Universidad de Valladolid

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

Desarrollo de una Spydercam

Autor:

Núñez Muñoz, Aitana

Tutor:

Zalama Casanova, Eduardo

Departamento de Ingeniería de Sistemas y Automática

Valladolid, septiembre 2025.

Contenido

Resumen

La presente memoria describe el diseño e implementación de una Spydercam (cámara tipo araña) de bajo costo, basada en una arquitectura distribuida con microcontroladores ESP32. El objetivo es que una plataforma sea capaz de desplazarse en un espacio tridimensional mediante el control coordinado de cuatro motores RMD-L-9025-35T. Para lograr esta tarea, se ha desarrollado una arquitectura distribuida basada en microcontroladores ESP32, con un nodo maestro que coordina el movimiento y cuatro nodos esclavos encargados del control individual de los motores. La comunicación entre nodos se realiza utilizando el protocolo inalámbrico ESP-NOW, que permite una trasmisión de datos sin depender de una red WiFi convencional.

Se han diseñado e integrado los módulos de control, la lógica de posicionamiento, y un algoritmo de interpolación cartesiana para el movimiento sincronizado de la plataforma suspendida. Se ha logrado establecer una comunicación fiable y eficiente entre los nodos, así como una respuesta estable de los motores. La plataforma suspendida es capaz de ejecutar movimientos suaves y controlados, demostrando la viabilidad técnica del sistema y su potencial de aplicación en entornos educativos, audiovisuales y de investigación.

Los resultados obtenidos confirman que es posible replicar funciones avanzadas de cámaras aéreas comerciales utilizando componentes accesibles, tecnologías abiertas y protocolos eficientes. Este proyecto abre la puerta a futuras mejoras como sistemas de estabilización para mejorar las tomas de la cámara o sincronización con dispositivos.

Capítulo 1: Motivación y objetivos

1.1 Spydercam

Una Spydercam es un sistema de cámara suspendida por cables de alta resistencia, controlado electrónicamente, que permite realizar movimientos tridimensionales precisos y repetibles a través de grandes espacios, como sets de filmación, estadios o escenarios (ver Figura 1). Utiliza una cámara montada en un *gimbal* estabilizado, el cual se mueve mediante motores computarizados que enrollan y desenrollan los cables según trayectorias programadas desde un software especializado, lo que permite capturar tomas aéreas dinámicas con una fluidez cinematográfica. A diferencia de otros sistemas de cámara como grúas, drones, la Spydercam puede realizar movimientos complejos con una estabilidad superior y sin interrupciones, siendo ideal para escenas de acción, efectos visuales, transmisiones deportivas, conciertos y producciones que requieren máxima precisión en la repetición de tomas. Su diseño modular permite adaptarla no solo al movimiento de cámaras, sino también al de utilería, luces e incluso actores, ofreciendo soluciones personalizadas según las necesidades de cada proyecto. Reconocida como una de las herramientas más avanzadas en la industria audiovisual, la Spydercam destaca por su versatilidad, seguridad y capacidad para llevar la narrativa visual a otro nivel [1][18].

Figura 1. Ejemplo de Spydercam [1]

A fecha de hoy hay varios tipos de Spydercam:

Spydercam LIGHT

El Spydercam LIGHT está diseñado para operar en espacios reducidos, como estudios de televisión o salas de noticias. Su diseño compacto y componentes de menor tamaño facilitan la instalación y adaptación en entornos con limitaciones de espacio. Gracias a su construcción liviana y al uso de cabrestantes potentes, esta versión puede alcanzar una altura de vuelo superior a la de otros sistemas comparables, lo que la convierte en una solución eficiente para producciones en interiores.

Spydercam FIELD

El sistema Spydercam FIELD está compuesto por cuatro cabrestantes, una plataforma rodante, una estación de control y un conjunto completo de funciones avanzadas de software y seguridad. Diseñado para cubrir áreas de hasta 250 x 250 metros, es ideal para grandes espacios como estadios o estudios de gran escala. Está equipado con cámaras de resolución 1080p o 4K y lentes gran angular, lo que lo convierte en la elección preferida para eventos deportivos y producciones que requieren cobertura aérea de alta calidad.

Spydercam MINI

El Spydercam MINI cubre superficies de hasta 60 x 60 metros, y gracias a su tamaño reducido, permite una instalación sencilla y rápida, especialmente en entornos de estudio. Su diseño compacto y eficiente lo hace ideal para producciones en interiores donde los requerimientos son diferentes a los de aplicaciones deportivas al aire libre, ofreciendo una solución versátil y fácil de adaptar.

Spydercam BOW

El Spydercam BOW funciona como un sistema de desplazamiento punto a punto, pero con la capacidad adicional de controlar también la altura de vuelo. Esto permite que los movimientos de la cámara no estén limitados a un nivel fijo, sino que puedan ajustarse dinámicamente durante la operación. El sistema incorpora un cabezal remoto totalmente estabilizado, garantizando imágenes fluidas y estables en todo momento, y ampliando las posibilidades creativas en producciones tanto en interiores como exteriores [1].

1.2 Motivación del trabajo

En las últimas décadas, el crecimiento de la robótica y la automatización ha dejado de ser una tendencia exclusiva de la industria para convertirse en una realidad cotidiana también en la educación, el entretenimiento y la investigación.

En este contexto, el desarrollo de sistemas robotizados por cable es un excelente ejemplo de integración tecnológica avanzada. Este proyecto propone el diseño y desarrollo de una Spydercam, construida a partir de una red de microcontroladores ESP32 conectados entre sí mediante el protocolo ESP-NOW y motores RMD-L-9025-35T. Se emplea una arquitectura maestro–esclavo en la que un nodo central coordina las acciones de cuatro nodos esclavos, cada uno encargado de controlar un motor. Este enfoque permite reproducir los movimientos tridimensionales típicos de una Spydercam profesional, pero con componentes más baratos y un diseño abierto, replicable en entornos educativos o de investigación.

Desde el punto de vista técnico, este proyecto representa un reto atractivo: requiere resolver problemas de control distribuido, sincronización inalámbrica, cinemática inversa, tensado de cables y diseño mecánico. Todo ello utilizando hardware y software accesibles, lo cual impulsa el ingenio y la optimización de recursos.

Permite abordar múltiples áreas del conocimiento: control de motores, programación en tiempo real, comunicación entre nodos, procesamiento de datos, integración de sensores, diseño 3D y electrónica de potencia. La robótica tiene la capacidad de generar aprendizaje significativo, porque combina teoría con resultados tangibles.

Este tipo de sistemas también tiene aplicaciones más allá de su uso audiovisual. El mismo concepto puede adaptarse a tareas de exploración en interiores, vigilancia de espacios industriales, asistencia en operaciones logísticas o como plataforma experimental en laboratorios de inteligencia artificial. Incluso puede evolucionar hacia sistemas autónomos que usen visión artificial o algoritmos de navegación para seguir objetivos o mapear entornos.

La creación de este sistema, además, genera una base sólida para futuras mejoras. Esta Spydercam puede evolucionar hacia soluciones más complejas mediante la incorporación de interfaces gráficas, control remoto mediante aplicaciones móviles, seguimiento automático de objetivos, sensores de posición más precisos o algoritmos de control predictivo. También se presta a integrar cámaras de análisis de movimiento, sistemas de realidad aumentada o plataformas de retrasmisión en tiempo real.

Este proyecto es, en esencia, una prueba de que con los recursos adecuados y una visión integradora, es posible crear tecnología útil, potente y al servicio del aprendizaje.

Además, este proyecto puede tener un impacto significativo en el proceso de enseñanza-aprendizaje en disciplinas técnicas. A menudo, los estudiantes enfrentan dificultades para conectar la teoría con la práctica, lo cual genera desmotivación o una comprensión superficial de los conceptos. Una plataforma como la Spydercam proporciona un hilo conductor para múltiples asignaturas, donde se pueden integrar conocimientos de física (dinámica, tensiones, vectores), matemáticas (trigonometría, interpolación, álgebra lineal), informática (estructuras de datos, programación concurrente, protocolos de comunicación) y diseño técnico (modelado 3D, análisis estructural).

Por otro lado, el uso del protocolo ESP-NOW y de microcontroladores ESP32 como base tecnológica del sistema representa una decisión estratégica que refuerza el objetivo de accesibilidad. Estos dispositivos, a pesar de su bajo coste, cuentan con una notable capacidad de procesamiento, conectividad inalámbrica integrada y una comunidad activa de desarrolladores. Esto permite no solo un diseño robusto, sino también fácilmente escalable y mantenible.

En resumen, la realización de este proyecto surge de una motivación doble: por un lado, el interés personal por explorar los desafíos técnicos asociados a los sistemas robóticos distribuidos, y por otro, el compromiso por desarrollar una herramienta educativa accesible que acerque la robótica avanzada a más personas. Diseñar una Spydercam de bajo coste representa no solo un reto ingenieril estimulante, sino también una oportunidad para fomentar el aprendizaje activo, reducir barreras tecnológicas y contribuir a una formación más inclusiva, creativa y conectada con los valores de la sociedad actual. Este trabajo busca demostrar que la innovación no depende únicamente de grandes presupuestos, sino también de la voluntad de compartir el conocimiento y hacerlo útil para otros.

1.3 Objetivos

1.2.1 Objetivo principal

Desarrollar un sistema funcional que busca controlar una plataforma móvil suspendida mediante cables, capaz de moverse en el espacio tridimensional controlando la longitud de los cuatro cables que la sostienen. Cada cable está vinculado a un motor controlado por un ESP32 esclavo, que recibe comandos del ESP32 maestro. El maestro interpreta las coordenadas deseadas y calcula cuánto debe enrollar o desenrollar cada motor para

llegar a la posición objetivo, empleando una comunicación inalámbrica eficiente y sincronizada.

1.2.2 Objetivos específicos

- Diseñar y construir la estructura física de la Spydercam, incluyendo el soporte y el anclaje de los motores.
- Implementar un sistema maestro-esclavo usando el protocolo ESP-NOW entre los ESP32, donde el maestro calcule las órdenes de movimiento y los esclavos controlen individualmente cada motor.
- Desarrollar una interfaz de usuario que permita manejar y operar con el movimiento de la plataforma.
- Desarrollar una interfaz de simulación para observar el comportamiento esperado de la estructura.
- Sincronizar el movimiento de los motores, de forma que la plataforma se desplace suavemente en el espacio tridimensional, manteniendo la estabilidad.
- Probar y ajustar el sistema de control, evaluando el comportamiento dinámico del conjunto bajo diferentes condiciones de carga y velocidad.

1.4 Impacto de mejora en la sociedad

El desarrollo de este sistema basado en microcontroladores ESP32 y motores con control distribuido representa una contribución en términos de democratización tecnológica. Al utilizar componentes de bajo costo y tecnologías abiertas como ESP-NOW, el proyecto permite acercar soluciones avanzadas de movimiento robótico a sectores que tradicionalmente no podrían acceder a tecnologías similares debido a sus altos costos o barreras técnicas. Esto incluye instituciones educativas, pequeños estudios de producción, etc.

