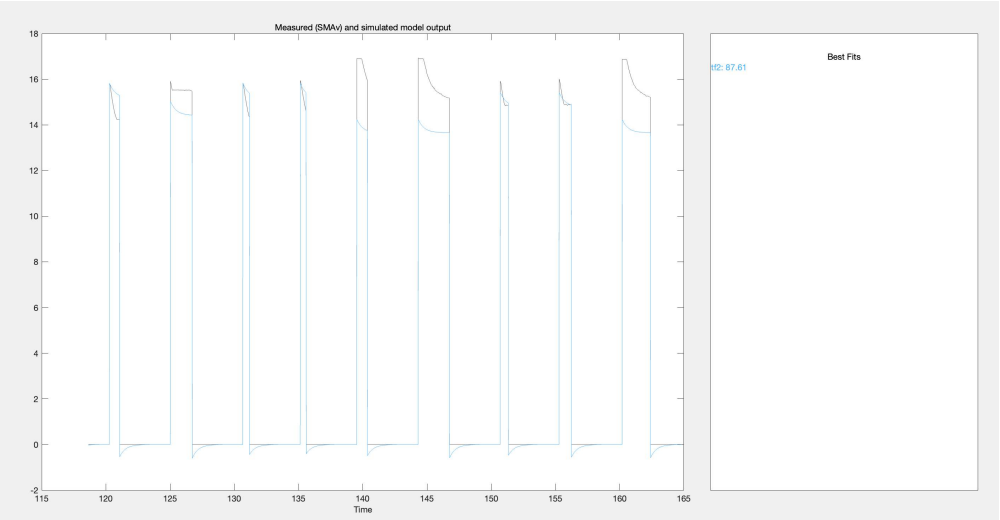


Figura 4.17: Resultados de ajuste de tf2.

Se probaron diferentes métodos para generar un modelo, pero todos presentaban problemas de *overfitting* (sobreajuste). Este fenómeno se ilustra claramente en la Figura 4.18 para el caso del modelo en espacio de estados. En la gráfica superior se observa cómo el modelo parece ajustarse perfectamente a los datos de estimación, capturando incluso los detalles más finos de la señal. Sin embargo, esta aparente precisión resulta engañosa, ya que al validar el modelo con datos independientes (como se muestra en la parte inferior de la figura), su capacidad predictiva es prácticamente nula, demostrando que había memorizado los datos de entrenamiento en lugar de aprender la dinámica subyacente del sistema.

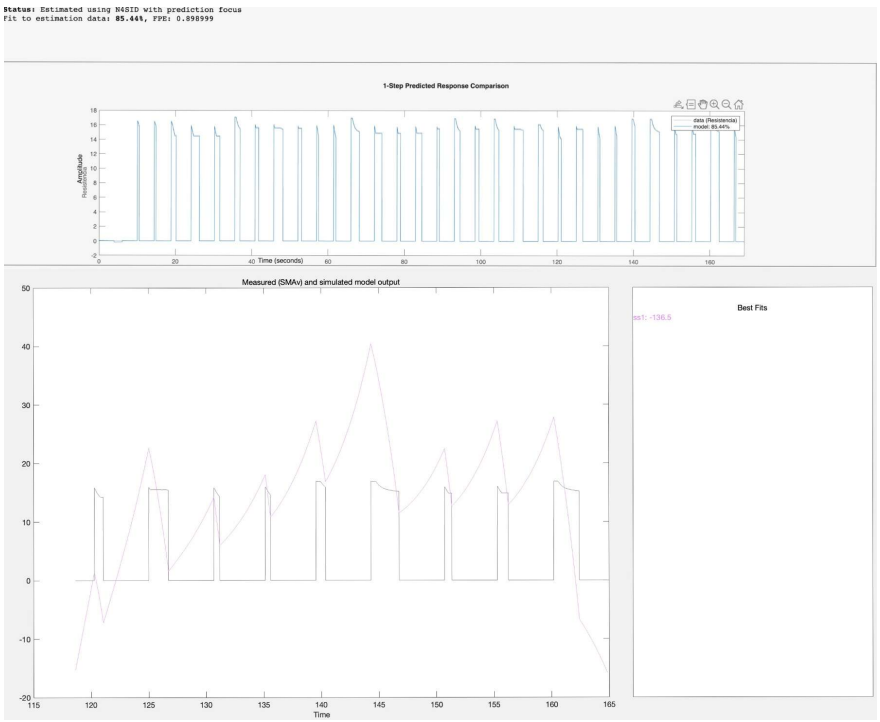


Figura 4.18: Ejemplo de *overfitting*.

4.3.2 Diseño del controlador

Para el control del diseño se implementó un enfoque de afinamiento automático basado en una función de transferencia y parámetros de diseño específicos, como el porcentaje de sobreimpulso (overshoot) y el tiempo de subida hasta alcanzar el 95% del valor de referencia. Se optó por utilizar el tiempo de subida en lugar del tiempo de estabilización (settling time) debido a las particularidades dinámicas y térmicas del sistema. En primer lugar, el movimiento de aleteo requiere una respuesta rápida en su fase inicial, por lo que el tiempo de subida resulta más adecuado para garantizar un arranque ágil del actuador. En segundo lugar, las características térmicas de la aleación con memoria de forma (SMA) introducen un retardo inherente entre el cambio en su resistencia eléctrica y el desplazamiento mecánico real. Para mitigar este efecto, es fundamental que el actuador alcance la temperatura de contracción lo más rápido posible, lo que justifica la selección de un tiempo de subida exigente. De este modo, la fase inicial de la respuesta impulsa el movimiento efectivo, mientras que el tiempo de estabilización debe corresponderse con la etapa final del desplazamiento, asegurando que el actuador complete su recorrido de manera controlada.

Para lograr este objetivo, se implementó el método de Ziegler-Nichols, una de las estrategias más populares para la sintonización de controladores PID. Este método proporciona un procedimiento sistemático para determinar los parámetros iniciales del controlador (ganancia proporcional Kp, tiempo integral Ti y tiempo derivativo Td) que garantizan una respuesta satisfactoria del sistema.

La técnica de Ziegler-Nichols ofrece dos enfoques principales para la sintonización de controladores: el método de la respuesta al escalón y el método de la frecuencia [33]. En este trabajo se ha implementado exclusivamente el método de la respuesta al escalón, el cual será el único detallado en esta explicación.

Método de Ziegler-Nichols de la Respuesta al Escalón [33]:

Paso 1: Inicialmente se debe obtener la respuesta al escalón de la planta en lazo abierto. Es importante que esta respuesta tenga una forma de "S", caracterizada por dos parámetros fundamentales: el tiempo de retraso L y la constante de tiempo T.

Paso 2: Se identifica el punto en la curva de respuesta al escalón donde la pendiente alcanza su valor máximo. En este punto se traza una línea tangente que se extiende hasta el eje del tiempo. La intersección de esta tangente con el eje temporal corresponde al parámetro L, Figura 4.19. El parámetro T (constante de tiempo) se determina mediante la intersección entre esta línea tangente y la línea horizontal $y(t) = K$, donde K representa la ganancia en estado estable de la planta.

Paso 3: una vez obtenidos los valores de L y T, junto con la ganancia K, se calcula la relación $a = KL/T$. Posteriormente, los parámetros Kp, Ti y Td del controlador PID se pueden determinar directamente utilizando las fórmulas de sintonización de Ziegler-Nichols para el método de respuesta al escalón, como se detalla en la Tabla 4.1, tomada de [33].

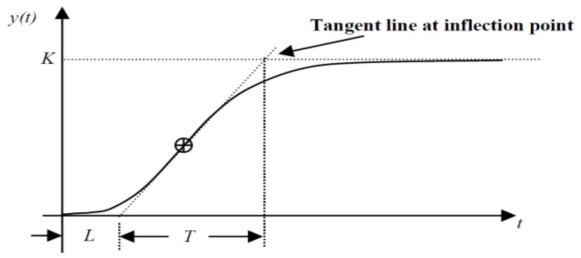


Figure 2. Open Loop System Step Response

Figura 4.19: Respuesta al escalón del sistema en lazo bierto, como se ilustra en [25, Fig. 2].

Tabla 4.1: Formulas de sintonización Ziegler-Nichols para el método de escalón.

Controlador	**Kp**	**Ti**	**Td**
P	1/a	-	-
PI	0.9/a	3L	-
PID	1.27/a	2L	L/2

Los valores generados mediante este método deben considerarse como una aproximación inicial para las ganancias del controlador. En la práctica, puede ser necesario realizar ajustes manuales adicionales una vez implementado el controlador en el sistema físico real, con el fin de optimizar su desempeño bajo condiciones operativas específicas.

Para implementar este método, se desarrolló un script completo en MATLAB que automatiza el proceso de sintonización. El código, junto con su documentación detallada, se encuentra disponible en el Anexo B.

El programa procesó la función de transferencia del sistema con los siguientes parámetros de diseño:

- Tiempo de subida (al 95% del valor de referencia): 0.2 s
- Sobreimpulso máximo permitido: 20%

Estos valores se seleccionaron para lograr una respuesta rápida y moderadamente agresiva del controlador. El proceso de sintonización automática generó los siguientes parámetros para el controlador PID:

- $Kp = 2.000$
- $Ki = 94.806$
- $Kd = 0.034$

El programa convierte las constantes Ti and Td a Ki y Kd usando las ecuaciones detalladas en [34].

- $Ki = Kp/Ti$
- $Kd = Kp * Td$

Para validar el desempeño del controlador, se implementó una simulación en SIMULINK utilizando una señal de entrada en escalón con amplitud de 15, Figura 4.20. En esta simulación se midieron los parámetros de diseño previamente establecidos (tiempo de subida y sobreimpulso) para verificar el cumplimiento de las especificaciones requeridas.

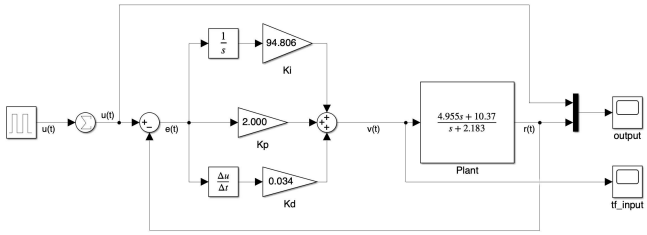


Figura 4.20: Simulación del controlador PID en SIMULINK.

En la salida del sistema, el sobreimpulso mostrado en la Figura 4.21 alcanza un valor máximo de 15.04, lo que corresponde a aproximadamente 0.26% respecto al valor final, cumpliendo con creces el criterio de diseño establecido.

Respecto al tiempo de subida al 95% del valor de referencia (14.25), la Figura 4.22 muestra dos elementos clave: la línea amarilla que marca el flanco positivo del pulso de entrada en 1.99 segundos, y la línea azul que representa la respuesta de la planta controlada. Esta última alcanza el nivel de 14.25 en 2.02 segundos, resultando en un tiempo de subida efectivo de 0.03 segundos, considerablemente menor al tiempo especificado en el diseño.

Adicionalmente, se observa que el error en estado estacionario es prácticamente nulo en esta configuración del controlador.

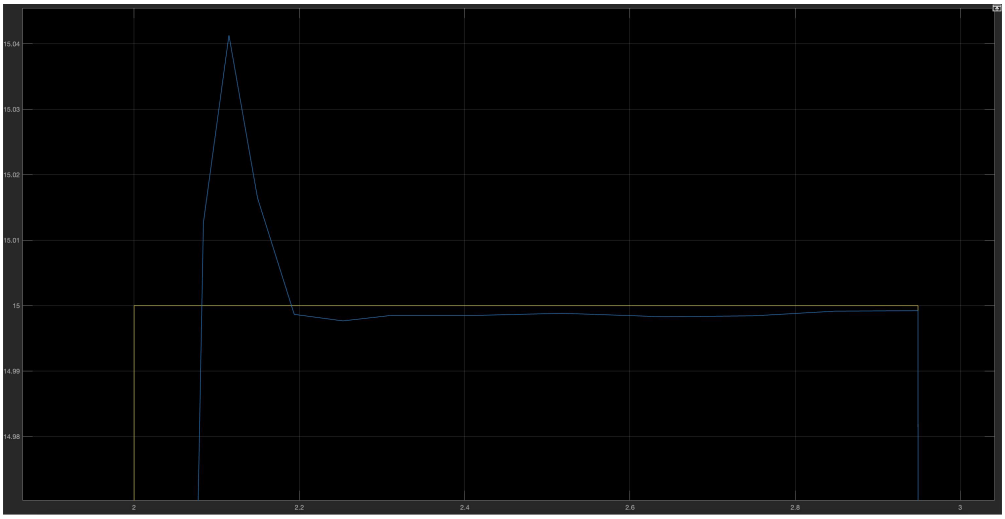


Figura 4.21: Sobreimpuslo resultante de la simulación del controlador.

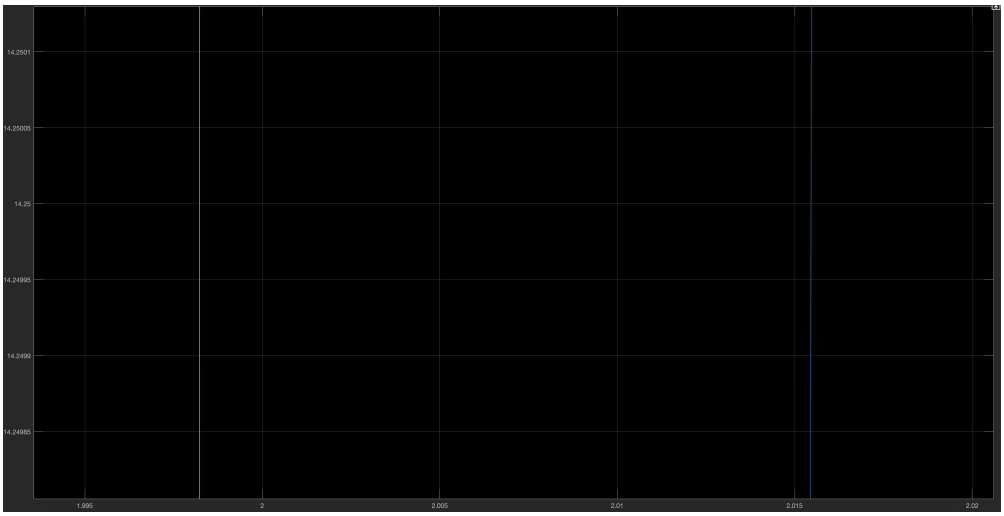


Figura 4.22: Tiempo de subida al 95% del valor de referencia (14.25).

Los resultados de la simulación muestran que el controlador cumple satisfactoriamente con las especificaciones de diseño establecidas.

4.4 Implementación práctica

4.4.1 Relación entre ángulo y resistencia

Para implementar el controlador, primero es necesario establecer un método de conversión entre los ángulos deseados y los valores de resistencia de referencia. Esta conversión permite alimentar al controlador con los valores de resistencia medidos y compararlos con los valores de referencia correspondientes. La relación entre ángulos y resistencias, representada por la ecuación 4.5, se obtiene mediante un proceso que se explicará en detalle en el siguiente capítulo. Esta relación es fundamental para el funcionamiento del sistema de control, ya que vincula directamente las posiciones angulares con las señales eléctricas del actuador.

El desarrollo de esta relación de conversión constituye un paso esencial previo a la implementación práctica del controlador, como se verá más adelante.

$$R = -0.2840 * \theta + 26.4085 \tag{4.5}$$

4.4.2 Implementación en Arduino IDE

El sistema de control se desarrolló en el entorno de desarrollo Arduino e incorpora los siguientes componentes y funcionalidades:

- Un generador de señales de prueba integrado, que permite validar el comportamiento del controlador.

- Mecanismos de seguridad específicamente diseñados para proteger el actuador contra condiciones de operación extremas.
- Sistema de monitoreo que proporciona métricas de rendimiento en tiempo real durante la operación del controlador.
- Circuito de calentamiento auxiliar que se activa cuando el controlador permanece inactivo, manteniendo el actuador en su rango operativo óptimo.

El sistema implementa cuatro canales de control independientes, permitiendo el manejo simultáneo de hasta cuatro actuadores SMA. Cada canal corresponde a una salida específica del conversor digital-analógico MCP4728, según la configuración presentada en la Tabla 4.2.

Tabla 4.2: Pines ESP32-S3 para medición de V1 y V3 por canal de DAC.

Canal	Medición	GPIO
A	V1	2
	V3	1
B	V1	4
	V3	3
C	V1	6
	V3	5
D	V1	17
	V3	7

Para la parte del controlador se utiliza la librería *Arduino PID Library* por Brett Beauregard, el funcionamiento de esta librería se puede leer detalladamente en [35] y descargar en `https://github.com/br3ttb/Arduino-PID-Library?tab=readme-ov-file`. La librería Arduino PID Library fue modificada para adaptarse al hardware utilizado, específicamente en el archivo PID_v1.cpp (línea 28), donde se cambió el límite superior de salida de 255 a 4000. Este ajuste fue necesario debido a que el ESP32-S3 y el DAC MCP4728 trabajan con una resolución de 12 bits (rango 0–4095), a diferencia del Arduino Uno (diseñado para 8 bits, con un rango de 0–255).

La modificación permite aprovechar la mayor resolución del sistema, mejorando la precisión del control, mientras se mantiene un margen de seguridad por debajo del valor máximo teórico (4095). Esta adaptación preserva todas las funcionalidades originales de la librería, pero ahora optimizadas para el ESP32-S3 y el DAC de 12 bits.

El sistema incluye un generador de señales programable capaz de producir diferentes formas de onda: sinusoidal, triangular, cuadrada, y diente de sierra, además de un modo manual que permite introducir valores de referencia directamente desde el Monitor Serial del IDE de Arduino.

Cada señal generada es completamente configurable mediante parámetros específicos como amplitud, frecuencia, desfase relativo entre canales, ciclo de trabajo (para las formas

de onda que lo requieran) y del desplazamiento en y si es necesario. Estas señales están matemáticamente definidas por:

- $Sinusoidal = Asin(2\pi ft + \phi) + C)$, dónde C es el desplazamiento.
- $Cuadrada = \begin{cases} +A & si \quad mod(ft + \phi/360, 1) < D \\ \\ 0 \end{cases}$, donde D es el ciclo de trabajo.
- $Triangular = A * \begin{cases} 4\phi & si \quad 0 \leq \phi < 0.25 \\ \\ 2 - 4\phi & si \quad 0.25 < \phi \leq 0.75 \\ \\ -4 + 4\phi & si \quad 0.75 \leq \phi < 1 \end{cases}$
- $Sierra = A * mod(ft, 1) + C$

Las señales generadas están restringidas a valores entre 0 y la amplitud dada por el usuario, ya que representan ángulos de referencia para el controlador PID. Esta limitación responde a la capacidad física del actuador, que no puede realizar movimientos angulares menores a cero. Todas las señales se definen matemáticamente para garantizar que cumplan con este rango operativo.

El sistema permite modificar tanto la configuración inicial como los parámetros en tiempo real mediante comandos seriales en el IDE de Arduino, con la capacidad de ajustar cada canal del DAC de forma independiente. Las instrucciones completas para el uso de estos comandos se encuentran documentadas en la sección de Anexos, incluyendo los formatos específicos y los parámetros configurables para cada tipo de señal.

El sistema incorpora dos protecciones críticas para el actuador SMA. La primera detecta circuitos abiertos mediante un temporizador configurable (5 segundos por defecto): si el PID está activando el DAC pero no se reciben mediciones, el sistema asume una desconexión del actuador o ruptura del SMA y lo detiene. La segunda protección previene sobrecalentamientos al monitorear continuamente la resistencia; cuando detecta valores inferiores a 14 ohmios, reduce inmediatamente la salida del DAC hasta que la resistencia supere los 15 ohmios, evitando así la operación prolongada en límites peligrosos.

Estos mecanismos operan independientemente del control principal, proporcionando una capa adicional de seguridad. El umbral de resistencia y el tiempo de detección de circuito abierto son parámetros ajustables, permitiendo adaptar las protecciones a diferentes configuraciones de SMA sin modificar el código base. La implementación garantiza tanto la seguridad del actuador como la del circuito electrónico asociado.

Aparte de estos mecanismos, el programa también le aplica una corriente pequeña, alrededor de 0.45 A, para mantener el actuador caliente y listo para contraerse con cualquier aumento en esta.

El sistema implementa un módulo de registro de datos que captura y muestra en formato CSV los siguientes parámetros clave de operación: el tiempo transcurrido hasta alcanzar el 95% del valor de referencia (tiempo de subida), el error en estado estable y el valor máximo registrado en la entrada. Estos datos se complementan con los valores actuales del DAC, el setpoint y la resistencia medida, proporcionando una visión completa del comportamiento del sistema.

El formato CSV estructurado permite un análisis posterior sencillo, con cada variable claramente identificada y separada por comas. Esta información se imprime periódicamente a través del puerto serial, facilitando tanto la monitorización en tiempo real como el almacenamiento para evaluación posterior.

El diagrama de flujo mostrado en la Figura 4.23 describe el proceso principal del sistema. En primer lugar, se realizan las mediciones de los voltajes V1 y V3 para todos los canales activos, utilizando estos valores para calcular la resistencia correspondiente de cada actuador mediante las ecuaciones 4.1, 4.2 y 4.3, donde R5 corresponde al valor conocido de 3.3 Ω.

Cuando hay una señal de prueba activa, el sistema calcula iterativamente (cada 5 ms) el nuevo valor de referencia, lo envía al controlador PID como un valor de resistencia y actualiza la salida del DAC. Luego registra e imprime las métricas de desempeño asociadas a cada actualización. En caso contrario, el sistema permanece en un estado de medición continua de resistencias, a la espera de que se active alguna señal de prueba mediante los comandos seriales correspondientes. Este diseño permite tanto la operación automática como el monitoreo constante del estado del sistema.

El código completo y con su documentación se pueden observar en la sección de anexos, Anexo C, junto al manual sobre escribir los comandos para el generador de señales, D y el manual sobre diseñar un controlador PID para actuadores a base de SMA (sin anexos ya que estos son los mismos que los anteriores), Anexo E.

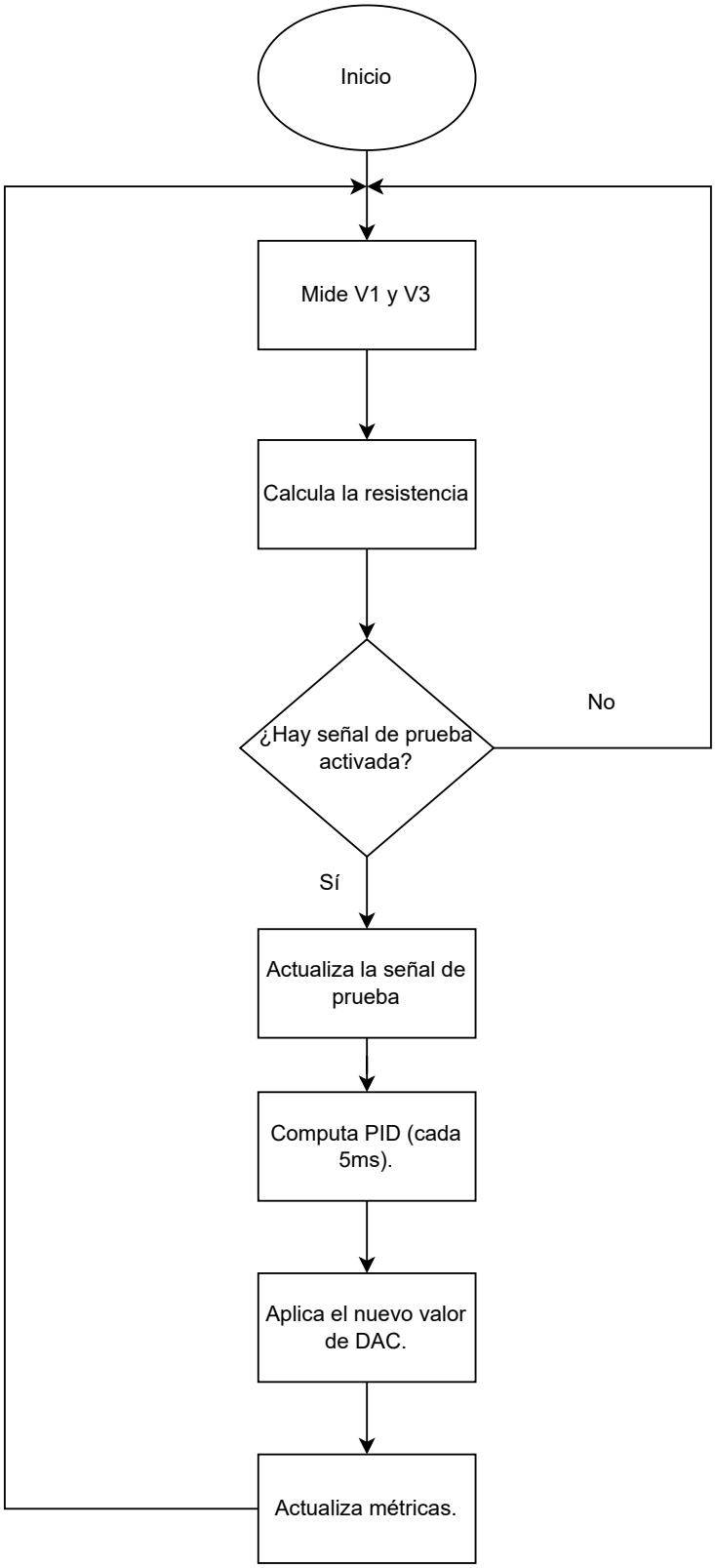


Figura 4.23: Flujo general del sistema de control.

Capítulo 5

Resultados y Análisis

Para garantizar la fiabilidad del sistema de control propuesto, es fundamental considerar aspectos técnicos y estadísticos. Un elemento clave es determinar el número adecuado de pruebas de validación.

En sistemas sin historial de pruebas previas, los principios de control estadístico recomiendan un mínimo de 10 pruebas para obtener una evaluación inicial válida y estimar la variabilidad del sistema [36]. Esta cantidad representa un equilibrio entre confiabilidad estadística y eficiencia operativa, particularmente cuando no existen datos históricos que sugieran baja variabilidad [37].

Al principio de este proyecto se había ideado evaluar la capacidad del sistema de reducir el tiempo de actuación en un 40%. Para validar estadísticamente este requerimiento, se diseñó una prueba de hipótesis unilateral izquierda, donde:

- H_0: El sistema de control no reduce el tiempo de actuación en un 40%.

- H_1: El sistema de control sí reduce el tiempo de actuación en un 40%.

Esta prueba de hipótesis nos funciona también para determinar el tamaño muestral de la segunda prueba. Para esto fue necesario caracterizar el comportamiento del sistema sin control, se llevó a cabo un protocolo experimental estructurado en tres etapas principales. Inicialmente, se realizó una caracterización del sistema en su estado base, ejecutando tres series de diez mediciones cada una. Para estas pruebas se aplicaron valores DAC específicos (3000, 3200 y 3800), previamente identificados en la primera prueba como generadores de desplazamientos angulares representativos de aproximadamente 15°, 25° y 35° respectivamente, lo que permite cubrir la mayor parte del rango operativo del actuador. Cada medición registró el tiempo requerido para alcanzar estos ángulos de referencia, obteniéndose los resultados que se presentan en la Tabla 5.1.

Tabla 5.1: Tiempo de movimiento del actuador hasta un ángulo cercano al de referencia.

N	$t_{\mathbf{15}}(\pm 0.1s)$	$t_{\mathbf{25}}(\pm 0.1s)$	$t_{\mathbf{35}}(\pm 0.1s)$
1	1.8	1.7	2.0
2	2.0	2.1	2.0
3	2.0	2.0	2.0
4	1.9	2.2	1.9
5	1.9	1.9	2.1
6	1.8	2.0	1.9
7	2.0	2.0	2.0
8	1.9	2.0	2.0
9	2.0	1.9	2.1
10	1.9	2.0	2.0
Promedio	1.9	2.0	2.0
Intervalo de confianza	[1.8,2.0]	[1.9,2.1]	[1.6,2.4]

La tabla muestra que el tiempo de aleteo es de aproximadamente 2 segundos. Si se busca reducirlo en un 40%, el valor objetivo sería de 1.2 segundos. Con esto, la hipótesis quedaría formulada de la siguiente manera:

- H_0: $\mu \geq 1.2$
- H_1: $\mu < 1.2$

Para rechazar H_0 con un 95% de confianza, se requirió un valor de $p < 0.05$. El tamaño de muestra necesario se calculó mediante la Ecuación 5.1, citada de [38]:

$$n = \frac{(z_\alpha + z_\beta)^2 \cdot \sigma^2}{\delta^2} \tag{5.1}$$

En este cálculo, se utilizó un nivel de significancia $\alpha = 0.05$ y una potencia del 80%, valores estándar en estudios de ingeniería [39]. Los valores críticos $z_\alpha = -1.645$ y $z_\beta = -0.842$ se obtuvieron de la Tabla A.3 de [38]. Con la mayor desviación estándar observada en los datos ($\sigma = 0.13$ s), correspondiente al ángulo de 25 grados) y una diferencia mínima a detectar ($\delta = 0.8s$), equivalente al 40% del tiempo base de 2 s), se determinó un tamaño de muestra teórico de $n \approx 9.69$, que se redondeó a 10 mediciones por ángulo de referencia para garantizar robustez en el análisis.

Cabe destacar que la incertidumbre de todas las mediciones reportadas en esta sección corresponde únicamente a su error estadístico, ya que cada conjunto incluye más de cinco repeticiones realizadas bajo las mismas condiciones y con instrumentos que no presentan una incertidumbre instrumental significativa [40]. En el caso particular de la resistencia, una magnitud indirecta por naturaleza, al derivar de mediciones digitales puntuales sin incertidumbre asociada, su incertidumbre se ha estimado mediante la desviación estándar de

las mediciones. Este enfoque refleja adecuadamente la variabilidad experimental mientras mantiene consistencia con la naturaleza de los datos obtenidos.

También que todos los intervalos de confianza calculados en este documento se hicieron para una confianza del 95%.

5.0.1 Pruebas desarrollados

La primera prueba resulta crucial para el desarrollo del sistema de control, ya que permite establecer la relación matemática precisa entre el ángulo de deflexión de la aleta y su resistencia eléctrica. Esta correlación, evaluada mediante el coeficiente de determinación R^2 (identificado como Métrica 8 en nuestro estudio), sirve como base para el diseño del algoritmo de control.

Para caracterizar este comportamiento, implementamos un protocolo experimental riguroso que consistió en aplicar cinco niveles de señal DAC distintos: 3000, 3200, 3400, 3600 y 3800. En cada uno de estos niveles, registramos mediciones de resistencia hasta obtener una secuencia estable de 10 valores consecutivos que presentaran una variación menor al 10% y se guarda el valor mínimo de resistencia, garantizando así condiciones de medición confiables y repetibles.

Paralelamente a las mediciones eléctricas, capturamos videos desde una vista superior de la aleta, realizando una grabación independiente para cada valor DAC aplicado. Para facilitar el análisis de movimiento, la aleta fue equipada con esferas plásticas de colores que permiten un tracking preciso mediante visión por computadora. De estos videos guardamos un valor máximo de ángulo.

Todo este proceso de repite diez veces.

La segunda prueba está relacionado con el resto de métricas. Para ello, implementamos una prueba dinámica donde aplicamos una señal cuadrada que alternaba cíclicamente entre 0° y 35° de deflexión angular.

Durante la duración de cada prueba, se realizan grabaciones de video desde la vista superior de la aleta, manteniendo la misma configuración de adquisición visual utilizada en experimentos anteriores. Para obtener una caracterización completa del rango de movimiento, replicamos este mismo procedimiento con amplitudes de 25° y 15°, cubriendo así tanto los extremos operativos como un punto intermedio de la capacidad total de la aleta.

Se grabaron los valores de resistencia medidos diez veces (calculados con la ecuación 5.1, el tiempo de subida al 95% del valor de referencia, de latencia de comunicación y de ángulo para su análisis y el tiempo de aleteo del actuador.

Un aspecto fundamental de esta y las demás pruebas fue el uso de un actuador completamente nuevo, sin historial de uso previo, manteniendo además la configuración original tanto del control como de la implementación física. Esta decisión metodológica nos permitió evaluar el comportamiento base del sistema sin influencias de desgaste o modificaciones previas.

5.0.2 Resultados de las pruebas desarrolladas

Los valores de resistencia y ángulo de movimiento se pueden ver resumidos en las Tablas 5.2 y 5.3.

Primera prueba

Tabla 5.2: Medidas de resistencia para la primera prueba

Medidas de Resistencia (Ω)					
N	$R_{\mathbf{3000}}(\pm 0.32)$	$R_{\mathbf{3200}}(\pm 0.06)$	$R_{\mathbf{3400}}(\pm 0.23)$	$R_{\mathbf{3600}}(\pm 0.33)$	$R_{\mathbf{3800}}(\pm 0.54)$
1	22.20	20.59	18.75	17.25	17.15
2	22.06	20.57	18.28	17.86	16.37
3	21.95	20.57	18.29	16.85	15.91
4	21.89	20.58	18.28	16.85	15.61
5	21.72	20.59	18.28	16.84	15.55
6	21.64	20.58	18.30	16.98	15.55
7	21.53	20.58	18.54	16.85	15.54
8	21.45	20.59	18.74	16.85	15.53
8	21.33	20.76	18.74	16.86	15.54
10	21.24	20.57	18.75	16.85	15.53
Promedio	21.70	20.59	18.49	17.00	15.83
Inter. confianza	[21.50,21.89]	[20.55,20.63]	[18.35,18.63]	[16.79,17.20]	[15.49,16.16]

Tabla 5.3: Medidas de ángulo para la primera prueba

Medidas de Ángulo (grados)					
N	$\theta_{\mathbf{3000}}(\pm 1.11)$	$\theta_{\mathbf{3200}}(\pm 1.18)$	$\theta_{\mathbf{3400}}(\pm 0.84)$	$\theta_{\mathbf{3600}}(\pm 0.51)$	$\theta_{\mathbf{3800}}(\pm 0.25)$
1	14.36	20.77	31.50	31.76	34.72
2	13.74	20.88	31.55	31.79	34.95
3	13.59	22.39	31.18	32.13	35.36
4	13.84	21.55	30.29	32.98	35.23
5	15.97	21.25	30.89	32.95	35.27
6	14.45	22.08	30.78	32.84	35.28
7	14.86	24.25	30.36	32.96	35.21
8	15.65	23.11	29.05	32.45	35.12
8	16.43	22.64	29.48	32.96	34.88
10	16.42	23.70	29.87	33.00	34.68
Promedio	14.93	22.26	30.50	32.58	35.07
Inter. confianza	[14.24,15.62]	[21.53,22.99]	[29.98,31.02]	[32.26,32.90]	[34.91,35.22]

Los datos angulares se obtuvieron utilizando el software *Tracker* de Open Source Physics [41], específicamente mediante la herramienta de *protractor*. El procedimiento consistió en

posicionar este transportador virtual sobre una línea negra de referencia que marca el punto inicial del actuador, asegurando que las puntas del mismo estuvieran alineadas con esta línea antes de iniciar la captura de video. Posteriormente, se estableció un segundo punto de referencia en la intersección entre esta línea base y el doblez característico de la aleta, tal como se ilustra en la Figura 5.1.

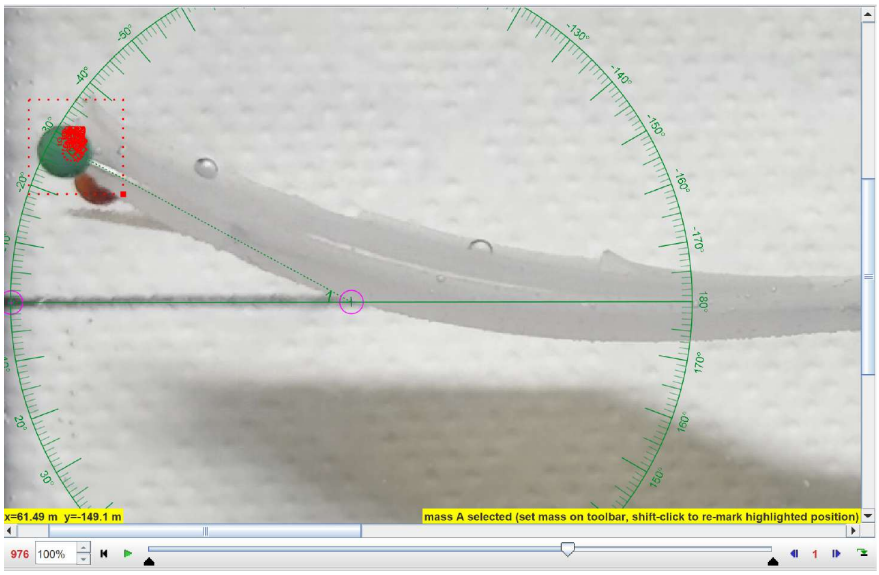


Figura 5.1: Medición de ángulos de movimiento en *Tracker*.

Una vez obtenidas las mediciones angulares, se calcularon los valores promedio que fueron empleados como puntos de datos. En la representación gráfica subsiguiente, estos valores promedio de ángulo se ubicaron en el eje x, mientras que los valores de resistencia correspondientes se situaron en el eje y. Esta disposición permitió visualizar y analizar la relación funcional existente entre ambas variables, lo que resulta fundamental para la caracterización del sistema.

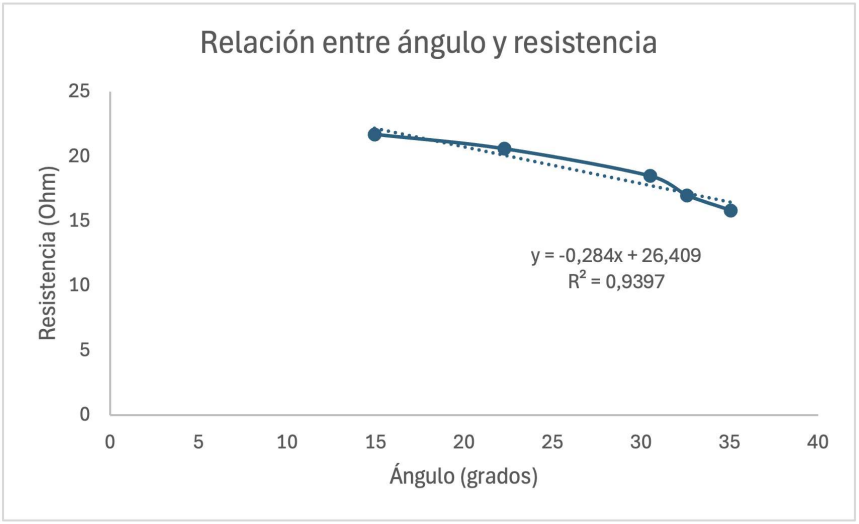


Figura 5.2: Gráfica de los valores de resistencia y ángulo medidos para la primera prueba.

En la Figura 5.2 se puede ver que la relación que existe entre estos datos es casi lineal y que se puede aproximar una ecuación con $y = mx + b$. Para esto se usaron las ecuaciones 5.2 y 5.3, donde n es la cantidad de puntos usados.

$$m = \frac{n\sum(x_i y_i) - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2} \tag{5.2}$$

$$b = \frac{\sum y_i - m\sum x_i}{n} \tag{5.3}$$

Estas fórmulas nos dieron como resultado la ecuación 4.5 ya expuesta anteriormente. Para evaluarla se tuvo que calcular el valor de R cuadrado, este se calcula mediante la ecuación 5.4 esta mide qué tan bien la línea de regresión explica los datos comparando dos cosas: en el numerador, las diferencias entre los valores reales (medidos) y los predichos por el modelo (los errores), y en el denominador, las diferencias entre los valores reales y su media. Si los errores son pequeños frente a la variabilidad total, el R2 se acerca a 1, indicando que el modelo es casi perfecto; en este caso, con un R2 de 0.94, la recta explica el 94% de la variación, demostrando un ajuste excelente.