Impacto social y contribución a los Objetivos de Desarrollo Sostenible (ODS)

El desarrollo de este sistema con tecnología abierta y de bajo coste contribuye directamente a los objetivos de la Agenda de 2030 para el Desarrollo Sostenible, al democratizar el acceso a tecnologías avanzadas, fomentar la educación técnica y promover la innovación inclusiva. En particular, este proyecto se alinea con varios ODS de manera significativa:

- **ODS4 – Educación de calidad**: Este sistema representa una herramienta práctica para el aprendizaje interdisciplinar en campos como la robótica, la programación, la electrónica y el control automático. Su uso en entornos educativos facilita la enseñanza basada en proyectos, potencia el pensamiento crítico y promueve el desarrollo del estudio de las áreas de ciencia y tecnología.
- **ODS 9 – Industria, innovación e infraestructura**: El proyecto fomenta una cultura de innovación accesible al desarrollar un sistema complejo con recursos

económicos y tecnológicos. Utilizar una arquitectura descentralizada con ESP32 y ESP-NOW demuestra cómo es posible construir sistemas inteligentes sin depender de grandes infraestructuras, facilitando la adopción de soluciones tecnológicas en regiones con recursos limitados.

- **ODS 12 – Producción y consumo responsables:** La reutilización de componentes electrónicos comunes, el enfoque modular del diseño y la optimización energética del sistema (baja latencia, bajo consumo) contribuyen a un uso más eficiente y responsable de los recursos tecnológicos. Además, al ser un sistema replicable [2].

En resumen, este proyecto no solo alcanza sus objetivos técnicos, sino que también actúa como catalizador para una transformación social y educativa más equitativa, colaborando activamente con los valores de sostenibilidad, igualdad e inclusión tecnológica que promueve la Agenda 2030.

1.5 Estructura de la memoria

Esta memoria se ha estructurado de forma lógica y progresiva para reflejar de manera clara el desarrollo del proyecto, desde su concepción hasta su implementación y validación. Cada capítulo aborda una etapa específica del trabajo, permitiendo al lector comprender tanto el contexto general como los detalles técnicos implicados. El objetivo de esta organización es facilitar la compresión del proceso seguido en cada fase del diseño y la construcción del sistema.

A continuación, se presenta una descripción general de los próximos capítulos, cada uno aborda un aspecto específico del proyecto, desde la motivación inicial hasta la documentación técnica final, permitiendo así un recorrido lógico y coherente por todas las etapas del proceso de diseño, implementación y validación del sistema.

El proyecto se estructura de la siguiente forma:

En el capítulo 2 se realiza una mención de los sistemas robotizados por cables existentes. Se revisan distintas configuraciones empleadas en la actualidad, destacando aspectos como el uso de cables tensados para el movimiento, su eficiencia en grandes volúmenes de trabajo y su aplicación en sectores como la industria, el entretenimiento o la investigación.

A partir de este estudio, se identifican las principales diferencias entre estos sistemas y el desarrollado en el presente proyecto, tanto en términos de diseño como de funcionalidad. Esta comparación permite justificar las decisiones tomadas durante el desarrollo y resaltar las particularidades e innovaciones del sistema propuesto.

En el capítulo 3 se aborda los fundamentos teóricos, todos los cálculos necesarios para el correcto desarrollo del proyecto.

En el capítulo 4 se detallan las explicaciones de todos los componentes del sistema y de las piezas diseñadas.

En el capítulo 5 se aborda el estudio de opciones para el desarrollo del sistema incluyendo comparativas técnicas y la elección para el diseño final y su implementación. Además, de una explicación detallada de su montaje.

El capítulo 6 describe el proceso de desarrollo de la interfaz gráfica del sistema, detalla la evolución desde los primeros prototipos hasta el desarrollo de la aplicación final y explica de forma breve su uso.

El capítulo 7 recopila y analiza todas las pruebas realizadas durante el desarrollo del proyecto. Se exponen los resultados obtenidos, evaluando el funcionamiento del sistema y validando su comportamiento en relación con los objetivos planteados.

El capítulo 8 presenta un resumen general de los aspectos más relevantes tratados a lo largo de la memoria, destacando los logros alcanzados y el cumplimiento de los objetivos propuestos. Además, se proponen una serie de posibles mejoras y líneas de trabajo futuras orientadas a optimizar y ampliar las capacidades del sistema desarrollado.

Un apartado con la bibliografía donde se puede encontrar todas las fuentes consultadas para la realización del proyecto.

Por último, los anejos donde se incluye toda la documentación técnica complementaria relacionada con el desarrollo del proyecto. Se incorporavel esquema eléctrico del sistema, así como una explicación detallada del código implementado y del funcionamiento de las interfaces desarrolladas.

Capítulo 2: Análisis de sistemas robotizados por cables

En las últimas décadas, la robótica y los sistemas mecatrónicos han experimentado un crecimiento vertiginoso, impulsado por la aparición de componentes electrónicos más potentes y accesibles. Entre los avances más significativos se encuentran los microcontroladores de bajo coste y los protocolos de comunicación inalámbrica que permiten la creación de arquitecturas distribuidas. Este ecosistema ha facilitado la experimentación y el desarrollo de soluciones complejas en entornos educativos y de investigación, sin necesidad de recurrir a sistemas industriales de alto presupuesto.

En términos generales, los sistemas de movimiento tridimensional suspendido, como los robots por cable, tienen una larga trayectoria en la ingeniería. Han sido utilizados en ámbitos tan diversos como el mantenimiento industrial, la rehabilitación física o la filmación aérea.

En el contexto audiovisual, la Spydercam se han consolidado como soluciones profesionales para la filmación aérea dinámica. En el ámbito deportivo, estas plataformas se utilizan ampliamente en estadios de fútbol, pistas de tenis o eventos de gran escala, donde permiten capturar tomas cenitales o desplazamientos rápidos a lo largo del campo. En cine, su capacidad para seguir secuencias dinámicas desde ángulos inusuales las convierte en herramientas expresivas de gran valor [1].

En cuanto al uso de ESP32, en los últimos años se ha demostrado como una plataforma idónea para sistemas embebidos distribuidos. Sus capacidades de comunicación Wi-Fi y Bluetooth, junto con su potencia de cálculo y bajo coste, han hecho que se utilice ampliamente en proyectos que requieren coordinación entre nodos. En particular, el protocolo ESP-NOW permite una comunicación directa entre dispositivos sin necesidad de infraestructura de red, lo que lo hace ideal para sistemas maestro–esclavo como el planteado en este proyecto. A diferencia del Wi-Fi convencional, ESP-NOW presenta baja latencia, consumo reducido y una configuración simplificada, aunque con ciertas limitaciones en la cantidad de nodos y el ancho de banda [5].

Un aspecto diferencial de este trabajo es precisamente la combinación de tecnologías existentes en un sistema replicable, funcional y accesible. A pesar de los múltiples proyectos de robot de cables documentados en, pocos abordan el control distribuido inalámbrico en tiempo real con una arquitectura maestro–esclavo optimizado. Además, la mayoría se centran en pruebas de concepto bidimensionales, sin cubrir el volumen tridimensional de movimiento ni integrar motores de grado semi-profesional.

Por tanto, este Trabajo Fin de Grado se justifica plenamente como una propuesta innovadora y necesaria, que combina herramientas existentes para resolver un problema específico: diseñar una plataforma robótica suspendida, funcional y

económica, que pueda utilizarse con fines pedagógicos, experimentales o incluso prácticos en espacios interiores.

2.1 Historia

El sistema Spydercam fue desarrollado por Jens C. Peters, fundador de CCSystems Inc., empresa que creó en el año 2000 con el propósito específico de diseñar un sistema de suspensión para cámaras basado en cables, capaz de ofrecer un movimiento tridimensional completo. A diferencia de otros sistemas, este permitía desplazar la cámara no solo de forma horizontal y lateral, sino también verticalmente, abriendo nuevas posibilidades en la captura de imágenes dinámicas. La primera prueba exitosa del sistema tuvo lugar en 2003, en un amplio salón de actos en Cartínia, Austria, y dos años más tarde, en 2005, se utilizó por primera vez en una producción televisiva en ese mismo país. Con el fin de perfeccionar el diseño y realizar mejoras técnicas, Jens C. Peters colaboró ese mismo año con la empresa alemana PMT Professional Motion Technology GmbH, especializada en tecnología de cámaras. Gracias a esta alianza, el sistema Spydercam fue finalmente optimizado y lanzado al mercado [15].

2.2 Robots de cables

COGIRO

COGIRO es un prototipo avanzado de robot paralelo accionado por cables, desarrollado por TECNALIA. Este sistema está diseñado para manejar estructuras complejas en un gran volumen de trabajo, combinando precisión, flexibilidad y capacidad de carga.

Cuenta con 6 grados de libertad controlados mediante ocho cables servocontrolados, lo que proporciona un movimiento altamente fluido y preciso (ver Figura 2). Su estructura mide aproximadamente 15x11x6 metros, con una carga máxima gestionable de 500kg y una repetibilidad submilimétrica(<1mm). Esta diseñado bajo una configuración tipo “grúa”; todos los puntos de anclaje de los cables se sitúan alrededor del volumen de trabajo, usando la gravedad para mantener la tensión [19].

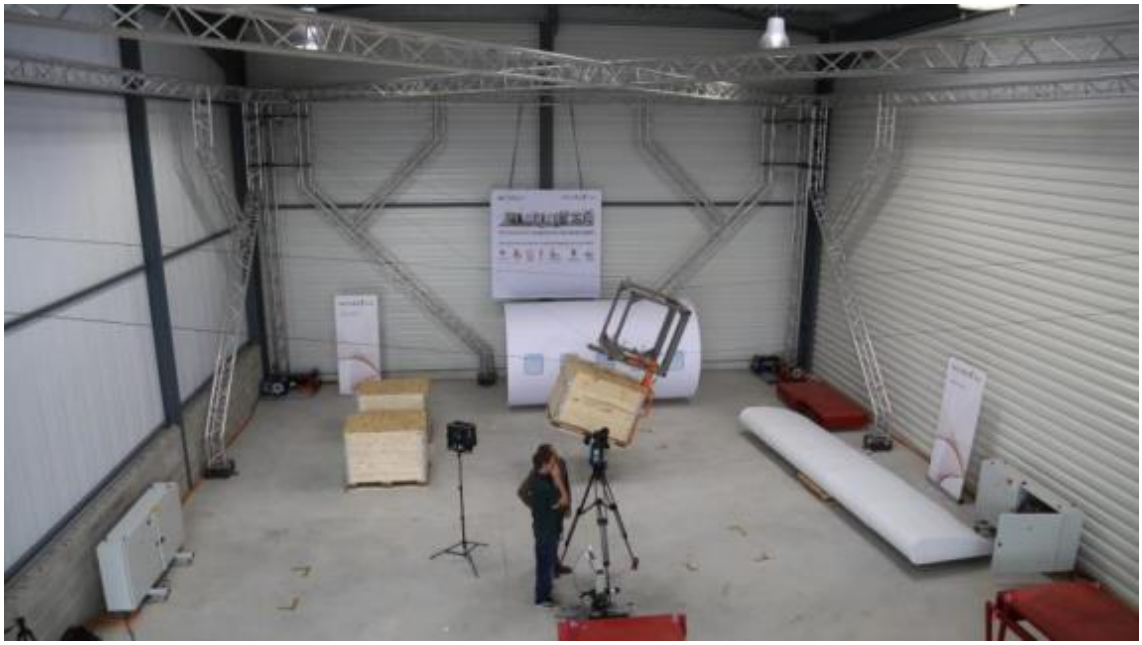


Figura 2. COGIRO – Robot de cables desarrollado por TECNALIA [19]

Aunque es concebido para aplicaciones industriales y de construcción, comparte principios fundamentales con una Spydercam ya que ambos se basan en la tecnología de robots paralelos accionados por cables, permitiendo movimientos precisos en tres dimensiones dentro de un volumen de trabajo amplio y controlado.

ROBOCRANE

El RoboCrane, desarrollado por el Laboratorio Nacional de Estándares e Innovación (NIST) de EE. UU., es uno de los ejemplos de robot accionado por cables (ver Figura 3). Su diseño imita la plataforma Stewart, pero en lugar de utilizar miembros rígidos, emplea un ensamblaje octaédrico de cables que aportan seis grados de libertad (traslación en x, y, z y rotación). Gracias a esta configuración, el RoboCrane puede pasar herramientas pesadas o delicadas con una precisión milimétrica, moviéndolas libremente en espacios tridimensionales amplios [3].

Figura 3. ROBOCRANE – Robot paralelo desarrollado por NIST

Al igual que la Spydercam, el RoboCrane se basa en la arquitectura de robots paralelos accionados por cables, permitiendo movimientos controlados en el espacio tridimensional; aunque sus aplicaciones son industriales, comparte los mismos principios fundamentales de suspensión y control.

Robot de cables para la limpieza de fachadas

El sistema consiste en una cesta equipada con un rodillo escamoteable, un sistema pulverizador de agua y un escurridor (ver Figura 4). La cesta se encuentra suspendida por dos grúas motorizadas, lo que permite posicionarla en cualquier punto de la superficie de la fachada. El control del sistema está distribuido en tres microcontroladores ESP32 que se comunican entre sí de manera inalámbrica utilizando el protocolo ESPNOW. La operación se realiza mediante un mando a distancia, que permite tanto el control manual de la cesta como la programación de trayectorias automatizadas a través de comandos G-code [4].

Figura 4. Robot de cables para limpieza de fachadas [4]

De manera similar, el sistema Spydercam diseñado comparte varios principios con el robot de limpieza de fachadas, especialmente en cuanto a la arquitectura distribuida de control y la comunicación inalámbrica. En este caso, también se emplean cuatro microcontroladores ESP32 que se comunican mediante el protocolo ESPNOW, lo que permite una coordinación eficiente entre los módulos. Al igual que en el robot de limpieza, el sistema de la Spydercam permite posicionar una carga suspendida con alta precisión dentro de un espacio tridimensional, controlando el movimiento mediante grúas motorizadas. Ambos proyectos demuestran la viabilidad de utilizar soluciones de bajo costo y tecnologías inalámbricas para sistemas de posicionamiento y control en aplicaciones prácticas y exigentes.

Cassino robot CDPR

Se han desarrollado sistemas CDPR cooperativos compuestos por múltiples grúas móviles, abordando desafíos como la localización conjunta, la evasión de obstáculos y el control adaptativo de la orientación de la carga útil. Por ejemplo, un CDPR modular desplegado mediante un rover, diseñado para operar a gran escala en entornos complejos. Este sistema modular, conocido como sistema de seguimiento Cassino (ver Figura 5), es una solución de bajo costo utilizada para evaluación y tareas de inspección en terrenos difíciles, además de contar con aplicaciones en procedimientos clínicos de diagnóstico y rehabilitación [17].

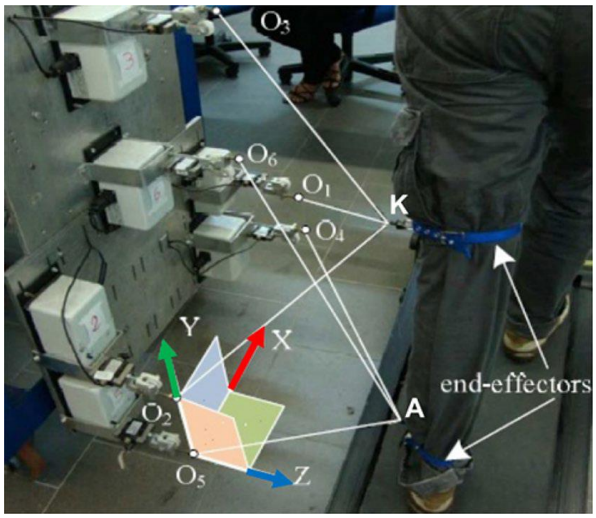


Figura 5. Robot de cables Cassino [17]

Capítulo 3: Fundamentos teóricos

3.1 Modelado Cinemático del Sistema

En esta primera etapa del desarrollo teórico se abordará el modelado cinemático del sistema, el cual constituye la base para el análisis y control del movimiento de la plataforma. El objetivo principal de este apartado es describir matemáticamente la relación entre la posición del punto de trabajo del robot y las longitudes de los cables que lo soportan y controlan su movimiento.

Para ello, se realizará un estudio detallado de la configuración geométrica del sistema, considerando las coordenadas del espacio de trabajo, los puntos de anclaje de los cables y las restricciones cinemáticas impuestas por la estructura del robot. Esta parametrización permitirá no solo identificar con precisión la posición y orientación de la plataforma en función de las variables de entrada, sino también sentar las bases para el posterior diseño de estrategias de control y simulación del sistema.

El modelado cinemático es esencial en robots de tipo cableado, ya que la precisión en el cálculo de las longitudes de los cables es determinante para garantizar la estabilidad, exactitud y seguridad durante la operación del robot. Por tanto, este análisis representa un componente clave dentro del desarrollo del proyecto.

3.1.1 Planteamiento

La idea del sistema es la que se plantea en la figura 6, un sistema de 4 motores unidos cada uno por cable a un nodo central. Para el correcto control de los motores deben recibir un dato de posición en grados y/o un dato de velocidad en revoluciones por minuto (rpm). En este apartado se va a explicar el correcto cálculo para poder controlar los motores dado una posición en el espacio.

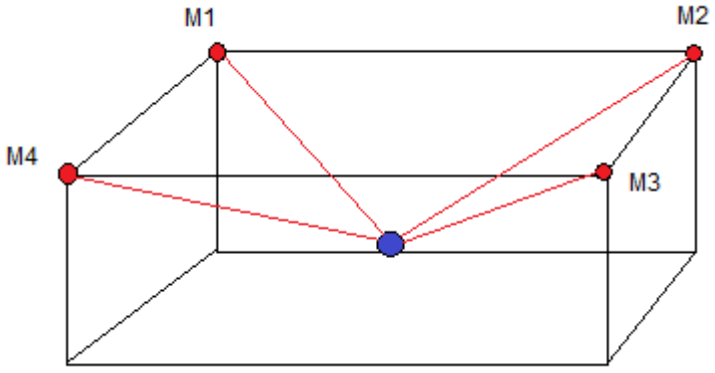


Figura 6. Planteamiento del sistema Spydercam

Los problemas fundamentales en el análisis de sistemas mecánicos, especialmente en robótica y animación son la cinemática directa y la cinemática inversa.

3.1.2 Cinemática inversa

La cinemática inversa consiste en calcular la posición del motor necesario para lograr llegar al punto en el espacio deseado. Consideramos un sistema con cuatro motores ubicados en posiciones fijas en el espacio, cada uno anclado en un punto diferente, definido por sus coordenadas en el plano. La plataforma móvil debe ubicarse en una posición dada por un punto en el espacio con coordenadas (x, y, z). Como observamos en la figura 7.

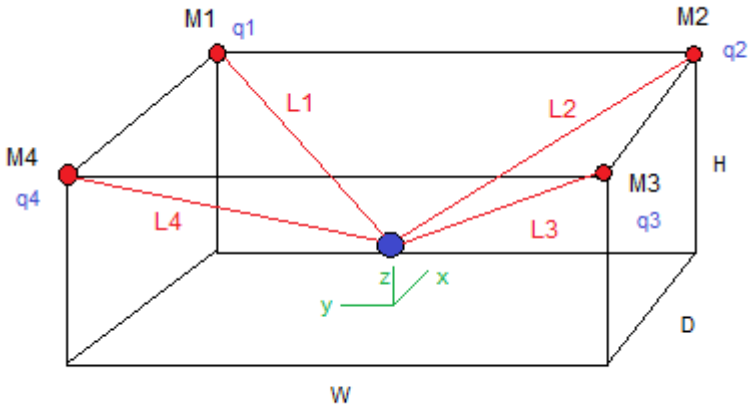


Figura 7. Planteamiento Spydercam

Mx – Motor x (1 a 4)

Lx – Longitud del cable x

qx – Posición de los motores en grados

H – Altura de los motores

D – distancia en metros de M3 a M2

W – distancia en metros de M4 a M3

R – Radio del motor

P – Distancia del centro de la plataforma a la sujeción del cable en la plataforma.

Para simplificar las ecuaciones se ha supuesto un paralelepípedo ideal, pero se puede realizar para una zona que no sea así. Lo único es en vez de usar las variables D y W para todas las longitudes en cada ecuación de la longitud correspondiente se deben sustituir D y W por la distancia en x e y al que va a corresponder al (0, 0, 0) del sistema.

Para calcular la longitud de cable necesaria para cada motor, se calcula la distancia entre la posición del motor y la posición deseada de la plataforma, es decir, el módulo del vector que conecta ambos puntos.

Dado que cada motor tiene una posición base ajustada por ciertos parámetros y dimensiones físicas del sistema, las longitudes se calculan con la siguiente fórmula general:

$$L_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} \qquad (3.1.1)$$

Donde (x_i, y_i, z_i) representa la posición fija en el espacio del anclaje del motor i, (x, y, z) es la posición objetivo de la plataforma.

Para los cálculos hay que tener en cuenta la posición exacta donde se sujetan los cables en la plataforma, y eso lo definimos con la variable P, donde se define la distancia del centro de la plataforma al anclaje.

En el sistema se consideran las dimensiones físicas, por tanto, las longitudes para los cuatro motores se calculan de la siguiente forma:

$$L1 = \sqrt{(x + P - \frac{W}{2})^2 + (y - P + \frac{D}{2})^2 + (z - H)^2} \qquad (3.1.2)$$

$$L2 = \sqrt{(x - P + \frac{W}{2})^2 + (y - P + \frac{D}{2})^2 + (z - H)^2} \qquad (3.1.3)$$

$$L3 = \sqrt{(x - P + \frac{W}{2})^2 + (y + P - \frac{D}{2})^2 + (z - H)^2} \qquad (3.1.4)$$

$$L4 = \sqrt{(x - P + \frac{W}{2})^2 + (y + P - \frac{D}{2})^2 + (z - H)^2} \qquad (3.1.5)$$

La posición de referencia del sistema se define en (0, 0, 0), que corresponde con a la configuración donde todos los motores están en posición cero.

La longitud inicial L_o del cable para esta posición se cálcula con la misma fórmula anterior, usando la fórmula de cualquiera de los motores, ya que se espera que las longitudes iniciales sean iguales para todos, si no fuera un paralelepípedo habría que calcular cada longitud inicial con su fórmula correspondiente.

$$L_o = \sqrt{(0 - P + \frac{W}{2})^2 + (0 + P - \frac{D}{2})^2 + (0 - H)^2} \qquad (3.1.6)$$

Esta longitud inicial sirve como base para calcular las variaciones de cable necesarias para alcanzar cualquier otra posición.

Para mover la plataforma, cada motor debe enrollar o desenrollar cable para ajustar su longitud desde la posición inicial hasta la nueva longitud objetivo. Esta variación se calcula como:

$$\Delta L_i = L_i - L_o \qquad (3.1.7)$$

Donde ΔL_i es la cantidad de cable que debe moverse (positiva para desenrollar, negativa para enrollar)

Con esta variación, se calcula el ángulo q_i que debe girar el motor para lograr dicha variación, considerando el radio R del tambor alrededor del cual se enrolla el cable:

$$q_i = \frac{\Delta L_i}{2\pi R} \cdot 360 \qquad (3.1.8)$$

Este ángulo q_i indica el desplazamiento angular necesario para que el motor modifique la longitud del cable en la cantidad requerida.