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \tag{5.4}$$

Segunda prueba

Las mediciones obtenidas en estas pruebas se encuentran resumidas en las tablas 5.4, 5.5, 5.6 y 5.7. Para determinar el tiempo de subida, se graficó la resistencia medida frente al valor objetivo (Figura 5.3), calculándose como la diferencia entre el instante en que la respuesta supera por primera vez dicho valor y el inicio del pulso de referencia. Por su parte, la latencia de comunicación se obtuvo restando el tiempo registrado antes de cada medición al tiempo de procesamiento en la programación de Arduino, utilizando para ello 10 valores consecutivos durante un pulso. El experimento consistió en operar el sistema hasta observar 10 ciclos completos de aleteo.

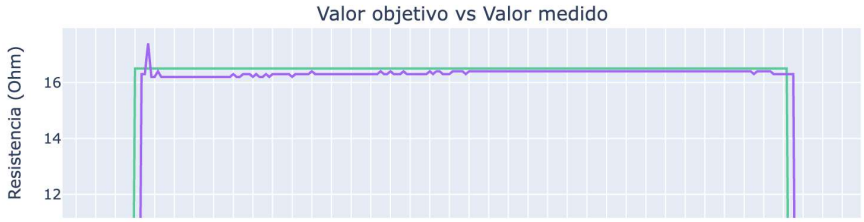


Figura 5.3: Ampliación de resistencia medida contra valor objetivo.

Tabla 5.4: Datos de validación con ángulo de referencia de 35 grados.

N	$t_{\mathbf{aleteo}}$ $(\pm 0.1s)$	$t_{\mathbf{95}}$ $(\pm 2ms)$	$Angulo_{\mathbf{max}}(\pm 0.8deg)$
1	1.8	25	29.3
2	2.0	25	30.1
3	2.0	30	30.9
4	2.1	25	31.0
5	1.8	25	31.3
6	2.0	25	31.4
7	2.0	25	31.5
8	1.9	25	31.6
9	2.0	30	31.6
10	2.0	25	31.9
Promedio	2.0	26	31.1
Intervalo de confianza	[1.9,2.0]	[25,27]	[31.1,32.1]

Tabla 5.5: Datos de validación con ángulo de referencia de 25 grados.

N	$t_{\mathbf{aleteo}}$ $(\pm 0.1s)$	$t_{\mathbf{95}}$ $(\pm 2ms)$	$Angulo_{\mathbf{max}}(\pm 1.3deg)$
1	2.0	30	20.5
2	1.9	35	22.7
3	2.0	30	21.3
4	2.1	30	24.1
5	2.0	30	19.7
6	2.0	30	21.8
7	1.9	30	22.3
8	2.0	30	20.9
9	2.0	35	20.2
10	2.0	30	21.7
Promedio	2.0	31	21.52
Intervalo de confianza	[1.9,2.0]	[30,32]	[20.7,22.3]

Tabla 5.6: Datos de validación con ángulo de referencia de 15 grados.

N	$t_{\textbf{aleteo}}$ $(\pm 0.1s)$	$t_{\textbf{95}}$ $(\pm 2ms)$	$Angulo_{\textbf{max}}(\pm 0.2deg)$
1	2.0	40	14.0
2	2.0	35	14.1
3	2.0	40	14.1
4	2.0	40	14.2
5	1.9	40	14.3
6	1.9	40	14.4
7	2,0	40	14.4
8	2.0	40	14.4
9	2.1	40	14.4
10	2.0	40	14.4
Promedio	2.0	40	14.3
Intervalo de confianza	[1.9,2.0]	[39,40]	[14.2,14.4]

Tabla 5.7: Mediciones de latencia de comunicación I2C a 400kHz.

N	**Latencia** $(\pm 19\mu s)$
1	201
2	169
3	187
4	156
5	203
6	172
7	201
8	169
9	161
10	157
Promedio	178
Intervalo de confianza	[166,189]

Tomando los valores promedio de los ángulos máximos se puede observar que hay cierta discrepancia entre estos y los valores de referencia, se calcularon los valores de error para cada uno y se obtuvieron los siguientes valores $e_{35} = 11.3\%$, $e_{25} = 14.0\%$ y $e_{15} = 5.0\%$ lo cual las pone dentro del rango marginal de la métrica número 3. El ángulo máximo de actuación se encontró que es 31.6 grados, el cual lo pone en el valor ideal de la métrica 1. Para el tiempo de subida (métrica 2) que en el peor de los casos está en promedio 40 ms, también está dentro del rango del valor ideal de esta métrica. En la latencia se tiene un valor mucho menor a 30 ms (178 μs) que también lo posiciona en el rango del valor ideal de la métrica 7, lo mismo con la métrica 8 y el valor de R^2 del modelo de ángulo-resistencia.

Para las métricas restantes (4, 5 y 6) se pueden validar con la implementación, ella tiene un sistema de detección de fallos que se activa cuando la resistencia medida es menor que

14 Ohm (valor marginal) y el funcionamiento estable para que funcione con dos o más actuadores se puede afirmar que esto funciona, ya que en la electrónica y programación cada canal del DAC opera independientemente del resto, son 4 circuitos independientes (uno para cada canal). Esto hace que cada actuador pueda funcionar adecuadamente siempre y cuando la fuente de poder pueda suministrar suficiente potencia para todos.

Para la hipótesis se puede ver claramente que para cada escenario de actuación no hubo reducción ninguna, es solo el valor del sistema actual, negando la suposición que se hizo al inicio del proyecto. Ningún tiempo objetivo aplica para esto ya que los valores mínimos son 1.9 s y el valor mínimo para la reducción era 1.2 s. Esto se puede ver claramente en los intervalos de confianza de estos valores, que ninguno contiene el valor objetivo, por lo tanto se confirma que el sistema de control no reduce el tiempo de movimiento del actuador.

Esto se debe a que el controlador actual es capaz de suministrar la corriente necesaria para modificar rápidamente la resistencia del SMA, pero esta resulta insuficiente para generar movimientos más acelerados, ya que estos requerirían una corriente mayor. Esta limitación tiene un doble origen: por un lado, el diseño del sistema subutiliza el rango operativo del DAC, concentrándose en valores intermedios sin aprovechar su capacidad máxima de salida; por otro lado, la estrategia de control se basa exclusivamente en el monitoreo de la resistencia, sin incorporar ningún tipo de realimentación sobre parámetros cinemáticos como la velocidad de desplazamiento o el ángulo alcanzado.

Aunque el sistema logró reducir significativamente el tiempo necesario para alcanzar el valor de resistencia objetivo —prácticamente de manera instantánea—, con resultados coherentes con las simulaciones, los datos de caracterización revelan un umbral crítico: el alambre requería una aplicación mínima de potencia de algún tiempo mayor a 0.5 s (valor mínimo configurable en el DAC) para observar un cambio medible en la resistencia. Este comportamiento se evidencia en las gráficas de caracterización, donde se aprecia que, por debajo de este tiempo, la respuesta del sistema era insuficiente.

Adicionalmente, se verificó que el controlador PID mantiene los valores de resistencia cercanos al objetivo una vez superado este umbral inicial, demostrando su eficacia en la regulación dinámica. Esto se puede observar en la Figura 5.4.

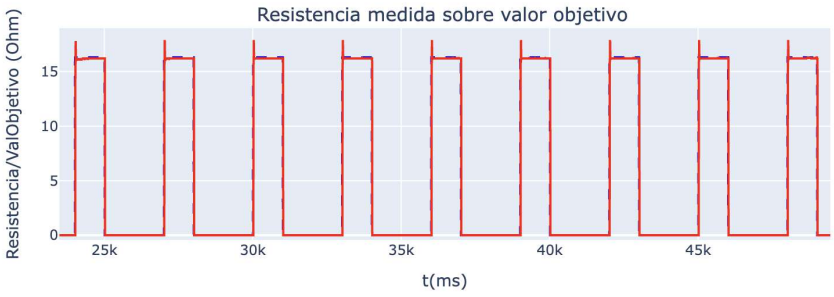


Figura 5.4: Salida del control.

En las figuras 5.5, 5.6, 5.7 se puede ver el resultado del movimiento controlado para 35, 25 y 15 grados, respectivamente. Se ve un movimiento oscilatorio que es repetible.

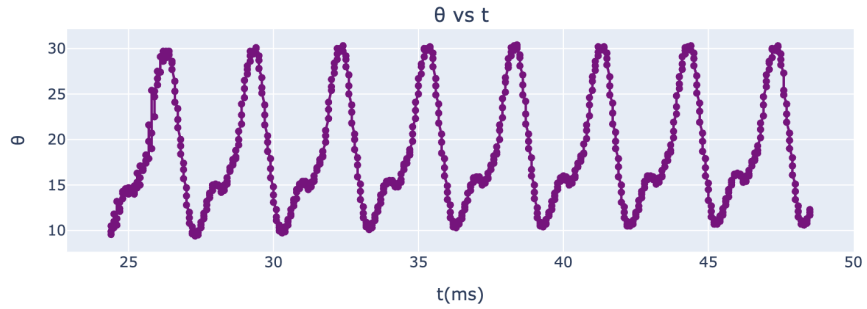


Figura 5.5: Movimiento de la aleta para pulsos de referencia de 35 grados.

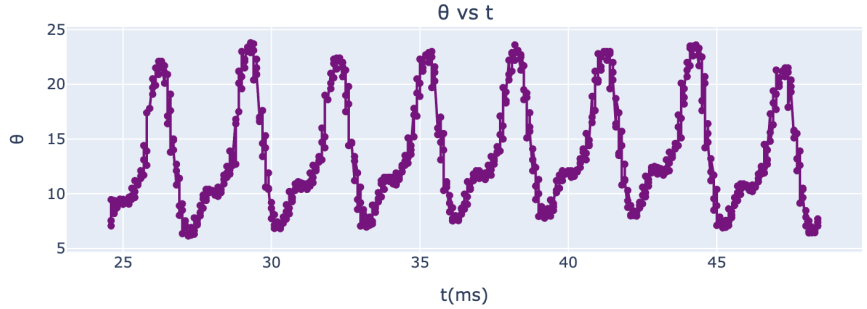


Figura 5.6: Movimiento de la aleta para pulsos de referencia de 25 grados.

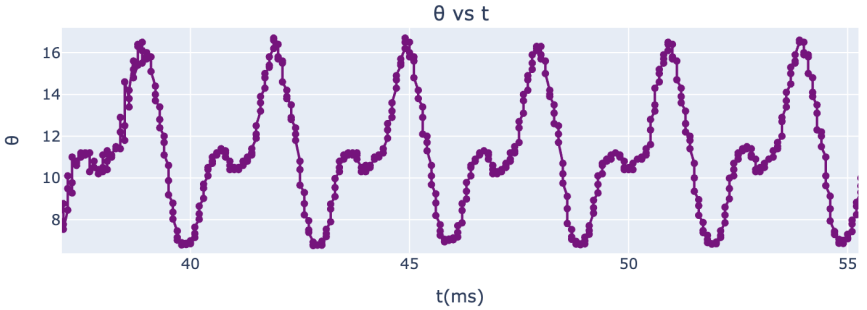


Figura 5.7: Movimiento de la aleta para pulsos de referencia de 15 grados.

El análisis de los valores promedio de los ángulos máximos revela discrepancias con los valores de referencia, situándolos dentro del rango marginal de la métrica 3. El ángulo máximo de actuación se estableció en 31.06 grados, valor que cumple con el ideal de la métrica 1. Respecto al tiempo de subida (métrica 2), el peor caso registrado fue de 39.5 ms en promedio, cumpliendo también con el rango ideal. La latencia satisface el criterio ideal de la métrica 7, al igual que el valor R^2 del modelo ángulo-resistencia (métrica 8).

Las métricas restantes (4, 5 y 6) se validan mediante la implementación: el sistema cuenta con detección de fallos que se activa cuando la resistencia medida es inferior a 14 Ω (valor marginal), mientras que el funcionamiento estable con múltiples actuadores está garantizado por la independencia de los 4 canales DAC, cada uno con su circuito independiente. Esto asegura un desempeño adecuado siempre que la fuente de alimentación provea suficiente potencia.

Respecto a la hipótesis inicial, los resultados obtenidos demuestran que no hubo reducción en los tiempos de actuación en ninguno de los escenarios evaluados. Los valores mínimos registrados fueron de 1.9 s, superando el objetivo inicial de 1.2 s. Para corroborar este hallazgo, se calculó el estadístico de prueba $t = 25.3$ (utilizando la distribución t de Student, dado que solo se dispone de la desviación estándar muestral). Este valor contrasta marcadamente con el valor crítico $t = -1.83$, correspondiente a un nivel de significancia $\alpha = 0.05$ y nueve grados de libertad. Dado que el estadístico se encuentra fuera de la región de rechazo (es decir, $25.3 > -1.83$), se confirma que no existe evidencia suficiente para afirmar que el tiempo de actuación se redujo por debajo de 1.2 s. Por lo tanto, no se rechaza H_0; esta conclusión se refuerza al analizar los intervalos de confianza, los cuales no incluyen el valor objetivo.

Esto indica la necesidad de implementar modificaciones en el sistema para alcanzar la meta establecida. Si bien no fue posible realizar estos ajustes durante el estudio debido a limitaciones de tiempo, en la sección Trabajo futuro del capítulo de Conclusiones se detallan algunas propuestas para optimizar el rendimiento del sistema.

5.1 Análisis económico

Esta sección presenta un balance detallado de los recursos invertidos y los resultados generados durante el desarrollo del proyecto. Al tratarse de una iniciativa de investigación sin fines comerciales, el análisis se centra exclusivamente en los costos monetarios incurridos y los aportes académicos obtenidos.

Los gastos en materiales para la fabricación de los actuadores se detallan en la Tabla 5.8. Es importante destacar que los componentes electrónicos no representaron un costo adicional, ya que fueron manufacturados previamente en el laboratorio. En cuanto al software, se utilizaron herramientas profesionales como Matlab bajo la licencia institucional proporcionada por el TEC [42], mientras que para el modelado 3D se empleó Shapr3D en su versión gratuita para estudiantes, aprovechando las facilidades académicas disponibles. Todos los equipos computacionales utilizados fueron de propiedad personal del investigador.

La inversión en mano de obra se calculó teóricamente equivalente a medio salario base de un ingeniero recién graduado en España (aproximadamente 1200 euros mensuales según [43]) durante cuatro meses de trabajo a medio tiempo. Sin embargo, este rubro no representó un gasto real para el CAR al tratarse de trabajo voluntario no remunerado. Los materiales principales incluyeron alambre de SMA (adquirido a 31.07 euros según [44]), que constituye el componente clave de los actuadores, junto con diversos insumos especializados adquiridos en Feroca, una tienda local especializada en materiales para resinas y siliconas ubicada cerca del centro de investigación. El costo total del proyecto, considerando todos estos factores, ascendió aproximadamente a 2 972 508 colones.

Aunque se trata de un proyecto de naturaleza investigativa, desarrollado en el ámbito universitario con fines académicos, y por tanto sin los insumos necesarios para generar un retorno económico directo, sus resultados representan valiosos aportes técnicos y académicos que

servirán como base para futuras investigaciones en el laboratorio.

- **Controlador PID para actuadores submarinos bioinspirados**: aunque este controlador en teoría no sea nada nuevo ni avanzado desde el punto de vista de teoría de control clásica, representa una innovación significativa al ser de los primeros desarrollos específicos para este tipo particular de actuadores que no existen en el mercado comercial. Esta implementación genera datos experimentales valiosos y establece parámetros de referencia para futuras investigaciones en el tema, representando un aporte relevante al conocimiento en esta área emergente.
- **Manual de diseño de controladores**: este manual, aunque breve y conciso, es sumamente valioso para el laboratorio ya que no solo documenta el conocimiento básico adquirido sobre el diseño de controladores para este tipo específico de actuadores, sino que también establece un proceso metodológico claro y reproducible que cualquier investigador futuro podrá seguir al implementar controladores similares. La documentación de este conocimiento tácito es fundamental para la continuidad de la investigación.
- **Avances en investigación de actuadores SMA**: el proyecto compiló y organizó conocimiento diverso sobre el funcionamiento y comportamiento de los actuadores de aleación con memoria de forma, estableciendo las bases para futuras publicaciones científicas y desarrollos académicos dentro del CAR. Los datos empíricos recolectados, las lecciones aprendidas y las metodologías desarrolladas representan un acervo importante para la siguiente fase de investigación en este campo.

Tabla 5.8: Inversión estimada durante desarrollo del proyecto.

Descripción	Cantidad	Precio (euros)	Precio (colones)
Alambre Actuador Dynalloy Muscle Wires 150 um LT, 5 m	1	31.07	18 497
Pol-Ease 2500. Desmoldeante universal en Aerosol	1	25.41	15 127
Barniz Acrílico en spray	1	19.24	11 455
PlatSil Gel-25 Silicona de Platino	1	48.28	28 742
Matlab Estudiante	1	69	41 079
Computadora personal	1	2400	1 428 840
Remuneración	4	600	1 428 768
Total (colones)	2 972 508		

5.2 Análisis de riesgo

El análisis de riesgos es una fase crítica en el desarrollo de proyectos de ingeniería, ya que permite anticipar posibles fallos y establecer estrategias para minimizar su impacto. En este

proyecto, se empleó un método basado en matrices de probabilidad-impacto (Figura 5.9), una herramienta ampliamente utilizada en gestión de riesgos por su capacidad para priorizar amenazas de manera visual y cuantitativa [45].

Cada riesgo se clasificó según dos parámetros en una escala del 1 al 5:

Probabilidad de que suceda:

- 1: Sería raro que ocurriera
- 2: Probabilidad baja
- 3: Probabilidad media
- 4: Probabilidad alta
- 5: Probabilidad muy alta

Impacto si sucede:

- 1: Despreciable
- 2: Marginal
- 3: Moderado
- 4: Crítico
- 5: Catastrófico

En la Tabla 5.9 se puede ver por completo

Tabla 5.9: Análisis de riesgo del proyecto.

Riesgo	Probabilidad (1-5)	Impacto (1-5)	Estraegias de mitigación
Modelos matemáticos imprecisos	3	4	Desarrollar modelos empíricos basados en datos experimentales
Retrasos en validación	3	4	Realizar pruebas piloto tempranas. Desarrollar simulaciones antes de las pruebas físicas.
Fallos en los actuadores	4	5	Tener actuadores de repuesto. Medir la resistencia del SMA para detectar cambios en el funcionamiento y prevenir fallos.
Desgaste prematuro de los actuadores	3	5	Limitar el rango de valores mínimos de resistencia para prevenir sobrecalentamiento
Limitada disponibilidad de los asesores.	4	3	Agendar reuinones semanales con anticipación. Documentar consultas a lo largo de la semana.

El análisis de riesgos realizado para el proyecto de diseño de control de actuadores SMA revela varios aspectos críticos que requieren especial atención. Entre los riesgos más significativos se encuentra la imprecisión de los modelos matemáticos, con una probabilidad media (3) pero un impacto crítico (4). Este riesgo surge fundamentalmente de la compleja naturaleza de la histéresis térmica y mecánica característica de las aleaciones con memoria de forma, que puede generar discrepancias entre los modelos teóricos y el comportamiento real del material. Para contrarrestar este problema, se ha establecido como estrategia de mitigación la validación sistemática de los modelos mediante datos experimentales, lo que permite reducir la incertidumbre y mejorar la precisión de las predicciones.