Con este procedimiento, el sistema puede controlar con precisión la posición de la plataforma o, si se desea, calcular los ángulos necesarios en función de la posición que el usuario quiera alcanzar.

3.1.3 Cinemática directa

La cinemática directa tiene como objetivo determinar la posición espacial de la plataforma móvil a partir de los ángulos de los motores y los parámetros geométricos del sistema. En otras palabras, dados los cuatro ángulos de los motores se busca calcular la posición del punto en el espacio. El proceso consiste en trasformar coordenadas angulares en coordenadas cartesianas:

Q(q1, q2, q3, q4) --> P(x, y, z)

Cada motor enrolla o desenrolla cable sobre un tambor de radio R. El ángulo de giro q_i (en grados) se relaciona directamente con el cambio en la longitud del cable ΔL_i:

$$\Delta L_i = \frac{q_i}{360} \cdot 2\pi R \qquad (3.1.8)$$

La longitud total de cable para el motor i es:

$$L_i = L_o + \Delta L_i \qquad (3.1.9)$$

Siendo L_o la misma longitud inicial que en el cálculo de cinemática inversa.

Cada motor se encuentra en una posición fija conocida (x_i, y_i, z_i) . La plataforma móvil, cuya posición (x, y, z) se desea encontrar, se conecta mediante cables a esos puntos.

La longitud del cable L_i es la distancia euclidiana entre el motor y la plataforma:

$$L_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} \quad (3.1.10)$$

Elevamos al cuadrado para facilitar el análisis:

$${L_i}^2 = (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 \quad (3.1.11)$$

Como hay cuatro motores, se obtiene un sistema de cuatro ecuaciones no lineales:

$$\begin{cases} {L_1}^2 = (x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 \\ {L_2}^2 = (x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 \\ {L_3}^2 = (x - x_3)^2 + (y - y_3)^2 + (z - z_3)^2 \\ {L_4}^2 = (x - x_4)^2 + (y - y_4)^2 + (z - z_4)^2 \end{cases} \quad (3.1.12)$$

Aquí, las únicas incógnitas son (x, y, z), la posición de la plataforma.

Para resolver este sistema de ecuaciones no lineales, se eliminan los términos cuadráticos restando una ecuación de referencia, en este caso se ha restado la primera, a las demás:

$${L_2}^2 - {L_1}^2 =$$

$$(x - x_2)^2 - (x - x_1)^2 + (y - y_2)^2 - (y - y_1)^2 + (z - z_2)^2 - (z - z_1)^2 \quad (3.1.13)$$

Esta resta elimina los términos x^2, y^2, z^2, resultando en una ecuación lineal. Al repetir este procedimiento con la tercera y cuarta ecuación, se obtiene un sistema lineal de tres ecuaciones con tres incógnitas:

$$A \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} = b \quad (3.1.14)$$

Donde:

$$A = \begin{pmatrix} 2(x_1 - x_2) & 2(y_1 - y_2) & 2(z_1 - z_2) \\ 2(x_1 - x_3) & 2(y_1 - y_3) & 2(z_1 - z_3) \\ 2(x_1 - x_4) & 2(y_1 - y_4) & 2(z_1 - z_4) \end{pmatrix}$$

$$b = \begin{pmatrix} {L_1}^2 - {L_2}^2 + {x_2}^2 - {x_1}^2 + {y_2}^2 - {y_1}^2 + {z_2}^2 - {z_1}^2 \\ {L_1}^2 - {L_3}^2 + {x_3}^2 - {x_1}^2 + {y_3}^2 - {y_1}^2 + {z_3}^2 - {z_1}^2 \\ {L_1}^2 - {L_4}^2 + {x_4}^2 - {x_1}^2 + {y_4}^2 - {y_1}^2 + {z_4}^2 - {z_1}^2 \end{pmatrix}$$

3.1.4 Calculo trayectorias

La trayectoria lineal 3D

La generación de una trayectoria lineal en el espacio tridimensional (3D) consiste en trazar un camino recto entre un punto de inicio (posición actual) y un punto de destino.

El primer paso para llevar a cabo esta trayectoria es calcular la interpolación lineal entre ambos puntos. La interpolación lineal en 3D permite obtener una serie de posiciones intermedias que definen el recorrido, manteniendo una transición suave y proporcional entre el inicio y el destino.

Las variables por utilizar en los cálculos son las siguientes:

N : Número total de pasos.

$\vec{P}_o = (x_o, y_o, z_o)$: Posición inicial.

$\vec{P}_i = (x_i, y_i, z_i)$: Posición intermedia, donde $i \in \{1, 2, \dots, N\}$.

$\vec{P}_f = (x_f, y_f, z_f)$: Posición final.

Para cada posición intermedia $\vec{P}_i$, se calcula un factor de interpolación:

$$t_i = \frac{i}{N} \qquad (3.1.15)$$

La posición interpolada en el espacio 3D se calcula como:

$$\vec{P}_i = \vec{P}_o + (\vec{P}_f - \vec{P}_o) \cdot t_i \qquad (3.1.16)$$

Entonces componente a componente queda de la siguiente forma:

$$x_i = x_o + (x_f - x_o) \cdot t_i$$

$$y_i = y_o + (y_f - y_o) \cdot t_i$$

$$z_i = z_o + (z_f - z_o) \cdot t_i$$

Este es el cálculo de interpolación lineal por componentes en un espacio tridimensional.

Para el cálculo de la distribución del tiempo del movimiento (suponemos velocidad constante)

Cada paso tiene una duración de:

$$\Delta t = \frac{T}{N} \qquad (3.1.17)$$

Donde T es el tiempo total de movimiento.

A partir de las posiciones intermedias obtenidas se calcula la cinemática inversa para el movimiento de los motores como se ha visto en el apartado *"3.1.2 Cinemática inversa"* de este documento.

En la figura 8 se puede ver un ejemplo de una trayectoria lineal generada mediante la interpolación lineal en el espacio 3D. Para ilustrar su funcionamiento, se ha implementado la simulación utilizando un script de Python, el cual permite visualizar el comportamiento de la trayectoria y analizar su evolución a lo largo del tiempo. Esta simulación además nos permite verificar que los cálculos realizados son correctos y que la interpolación se comporta conforme a lo esperado.

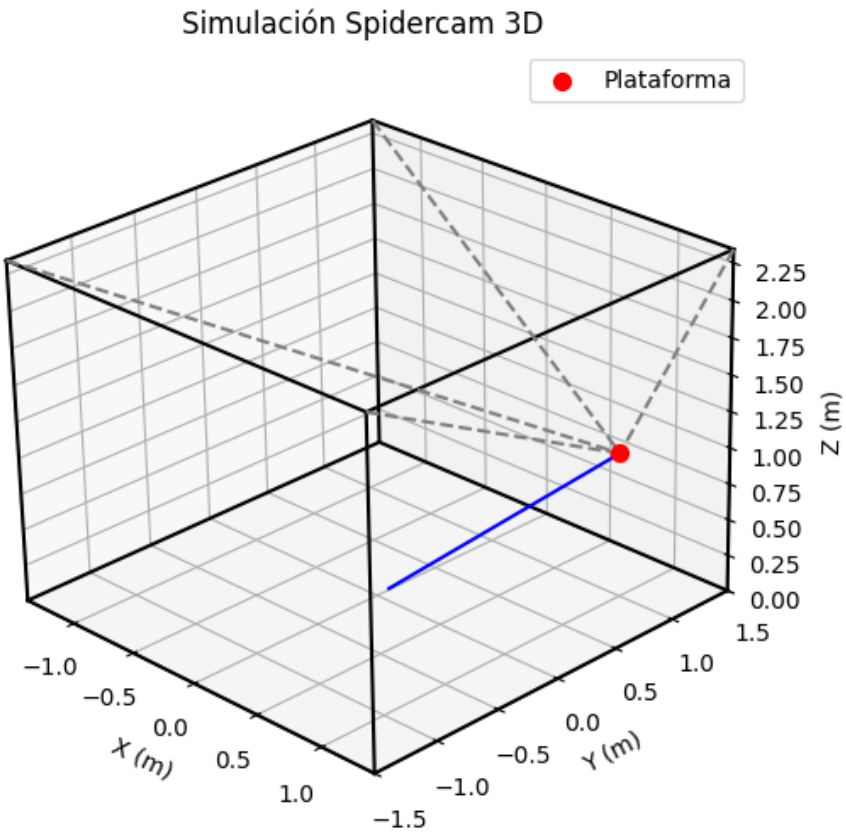


Figura 8. Trayectoria Lineal – simulación en script de Python.

3.1.5 Curvas de aceleración

Con el objetivo de lograr una transición más natural en la velocidad, se trasforma el parámetro t mediante una función cúbica por tramos, de esta forma se modifica el comportamiento del movimiento en función del tiempo, permitiendo así un control más preciso sobre la aceleración y desaceleración.

$$f(t) = \begin{cases} 4t^3 & si\ 0 \le t \le \frac{1}{2} \\ 1 - \frac{1}{2}(-2t+2)^3 & si\ \frac{1}{2} \le t \le 1 \end{cases} \qquad (3.1.19)$$

Esta función es continua y suavemente derivable, y tiene las siguientes propiedades:

$f(0) = 0 \quad f(1) = 1 \rightarrow$ inicia y termina en los extremos correctamente.

$f'(0) = 0 \quad f'(1) = 0 \rightarrow$ la velocidad es cero al inicio y al final.

$f''(t) > 0 \quad cuando \quad t < 0.5 \rightarrow$ aceleración positiva.

$f''(t) < 0 \quad cuando \quad t > 0.5 \rightarrow$ aceleración negativa (desaceleración).

Esto permite modelar un movimiento suavizado simétrico, en las siguientes figuras (Figuras 9, 10 y 11) se muestra la representación de la función utilizada y sus derivadas.

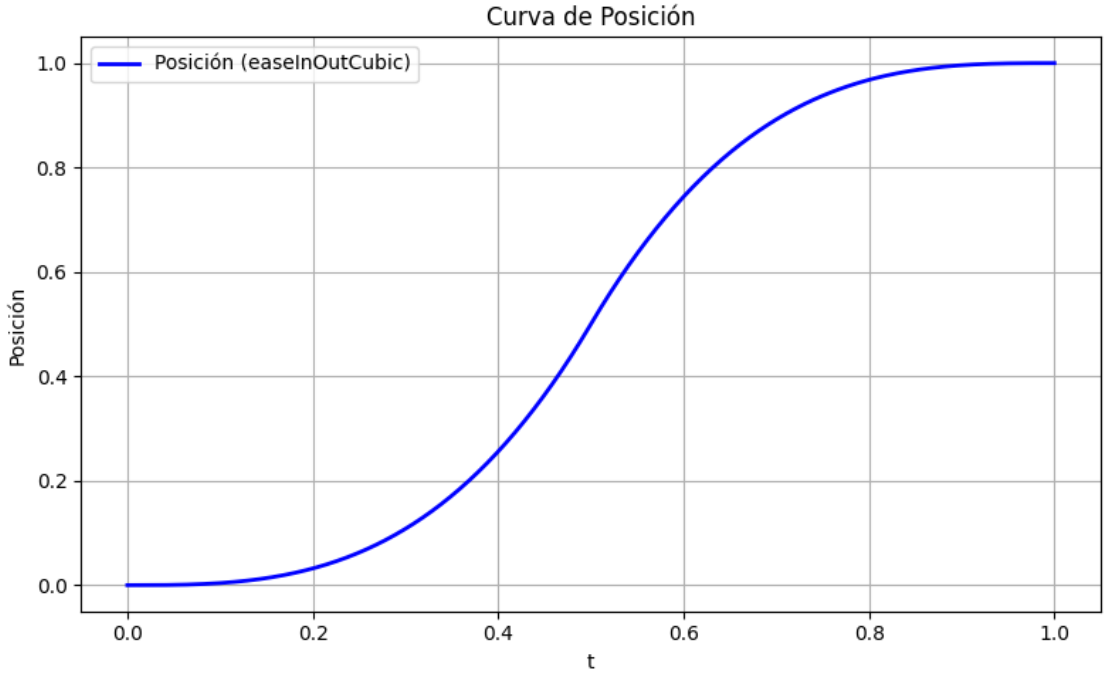


Figura 9. Curva de posición

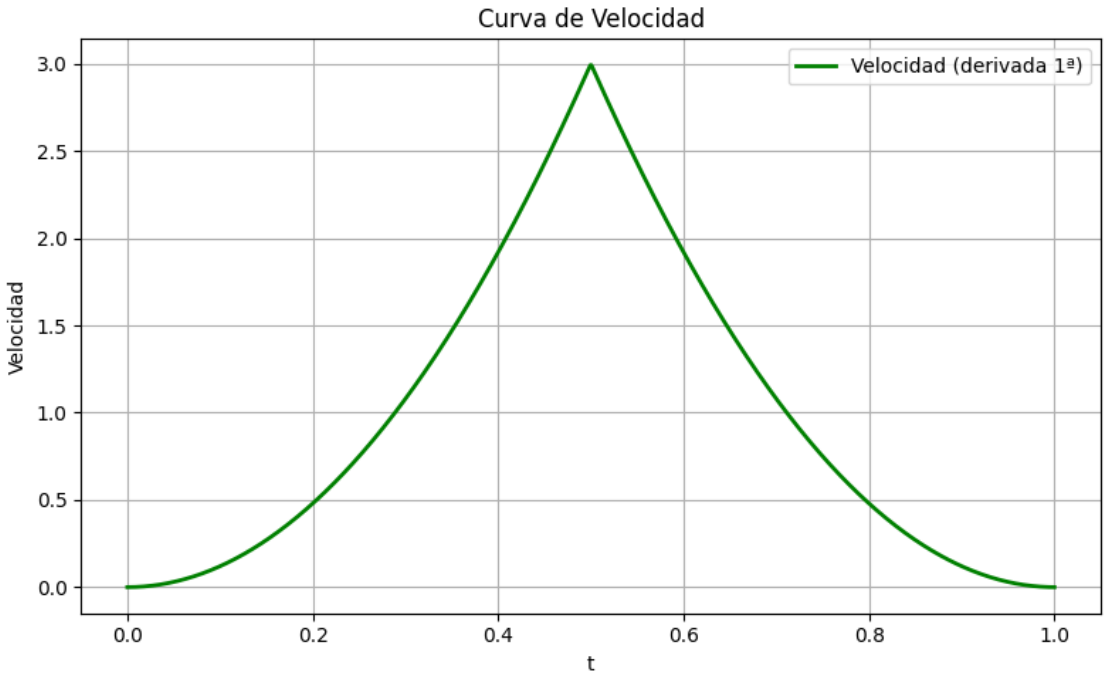


Figura 10. Curva de velocidad

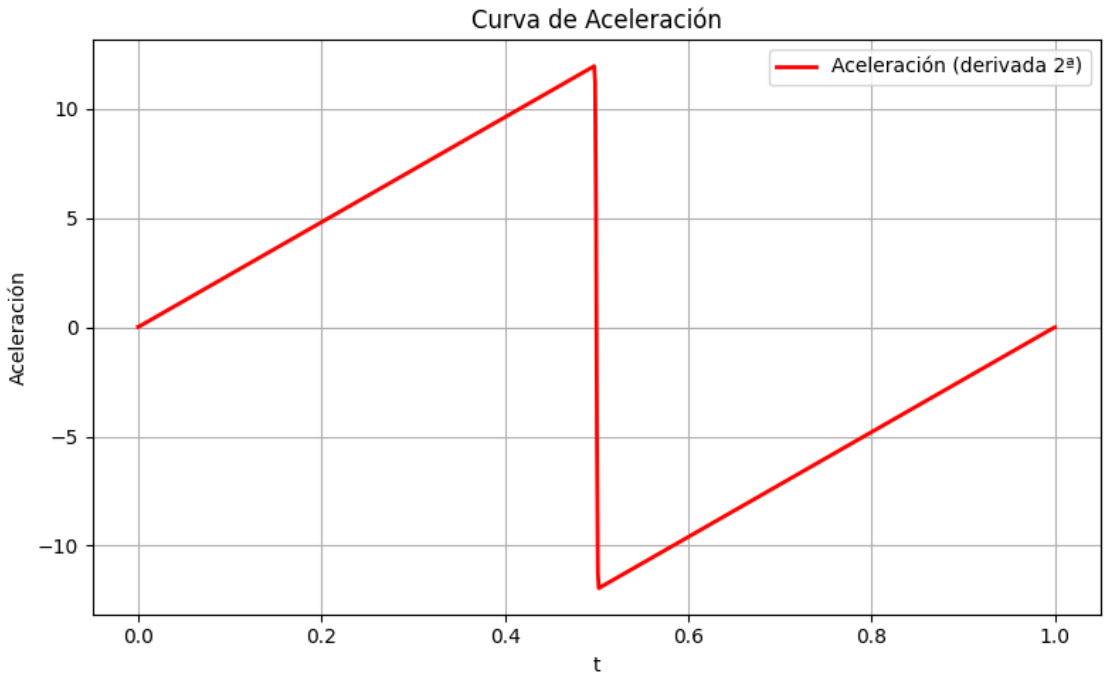


Figura 11. Curva de aceleración

Estas curvas permiten un mejor movimiento en los desplazamientos cortos, ya que debido a su longitud no se puede utilizar un segmento con velocidad constante.

3.1.6 Perfil trapezoidal

En el control de movimiento de sistemas robóticos, es fundamental garantizar que los desplazamientos se realicen de forma suave, controlada y segura. Para ello, se utilizan perfiles de velocidad que determinan cómo debe evolucionar a posición del sistema en el tiempo.

Uno de los perfiles más comunes y eficaces es el perfil de velocidad trapezoidal, el cual combina aceleración, velocidad y desaceleración constantes.

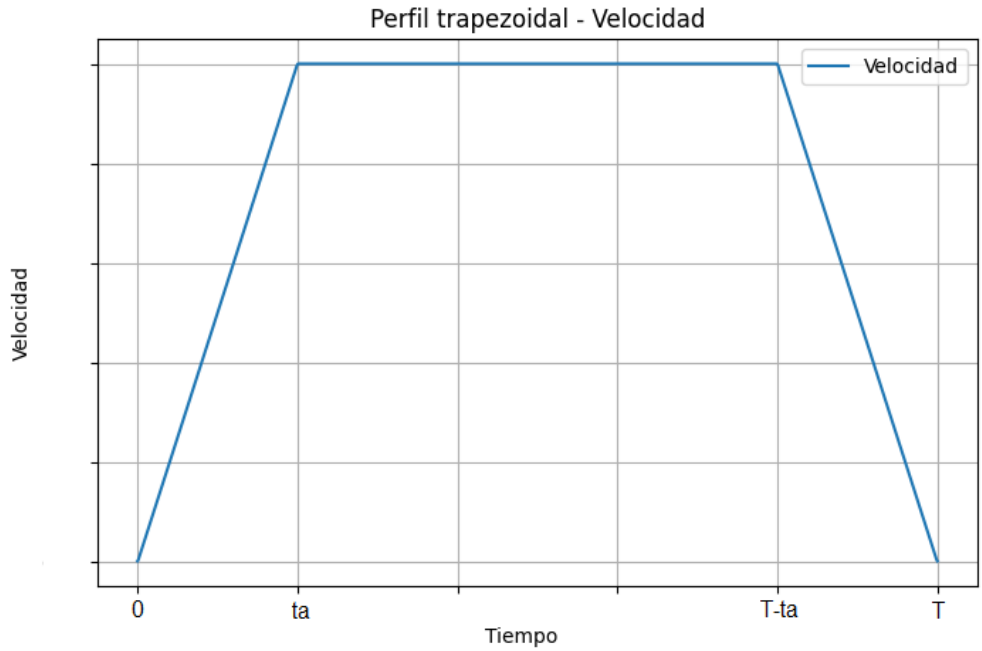


Figura 12. Perfil de velocidad trapezoidal

Como se observa en la Figura 12 el perfil trapezoidal se divide en tres fases claramente diferenciadas:

- $0 \leq t \leq t_a$ Aceleración constante.
- $t_a \leq t \leq \mathrm{T} - t_a$ Movimiento a velocidad constante.
- $T - t_a \leq t \leq T$ Desaceleración constante.

Donde:

T: tiempo total del movimiento.

t_a : duración de la aceleración (también igual a la duración de la desaceleración).

D: distancia total a recorrer, para los cálculos se normaliza a D=1.

Etapa de aceleración $(0 \leq t \leq t_a)$

$$a = \frac{v}{t_a} \rightarrow x(t) = \frac{1}{2}at^2 = \frac{1}{2} \cdot \frac{v}{t_a} \cdot t^2 \qquad (3.1.20)$$

Etapa de velocidad constante $(t_a \leq t \leq \mathrm{T} - t_a)$

$$x(t) = x(t_a) + v \cdot (t - t_a) \quad donde \; x(t_a) = \frac{1}{2}vt_a$$

$$x(t) = \frac{1}{2}vt_a + v(t - t_a) = v\left(t - \frac{t_a}{2}\right) \qquad (3.1.21)$$

Etapa de desaceleración $(T - t_a \leq t \leq T)$

$$x(t) = D - \frac{1}{2}(T - t)^2 = 1 - \frac{1}{2} \cdot \frac{v}{t_a} \cdot (T - t)^2 \qquad (3.1.22)$$

<u>Cálculo de la velocidad:</u>

Sabemos que el área bajo la curva de velocidad es la distancia total:

$$D = Área = \frac{1}{2}vt_a + v(T - 2t_a) + \frac{1}{2}vt_a = v(t - t_a)$$

$$v = \frac{1}{T - t_a} \qquad (3.1.23)$$

Función completa de la posición x(t).

La posición x(t) en cualquier instante se define como:

$$x(t) = \begin{cases} \frac{1}{2} \cdot \frac{v}{t_a} \cdot t^2 & si \; 0 \leq t \leq t_a \\ v\left(t - \frac{t_a}{2}\right) & si \; t_a \leq t \leq \mathrm{T} - t_a \\ 1 - \frac{1}{2} \cdot \frac{v}{t_a} \cdot (T - t)^2 & si \; T - t_a \leq t \leq T \end{cases} \qquad (3.1.24)$$

<u>Derivadas: velocidad y aceleración</u>

Velocidad: $v(t) = \frac{dx(t)}{dt}$

$$v(t) = \begin{cases} \frac{v}{t_a} \cdot t & si \; 0 \leq t \leq t_a \\ v & si \; t_a \leq t \leq \mathrm{T} - t_a \\ \frac{v}{t_a} & si \; T - t_a \leq t \leq T \end{cases} \qquad (3.1.25)$$

Aceleración $a(t) = \frac{dv(t)}{dt}$

$$a(t) = \begin{cases} \frac{v}{t_a} & si\ 0 \leq t \leq t_a \\ 0 & si\ t_a \leq t \leq \mathrm{T} - t_a \\ -\frac{v}{t_a} & si\ T - t_a \leq t \leq T \end{cases} \qquad (3.1.26)$$

A continuación, se pueden observar las gráficas obtenidas correspondientes a un tiempo de cinco segundos en las Figuras 13, 14 y 15.