Otro riesgo destacable son los posibles retrasos en el proceso de validación, que también presenta una probabilidad media (3) y un impacto crítico (4). La principal causa de este riesgo radica en la dependencia de pruebas físicas y la disponibilidad de recursos de laboratorio. Para minimizar este efecto, se ha implementado un enfoque basado en simulaciones numéricas preliminares y el desarrollo de prototipos en etapas tempranas del proyecto, estrategias que permiten agilizar significativamente el proceso de validación.

El riesgo más crítico identificado corresponde a los fallos en los actuadores, con una alta probabilidad de ocurrencia (4) y un impacto catastrófico (5). Este riesgo está directamente asociado a la fatiga del material SMA, provocada por los repetidos ciclos térmicos y

posibles sobrecargas mecánicas durante la operación. Como medidas de mitigación, se ha implementado un sistema de monitoreo continuo de la resistencia eléctrica, que sirve como indicador temprano de degradación, complementado con el mantenimiento de un inventario de actuadores de repuesto para garantizar la continuidad operativa.

El desgaste prematuro de los actuadores constituye otro riesgo importante, con probabilidad media (3) pero impacto catastrófico (5). Su origen se encuentra principalmente en la operación fuera de los rangos seguros, particularmente en situaciones de sobrecalentamiento. Para abordar este problema, se han establecido límites estrictos en los parámetros de operación.

Finalmente, la limitada disponibilidad de los asesores, con alta probabilidad (4) pero impacto moderado (3), surge de los conflictos de agenda típicos en equipos. Este riesgo se gestiona mediante una cuidadosa planificación anticipada de reuniones y la implementación de un sistema de documentación estructurado para registrar consultas y decisiones a lo largo del desarrollo del proyecto.

Capítulo 6

Conclusiones

Este capítulo presenta las conclusiones y recomendaciones finales del proyecto. Las conclusiones reflejan el cumplimiento de los objetivos y la satisfacción de los requerimientos del cliente, mientras que las recomendaciones detallan los siguientes pasos, posibles aplicaciones y aprendizajes relevantes para proyectos futuros. Asimismo, se incluyen hallazgos derivados de la metodología empleada, las validaciones realizadas y el análisis económico.

El sistema de control para actuadores submarinos bioinspirados en SMA, desarrollado en el CAR, ha alcanzado los objetivos propuestos (excepto la reducción en el tiempo del aleteo del actuador), ofreciendo una solución tecnológica innovadora que optimiza los procesos de investigación y caracterización.

6.1 Conclusiones

Los objetivos del proyecto, descritos en la sección 1.5, se desarrollaron de la siguiente manera:

El primer objetivo, relacionado con la evaluación del proceso de identificación de necesidades del cliente y la definición de requisitos técnicos, se abordó en el capítulo 3. El análisis realizado demostró que todas las necesidades identificadas recibieron una calificación de importancia moderada o superior, lo que confirma que se captaron adecuadamente los requerimientos clave del cliente. Este proceso permitió establecer un conjunto de métricas técnicas fundamentales para evaluar el desempeño del sistema de control, asegurando que el diseño respondiera a las expectativas planteadas.

El segundo objetivo, centrado en el diseño del sistema de control, tuvo como entregable principal este documento técnico, junto con indicadores de desempeño específicos. Aunque se planteó como indicador una mejora en el movimiento del robot completo, factores externos mencionados en el alcance del proyecto impidieron esta evaluación. Sin embargo, para el actuador individual se verificó el cumplimiento de todas las métricas establecidas en el capítulo 3, aunque no siempre con los valores ideales. Los resultados demostraron que el actuador desarrollado satisface los requerimientos básicos de funcionamiento.

Este segundo objetivo está estrechamente vinculado con el tercero, que consistía en la validación del sistema. La validación realizada confirmó que el sistema de control funciona correctamente, siendo capaz de generar movimientos estables y repetibles en los actuadores mientras mantiene su resistencia cercana al valor de referencia. No obstante, no se logró mejorar el tiempo de respuesta del actuador, que se mantuvo sin cambios significativos respecto a su estado inicial. Esto se explica en el capítulo 5.

Los resultados de validación demuestran que, a pesar de ciertas limitaciones, el sistema cumple satisfactoriamente su función principal de controlar los actuadores submarinos bioinspirados. Se observó un error de posición que, aunque no alcanza el valor ideal, se mantiene por debajo del 15%, nivel considerado aceptable para las aplicaciones previstas de estos actuadores.

En cuanto al desempeño dinámico, el sistema presenta un tiempo de respuesta notablemente rápido para generar cambios en la resistencia del actuador, superando las expectativas iniciales. Adicionalmente, el diseño del actuador demostró capacidad para alcanzar ángulos de desplazamiento superiores al valor teórico ideal, lo que representa una ventaja operativa significativa.

Estos resultados globales indican que, dentro de los parámetros operacionales requeridos, el sistema desarrollado ofrece un rendimiento adecuado y en algunos aspectos (como velocidad de respuesta y rango angular) supera los requisitos inicialmente planteados, compensando así aquellas métricas donde no se alcanzaron los valores óptimos.

6.2 Trabajo Futuro

Como conclusión del proyecto, se proponen las siguientes recomendaciones para futuros desarrollos que permitirían optimizar el sistema de control y mejorar el rendimiento del actuador.

6.2.1 Aumentar el diámetro de los alambres SMA

La fuerza de actuación de los alambres SMA depende no solo de su longitud, sino también de su diámetro. Utilizar alambres con un diámetro mayor a 150 μm (por ejemplo, 200 μm) incrementaría significativamente su fuerza de contracción, facilitando la superación de la resistencia mecánica impuesta por la silicona y mejorando así la eficiencia del movimiento.

6.2.2 Diseñar un control para corriente

Durante el desarrollo, se observó que la corriente aplicada al SMA tiene una relación inversa con su resistencia: a mayor corriente, menor resistencia. Además, el movimiento del actuador depende directamente de la intensidad de corriente suministrada, permitiendo contracciones más rápidas y pronunciadas. Por lo tanto, se recomienda desarrollar un controlador de

corriente que permita ajustar con precisión el suministro eléctrico para alcanzar ángulos y tiempos de respuesta específicos.

Esta estrategia ofrecería varias ventajas:

- Independencia de la temperatura: A diferencia del control por resistencia, que se ve afectado por el calentamiento gradual del sistema, el control por corriente permite respuestas inmediatas al calentar el actuador más agresivamente.
- Mayor eficiencia energética: Mediante pulsos cortos de alta corriente, se podrían lograr movimientos rápidos sin sobrecalentar el controlador.
- Repetibilidad: Establecer valores de corriente específicos para posiciones clave mejoraría la consistencia del actuador.

6.2.3 Cambio de diseño del actuador

En las pruebas realizadas, se detectó que, cuando el alambre SMA alcanza temperaturas elevadas, su velocidad de relajación disminuye, lo que impide un retorno rápido a la posición inicial. Para solucionar este problema, se propone implementar un diseño antagonista que emplee dos alambres SMA opuestos.

Este enfoque permitiría:

- Movimiento más rápido y controlado: Mientras un alambre se contrae (activado), el otro se relaja (desactivado), mejorando la velocidad de respuesta general.
- Mayor precisión: El sistema antagonista ofrece un mejor control sobre la posición del actuador al equilibrar las fuerzas aplicadas.

Sin embargo, es crucial garantizar que solo un alambre esté activado en cada momento, ya que la activación simultánea podría generar esfuerzos mecánicos excesivos y dañar el sistema.

6.2.4 Estandarizar el proceso de fabricación de actuadores

Un aspecto crítico identificado durante el proyecto fue la variabilidad en las características de funcionamiento entre actuadores fabricados con los mismos materiales y procedimientos. Esta inconsistencia surge principalmente por diferencias en el tensado de los alambres SMA durante el montaje, un factor difícil de controlar incluso cuando el proceso es realizado por la misma persona. Adicionalmente, se observó un considerable desperdicio de material durante la fase de aprendizaje del proceso de facturación.

Para abordar estos problemas, resulta fundamental desarrollar métodos que garanticen una tensión uniforme en los alambres SMA durante la fabricación. Una solución práctica sería

implementar un sistema de montaje asistido que permita aplicar y medir la tensión de manera precisa y reproducible.

La implementación de protocolos de fabricación detallados sería igualmente importante. Dichos protocolos deberían especificar parámetros clave como la secuencia exacta de operaciones, valores de tensión objetivo, herramientas específicas a utilizar y puntos de verificación de calidad. Este enfoque permitiría reducir significativamente la variabilidad entre actuadores, optimizar el uso de materiales y facilitar la capacitación de nuevos operarios.

La estandarización del proceso de fabricación no solo mejoraría la consistencia en el desempeño de los actuadores, sino que también contribuiría a la reproducibilidad de los resultados experimentales. Este avance sería particularmente valioso para futuras investigaciones y desarrollos que requieran comparar datos entre diferentes prototipos o validar mejoras en el diseño.

Bibliografía

[1] U. P. de Madrid. Centro de Automática y Robótica (CAR). Centro Mixto UPM-CSIC. [En línea]. Disponible: https://www.upm.es/recursosidi/map/centro-de-automatica-y-robotica-car-centro-mixto-upm-csic/

[2] C. de Automática y Robótica. Centre for automation and robotics. [En línea]. Disponible: https://www.car.upm-csic.es/about-us/

[3] R. E. Shadwick and G. V. Lauder, *Fish Biomechanics*. Elsevier Academic Press, 2006.

[4] E. G. Drucker and G. V. Lauder, "Function of pectoral fins in rainbow trout: behavioral repertoire and hydrodynamic forces," *The Journal of Experimental Biology*, vol. 206, DOI 10.1242/jeb.00139, pp. 813–826, 2002.

[5] E. G. Drucker and G. V. Lauder, "Wake dynamics and locomotor function in fishes: Interpreting evolutionary patterns in pectoral fin design," *Integrative and Comparative Biology*, vol. 47, pp. 991–1008, 2002.

[6] J. A. W. EMark W. Westneat, Dean H. Thorsen and M. E. Hale, "Structure, function, and neural control of pectoral fins in fishes," *IEEE Journal of Oceanic Engineering*, vol. 29, no. 3, pp. 647–683, 2004.

[7] D. H. Thorsen and M. W. Westneat, "Diversity of pectoral fin structure and function in fishes with labriform propulsion," *Journal of Morphology*, 2005.

[8] E. G. Drucker and J. S. Jensen, "Kinematic and electromyographic analysis of steady pectoral fin swimming in the surfperches," *The Journal of Experimental Biology*, vol. 200, pp. 1709–1723, 1997.

[9] J. L. Tangorra, G. V. Lauder, I. W. Hunter, R. Mittal, P. G. A. Madden, and M. Bozkurttas, "The effect of fin ray flexural rigidity on the propulsive forces generated by a biorobotic fish pectoral fin," *The Journal of Experimental Biology*, vol. 213, pp. 4043–4054, 2010.

[10] G. L. Chris Phelan, James Tangorra and M. Hale, "A biorobotic model of the sunfish pectoral fin for investigations of fin sensorimotor control," DOI 10.1088/1748-3182/5/3/035003.

[11] R. M. P. M. M. Bozkurttas, H. Dong. and G. Lauder, "Hydrodynamic performance of deformable fish fins and flapping foils," *AIAA Aerospace Sciences Meeting and Exhibit*, vol. 44, 2006.

[12] G. V. Lauder and B. C. Jayne, "Pectoral fin locomotion in fishes: Testing drag based models using three-dimensional kinematics," *American Zoologist*, vol. 36, pp. 567–581, 1996.

[13] E. M. Standen and G. V. Lauder, "Dorsal and anal fin function in bluegill sunfish lepomis macrochirus: three-dimensional kinematics during propulsion and maneuvering," *The Journal of Experimental Biology*, vol. 208, pp. 2753–2763, 2005.

[14] Y. H. Teh and R. Featherstone, "An architecture for fast and accurate control of shape memory alloy actuators," *The International Journal of Robotics Research*, vol. 27, DOI 10.1177/0278364908090951, no. 5, pp. 595–611, 2008.

[15] W. Coral and C. Rossi, "Soft dorsal/anal fins pairs for roll and yaw motion in robotic fish," *Bioinspiration Biomimetics*, vol. 18, DOI 10.1088/1748-3190/aca132, no. 1, p. 016008, 2023.

[16] SIMSCALE. What is Joule Heating (Joule Effect?). [En línea]. Disponible: https://www.simscale.com/docs/simwiki/heat-transfer-thermal-analysis/what-is-joule-heating/

[17] K. Ogata, *Ingeniería de control moderna*. Person Education, 2010.

[18] S. Majima, K. Kodama, and T. Hasegawa, "Modeling of shape memory alloy actuator and tracking control system with the model," *IEEE Transactions on Control Systems Technology*, vol. 9, DOI 10.1109/87.896745, no. 1, pp. 54–59, 2001.

[19] K. Signh, J. Sirohi, and I. Chopra, "An improved shape memory alloy actuator for rotor blade tracking," *Journal of intelligent material systems and structures*, vol. 14, DOI 10.1177/104538903039134, no. 12, pp. 767–786, 2003.

[20] Y. H. Teh. Fast, accurate force and position control of shape memory alloy actuators. [En línea]. Disponible: https://www.semanticscholar.org/paper/Fast%2C-accurate-force-and-position-control-of-shape-Teh/3124cb486c50640997e6391a6f436d225dc336fd

[21] K. U. y S.D. Eppinger, *Diseño y Desarrollo de Productos*. McGraw-Hill, 2013.

[22] J. Crespo, *Módulo 1. sesión 2: "fase 0"*. Presentación en clase, 2024.

[23] C. Rossi and W. Coral, *Bending formulas*, 2021.

[24] E. Potapov, Pavel Silva, "Time response of shape memory alloy actuators," *Journal of intelligent material systems and structures*, vol. 11, DOI 10.1106/XH1H-FH3Q-1YEX-4H3F, pp. 125–134, 2000.

[25] Dynalloy. Technical sheets. [En línea]. Disponible: https://dynalloy.com/tech-sheets/

[26] J. Tan. Intro to time-sensitive networking for m2m communication in iot. [En línea]. Disponible: https://www.seeedstudio.com/blog/2021/07/19/intro-to-time-sensitive-networking-for-m2m-communication-in-iot/#:~:text=Industrial%20control%20systems%20are%20required%20to%20process,more%20effective%20the%20resulting%20action%20will%20be.

[27] Eseye. High data low latency iot connectivity. [En línea]. Disponible: https://www.eseye.com/resources/blogs/high-data-low-latency-iot-connectivity/#:~:text=Connectivity%20requirements%20for%20high%20data,in%20case%20of%20an%20outages.

[28] K. Sarraf. Re: What is the acceptable r-squared value?. [En línea]. Disponible: https://www.researchgate.net/post/what_is_the_acceptable_r-squared_value/65d4743e7a7c2bf1e70c2876/citation/download.

[29] I. Valchanov. Measuring explanatory power with the r-squared. [En línea]. Disponible: https://365datascience.com/tutorials/statistics-tutorials/r-squared/.

[30] R. N. Tutorials. Esp32: How to log data (10 different ways). [En línea]. Disponible: https://randomnerdtutorials.com/esp32-how-to-log-data/#microsd-card

[31] T. Instruments. Ina333 micro-power (50a), zerø-drift, rail-to-rail out instrumentation amplifier. [En línea]. Disponible: https://www.ti.com/lit/ds/symlink/ina333.pdf?ts=1752206410124&ref_url=https%253A%252F%252Fwww.mouser.fi%252F

[32] T. Semiconductor. Bc817-16/-25/-40. [En línea]. Disponible: https://services.taiwansemi.com/storage/resources/datasheet/BC817-16%20SERIES_K2001.pdf

[33] M. A. D. S. Ashwaq Abdulameer, Marizan Sulaiman, "Gui based control system analysis using pid controller for education," *JIndonesian Journal of Electrical Engineering and Computer Science.*, vol. 3, DOI 10.11591/ijeecs.v3.i1.pages, no. 1, pp. 91–101, 2016.

[34] S. Chak. Pid tuning via ziegler nichols method. [En línea]. Disponible: https://es.mathworks.com/matlabcentral/answers/2069011-pid-tuning-via-ziegler-nichols-method

[35] B. Beauregard. Improving the beginner's pid. [En línea]. Disponible: http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

[36] D. C. Montgomery, *Design and analysis of experiments.* John Wiley Sons, Inc, 2017.

[37] H. W. Coleman and W. G. Steele, *Experimentation, validation, and uncertainty analysis for engineers.* John Wiley Sons, Inc, 2009.

[38] R. E.Walpole, R. H. Myers, S. L. Myers, and K. Ye, *Probabilidad y estadística para ingeniería y ciencias. Novena Edición.* PEARSON EDUCACIÓN, 2012.

[39] R. Ferrero. ¿qué es el tamaño muestral y por qué importa? [En línea]. Disponible: https://www.maximaformacion.es/blog-dat/tamano-muestral-y-potencia-estadistica/#:~:text=%C2%BFCu%C3%A1l%20ser%C3%ADa%20un%20valor%20aceptable,el%2080%25%20de%20las%20veces.

[40] M. E. Vílchez, *Teoría de mediciones y otros conceptos básicos.* Instituto Tecnológico de Costa Rica, 2008.

[41] O. S. Physics, "Tracker," 2024. [En línea]. Disponible: https://opensourcephysics.github.io/tracker-website/

[42] Mathworks. Nueva licencia de matlab student. [En línea]. Disponible: https://es.mathworks.com/store/link/products/student/SV?s_tid=ac_buy_sv_but1

[43] Sipem. ¿cuánto cobra un ingeniero recién graduado en españa? [En línea]. Disponible: https://sipem.upct.es/noticia/cuanto-cobra-un-ingeniero-recien-graduado-en-espana

[44] Robotshop. Alambre actuador dynalloy muscle wires 150 um lt, 5 m. [En línea]. Disponible: https://eu.robotshop.com/es/products/cable-actuador-dynalloy-muscle-wires-150-m-lt-5-m?qd=8b9796c7187a050f5cdacab3339e18cf

[45] A. Conocimiento. Matriz probabilidad-impacto. [En línea]. Disponible: https://activaconocimiento.es/matriz-probabilidad-impacto/

Apéndice A

Proceso de facturación de los actuadores

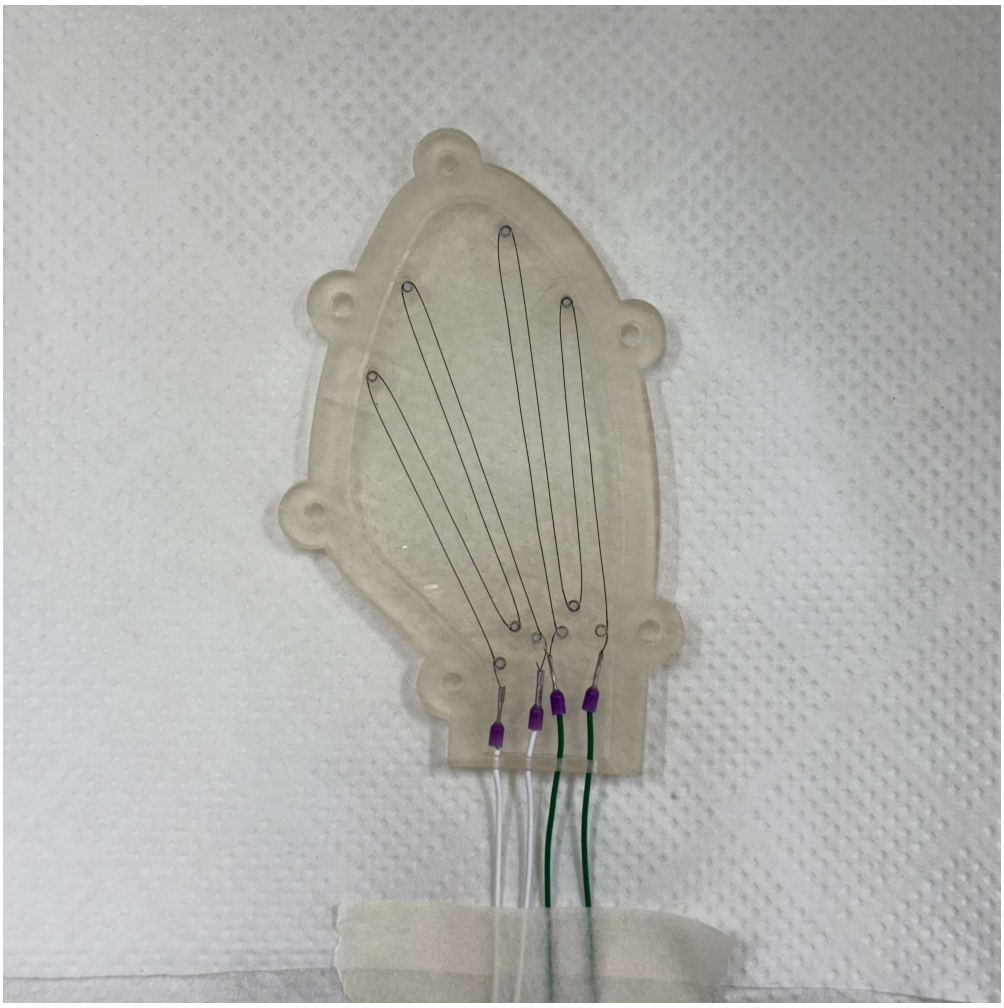


Figura A.1: Resultado de 1, 2 y 3.

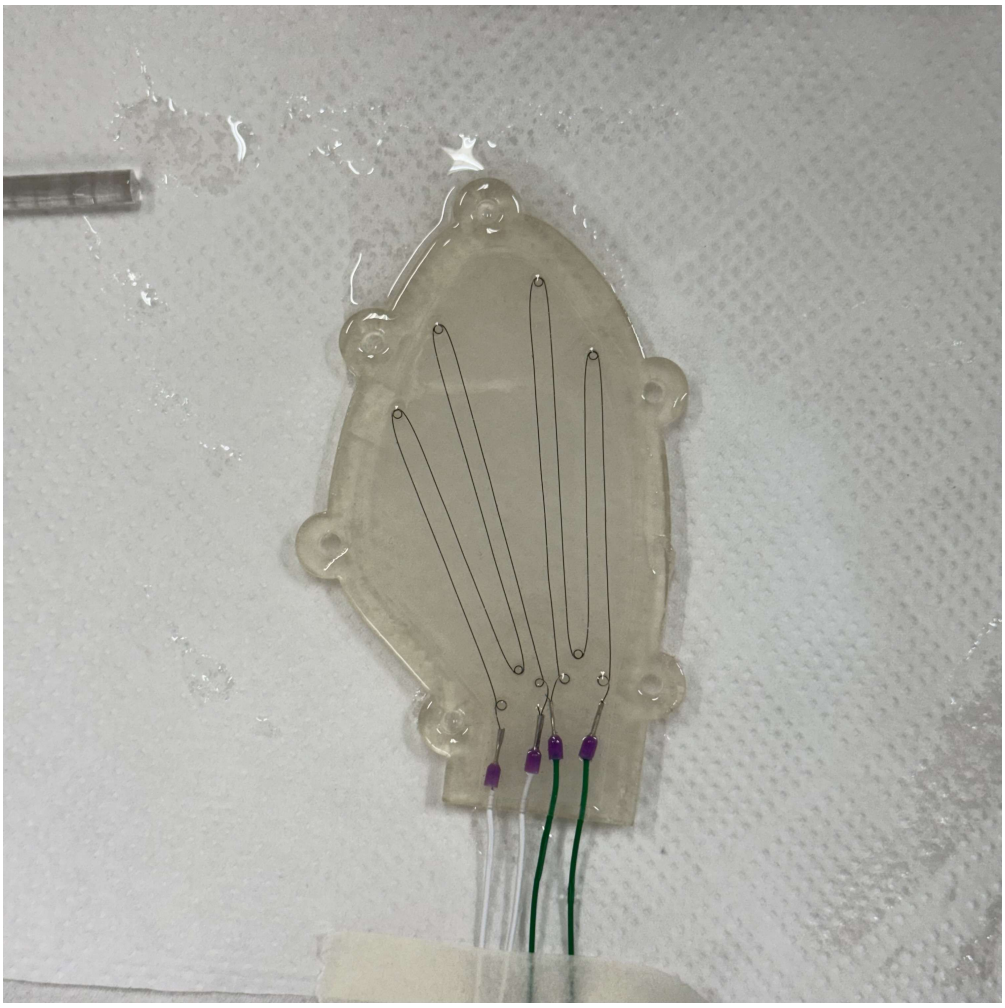


Figura A.2: Resultado de paso 4.

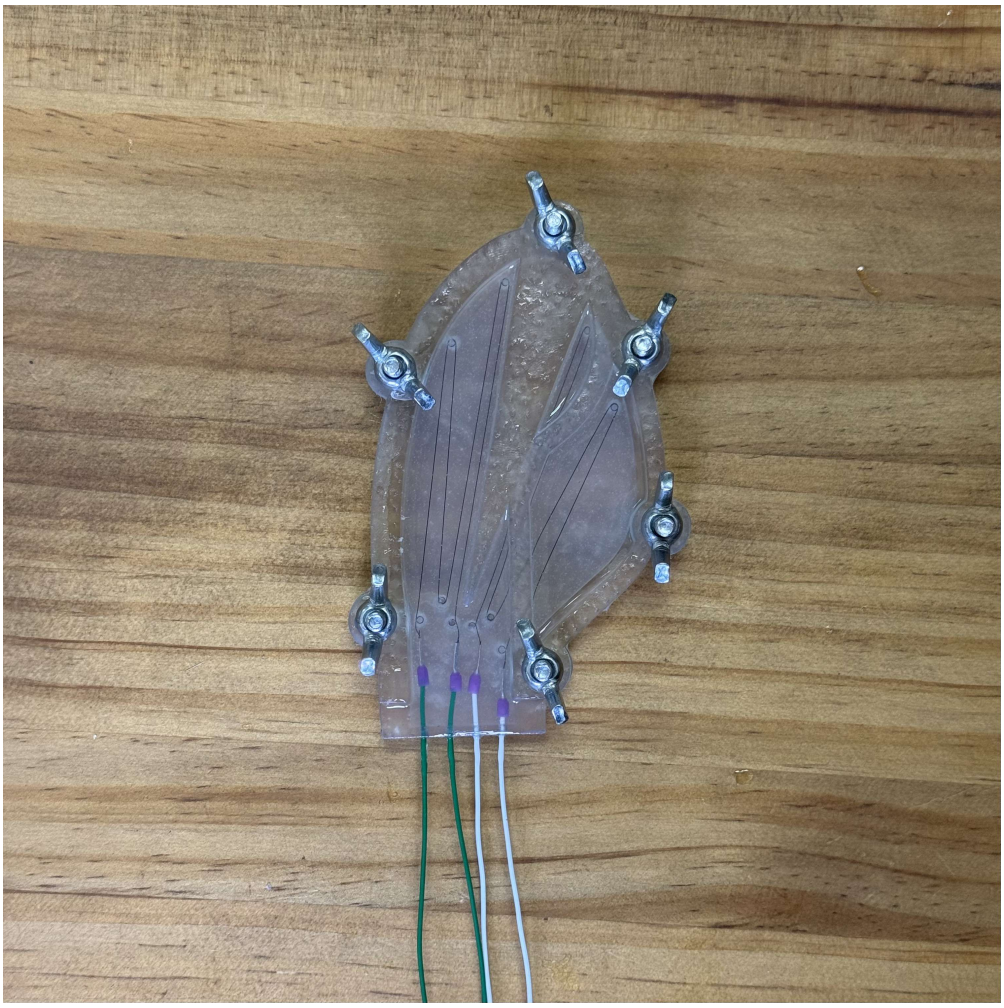


Figura A.3: Curado de paso 5.

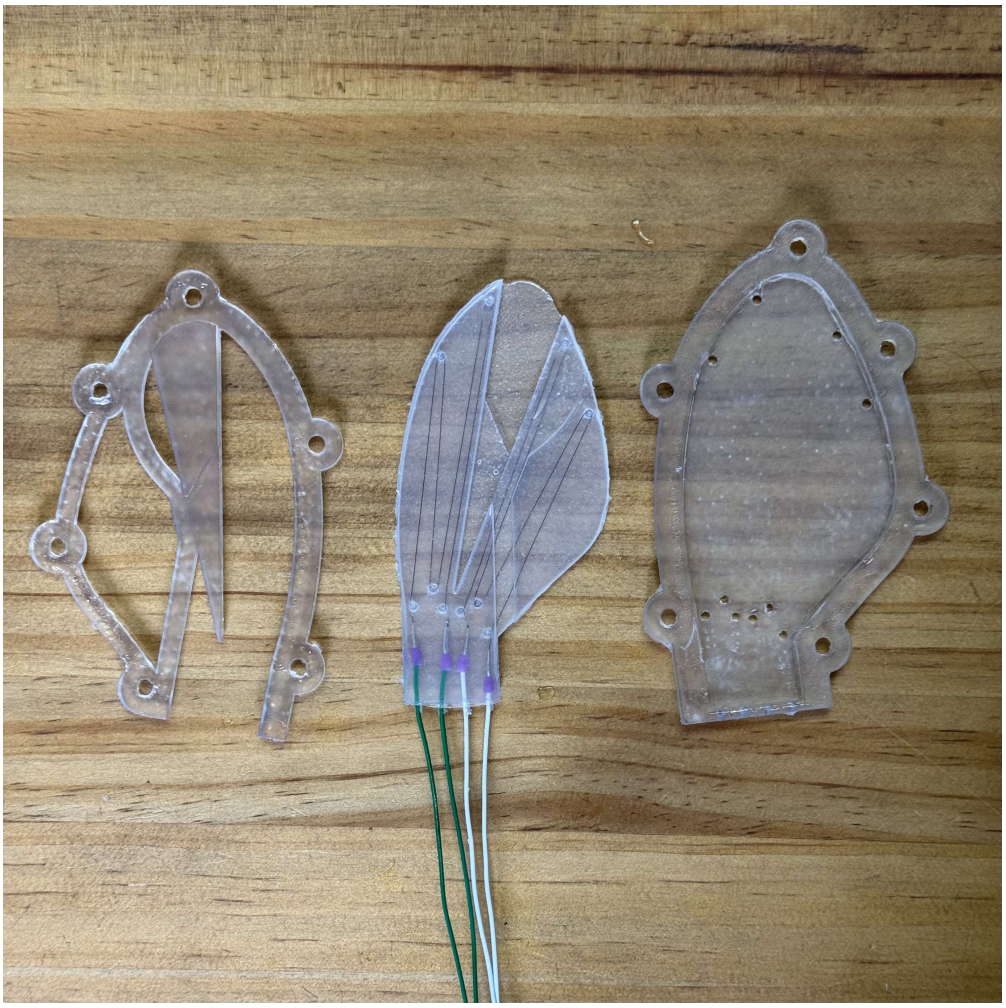


Figura A.4: Desmolde 5.

Apéndice B

Código de Ziegler-Nichols en MATLAB

```
% PID Controller Tuning using Modified Ziegler-Nichols Step
   Response Method
% This script demonstrates an aggressive PID tuning approach to
   achieve
% faster response times while limiting overshoot

% Clear workspace and figures for a fresh start
clc;          % Clear command window
clear;        % Clear workspace variables
close all;    % Close all open figures

%% SYSTEM DEFINITION
% Define the plant transfer function
num = [4.955 10.37];  % Numerator coefficients of transfer
   function
den = [1 2.183];      % Denominator coefficients of transfer
   function
sys = tf(num, den);   % Create transfer function object

% Plot the open-loop step response to understand system behavior
figure;
step(sys);            % Plot step response
title('Original System Step Response');
grid on;              % Add grid for better readability

%% USER SPECIFICATIONS
% Set desired performance criteria
desired_rise_time = 0.2;  % Target time to reach 95% of final
```

```
   value (seconds)
max_overshoot = 20;        % Maximum allowed overshoot (percentage
   )

%% ZIEGLER-NICHOLS STEP RESPONSE TUNING (MODIFIED)
% Extract step response data for analysis
[y, t] = step(sys);        % Get step response output and time
   vectors
final_value = y(end);      % Determine steady-state value

% Calculate dead time (L) - time to reach 28.3% of final value
L = t(find(y > 0.283*final_value, 1));

% Calculate time constant (T) - time between 28.3% and 63.2% of
   final value
T = t(find(y > 0.632*final_value, 1)) - L;

% Ensure non-zero values for stability in calculations
L = max(L, t(2));  % Prevent L from being smaller than second
   time sample
T = max(T, t(2));  % Prevent T from being smaller than second
   time sample

% Modified Ziegler-Nichols parameters for aggressive tuning:
% - Increased proportional gain for faster response
% - Reduced integral time to decrease lag
% - Increased derivative time for better damping
Kp = 2.0 / (L/T);       % Proportional gain (increased from
   standard 1.2)
Ti = 1.0 * L;           % Integral time (reduced from standard 2L)
Td = 0.8 * L;           % Derivative time (increased from standard
    L/2)

%% PID CONTROLLER IMPLEMENTATION
s = tf('s');  % Create Laplace variable for transfer function

% Create PID controller in parallel form: Kp + Ki/s + Kd*s
pid_aggressive = Kp * (1 + 1/(Ti*s) + Td*s);

% Form closed-loop system with negative feedback
cl_agg = feedback(pid_aggressive * sys, 1);

% Measure system performance
```

```
step_info = stepinfo(cl_agg);  % Get step response
   characteristics
rise_time_achieved = step_info.RiseTime * 1.5;  % Approximate 95%
    rise time

% Display initial tuning results
fprintf('Initial Aggressive Tuning:\n');
fprintf('Kp = %.3f, Ki = %.3f, Kd = %.3f\n', Kp, Kp/Ti, Kp*Td);
fprintf('Achieved 95%% Rise Time = %.3f sec\n',
   rise_time_achieved);

%% ITERATIVE TUNING FOR IMPROVED PERFORMANCE
if rise_time_achieved > desired_rise_time
    fprintf('\n--- Optimizing for Faster Response ---\n');

    % Tuning adjustment factors:
    Kp_scale = 1.1;    % 10% increase in Kp each iteration for
       faster response
    Td_scale = 1.05;   % 5% increase in Td each iteration for
       better damping
    Ti_scale = 0.95;   % 5% decrease in Ti each iteration to
       reduce lag

    % Perform up to 20 tuning iterations
    for i = 1:20
        % Adjust PID parameters
        Kp = Kp * Kp_scale;  % Increase proportional gain
        Td = Td * Td_scale;  % Increase derivative time
        Ti = Ti * Ti_scale;  % Decrease integral time

        % Rebuild PID controller with new parameters
        pid_aggressive = Kp * (1 + 1/(Ti*s) + Td*s);
        cl_agg = feedback(pid_aggressive * sys, 1);

        % Evaluate new performance
        step_info = stepinfo(cl_agg);
        rise_time_achieved = step_info.RiseTime * 1.5;
        overshoot = step_info.Overshoot;

        % Display iteration results
        fprintf('Iteration %d: Rise Time = %.3f sec, Overshoot =
           %.1f%%\n', ...
                i, rise_time_achieved, overshoot);
```

```
        % Stop if specifications are met or overshoot exceeds
           limit
        if rise_time_achieved <= desired_rise_time || overshoot >
            max_overshoot
            break;
        end
    end
end

%% FINAL RESULTS
fprintf('\n--- Final Tuned PID ---\n');
fprintf('Kp = %.3f, Ki = %.3f, Kd = %.3f\n', Kp, Kp/Ti, Kp*Td);
fprintf('Achieved 95%% Rise Time = %.3f sec\n',
   rise_time_achieved);
fprintf('Overshoot = %.1f%%\n', step_info.Overshoot);

%% PLOT FINAL RESPONSE
figure;
step(cl_agg);  % Plot step response of tuned system
hold on;
% Add line indicating 95% of final value
yline(0.95 * final_value, '--r', '95% Target');
title(sprintf('Aggressive PID: %.3fs Rise Time, %.1f%% Overshoot'
   , ...
       rise_time_achieved, step_info.Overshoot));
grid on;
legend('Step Response', '95% Target');

%% COMPARE WITH ORIGINAL SYSTEM
figure;
% Plot both original and tuned responses for comparison
% CORRECTED PLOTTING SYNTAX:
[y_orig, t_orig] = step(sys);
[y_tuned, t_tuned] = step(cl_agg);
plot(t_orig, y_orig, 'b--', 'LineWidth', 1.5);
hold on;
plot(t_tuned, y_tuned, 'r', 'LineWidth', 1.5);
yline(0.95 * final_value, '--k', '95% Target');
title('Comparison: Original vs. Aggressive PID');
xlabel('Time (seconds)');
ylabel('Amplitude');
legend('Original', 'Aggressive PID', '95% Target');
```

```
grid on;
```

Apéndice C

Código de implementación en Arduino

C.0.1 PID_SMA.ino

```
1    /**
2  * Main control program for PID-based actuator positioning system
3  *
4  * System Features:
5  * - 4 independent control channels (A-D)
6  * - Real-time PID control with configurable parameters
7  * - Multiple waveform generation for testing
8  * - Safety monitoring and emergency shutdown
9  * - Performance metrics collection
10 *
11 * Hardware Requirements:
12 * - Adafruit MCP4728 12-bit DAC
13 * - Multiple ADC channels for voltage measurement
14 * - Current shunt resistors for each channel
15 *
16 * Operation:
17 * 1. Initializes DAC and all control channels
18 * 2. Runs control loop at fixed interval (PID_SAMPLE_TIME)
19 * 3. Processes serial commands for configuration
20 * 4. Monitors safety conditions
21 */
22 #include <Wire.h>
23 #include <Adafruit_MCP4728.h>
24 #include "PID_v1.h"
25 #include "Config.h"
26 #include "TestSignalGenerator.h"
```

```
27 #include "CommandHandler.h"
28
29 // Global DAC object for all channels
30 Adafruit_MCP4728 dac;
31
32 void setup() {
33   Serial.begin(SERIAL_BAUD_RATE);
34   Wire.begin(8, 9);  // I2C pins
35
36   // Initialize DAC - critical for system operation
37   if (!dac.begin(MCP4728_ADDR)) {
38     Serial.println("DAC␣INIT␣FAILED␣-␣CHECK␣CONNECTIONS");
39     while(1);  // Halt if DAC communication fails
40   }
41
42   // Safety initialization - set all DAC outputs to minimum
43   for (int i = 0; i < 4; i++) {
44     dac.setChannelValue(channels[i].dacChannel, DAC_MIN);
45   }
46
47   // Initialize channel controllers and signal generators
48   initializeChannels();
49   Serial.println("SYSTEM␣INITIALIZED");
50   printHelp();  // Show command reference
51 }
52
53 void loop() {
54   static unsigned long lastControlTime = 0;
55   static unsigned long masterTime = millis();
56
57   // Fixed-interval control loop (PID_SAMPLE_TIME ms)
58   if (millis() - lastControlTime >= PID_SAMPLE_TIME) {
59     lastControlTime = millis();
60     masterTime = millis();
61
62     // Update all enabled channels
63     for (int i = 0; i < 4; i++) {
64       if (!channels[i].enabled) continue;
65
66       // Skip update if in idle mode (already handled)
67       if (!channels[i].inIdleMode) {
68         updateChannel(i, masterTime);
69       }
```

```
70      }
71    }
72
73    // Process any incoming serial commands
74    handleSerialCommands();
75  }
```