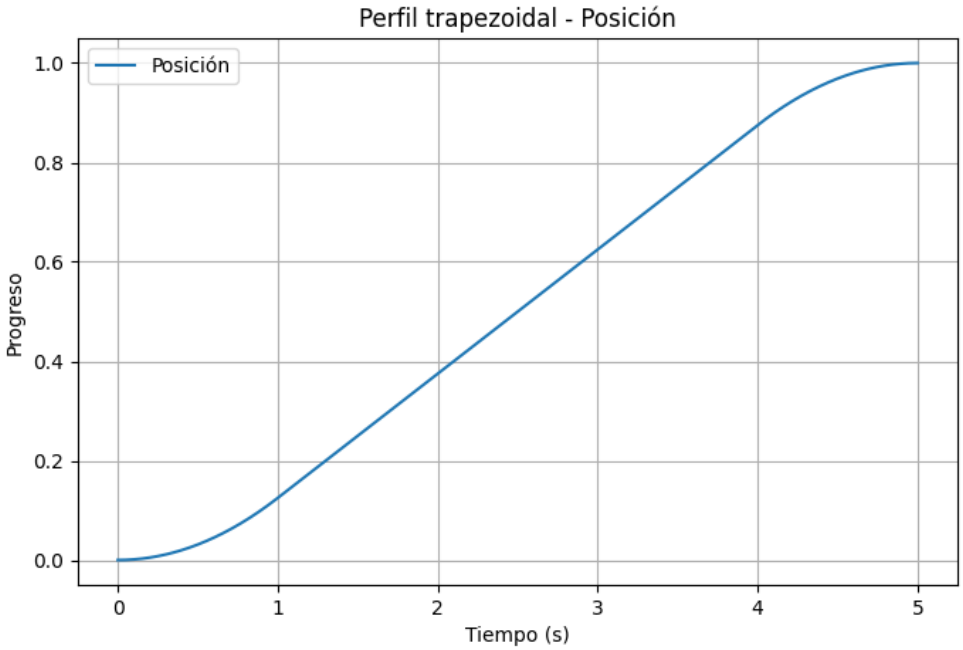


Figura 13. Perfil trapezoidal - posición

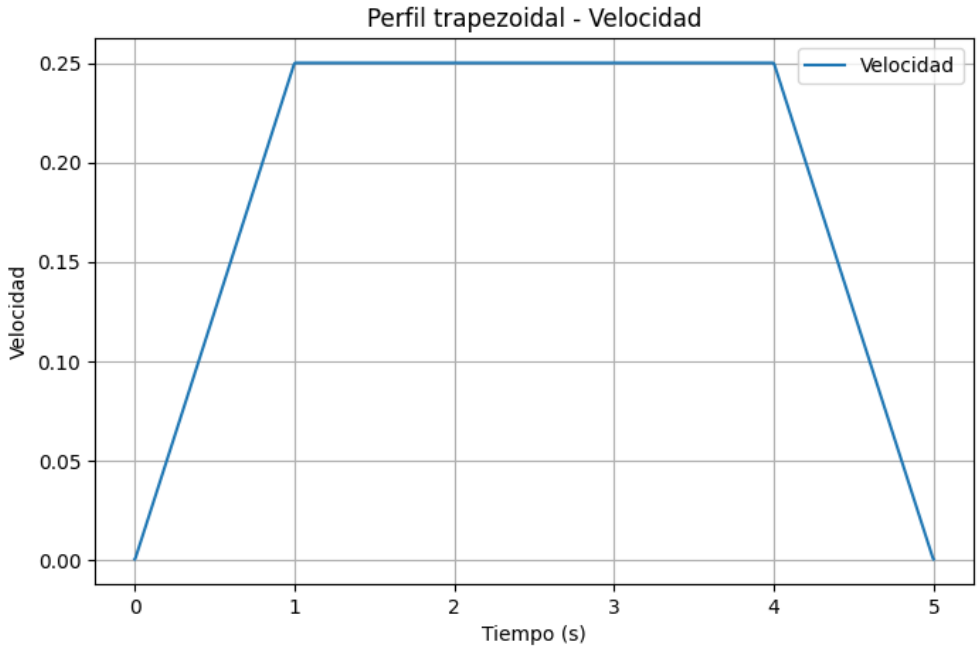


Figura 14. Perfil trapezoidal - velocidad

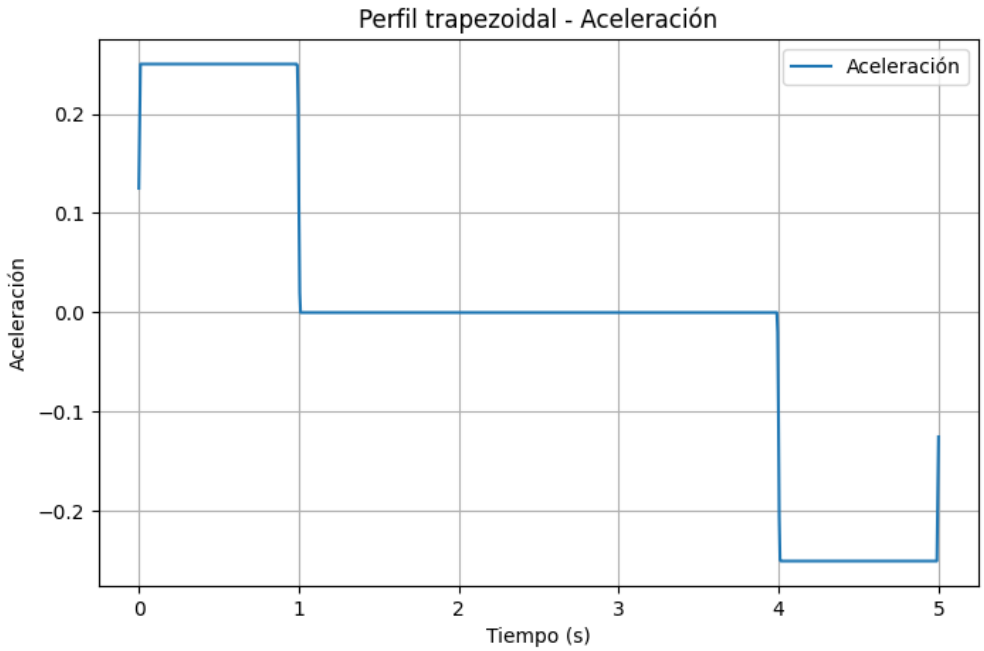


Figura 15. Perfil trapezoidal - aceleración

3.1.7 Cálculo de prestaciones de los motores

Para los cálculos se van a utilizar los datos de su hoja de características [7]:

Tensión nominal V	12 V
Corriente nominal In	3.46 A
Par nominal Tn	2.79 Nm
Velocidad nominal wn	130 rpm
Par máximo	5.8 Nm
Corriente máxima	7.6 A
Constante de par Kt	0.76 Nm/A
Constante de velocidad Kv	12 rpm/v
Potencia nominal	200 W
Inercia del rotor	4656 gcm2

Tabla 1. Características motor RMD-L-9025-35T

Potencia consumida:

$$P = V \cdot I = 12 \cdot 3.46 = 41.52\ W \qquad (3.1.\ 27)$$

El motor consume 42 W a plena carga nominal.

Potencia mecánica entregada:

$$P = T\ x\ w = 2.79 \cdot \frac{2\pi \cdot 130}{60} = 37.97\ W \quad (3.1.28)$$

La potencia mecánica entregada es de 38 W.

Eficiencia estimada

$$\eta = \frac{P_{mecanica}}{P_{electrica}} \cdot 100 = \frac{37.97}{41.52} \cdot 100 = 90.36\% \qquad (3.1.29)$$

Eficiencia estimada al operar a par y velocidad nominal es de 90.36%.

Capítulo 4: Herramientas Utilizadas

4.1 Arduino

Para controlar el robot de cables se ha utilizado el ESP32, se utiliza cinco en total, uno por cada motor y otro para mandarles las señales.

El ESP32 es un microcontrolador de alto rendimiento desarrollado por Espressif Systems. Este chip ofrece una combinación excepcional de potencia de cómputo, conectividad inalámbrica y versatilidad de entradas/salidas, lo que lo convierte en una elección idónea para aplicaciones embebidas complejas. Podemos verlo en la figura 16.

Figura 16. Imagen de ESP32

Las razones principales de su elección son:

- Conectividad dual Wi-Fi + Bluetooth, incluyendo soporte para protocolo como ESP-NOW, que es utilizado para la comunicación inalámbrica entre los nodos del sistema y es el que se utiliza en este proyecto.
- Bajo consumo energético, con modos de suspensión que permiten optimizar la eficiencia en sistemas que funcionan con alimentación limitada.
- Capacidad de procesamiento multinúcleo, facilitando la ejecución simultánea de tareas como control de motores, recepción de comandos y envío de datos.
- Gran número de GPIOs, buses UART/SPI/I2C, y capacidad de PWM, útiles para controlar actuadores, sensores y otros módulos periféricos.

Cada nodo del sistema distribuido, tanto el maestro como los esclavos, está construido alrededor de un ESP32, que se encarga de la comunicación, la interpretación de comandos, el control de los motores, y la respuesta ante eventos del sistema.

En cuanto a la alimentación eléctrica, el ESP32 opera en 3.3V o 5V.

Respecto a la distribución de sus pines, el ESP32 dispone de una amplia gama de GPIOs (General Purpose Input/Output), muchos de los cuales permiten múltiples funciones dependiendo de la configuración del firmware. Los pines se pueden agrupar según su función principal en las siguientes categorías:

- Pines de alimentación: incluyen las entradas de 3.3V, GND y pines habilitadores como EN (Enable).
- Pines digitales de propósito general (GPIO): utilizados para controlar motores, leer sensores o generar señales PWM.
- Pines analógicos: permiten la lectura de señales analógicas a través del ADC (Convertidor Analógico-Digital).
- Pines de comunicación: incluyen interfaces como UART, SPI, I2C, y PWM, esenciales para la interacción con módulos externos y periféricos.

En cuanto a las capacidades de comunicación, el ESP32 se destaca por ofrecer múltiples interfaces de conectividad tanto cableadas como inalámbricas. Entre ellas se encuentran:

- **UART** (Universal Asynchronous Receiver-Transmitter): usada comúnmente para depuración o conexión con módulos seriales.
- **I2C** (Inter-Integrated Circuit): útil para comunicar varios dispositivos usando solo dos líneas (SDA y SCL).
- **SPI** (Serial Peripheral Interface): ideal para comunicaciones de alta velocidad con periféricos como pantallas o memorias.
- **Bluetooth**: compatible con versiones clásicas y BLE (Bluetooth Low Energy).
- **Wi-Fi**: conexión directa a redes inalámbricas o creación de redes propias en modo punto de acceso [5].

Sin embargo, para este proyecto se ha optado por utilizar el protocolo ESP-NOW, una tecnología de comunicación inalámbrica desarrollada por Espressif que permite el envío de datos entre dispositivos ESP32 sin necesidad de una red Wi-Fi tradicional. Esta elección se debe a sus ventajas en términos de bajo consumo, baja latencia y facilidad de implementación en entornos distribuidos. El funcionamiento detallado de este protocolo se describe en el siguiente apartado.