C.0.2 Config.h

```
1  /**
2   * System Configuration
3   *
4   * Centralized configuration parameters for:
5   * - Hardware pin assignments
6   * - Control parameters
7   * - Safety limits
8   * - Communication settings
9   */
10 #ifndef CONFIG_H
11 #define CONFIG_H
12
13 /*******************************
14  * HARDWARE CONFIGURATION
15  *******************************/
16 #define MCP4728_ADDR          0x60    // I2C address of DAC
17
18 // Channel A
19 #define ADC_V1_PIN_A          2       // Shunt voltage pin for
       channel A
20 #define ADC_V3_PIN_A          1       // Output voltage pin for
       channel A
21
22 // Channel B
23 #define ADC_V1_PIN_B          4       // Shunt voltage pin for
       channel B
24 #define ADC_V3_PIN_B          3       // Output voltage pin for
       channel B
25
26 // Channel C
27 #define ADC_V1_PIN_C          6       // Shunt voltage pin for
       channel C
28 #define ADC_V3_PIN_C          5       // Output voltage pin for
       channel C
```

```
29
30 // Channel D
31 #define ADC_V1_PIN_D          17      // Shunt voltage pin for
     channel D
32 #define ADC_V3_PIN_D          7       // Output voltage pin for
     channel D
33
34 #define SHUNT_R               3.3     // Shunt resistor value (
     ohms)
35 #define ADC_VREF              3.3     // Reference voltage for
     ADC
36 #define ADC_MAX_VALUE         4095.0  // 12-bit ADC resolution
37
38 /*******************************
39  * IDLE CURRENT CONTROL
40  *******************************/
41 #define IDLE_CURRENT          0.45     // Maintenance current (A)
42 #define IDLE_V1_TARGET        (IDLE_CURRENT * SHUNT_R)  // Target
     V1 voltage
43 #define IDLE_V1_TOLERANCE     0.05     //  5  percent tolerance
     for V1
44 #define IDLE_CURRENT_TIMEOUT 5000     // Time before entering
     idle mode (ms)
45
46 /*******************************
47  * PID CONTROL PARAMETERS
48  *******************************/
49 #define PID_MIN               0        // Minimum PID output value
50 #define PID_MAX               4000     // Maximum PID output value
51 #define DAC_MIN               0        // Minimum effective DAC
     value
52 #define DAC_MAX               4000     // Maximum DAC value (12-
     bit)
53
54 // Default PID Tuning Parameters
55 #define DEFAULT_KP            2.0
56 #define DEFAULT_KI            98.8
57 #define DEFAULT_KD            0.034
58 #define PID_SAMPLE_TIME       5        // PID calculation interval
      (ms)
59
60 // PID Direction (REVERSE for this application)
61 #define PID_DIRECTION         REVERSE
```

```
62
63 /*******************************
64  * SYSTEM PARAMETERS
65  *******************************/
66 #define MIN_OPERATING_CURRENT 0.45    // Minimum current (A) for
     valid measurements
67 #define DEFAULT_TARGET_R      17.0    // Default target
     resistance (ohms)
68 #define MIN_RESISTANCE        14.0    // Minimum allowed
     resistance (ohms)
69 #define MAX_RESISTANCE        50      // Maximum allowed
     resistance (ohms)
70
71 /*******************************
72  * DISCONNECTION DETECTION
73  *******************************/
74 #define DISCONNECTION_TIMEOUT_MS  5000    // Time without current
      to consider disconnected
75 #define MIN_OPERATING_VOLTAGE     0.25     // Minimum expected
     voltage when active
76
77 /*******************************
78  * SERIAL COMMUNICATION
79  *******************************/
80 #define SERIAL_BAUD_RATE     115200  // Serial communication
     speed
81
82 /*******************************
83  * SAFETY LIMITS
84  *******************************/
85 #define EMERGENCY_HYSTERESIS  1.0     // Resistance must recover
     by this much
86 #define MAX_CURRENT           1.0     // Absolute maximum allowed
      current (A)
87 #define MAX_DAC_RAMP_STEP     1000    // Maximum step size during
      DAC ramp-up
88
89 #endif // CONFIG_H
```

C.0.3 TestSignalGenerator.h

```
1 /**
2  * Test Signal Generator
```

```
3  *
4  * Generates various waveform patterns for system testing
5  *
6  * Supported Waveforms:
7  * - Constant
8  * - Pulse
9  * - Sine
10 * - Triangle
11 * - Sawtooth
12 * - Square
13 *
14 * Features:
15 * - Adjustable frequency, amplitude, offset
16 * - Phase control
17 * - Phase synchronization between channels
18 */
19 #ifndef TESTSIGNALGENERATOR_H
20 #define TESTSIGNALGENERATOR_H
21
22 #include <Arduino.h>
23
24 // Signal type enumeration
25 enum SignalType {
26   SIGNAL_MANUAL = 0,     // Manual value
27   SIGNAL_CONSTANT = 1,   // Constant output
28   SIGNAL_PULSE = 2,      // Pulse wave
29   SIGNAL_SINE = 3,       // Sine wave
30   SIGNAL_TRIANGLE = 4,   // Triangle wave
31   SIGNAL_SAWTOOTH = 5,   // Sawtooth wave
32   SIGNAL_SQUARE = 6      // Square wave
33 };
34
35 class TestSignalGenerator {
36 public:
37   TestSignalGenerator();
38
39   // Configuration
40   void setType(SignalType type) { _type = type; }  // Set
       waveform type
41   void setFrequency(float freqHz);                 // Set
       frequency (Hz)
42   void setAmplitude(double amplitudeDegrees) { _amplitude =
       amplitudeDegrees; } // Set amplitude (degrees)
```

```
43   void setOffset(double offsetDegrees) { _offset = offsetDegrees; } // Set DC offset (degrees)
44   void setPhase(float phaseDegrees);              // Set phase (degrees)
45   void setDutyCycle(float duty) { _dutyCycle = duty; } // Set duty cycle (0-1)
46   void setManualValue(double degrees) { _manualValue = degrees; } // Set manual value
47   void setPhaseReference(const TestSignalGenerator* refSignal); // Set phase reference
48
49   // Execution
50   double update(unsigned long currentTimeMs); // Update signal generator
51
52   // Debug
53   void debugPrint(); // Print current state
54   bool getCurrentState() const { return _currentState; } // Get current state (for square/pulse waves)
55
56 private:
57   // Signal parameters
58   SignalType _type = SIGNAL_MANUAL; // Current signal type
59   double _manualValue = 0;          // Manual value
60   float _frequency = 1.0;           // Frequency (Hz)
61   double _amplitude = 50.0;         // Amplitude (degrees)
62   double _offset = 0;               // DC offset (degrees)
63   float _phase = 0;                 // Phase offset (degrees)
64   float _dutyCycle = 0.5;           // Duty cycle (0-1)
65
66   // Timing variables
67   float _periodMs = 1000.0;         // Wave period (ms)
68   unsigned long _phaseOffsetMs = 0;  // Phase offset (ms)
69   unsigned long _refStartTime = 0;   // Reference start time
70
71   // State tracking
72   const TestSignalGenerator* _phaseRef = nullptr; // Phase reference
73   bool _currentState = false;        // Current state (for square/pulse)
74   unsigned long _lastTransitionTime = 0; // Last transition time
75   double _accumulatedPhase = 0;      // Accumulated phase (radians)
```

```
76   unsigned long _lastUpdateTime = 0; // Last update time
77
78   // Waveform generation
79   double _generateWave(double phase); // Generate waveform value
80 };
81 #endif
```

C.0.4 TestSignalGenerator.cpp

```
1  /**
2   * Test Signal Generator Implementation
3   *
4   * Implements waveform generation for system testing
5   */
6  #include "TestSignalGenerator.h"
7  #include <math.h>
8
9  #ifndef PI
10 #define PI 3.14159265358979323846
11 #endif
12
13 TestSignalGenerator::TestSignalGenerator() {}
14
15 /**
16  * Set frequency
17  *
18  * @param freqHz Frequency in Hz (0.001-100.0)
19  */
20 void TestSignalGenerator::setFrequency(float freqHz) {
21     _frequency = constrain(freqHz, 0.001, 100.0);
22 }
23
24 /**
25  * Set phase
26  *
27  * @param phaseDegrees Phase in degrees (normalized to 0-360)
28  */
29 void TestSignalGenerator::setPhase(float phaseDegrees) {
30     _phase = fmod(phaseDegrees, 360.0);
31     if (_phase < 0) _phase += 360.0;
32 }
33
34 /**
```

```
35  * Set phase reference
36  *
37  * Synchronizes this generator's phase to another generator
38  *
39  * @param refSignal Reference signal generator
40  */
41 void TestSignalGenerator::setPhaseReference(const
      TestSignalGenerator* refSignal) {
42     _phaseRef = refSignal;
43     if (refSignal) {
44         // Sync phase with reference
45         _accumulatedPhase = refSignal->_accumulatedPhase + (
               _phase * PI / 180.0);
46     }
47 }
48
49 /**
50  * Update signal generator
51  *
52  * Calculates current waveform value based on time
53  *
54  * @param currentTimeMs Current system time (ms)
55  * @return Current waveform value
56  */
57 double TestSignalGenerator::update(unsigned long currentTimeMs) {
58     if (_type == SIGNAL_MANUAL) {
59         return _manualValue;
60     }
61
62     // Calculate time elapsed since last update (seconds)
63     double deltaT = (_lastUpdateTime == 0) ? 0 :
64                     (currentTimeMs - _lastUpdateTime) / 1000.0;
65     _lastUpdateTime = currentTimeMs;
66
67     // Phase accumulation
68     if (_phaseRef) {
69         // Phase-locked to reference signal
70         _accumulatedPhase = _phaseRef->_accumulatedPhase + (
               _phase * PI / 180.0);
71     } else {
72         // Normal phase accumulation
73         _accumulatedPhase += 2.0 * PI * _frequency * deltaT;
74     }
```

```
75
76      // Normalize phase to [0, 2  ]
77      _accumulatedPhase = fmod(_accumulatedPhase, 2.0 * PI);
78      if (_accumulatedPhase < 0) _accumulatedPhase += 2.0 * PI;
79
80      // Generate waveform value
81      double normalizedPhase = _accumulatedPhase / (2.0 * PI);
82      double wave = _generateWave(normalizedPhase);
83      double angle = (_amplitude * wave) + _offset;
84
85      // Track transitions for square/pulse waves
86      bool newState = wave > 0.5;
87      if (newState != _currentState) {
88          _lastTransitionTime = currentTimeMs;
89          _currentState = newState;
90      }
91
92      return constrain(angle, 0.0, 50.0);
93  }
94
95  /**
96   * Generate waveform
97   *
98   * @param phase Normalized phase [0-1]
99   * @return Waveform value [0-1]
100  */
101 double TestSignalGenerator::_generateWave(double phase) {
102     switch (_type) {
103         case SIGNAL_CONSTANT:  return 1.0;
104         case SIGNAL_PULSE:
105         case SIGNAL_SQUARE:    return (phase < _dutyCycle) ? 1.0
               : 0.0;
106         case SIGNAL_SINE:      return 0.5 * (1.0 + sin(phase *
               2.0 * PI));
107         case SIGNAL_TRIANGLE:
108             if (phase < 0.25)  return phase * 4.0;
109             if (phase < 0.75)  return 2.0 - (phase * 4.0);
110             return -4.0 + (phase * 4.0);
111         case SIGNAL_SAWTOOTH:  return phase;
112         default:               return 0.0;
113     }
114 }
115
```

```
116 /**
117  * Debug print
118  *
119  * Prints current generator state
120  */
121 void TestSignalGenerator::debugPrint() {
122     Serial.print("Type:");
123     switch(_type) {
124         case SIGNAL_SINE: Serial.print("SINE"); break;
125         case SIGNAL_SQUARE: Serial.print("SQUARE"); break;
126         // ... other types ...
127     }
128     Serial.print("␣Phase:");
129     Serial.print(_accumulatedPhase * 180.0 / PI);
130     Serial.print("␣Freq:");
131     Serial.print(_frequency);
132     Serial.print("Hz␣State:");
133     Serial.println(_currentState ? "HIGH" : "LOW");
134 }
```

C.0.5 CommandHandler.h

```
1  /**
2   * Command Handler Interface
3   *
4   * Manages all channel control and serial communication functions
5   *
6   * Key Components:
7   * - DAC calibration structure
8   * - PID metrics tracking
9   * - Channel control structure
10  * - Function declarations for channel management
11  *
12  * Data Structures:
13  * - DACCalibrationPoint: Maps DAC values to physical
       measurements
14  * - PIDMetrics: Tracks control performance statistics
15  * - Channel: Contains all state and configuration for a control
       channel
16  */
17 #ifndef COMMANDHANDLER_H
18 #define COMMANDHANDLER_H
19
```

```
20 #include <Arduino.h>
21 #include <Wire.h>
22 #include <Adafruit_MCP4728.h>
23 #include "PID_v1.h"
24 #include "Config.h"
25 #include "TestSignalGenerator.h"
26
27 // Forward declarations
28 class PID;
29
30 // Global DAC object
31 extern Adafruit_MCP4728 dac;
32
33 /**
34  * DAC Calibration Point Structure
35  *
36  * Maps specific DAC values to measured physical quantities
37  * Used for accurate resistance/angle control
38  */
39 struct DACCalibrationPoint {
40   uint16_t dacValue;     // Raw DAC output value (0-4000)
41   double angle;          // Corresponding actuator angle (degrees)
42   double resistance;     // Corresponding resistance measurement (
       ohms)
43 };
44
45 /**
46  * PID Performance Metrics Structure
47  *
48  * Tracks control loop performance characteristics
49  */
50 struct PIDMetrics {
51   unsigned long responseTime95;      // Time to reach 95% of
       setpoint (ms)
52   unsigned long settlingTime;        // Time to stay within  5 %
       of setpoint (ms)
53   double steadyStateError;           // Average error after
       settling (ohms)
54   unsigned long lastUpdateTime;      // Last measurement timestamp
        (ms)
55   double peakError;                  // Maximum observed error (
       ohms)
56   bool targetReached;                // Flag if target was reached
```

```
57   bool isMeasuring;                  // Flag if currently
       measuring
58   unsigned long measurementStart;    // When measurement began (ms
       )
59   unsigned long loopTime;            // Time to complete full
       control loop ( s )
60   unsigned long feedbackLatency;     // Time from measurement to
       output ( s )
61   unsigned long lastLoopStart;       // For timing measurements (
        s )
62 };
63
64 /**
65  * Channel Control Structure
66  *
67  * Contains complete state and configuration for one control
      channel
68  */
69 struct Channel {
70   bool enabled;                     // Channel enable flag
71   double measuredR;                 // Current measured resistance
        (ohms)
72   double pidOutput;                 // PID controller output
       (0-4000)
73   double angleSetpoint;             // Desired angle (degrees)
74   double targetR;                   // Target resistance (ohms)
75   PID* controller;                  // PID controller instance
76   uint8_t v1Pin;                    // Shunt voltage ADC pin
77   uint8_t v3Pin;                    // Output voltage ADC pin
78   MCP4728_channel_t dacChannel;     // DAC channel
79   TestSignalGenerator signalGen;     // Test signal generator
80   SignalType currentSignalType;      // Current signal type
81   bool useTestSignal;               // Test signal enable flag
82   bool emergencyShutdown;           // Emergency state flag
83   bool actuatorDisconnected;         // Actuator connection state
84   bool inIdleMode;                  // Idle current mode flag
85   unsigned long lastActiveTime;      // Last active timestamp (ms)
86   uint16_t idleDacValue;            // DAC value for idle current
87   unsigned long lastValidReadingTime; // Last valid measurement (
       ms)
88   unsigned long disconnectionTimeout; // Disconnection timeout (
       ms)
89   PIDMetrics metrics;               // Performance metrics
```

```
};

// Global channel array
extern Channel channels[4];

// Function declarations
void initializeChannels();
void updateChannel(int chIndex, unsigned long masterTime);
void handleSerialCommands();
void printHelp();
double angleToResistance(double angle);
uint16_t resistanceToDAC(double resistance);
double measureResistance(uint8_t v1Pin, uint8_t v3Pin, double
    lastValidR);
void resetMetrics(int chIndex);
void updateMetrics(int chIndex);

#endif
```

C.0.6 CommandHandler.cpp

```
/**
 * Command Handler Implementation
 *
 * Implements all channel control and measurement functions
 *
 * Key Functionality:
 * - Channel initialization and PID setup
 * - Resistance measurement and calculation
 * - DAC value conversion
 * - Angle/resistance conversion
 * - Safety monitoring
 * - Performance metrics
 * - Serial command processing
 */
#include "CommandHandler.h"
#include <Adafruit_MCP4728.h>
#include "PID_v1.h"

// DAC calibration table - maps DAC values to resistance/angle
    measurements
const DACCalibrationPoint calibrationTable[] = {
```

```
21   {3000, 16.42, 21.70},  // DAC value, angle (deg), resistance (
       ohms)
22   {3200, 23.70, 20.57},
23   {3400, 29.87, 18.49},
24   {3600, 33.00, 16.90},
25   {3800, 34.68, 15.80}
26 };
27 const int calibrationPoints = sizeof(calibrationTable) / sizeof(
     calibrationTable[0]);
28
29 // Global channel array initialization
30 Channel channels[4] = {
31   {false, DEFAULT_TARGET_R, PID_MIN, 0, angleToResistance(0),
       nullptr, ADC_V1_PIN_A, ADC_V3_PIN_A, MCP4728_CHANNEL_A,
       TestSignalGenerator(), SIGNAL_MANUAL, false, false, false,
       false, 0, 0, 2000, {}},
32   {false, DEFAULT_TARGET_R, PID_MIN, 0, angleToResistance(0),
       nullptr, ADC_V1_PIN_B, ADC_V3_PIN_B, MCP4728_CHANNEL_B,
       TestSignalGenerator(), SIGNAL_MANUAL, false, false, false,
       false, 0, 0, 2000, {}},
33   {true, DEFAULT_TARGET_R, PID_MIN, 0, angleToResistance(0),
       nullptr, ADC_V1_PIN_C, ADC_V3_PIN_C, MCP4728_CHANNEL_C,
       TestSignalGenerator(), SIGNAL_MANUAL, false, false, false,
       false, 0, 0, 2000, {}},
34   {true, DEFAULT_TARGET_R, PID_MIN, 0, angleToResistance(0),
       nullptr, ADC_V1_PIN_D, ADC_V3_PIN_D, MCP4728_CHANNEL_D,
       TestSignalGenerator(), SIGNAL_MANUAL, false, false, false,
       false, 0, 0, 2000, {}}
35 };
36
37 /**
38  * Initialize all control channels
39  *
40  * Sets up PID controllers and signal generators for each enabled
       channel
41  * Configures:
42  * - PID parameters (Kp, Ki, Kd)
43  * - Signal generator settings
44  * - Safety parameters
45  */
46 void initializeChannels() {
47   for (int i = 0; i < 4; i++) {
48     if (channels[i].enabled) {
```

```
49        // Initialize PID controller
50        channels[i].controller = new PID(
51          &channels[i].measuredR,
52          &channels[i].pidOutput,
53          &channels[i].targetR,
54          DEFAULT_KP,
55          DEFAULT_KI,
56          DEFAULT_KD,
57          PID_DIRECTION
58        );
59
60        // Configure PID controller
61        channels[i].controller->SetMode(AUTOMATIC);
62        channels[i].controller->SetOutputLimits(PID_MIN, PID_MAX);
63        channels[i].controller->SetSampleTime(PID_SAMPLE_TIME);
64
65        // Configure signal generator
66        channels[i].signalGen.setAmplitude(35.0);  // 0-35 degrees
             range
67        channels[i].signalGen.setOffset(0);
68        channels[i].useTestSignal = true;
69        channels[i].signalGen.setFrequency(0.5);   // 0.5Hz
             frequency
70        channels[i].signalGen.setType(SIGNAL_PULSE);
71        channels[i].currentSignalType = SIGNAL_PULSE;
72        channels[i].signalGen.setPhase(0); // Reference phase
73
74        // Initialize safety parameters
75        channels[i].actuatorDisconnected = false;
76        channels[i].inIdleMode = false;
77        channels[i].lastValidReadingTime = 0;
78        channels[i].lastActiveTime = millis();
79        channels[i].disconnectionTimeout = DISCONNECTION_TIMEOUT_MS
             ;
80
81        resetMetrics(i);
82      }
83    }
84 }
85
86 /**
87  * Reset performance metrics for a channel
88  *
```

```
89  * @param chIndex Channel index (0-3)
90  */
91 void resetMetrics(int chIndex) {
92   Channel& ch = channels[chIndex];
93   ch.metrics.responseTime95 = 0;
94   ch.metrics.settlingTime = 0;
95   ch.metrics.steadyStateError = 0;
96   ch.metrics.lastUpdateTime = 0;
97   ch.metrics.peakError = 0;
98   ch.metrics.targetReached = false;
99   ch.metrics.isMeasuring = false;
100  ch.metrics.measurementStart = 0;
101  ch.metrics.loopTime = 0;
102  ch.metrics.feedbackLatency = 0;
103  ch.metrics.lastLoopStart = 0;
104 }
105
106 /**
107  * Update performance metrics for a channel
108  *
109  * Calculates:
110  * - Response time (95% of target)
111  * - Steady-state error
112  * - Peak error
113  *
114  * @param chIndex Channel index (0-3)
115  */
116 void updateMetrics(int chIndex) {
117   Channel& ch = channels[chIndex];
118   unsigned long currentTime = millis();
119   double error = fabs(ch.targetR - ch.measuredR);
120
121   // Track peak error
122   if (error > ch.metrics.peakError) {
123     ch.metrics.peakError = error;
124   }
125
126   // Start measurement when significant error detected
127   if (!ch.metrics.isMeasuring && error > 0.1 * ch.targetR) {
128     ch.metrics.isMeasuring = true;
129     ch.metrics.measurementStart = currentTime;
130     ch.metrics.targetReached = false;
131     ch.metrics.peakError = 0;
```

```
  }

  // Detect when 95% of target is reached
  if (!ch.metrics.targetReached && error <= 0.05 * ch.targetR) {
    ch.metrics.responseTime95 = currentTime - ch.metrics.
       measurementStart;
    ch.metrics.targetReached = true;
  }

  // Calculate steady-state error (exponential moving average)
  if (ch.metrics.targetReached) {
    ch.metrics.steadyStateError = 0.9 * ch.metrics.
       steadyStateError + 0.1 * error;
  }

  ch.metrics.lastUpdateTime = currentTime;
}

/**
 * Convert angle to resistance
 *
 * Uses linear approximation based on actuator characteristics
 *
 * @param angle Desired angle (degrees)
 * @return Corresponding resistance (ohms)
 */
double angleToResistance(double angle) {
    if (angle <= 0) {
        return MAX_RESISTANCE;
    }
    return constrain(-0.2840 * angle + 26.4085, MIN_RESISTANCE,
       MAX_RESISTANCE);
}

/**
 * Convert resistance to DAC value
 *
 * Uses calibration table with linear interpolation
 *
 * @param resistance Target resistance (ohms)
 * @return DAC value (0-4000)
 */
uint16_t resistanceToDAC(double resistance) {
```

```
172     if (resistance >= MAX_RESISTANCE) {
173         return DAC_MIN;
174     }
175
176     // Find calibration points for interpolation
177     int lowerIdx = -1, upperIdx = -1;
178     for (int i = 0; i < calibrationPoints; i++) {
179         if (calibrationTable[i].resistance >= resistance) {
180             lowerIdx = i;
181         }
182         if (calibrationTable[i].resistance <= resistance &&
               upperIdx == -1) {
183             upperIdx = i;
184         }
185     }
186
187     // Handle boundary conditions
188     if (lowerIdx == -1) return calibrationTable[0].dacValue;
189     if (upperIdx == -1) return calibrationTable[calibrationPoints
           -1].dacValue;
190
191     // Linear interpolation
192     double x0 = calibrationTable[lowerIdx].resistance;
193     double x1 = calibrationTable[upperIdx].resistance;
194     double y0 = calibrationTable[lowerIdx].dacValue;
195     double y1 = calibrationTable[upperIdx].dacValue;
196
197     uint16_t dacValue = (uint16_t)(y0 + (resistance - x0) * (y1 -
            y0) / (x1 - x0));
198     return constrain(dacValue, DAC_MIN, 4000);
199 }
200
201 /**
202  * Measure resistance from analog inputs
203  *
204  * Uses shunt voltage (V1) and output voltage (V3) measurements
205  *
206  * @param v1Pin Shunt voltage pin
207  * @param v3Pin Output voltage pin
208  * @param lastValidR Last valid resistance reading
209  * @return Measured resistance (ohms)
210  */
211 double measureResistance(uint8_t v1Pin, uint8_t v3Pin, double
```

```
    lastValidR) {
212     float v1 = analogRead(v1Pin) * (ADC_VREF/ADC_MAX_VALUE);
213     float v3 = analogRead(v3Pin) * (ADC_VREF/ADC_MAX_VALUE);
214     float current = v1 / SHUNT_R;
215 
216     if (current > MIN_OPERATING_CURRENT) {
217         double newR = ((((v3/10000.0)*30000.0)+v3)-v1)/current;
218         return constrain(newR, MIN_RESISTANCE, MAX_RESISTANCE);
219     }
220     return lastValidR;
221 }
222 
223 /**
224  * Update channel control loop
225  *
226  * Performs complete control cycle for one channel:
227  * 1. Voltage measurement
228  * 2. Disconnection detection
229  * 3. Safety checks
230  * 4. Idle mode management
231  * 5. Test signal handling
232  * 6. PID computation
233  * 7. Metrics update
234  * 8. DAC output
235  *
236  * @param chIndex Channel index (0-3)
237  * @param masterTime Current system time (ms)
238  */
239 void updateChannel(int chIndex, unsigned long masterTime) {
240     Channel& ch = channels[chIndex];
241     unsigned long loopStart = micros();
242 
243     // 1. Voltage measurement
244     unsigned long measureStart = micros();
245     float v1 = analogRead(ch.v1Pin) * (ADC_VREF/ADC_MAX_VALUE);
246     float v3 = analogRead(ch.v3Pin) * (ADC_VREF/ADC_MAX_VALUE);
247     float current = v1 / SHUNT_R;
248     unsigned long measureEnd = micros();
249 
250     // 2. Calculate DAC value early for disconnection detection
251     uint16_t dacValue;
252     if (ch.angleSetpoint <= 0) {
253         dacValue = DAC_MIN;
```

```
254     } else {
255         dacValue = map(ch.pidOutput, PID_MIN, PID_MAX, DAC_MIN,
              4000);
256         dacValue = constrain(dacValue, DAC_MIN, 4000);
257     }
258
259     // 3. Disconnection detection
260     if (current <= MIN_OPERATING_CURRENT) {
261         if ((dacValue > DAC_MIN || ch.pidOutput > PID_MIN * 1.1)
              && !ch.emergencyShutdown) {
262             if (ch.lastValidReadingTime == 0) {
263                 ch.lastValidReadingTime = masterTime;
264             } else if (masterTime - ch.lastValidReadingTime > ch.
                  disconnectionTimeout) {
265                 ch.actuatorDisconnected = true;
266                 Serial.print("ACTUATOR␣DISCONNECTED␣Ch");
267                 Serial.print("ABCD"[chIndex]);
268                 Serial.println("␣-␣No␣current␣detected␣with␣
                      active␣control␣signal");
269             }
270         } else {
271             ch.lastValidReadingTime = 0; // Reset if not trying
                  to drive current
272         }
273     } else {
274         ch.lastValidReadingTime = masterTime;
275         ch.actuatorDisconnected = false;
276     }
277
278     // Calculate resistance if current is sufficient
279     if (current > MIN_OPERATING_CURRENT) {
280         ch.measuredR = ((((v3/10000.0)*30000.0)+v3)-v1)/current;
281     } else {
282         ch.measuredR = MAX_RESISTANCE;
283     }
284
285     // 4. Safety checks
286     if (ch.measuredR < MIN_RESISTANCE || ch.actuatorDisconnected)
          {
287         if (!ch.emergencyShutdown) {
288             ch.emergencyShutdown = true;
289             ch.inIdleMode = false;
290             dac.setChannelValue(ch.dacChannel, DAC_MIN);
```

```
291             if (ch.controller != nullptr) {
292                 ch.controller->SetMode(MANUAL);
293             }
294             ch.pidOutput = PID_MIN;
295             Serial.print("EMERGENCY␣SHUTDOWN␣Ch");
296             Serial.print("ABCD"[chIndex]);
297             if (ch.actuatorDisconnected) {
298                 Serial.println("␣-␣Actuator␣disconnected");
299             } else {
300                 Serial.print("␣-␣Resistance␣");
301                 Serial.print(ch.measuredR);
302                 Serial.println("  ␣(below␣minimum)");
303             }
304         }
305         return;
306     }
307     else if (ch.emergencyShutdown) {
308         if (ch.measuredR >= (MIN_RESISTANCE +
               EMERGENCY_HYSTERESIS) && !ch.actuatorDisconnected) {
309             ch.emergencyShutdown = false;
310             if (ch.controller != nullptr) {
311                 ch.controller->SetMode(AUTOMATIC);
312             }
313             Serial.print("Safety␣reset␣Ch");
314             Serial.print("ABCD"[chIndex]);
315             Serial.println("␣-␣Resistance␣recovered");
316         }
317         else {
318             return;
319         }
320     }
321
322     // 5. Idle mode management
323     bool shouldBeIdle = (ch.pidOutput <= PID_MIN * 1.1) &&
324                         !ch.emergencyShutdown &&
325                         !ch.actuatorDisconnected &&
326                         (masterTime - ch.lastActiveTime >
                              IDLE_CURRENT_TIMEOUT);
327
328     if (shouldBeIdle && !ch.inIdleMode) {
329         ch.inIdleMode = true;
330         ch.idleDacValue = resistanceToDAC(angleToResistance(0)) +
              (IDLE_V1_TARGET * 1000);
```

```
331         dac.setChannelValue(ch.dacChannel, constrain(ch.
                idleDacValue, DAC_MIN, DAC_MAX));
332         Serial.print("Channel "); Serial.print("ABCD"[chIndex]);
333         Serial.println(" entering idle mode (maintenance current)
                ");
334     }
335     else if (!shouldBeIdle && ch.inIdleMode) {
336         ch.inIdleMode = false;
337         ch.lastActiveTime = masterTime;
338     }
339
340     // 6. Idle current regulation
341     if (ch.inIdleMode) {
342         float v1Error = IDLE_V1_TARGET - v1;
343
344         if (fabs(v1Error) > IDLE_V1_TOLERANCE) {
345             ch.idleDacValue += v1Error * 100;
346             ch.idleDacValue = constrain(ch.idleDacValue, DAC_MIN,
                    DAC_MAX);
347             dac.setChannelValue(ch.dacChannel, ch.idleDacValue);
348         }
349
350         ch.metrics.lastUpdateTime = masterTime;
351         return;
352     }
353
354     // 7. Test signal handling
355     if (ch.useTestSignal) {
356       double angle = ch.signalGen.update(masterTime);
357       ch.angleSetpoint = angle;
358       ch.targetR = angleToResistance(angle);
359     }
360
361     // 8. PID computation
362     ch.controller->Compute();
363
364     // 9. Update performance metrics
365     unsigned long beforeMetrics = micros();
366     updateMetrics(chIndex);
367     ch.metrics.feedbackLatency = micros() - measureStart;
368
369     // 10. Apply DAC output
370     dac.setChannelValue(ch.dacChannel, dacValue);
```

```
371
372     // Update active time if output is significant
373     if (ch.pidOutput > PID_MIN * 1.1) {
374         ch.lastActiveTime = masterTime;
375     }
376
377     // Calculate full loop time
378     ch.metrics.loopTime = micros() - loopStart;
379
380     // 11. CSV Serial output
381     Serial.print(masterTime); Serial.print(",");
382     Serial.print("ABCD"[chIndex]); Serial.print(",");
383     Serial.print(ch.angleSetpoint,1); Serial.print(",");
384     Serial.print(ch.targetR,1); Serial.print(",");
385     Serial.print(ch.measuredR,1); Serial.print(",");
386     Serial.print(ch.pidOutput); Serial.print(",");
387     Serial.print(dacValue); Serial.print(",");
388     Serial.print(ch.inIdleMode ? "IDLE" : "ACTIVE"); Serial.print
        (",");
389     Serial.print(ch.metrics.loopTime); Serial.print(",");
390     Serial.print(ch.metrics.feedbackLatency); Serial.print(",");
391     Serial.print(measureEnd - measureStart); Serial.print(",");
392
393     // Performance metrics
394     if (ch.metrics.isMeasuring || ch.metrics.targetReached) {
395       if (ch.metrics.responseTime95 > 0) {
396          Serial.print(ch.metrics.responseTime95);
397       } else {
398           Serial.print("--");
399       }
400       Serial.print(",");
401       Serial.print(ch.metrics.steadyStateError, 3); Serial.print(
            ",");
402       Serial.print(ch.metrics.peakError, 3);
403     } else {
404         Serial.print("--,--,--");
405     }
406   Serial.println();
407 }
408
409 /**
410  * Handle serial commands
411  *
```

```
412  * Processes incoming serial commands with format:
413  * [channel][command][value]
414  *
415  * Example: 'cm15' = Channel C manual 15
416  */
417 void handleSerialCommands() {
418   if (!Serial.available()) return;
419
420   char channelChar = Serial.read();
421   int ch = -1;
422
423   if (channelChar >= 'a' && channelChar <= 'd') {
424     ch = channelChar - 'a';
425     channels[ch].lastActiveTime = millis();
426     channels[ch].inIdleMode = false;
427
428     if (Serial.available()) {
429       char cmd = Serial.read();
430       float value = 0;
431
432       switch(cmd) {
433         case 'm': // Manual angle setpoint
434           if (Serial.available()) {
435             value = Serial.parseFloat();
436             channels[ch].angleSetpoint = value;
437             channels[ch].targetR = angleToResistance(value);
438             channels[ch].useTestSignal = false;
439             resetMetrics(ch);
440             Serial.print("Channel "); Serial.print((char)('A'+ch)
                  );
441             Serial.print(" manual angle set to "); Serial.print(
                  value,1);
442             Serial.print("  ("); Serial.print(channels[ch].
                  targetR,1);
443             Serial.println("  )");
444           }
445           break;
446
447         case 't': // Signal type
448           if (Serial.available()) {
449             int type = Serial.parseInt();
450             if (type >= SIGNAL_MANUAL && type <= SIGNAL_SQUARE) {
451               channels[ch].currentSignalType = (SignalType)type;
```

```
452             channels[ch].signalGen.setType((SignalType)type);
453             channels[ch].useTestSignal = (type != SIGNAL_MANUAL
                    );
454             resetMetrics(ch);
455             Serial.print("Channel␣"); Serial.print((char)('A'+
                    ch));
456             Serial.print("␣signal␣type␣set␣to␣");
457             switch(type) {
458               case SIGNAL_SINE: Serial.println("SINE"); break;
459               case SIGNAL_TRIANGLE: Serial.println("TRIANGLE");
                      break;
460               case SIGNAL_SAWTOOTH: Serial.println("SAWTOOTH");
                      break;
461               case SIGNAL_PULSE: Serial.println("PULSE"); break
                    ;
462               case SIGNAL_SQUARE: Serial.println("SQUARE");
                    break;
463               default: Serial.println("MANUAL"); break;
464             }
465           }
466         }
467         break;
468
469       case 'f': // Frequency
470         if (Serial.available()) {
471           value = Serial.parseFloat();
472           channels[ch].signalGen.setFrequency(value);
473           resetMetrics(ch);
474           Serial.print("Channel␣"); Serial.print((char)('A'+ch)
                  );
475           Serial.print("␣frequency␣set␣to␣"); Serial.print(
                  value);
476           Serial.println("Hz");
477         }
478         break;
479
480       case 'a': // Amplitude
481         if (Serial.available()) {
482           value = Serial.parseFloat();
483           channels[ch].signalGen.setAmplitude(value);
484           resetMetrics(ch);
485           Serial.print("Channel␣"); Serial.print((char)('A'+ch)
                  );
```

```
486             Serial.print("␣amplitude␣set␣to␣"); Serial.println(
                    value);
487           }
488           break;
489
490         case 'o': // Offset
491           if (Serial.available()) {
492             value = Serial.parseFloat();
493             channels[ch].signalGen.setOffset(value);
494             resetMetrics(ch);
495             Serial.print("Channel␣"); Serial.print((char)('A'+ch)
                    );
496             Serial.print("␣offset␣set␣to␣"); Serial.println(value
                    );
497           }
498           break;
499
500         case 'p': // Phase
501           if (Serial.available()) {
502             value = Serial.parseFloat();
503             channels[ch].signalGen.setPhase(value);
504             resetMetrics(ch);
505             Serial.print("Channel␣"); Serial.print((char)('A'+ch)
                    );
506             Serial.print("␣phase␣set␣to␣"); Serial.print(value);
507             Serial.println("  ");
508           }
509           break;
510
511         case 'd': // Duty cycle
512           if (Serial.available()) {
513             value = Serial.parseFloat();
514             value = constrain(value, 0.0, 1.0);
515             channels[ch].signalGen.setDutyCycle(value);
516             resetMetrics(ch);
517             Serial.print("Channel␣"); Serial.print((char)('A'+ch)
                    );
518             Serial.print("␣duty␣cycle␣set␣to␣"); Serial.println(
                    value);
519           }
520           break;
521
522         case 's': // Toggle test signal
```

```
523           channels[ch].useTestSignal = !channels[ch].
                 useTestSignal;
524           resetMetrics(ch);
525           Serial.print("Channel␣"); Serial.print((char)('A'+ch));
526           Serial.print("␣test␣signal␣");
527           Serial.println(channels[ch].useTestSignal ? "ON" : "OFF
                 ");
528           break;
529
530         case 'e': // Emergency reset
531           if (channels[ch].emergencyShutdown) {
532             channels[ch].emergencyShutdown = false;
533             resetMetrics(ch);
534             Serial.print("Channel␣"); Serial.print((char)('A'+ch)
                   );
535             Serial.println("␣emergency␣reset");
536           }
537           break;
538
539         case 'r': // Phase reference
540           if (Serial.available()) {
541             char refChar = Serial.read();
542             if (refChar >= 'a' && refChar <= 'd') {
543               int refCh = refChar - 'a';
544               channels[ch].signalGen.setPhaseReference(&channels[
                     refCh].signalGen);
545               resetMetrics(ch);
546               Serial.print("Channel␣"); Serial.print((char)('A'+
                     ch));
547               Serial.print("␣phase␣referenced␣to␣channel␣");
548               Serial.println((char)('A'+refCh));
549             }
550             else {
551               channels[ch].signalGen.setPhaseReference(nullptr);
552               resetMetrics(ch);
553               Serial.print("Channel␣"); Serial.print((char)('A'+
                     ch));
554               Serial.println("␣phase␣reference␣cleared");
555             }
556           }
557           break;
558
559         case '?': // Help
```

```
560           printHelp();
561           break;
562
563         case 'i': // Idle status
564           Serial.print("Channel "); Serial.print((char)('A'+ch));
565           Serial.print(" idle status: ");
566           Serial.println(channels[ch].inIdleMode ? "ACTIVE" : "
                INACTIVE");
567           break;
568
569         default:
570           Serial.println("Unknown command");
571           printHelp();
572       }
573     }
574   }
575   while (Serial.available()) Serial.read();
576 }
577
578 /**
579  * Print command help
580  *
581  * Displays all available commands and their formats
582  */
583 void printHelp() {
584   Serial.println("timestamp,channel,angle,targetR,measuredR,
        pidOutput,dacValue,mode,loopTime,feedbackLatency,measureTime
        ,responseTime95,steadyStateError,peakError");
585   Serial.println("Command format: [channel][command][value]");
586   Serial.println("Example: 'cm15' = Channel C manual 15  ");
587   Serial.println("Commands:");
588   Serial.println("m=manual angle, t=signal type (0-6)");
589   Serial.println("f=frequency(Hz), a=amplitude(  )");
590   Serial.println("o=offset(  ), p=phase(  ), d=duty(0-1)");
591   Serial.println("s=toggle signal, e=emergency reset");
592   Serial.println("rX=phase ref (X=a-d), i=idle status");
593   Serial.println("?=help");
594   Serial.println("Signal types: 0=Manual, 1=Const, 2=Pulse");
595   Serial.println("3=Sine, 4=Triangle, 5=Sawtooth, 6=Square");
596   Serial.println("System automatically maintains idle current
        when inactive");
597   Serial.println("Performance metrics are automatically shown
        when available:");
```

```
598   Serial.println("Resp95% = Time to reach 95% of target (ms)");
599   Serial.println("SSErr = Steady state error (  )");
600   Serial.println("PeakErr = Maximum observed error (  )");
601   Serial.println("Timing metrics:");
602   Serial.println("Loop = Full control loop execution time ( s )")
        ;
603   Serial.println("Feedback = Measurement to output latency ( s )"
        );
604   Serial.println("Measure = Analog measurement time ( s )");
605 }
```