4.1.1 Estructura de control maestro-esclavo: Protocolo ESP-NOW

El protocolo de comunicaciones ESP-NOW, desarrollado por Espressif Systems, se utiliza en este proyecto como solución inalámbrica principal, debido a sus ventajas en términos de bajo consumo energético, baja latencia y facilidad de implementación. Este protocolo permite la transmisión directa de datos entre dispositivos ESP32 sin necesidad de una infraestructura de red Wi-Fi tradicional, lo que lo convierte en una opción especialmente adecuada para establecer la comunicación entre los distintos nodos del sistema desarrollado. Esta elección no ha sido arbitraria, sino el resultado de un estudio detallado de las necesidades específicas del sistema, especialmente en cuanto a velocidad de transmisión, fiabilidad, bajo consumo energético, y ausencia de infraestructura externa o redes WiFi.

ESP-NOW es un protocolo propietario, basado en la banda ISM de 2.4 GHz, que permite la comunicación directa entre varios dispositivos ESP32, ESP8266 u otros microcontroladores de la misma familia. Una de sus principales ventajas es que no requiere establecer una conexión WiFi tradicional ni realizar el intercambio de datos mediante un punto de acceso (AP), lo que simplifica la configuración inicial y reduce significativamente la latencia en la transmisión de mensajes.

A nivel conceptual, el protocolo puede entenderse como un sistema de mensajería ligera orientado a eventos, con una arquitectura maestro–esclavo o peer-to-peer según se configure, en el que un nodo central (en este caso, el ESP32 maestro) puede enviar y recibir datos de múltiples dispositivos periféricos [5][6].

Arquitectura de red: Maestro y Esclavos

El diseño de este sistema de control distribuido emplea una topología de un maestro y múltiples esclavos (1:4 en este caso), en la que el nodo maestro centraliza la lógica de control y cálculo cinemático, y los nodos esclavos ejecutan las órdenes recibidas. Cada nodo esclavo está asociado al control de un motor específico (en este caso, motores RMD-L-9025-35T), actuando como interfaz entre el hardware de movimiento y el sistema de supervisión central.

Características técnicas del protocolo ESP-NOW

El protocolo ofrece un conjunto de capacidades que lo hacen especialmente adecuado para sistemas embebidos que requieren comunicaciones inalámbricas ligeras y eficientes:

- Baja latencia en la transmisión de datos, ideal para sistemas en tiempo casi real.
- Capacidad para transmitir mensajes de hasta 250 bytes de carga útil por paquete.

- Soporte para múltiples tipos de comunicación: unicast, multicast y en modo mixto.
- Flexibilidad para operar con dispositivos con y sin cifrado simultáneamente.
- Implementación de llamadas de confirmación, que notifican al microcontrolador si la transmisión ha sido exitosa o ha fallado. [5][6]

Estas características facilitan una programación modular y robusta, donde cada nodo puede operar de forma autónoma dentro de su función, pero completamente sincronizado con el sistema global mediante el canal inalámbrico establecido por ESP-NOW.

Limitaciones operativas

No obstante, el protocolo presenta ciertas restricciones que deben ser tenidas en cuenta durante el diseño:

- El número máximo de dispositivos emparejados con cifrado activo es limitado: hasta 10 nodos en modo estación (STA), y en torno a 6 en modo punto de acceso (AP) o configuración mixta.
- El número total de nodos (con o sin cifrado) conectables al maestro no debe exceder 20 dispositivos.
- El tamaño del mensaje está restringido a un máximo de 250 bytes, lo que limita la complejidad de los comandos o paquetes de datos transmitidos [6].

En el contexto del presente sistema, estas limitaciones no suponen un obstáculo, ya que la información intercambiada consiste en comandos compactos, parámetros cinemáticos y confirmaciones de estado, todos ellos fácilmente codificables en menos de 250 bytes.

Configuración del emparejamiento

La comunicación mediante ESP-NOW se basa en la identificación por dirección MAC de los dispositivos, por lo que es necesario realizar un proceso inicial de emparejamiento, en el que el maestro registra las direcciones MAC de los esclavos con los que desea comunicarse. Este emparejamiento se guarda en la memoria del microcontrolador y no necesita repetirse mientras no cambien las direcciones físicas ni se modifique la configuración de red.

Una vez completado este proceso, el envío de mensajes entre nodos es inmediato y no requiere negociación de sesión ni mantenimiento de una conexión permanente como ocurre con los protocolos basados en sockets TCP/IP.

Aplicación en el sistema Spydercam

El protocolo ESP-NOW ha sido clave en la viabilidad técnica de la arquitectura inalámbrica distribuida del sistema Spydercam, ya que ha permitido prescindir de cableado físico entre el nodo central de control y los controladores de motor, lo cual habría supuesto un importante problema de movilidad y peso en una plataforma suspendida por cables. La latencia mínima, el bajo consumo energético y la fiabilidad de las transmisiones lo convierten en una solución ideal para aplicaciones robóticas, cinematográficas y de automatización ligera.

El esquema implementado puede representarse gráficamente en la figura 17.

Figura 17. Esquema protocolo ESP-NOW

En esta configuración, cada nodo esclavo recibe comandos cinemáticos calculados por el maestro y los ejecuta mediante el controlador integrado en los motores RMD-L.

4.2 Motores RMD-L-9025-35T: Características y Justificación de uso.

El sistema del robot de cables diseñado se basa en el uso de motores RMD-L-9025-35T (ver figura 18), fabricados por MyActuator. Estos motores son los que van a controlar la longitud del cable, es decir, dependiendo de cuanto le manden girar el cable se acortara o alargara. A continuación, se detallan sus principales características, la integración al sistema y su control.

Figura 18. Motor RMD-L-9025-35T

Características técnicas

Los motores RMD-L-9025-35T son unidades compactas de alto rendimiento que combinan motor, codificador absoluto, controlador interno y controladora CAN en un solo módulo. Esto los convierte en una solución altamente integrada y lista para su implementación en sistemas embebidos de control distribuido. Sus especificaciones más relevantes son:

- Tensión de alimentación: 12-24 V DC.
- Sistema de retroalimentación: Enconder magnético de alta resolución incorporado.
- Interfaz de comunicación: protocolo CAN, con comandos compatibles con los estándares de la industria.
- Protecciones internas: sobrecorriente, sobretensión, sobretemperatura. [7]

Esta combinación de prestaciones ofrece una solución robusta, precisa y compacta, ideal para aplicaciones mecatrónicas móviles como una cámara suspendida por cables, donde la precisión de posicionamiento y respuesta dinámica son fundamentales.

Introducción al protocolo CAN

El protocolo CAN (Controller Area Network) es un estándar de comunicación en red ampliamente utilizado en entornos industriales. Fue desarrollado originalmente por Robert Bosch GmbH en los años 80 para facilitar la comunicación robusta entre múltiples dispositivos electrónicos sin necesidad de una arquitectura centralizada.

El bus CAN opera mediante un modelo de comunicación distribuido, donde todos los nodos comparten el mismo canal físico (dos cables: CAN_H y CAN_L) y pueden emitir o recibir mensajes. El sistema implementa una técnica de arbitraje por prioridad basada en el identificador del mensaje, lo que permite que los datos más importantes lleguen primero sin colisiones. Entre sus principales características se incluyen:

- Velocidades de transmisión de hasta 1 Mbps en buses estándar, con versiones más modernas (CAN FD) que alcanzan cifras superiores.
- Topología en bus con cableado mínimo (par trenzado).
- Alta fiabilidad, incluso en entornos ruidosos o con interferencias electromagnéticas.
- Mensajes con prioridad, útiles para sistemas en tiempo real.
- Formato estándar de trama con campos de ID, longitud, datos (hasta 8 bytes), CRC, ACK, etc [20].

En comparación con otros protocolos como UART, I2C o SPI, el CAN se distingue por su robustez, escalabilidad, y capacidad para gestionar múltiples dispositivos conectados sin necesidad de maestro esclavo rígido.

Aplicación del protocolo CAN en el sistema Spydercam

En este proyecto, el protocolo CAN se emplea como canal principal de comunicación entre los microcontroladores ESP32 y los motores RMD-L-9025-35T. La elección del protocolo no ha sido arbitraria: los motores seleccionados incluyen de forma nativa una interfaz CAN integrada, que permite acceder a sus funciones internas mediante comandos estandarizados.

Cada nodo esclavo del sistema (ESP32 individual) se conecta a su respectivo motor mediante una placa MCP2515_CAN, que actúa como controlador CAN externo, ya que el ESP32 no incorpora esta funcionalidad de forma nativa. Ul módulo MCP2515, proporciona la capa física y de enlace necesaria para establecer una comunicación fiable entre el microcontrolador y el motor.

Esquema básico de integración:

- ESP32 esclavo: Comunicación SPI con MCP2515 (para enviar/recibir mensajes CAN)
- MCP2515: Traduce los comandos SPI a señales CAN.
- Motor RMD-L-9025-35T: Recibe comandos CAN y devuelve datos de estado.

Este diseño modular permite que cada nodo actúe como una unidad autónoma de control de motor, mientras que el nodo maestro (otro ESP32) se encarga de enviar comandos globales, procesar trayectorias y coordinar el comportamiento de todo el sistema.

Comandos e integración funcional

Los motores RMD permiten recibir diversos comandos CAN y cada mensaje enviado desde el ESP32 incluye:

- Un identificador (ID) del dispositivo destino.
- Un conjunto de bytes con los parámetros de control (p. ej., RPM deseados).
- Un chequeo de integridad para asegurar la transmisión.

Por otro lado, el motor responde con mensajes CAN que incluyen:

- Valor actual de posición angular.
- Velocidad instantánea.
- Estado térmico del motor.
- Fallos o códigos de error (en caso de detección interna).

La utilización del bus CAN aporta múltiples beneficios clave en el contexto del sistema Spydercam:

- Robustez eléctrica: CAN soporta condiciones de ruido, crucial en aplicaciones con motores de potencia.
- Flexibilidad escalable: se pueden añadir más nodos sin rediseñar el bus.
- Diagnóstico en tiempo real: se pueden supervisar errores o estados directamente desde el nodo maestro.

En resumen, la integración del protocolo CAN en el sistema embebido permite una comunicación precisa, rápida y fiable entre los ESP32 y los motores inteligentes RMD-L-9025-35T, aportando una infraestructura sólida para el control sincronizado de los ejes del sistema Spydercam.

4.3 MCP2515 CAN – Módulo de comunicación por bus CAN

Para facilitar la comunicación con los motores RMD-L-9025-35T, que utilizan el protocolo CAN Bus, se emplea una placa basada en el controlador MCP2515 (ver Figura 19).

Figura 19. Módulo MCP2515 CAN [21]

Este módulo convierte las señales SPI del ESP32 en comandos válidos para la red Controller Area Network (CAN), que es ampliamente utilizada en entornos industriales y automotrices debido a su robustez y resistencia a interferencias.

Características destacables [21]:

- Comunicación fiable en entornos eléctricos ruidosos, gracias al diseño diferencial del bus CAN.
- Soporte para múltiples nodos en la misma línea de comunicación, ideal para sistemas distribuidos.
- Compatible con velocidades de 1 Mbps, suficiente para la transmisión de comandos de control y monitoreo en tiempo real.

En el sistema Spydercam, cada nodo esclavo (ESP32) se comunica con su respectivo motor mediante el módulo MCP2515, lo que permite un control preciso de la velocidad y posición del motor.

4.4 LM2596S – Módulo regulador de voltaje DC-DC

El sistema de alimentación se completa con el uso de módulos step-down (buck) basados en el regulador LM2596S (ver Figura 20), que permiten adaptar tensiones de entrada superiores (por ejemplo, de una batería de 12V o fuente de alimentación general) a niveles compatibles con los componentes electrónicos, como los 5V o 3.3V requeridos por algunos módulos y sensores. Este módulo es el encargado de en base a

los 12V de la fuente de alimentación de generar los 5V necesarios para alimentar al ESP32 correspondiente.