Apéndice D

Manual de comandos del generador de señales

```
==========================================
          TEST SIGNAL COMMANDS GUIDE
==========================================

 COMMAND FORMAT:
  [channel][command1][value1][command2][value2]...
  • No spaces between commands
  • All in one continuous line

 CHANNEL SELECTION:
  a = Channel A       c = Channel C
  b = Channel B       d = Channel D


             COMMAND REFERENCE

 COMMAND    PARAMETER     EXAMPLE

 t[0-6]     Signal type   at3 (Sine)
            0=Manual      bt6 (Square)
            1=Constant
            2=Pulse
            3=Sine
            4=Triangle
            5=Sawtooth
            6=Square

 f[Hz]      Frequency     af1 (1Hz)
```

```
                          bf0.5 (0.5Hz)

 a[deg]      Amplitude     ca10 (±10°)
                           da5 (±5°)

 o[deg]      Offset        ao5 (+5°)
                           bo-2 (-2°)

 p[deg]      Phase         cp90 (90°)
             (0-360)       dp180 (180°)

 d[0.0-1.0] Duty cycle     ad0.3 (30%)
                           bd0.7 (70%)

 r[a-d]      Phase ref     cra (ref A)
 r           (clear ref)   dr (clear)

 s           Toggle signal as (toggle A)

 m[deg]      Manual angle  bm45 (45°)

 e           Emerg. reset  ce (reset C)

 ?           Show help     ?


 QUICK EXAMPLES:
1. Channel A: 1Hz Sine wave ±10° enabled
   → at3f1a10s

2. Channel B: Square wave @2Hz, 25% duty, synced to A
   → bt6f2d0.25ras

3. Channel C: Manual 45° angle
   → cm45

4. Channel D: Triangle wave 0.5Hz ±15° 90° phase
   → dt4f0.5a15p90s

 TROUBLESHOOTING:
• Ensure no spaces between commands
• Use periods for decimals (0.5 not 0,5)
• Check serial monitor for error messages
```

```
• Emergency shutdowns auto-report in monitor

==========================================
```

Apéndice E

Manual de diseño de controlador

Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Mecatrónica

Design and Tuning of a PID Controller for Bioinspired SMA-Based Robotic Fish Actuators at CAR UPM-CSIC

Author:
David José Rodríguez Camacho
dajoroca64@gmail.com

July 2025

Contents

0.1 Introduction

This manual provides a step-by-step guide for designing a PID (Proportional-Integral-Derivative) controller for Shape Memory Alloy (SMA) actuators, from system identification to practical implementation. The process includes:

- System Identification: Deriving a mathematical model of the SMA actuator.
- Controller Design: Tuning the PID parameters using the Ziegler-Nichols method.
- Implementation: Integrating the controller with an ESP32-S3 and DAC (MCP4728) using Arduino IDE.

0.2 Electronics

The electronic platform, designed by William Coral, combines an ESP32-S3 microcontroller with an MCP4728 12-bit DAC to create a sophisticated SMA control and monitoring system. The microcontroller drives up to four shape memory alloy actuators independently through the DAC's analog outputs, with current regulation achieved by programming digital values (0-4095) via I2C communication. A 3.3 Ω reference resistor (R5) serves as the current sensing element for the system.

Each electrical measurement (V1 and V3) connects to dedicated GPIO pins on the ESP32-S3, as specified in Table 1. These connections enable real-time monitoring of the SMA's dynamic resistance through the measurement framework established in equations 1, 2 and 3. Importantly, while the DAC values determine the excitation current, the resistance calculation operates independently through these GPIO measurement channels.

For a more detailed explanation of the GPIO mapping and the functionality of this board, refer to [1]. This architecture provides researchers with both precise current control through the DAC and simultaneous resistance monitoring via the GPIO measurement system, enabling comprehensive characterization of SMA behavior during actuation cycles.

- $$I_{\text{SMA}} = \frac{V1}{R5} \tag{1}$$
- $$V_{\text{SMA}} = (\frac{V3}{R7} * R6 + V3) - V1 \tag{2}$$
- $$R_{\text{SMA}} = \frac{V_{\text{SMA}}}{I_{\text{SMA}}} \tag{3}$$

Table 1: ESP32-S3 GPIO Pins for Measuring V1 and V3 from MCP4728 Channels.

Channel	Measurement	GPIO
A	V1	2
	V3	1
B	V1	4
	V3	3
C	V1	6
	V3	5
D	V1	17
	V3	7

0.3 Sytem Identification

To develop an effective PID controller for our SMA system, we first need to characterize the actuator's response through careful experimentation. The process involves applying variable electrical inputs while monitoring the real-time behavior. We stimulate the actuator using random DAC values within a specific operational range, typically from 3000 up to the maximum value, with each pulse lasting between 0.5 to 1.5 seconds. This random variation helps capture the system's dynamics across different excitation levels and durations. All testing must be performed underwater to maintain proper thermal conditions, as the water provides essential heat dissipation that significantly affects the SMA's response characteristics.

During testing, we deliberately exceed the actuator's nominal current ratings by applying a maximum current between 0.7A and 1.0A. This practical current limit, which often surpasses manufacturer specifications, becomes our operational ceiling for control applications. Throughout the experiment, we continuously monitor the actuator's dynamic resistance as our primary feedback metric. This resistance measurement provides crucial insight into the SMA's state transformation during both heating and cooling phases. The collected data of input currents (via DAC values) versus resistance responses forms the foundation for deriving our transfer function, it is also important to record the timestamp of each measurement to be able to determine the sampling time.

This experimental approach allows us to properly model the SMA's behavior for PID controller design while working within the actuator's true operational limits under realistic conditions. The random input pattern prevents system settling into predictable states, while the underwater testing maintains consistent thermal boundary conditions. The resulting characterization captures the full dynamics of the phase transformation, enabling effective controller development.

The measured response signal should exhibit a characteristic shape matching the profile shown in Figure 1. A distinct resistance drop must be observable during the phase transformation, with the magnitude of this decrease falling within

the range of 7% to 10% of the peak resistance value. This specific resistance variation serves as the key indicator of successful SMA activation and provides the essential dynamic response needed for effective PID control implementation.

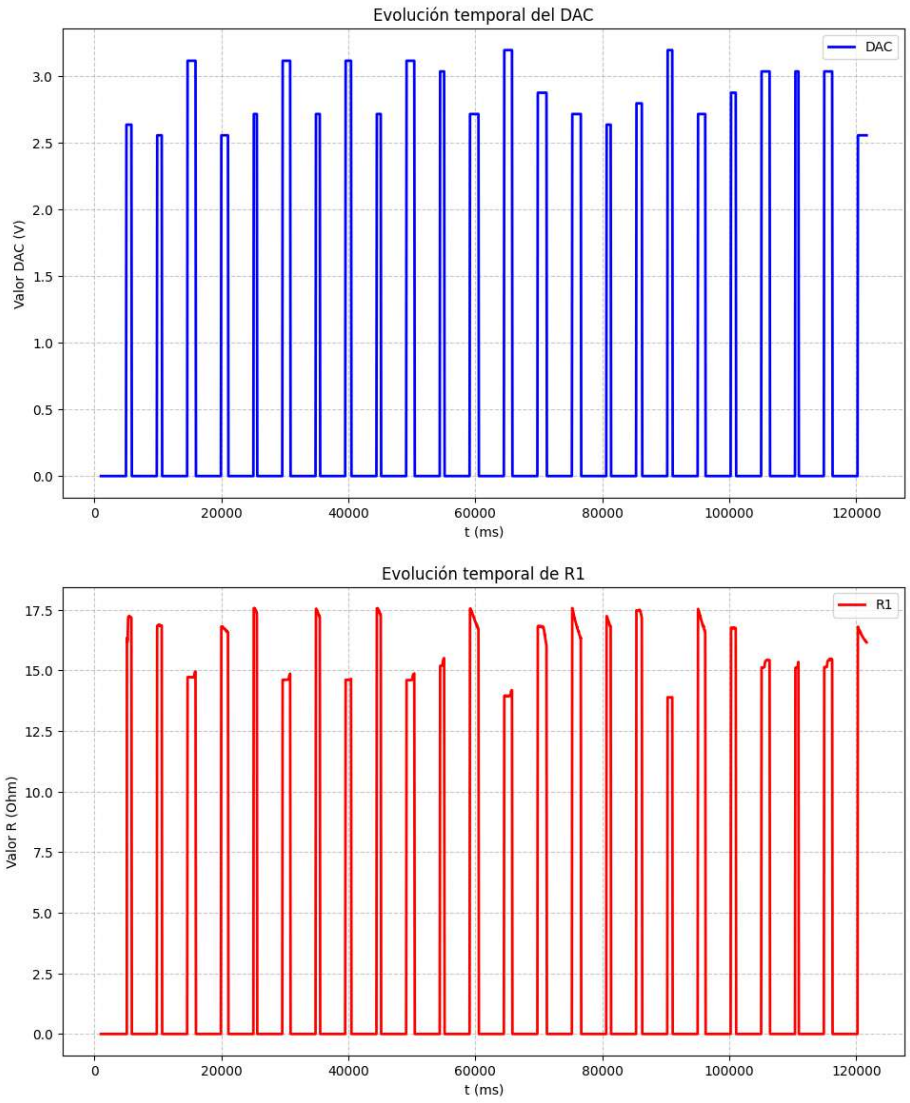


Figure 1: Resulting input and output signals.

We first organize the experimental data into MATLAB-compatible formats (.csv is recommended). The recorded measurements should be imported as column vectors, with each vector maintaining exactly the same name as its corresponding column header in the original CSV file. This naming consistency

is crucial for proper variable referencing throughout our analysis.

Once the data vectors are properly loaded into MATLAB's workspace, we configure the system representation using these three essential components:

- Input vector (containing our excitation signals)
- Output vector (recording the system responses)
- Sampling time parameter (defining our measurement intervals)

We can achive this with the following code:

```
deltaT = (Time(2)-Time(1))/1000;
SMA = iddata(R,IN,deltaT);
SMA.inputname = {'DAC_V'};
SMA.outputname = {'Resistance'};
```

Once the input, output, and time vectors are prepared in MATLAB, we can proceed with system identification by running either the ident or systemIdentification command, depending on the installed MATLAB version. This command requires the System Identification Toolbox to be available. Upon execution, the tool will open an interactive window matching the interface shown in Figure 2.

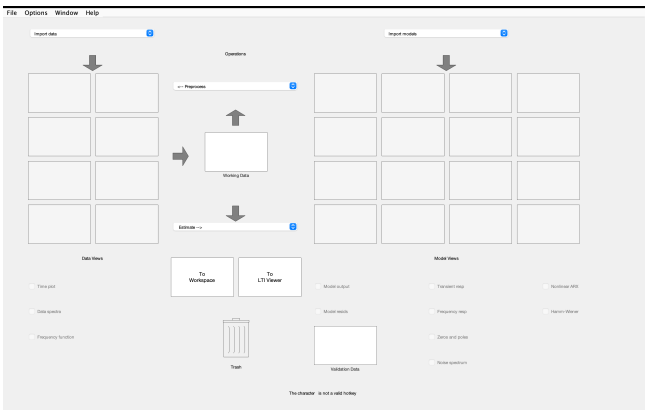


Figure 2: System Identification window.

To load our experimental data, we first click Import Data in the System Identification Tool window and select the Data Object option. This opens a configuration dialog where we specify our workspace variable by entering "SMA" in the Workspace Variable field - exactly matching our data vector's name. The tool automatically loads all the system characteristics we configured earlier. Before completing the import, we simply need to adjust the Starting Time parameter to zero, then finalize the process by clicking the Import button. This ensures our data is properly formatted with correct timing references for the subsequent system identification analysis.

We begin by accessing the Preprocess menu and selecting the Select Range option. This action opens a graphical interface displaying our experimental data. Using the mouse, we perform two square-drag selections to create our data partitions:

First, we select approximately 70%of the data points and name this range SMAe (our estimation dataset), confirming the selection by clicking Insert. Next, we repeat the process for the remaining 30% of data points, naming this range SMAv (our validation dataset) and clicking Insert again.

After closing this window with the Close button, we complete the setup by:

- Dragging the SMAe dataset to the Working Data box
- Dragging the SMAv dataset to the Validation Data box

The System Identification Tool interface should now match the configuration displayed in Figure 3, with both datasets properly assigned to their respective roles.

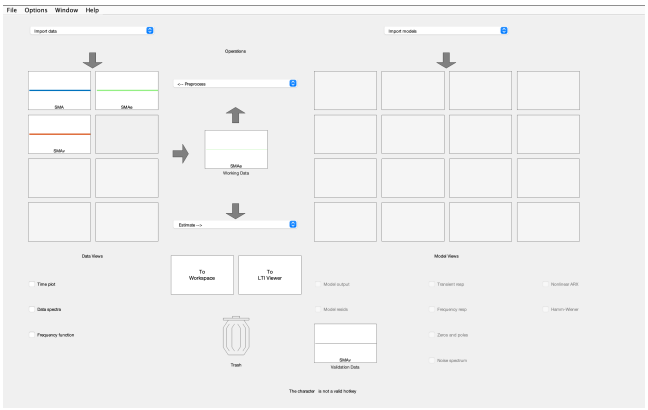


Figure 3: System Identification window after data preprocessing.

To begin modeling our system, we select the Estimate option and choose Transfer Function as our estimation method. For this analysis, we recommend starting with simple models and gradually increasing complexity - begin by generating multiple transfer functions with varying numbers of poles and zeros. Through empirical testing with this particular dataset, we've found that a model structure with one pole and one zero provides satisfactory results. Ensure the Continuous Time option remains selected throughout this process.

After specifying the desired model order (number of poles and zeros), clicking Estimate initiates the calculation. While the tool provides an initial fit value upon completion, we recommend disregarding this estimation as it's based solely on the training data (SMAe). For a more reliable validation:

- 1. Select the transfer function models you wish to evaluate (indicated by bold colors and thick lines in the interface)

- 2. Click the Model Output option to generate a comparative plot
- 3. Analyze the overlay between the model output and your validation data (SMAv)
- 4. Note the calculated fit percentage

For practical applications a fit above 70% is acceptable but ideally we should target models achieving $> 80\%$ fit with validation data. We must carefully balance model complexity against fit quality. While highly complex systems can achieve excellent fit percentages, this often comes at the cost of practical utility. The most effective models maintain simplicity while still delivering sufficient accuracy for control purposes.

Once you've identified a satisfactory transfer function, finalize the process by dragging the selected model to the To Workspace box for subsequent use in your control system design and implementation.

0.4 PID Tuning

The program documented in Appendix A generates initial PID gain values using the Ziegler-Nichols method [2]. To utilize this tool, you need to input the numerator coefficients of your selected transfer function into the variable 'num' and the denominator coefficients into 'den'. The method requires specifying two key design parameters: the maximum allowable overshoot percentage and the desired rise time to reach 95% of the final value. There exists a fundamental trade-off in controller design - achieving faster response times requires accepting greater overshoot, as the control strategy prioritizes speed over absolute precision.

After obtaining the PID gains, it's advisable to conduct a basic simulation in SIMULINK (Figure 4) to verify the expected performance. While controller discretization would typically be necessary for microcontroller implementation, the relatively slow frequency response of SMA actuators often allows the system to be treated as continuous, provided the sampling rate and PID calculation loop operate at sufficiently high speeds relative to the mechanical response times.

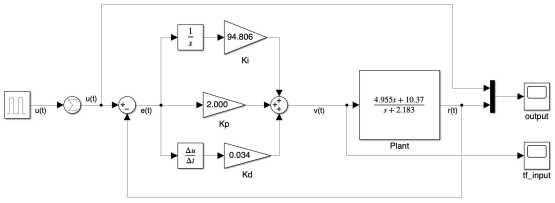


Figure 4: PID Simulation in Simulink.

This tunning just gives you a starting point, it might be necessary to slightly change the gains manually in the practical implementation.

0.5 Arduino Implementation

The control system implementation utilizes the Arduino PID Library by Brett Beauregard, with comprehensive documentation available in the cited reference and downloadable from the specified GitHub repository. This library was specifically modified to accommodate our hardware configuration, particularly in the PID_v1.cpp file where line 28 was altered to increase the output upper limit from 255 to 4000. This critical adjustment was necessary because the ESP32-S3 microcontroller and MCP4728 DAC combination operates with 12-bit resolution (0-4095 range), contrasting with the Arduino Uno's native 8-bit architecture (0-255 range). The modification fully leverages the system's enhanced resolution while prudently maintaining a safety margin below the theoretical maximum value of 4095, preserving all original library functionality while optimizing performance for our specific hardware setup.

The system incorporates a sophisticated programmable signal generator capable of producing various waveform types including sinusoidal, triangular, square, and sawtooth patterns, along with a manual input mode accessible through the Arduino IDE Serial Monitor. Each waveform type is mathematically defined with fully configurable parameters - amplitude (A), frequency (f), phase shift (ϕ), duty cycle (D for square waves), and vertical offset (C). These mathematical formulations are carefully designed to constrain all output signals within the physically meaningful range of 0 to the user-specified amplitude, respecting the actuator's mechanical limitations while providing flexible control options.

Critical safety mechanisms protect the SMA actuator through two primary functions. The first involves open-circuit detection, which activates after a configurable timeout period (defaulting to 5 seconds) when the system detects continued DAC activation without corresponding resistance measurements. The second protection mechanism prevents overheating by continuously monitoring resistance and automatically reducing DAC output whenever values fall below 14 Ω, maintaining this reduced output state until resistance recovers above 15 Ω. These safeguards operate independently of the main control loop, providing robust protection for both the actuator and associated circuitry.

A standby current of approximately 0.45 A is maintained to keep the actuator in a preheated state, ensuring rapid response capability when activation signals are received. The system's comprehensive data logging functionality captures and formats key operational parameters including rise time to 95% of reference value, steady-state error, maximum input values, current DAC outputs, setpoints, and measured resistances into structured CSV output for both real-time monitoring and post-analysis.

The core operational process begins with continuous voltage measurements (V1 and V3) across all active channels, from which actuator resistances are

calculated using the specified equations with the known 3.3 Ω reference resistor value. When test signals are active, the system iteratively updates reference values every 5 milliseconds, feeding these to the PID controller and adjusting DAC outputs accordingly while logging all relevant performance metrics. During inactive periods, the system maintains continuous resistance monitoring, ready to respond to serial command inputs. The complete implementation, including fully documented code (except the library files) and a comprehensive manual for the test signal generator commands, is available in the Appendix (B and C) section for reference and further development.