Figura 20. Módulo regulador LM2596S

Ventajas del LM2596S:

- Regulación ajustable de voltaje de salida mediante potenciómetro.
- Alta eficiencia (>75%), reduciendo la generación de calor y pérdidas energéticas.
- Capacidad de corriente de hasta 2A continuos, más que suficiente para alimentar un ESP32 y su circuito auxiliar.

La incorporación de este módulo garantiza la estabilidad eléctrica del sistema, evitando fluctuaciones o caídas de tensión que podrían comprometer el correcto funcionamiento del microcontrolador o la comunicación con los motores.

4.5 Diseño de piezas e impresión

Para desarrollar el prototipo se han elaborado las piezas que lo componen con el método de impresión 3D.

El prototipo se compone de diversas piezas cuyas funciones no implican la resistencia a grandes esfuerzos mecánicos ni requieren propiedades específicas. En este contexto, la fabricación mediante impresión 3D se presenta como la solución más adecuada, ya que permite producir elementos con geometrías complejas de manera eficiente y económica. Además, esta tecnología ofrece una notable versatilidad en el diseño, facilitando la iteración y personalización de las piezas según las necesidades del proyecto, sin que ello suponga un incremento significativo en los costes de producción ni en los tiempos de desarrollo.

La impresión 3D es una tecnología que permite crear objetos tridimensionales a partir de un modelo digital. A diferencia de los métodos de fabricación tradicionales que eliminan material (como el mecanizado), la impresión 3D construye las piezas capa por capa, depositando material únicamente donde sea necesario. Esto la convierte en una técnica eficiente, versátil y adecuada para la producción rápida de prototipos o piezas personalizadas.

La impresora interpreta un archivo 3D, lo convierte en capas mediante un software de laminado y, posteriormente, ejecuta la fabricación capa por capa utilizando diversos materiales, como plásticos, resinas, metales o compuestos [22].

Capítulo 5: Desarrollo de un Sistema Spydercam Autónomo basado en ESP-NOW

En este capítulo se expone el proceso de desarrollo del sistema, incluyendo el planteamiento inicial, la identificación de los requisitos técnicos y el análisis de las distintas alternativas consideradas. Aunque los componentes finales ya fueron descritos en el capítulo anterior, aquí se justifica su elección en base a criterios técnicos y funcionales. Además, se detalla paso a paso el desarrollo, montaje e integración del sistema, desde la selección de soluciones hasta la implementación final.

5.1 Planteamiento del Sistema y Análisis de Alternativas

5.1.1 Objetivos funcionales del sistema spydercam

El objetivo principal de este proyecto es el desarrollo de una spydercam autónoma capaz de desplazarse con precisión en un espacio tridimensional mediante un sistema de cables accionados por motores. Este sistema debe ser capaz de:

- Posicionarse con precisión en un punto determinado del espacio.
- Ejecutar trayectorias suaves y controladas.
- Ser controlado remotamente desde una interfaz de usuario.
- Operar de forma estable y segura sin intervención manual directa.

El sistema actualmente solo está siendo usado para el movimiento de una plataforma rígida, pero podría incorporársele sin problemas una cámara para poderse aplicar en grabación audiovisual o inspección aérea de interiores.

5.1.2 Requisitos técnicos

Para alcanzar los objetivos anteriores, se han definido los siguientes requisitos técnicos:

- Precisión de posicionamiento: error máximo <2cm.
- Área de cobertura: 3x2.34x2.8 m.
- Capacidad de carga útil: mínimo de 300g.
- Sistema de control remoto: comunicación fiable desde un PC.
- Respuesta rápida: baja latencia en la comunicación y control.
- Alimentación externa: fuente de alimentación.

5.1.3 Tecnologías candidatas

Durante la fase de investigación previa se evaluaron varias alternativas:

- Microcontroladores: ESP32, Arduino Mega…
- Protocolos de comunicación: UART, I2C, WiFi (TCP/IP), Bluetooth, ESP-NOW…

- Motores: estándar de rotación continua, motores DC con encoder, Motores RMD-L-9025-35T con controlador integrado.
- Diseño mecánico: estructura de impresora 3D, sistema de poleas…

5.1.4 Comparativa de soluciones: ventajas y limitaciones

Durante el proceso de diseño se evaluaron varias alternativas. La siguiente tabla resume las principales opciones evaluadas, junto con sus ventajas y limitaciones más relevantes:

Componente	Alternativa Evaluada	Ventajas	Limitaciones
Microcontrolador	Arduino Mega ESP32	- Arduino: fácil de usar, comunidad amplia - ESP32: potente, WiFi y BT	- Arduino: limitado en velocidad y RAM
Protocolo de comunicación	WiFi (TCP/IP) Bluetooth UART ESP-NOW	- ESP-NOW: baja latencia, sin router - UART: simple y fiable punto a punto	- Bluetooth: limitado alcance y fiabilidad - WiFi: necesita infraestructura o punto de acceso
Actuadores	Servomotores Motores DC + encoder Motores RMD-L-9025-35T	- RMD: alta precisión, controlador integrado, comunicación por UART o CAN - DC + encoder: buena respuesta y versatilidad	- RMD: mayor coste y configuración más compleja - Servos: baja potencia y precisión insuficiente
Estructura mecánica	Estructura impresión 3D	- Impresión 3D: bajo coste, ligereza, personalización	- 3D: requiere refuerzos en piezas sometidas a carga

Tabla 2. Comparativa de soluciones

La elección final se realizó considerando no solo el rendimiento, sino también la disponibilidad de componentes, facilidad de integración. Al final fue una decisión entre precisión, complejidad y coste.

5.1.5 Solución seleccionada: criterios y justificativos

Finalmente se optó por la siguiente configuración que son los que se explicaron en el capítulo anterior:

- ESP32 como microcontrolador principal por su versatilidad, doble núcleo, conectividad WiFi y compatibilidad con ESP-NOW.
- Protocolo ESP-NOW para la comunicación entre el nodo maestro y los cuatro nodos esclavos, por su baja latencia y simplicidad de configuración sin necesidad de infraestructura de red adicional.

- Motores RMD-L-9025-35T, que incorporan controladores integrados y permiten un control preciso de posición y velocidad, simplificando la electrónica de potencia.
- Diseño mecánico personalizado en Autodesk Inventor, fabricado mediante impresión 3D para maximizar la adaptabilidad, minimizar peso y facilitar el montaje.

Esta solución proporciona un equilibrio entre precisión, fiabilidad, coste razonable y facilidad de desarrollo, siendo viable tanto para pruebas en laboratorio como para demostraciones funcionales en campo controlado.

5.2. Diseño mecánico y fabricación

Este apartado recoge el desarrollo estructural y la fabricación de los componentes físicos que conforman el sistema Spydercam. Se abordan tanto los criterios considerados durante la fase de diseño mecánico como las decisiones adoptadas en relación con los materiales, tecnologías de manufactura y soluciones constructivas implementadas. Asimismo, se detalla el proceso de validación del conjunto, con énfasis en el comportamiento dinámico y la integración mecánica con el sistema de control. El objetivo es garantizar que la estructura sea funcional, precisa y fiable dentro del contexto robótico para el que ha sido diseñada.

5.2.1 Diseño CAD mediante Autodesk Inventor

El diseño mecánico del sistema ha sido desarrollado íntegramente en Autodesk Inventor, un software de modelado paramétrico en 3D que ha permitido representar con precisión cada uno de los componentes del proyecto. Se modelaron tanto las partes estructurales como los soportes para los motores, el carrete de cable y los alojamientos destinados a los elementos electrónicos.

El entorno CAD ha facilitado la verificación dimensional y geométrica del conjunto, asegurando la correcta alineación de las piezas móviles y permitiendo realizar ajustes interactivos en las dimensiones antes del proceso de fabricación. Esta metodología ha contribuido significativamente a minimizar errores y ayudar a modificaciones posteriores.

Entre las piezas diseñadas destacan los elementos de la estructura de soporte de los motores, que incluyen: el carrete en el que se enrolla el cable, la estructura de fijación del motor, y el sistema de anclaje a pared. Este último incorpora un mecanismo basado en una varilla que permite un montaje y desmontaje más sencillo y seguro del conjunto.

5.2.2 Montaje y ensamblaje final del sistema

A continuación, se detalla el proceso de montaje, junto con una explicación de los distintos componentes que lo conforman. Con el objetivo de mejorar la comprensión visual, se incorporan imágenes tanto de los modelos CAD como de las piezas físicas una vez impresas, lo que permite observar su correspondencia y ensamblaje en el sistema final.

Inserción de los motores

Los motores RMD serán los encargados de proporcionar el movimiento al sistema, ya que son los que ejecutan los movimientos para enrollar o desenrollar el cable.

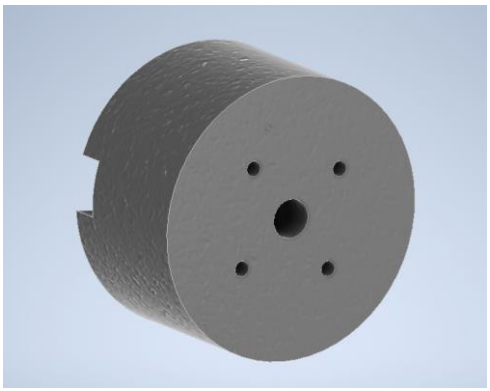


Figura 21. Motor RMD-L-9025-35T en formato CAD

Para una mejor visualización de como quedaría el montaje se ha diseñado también el motor en Autodesk Inventor, en la figura 21 se puede observar el modelo CAD del motor.

Lo primero que se diseño fue el carrete para enrollar el cable en el giro del motor, lo podemos observar en la figura 22.

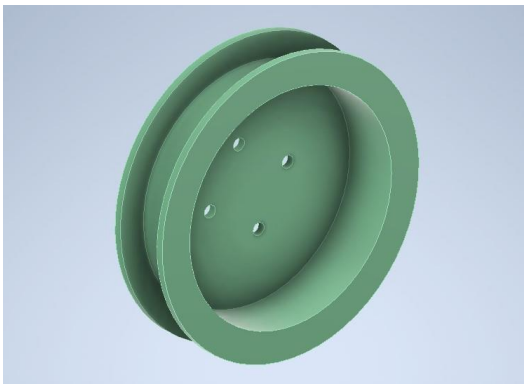


Figura 22. Carrete en formato CAD

Se ha diseñado con el mínimo material posible con ranuras para poder atornillarlo al motor de forma efectiva. En este carrete se acopla en la parte central el motor y en la parte superior se enrolla el cable.

Lo siguiente que se diseño es la estructura para sujetar el motor y anclarlo a la pared, está dividido en una bisagra que está sujeta por un soporte.

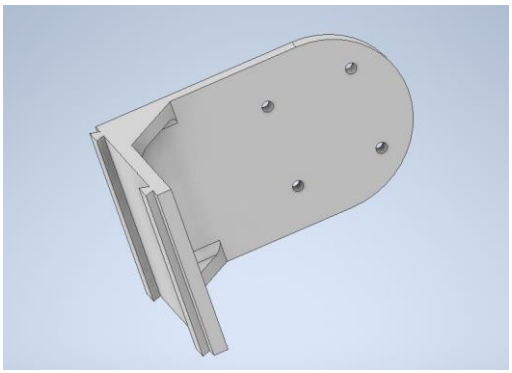


Figura 23. Bisagra en formato CAD

En esta figura 23 vemos el soporte que sujeta el motor junto con el carrete, se atornilla al motor, esta pieza va sujetada a la bisagra de la siguiente figura 24 con una cola de milano.

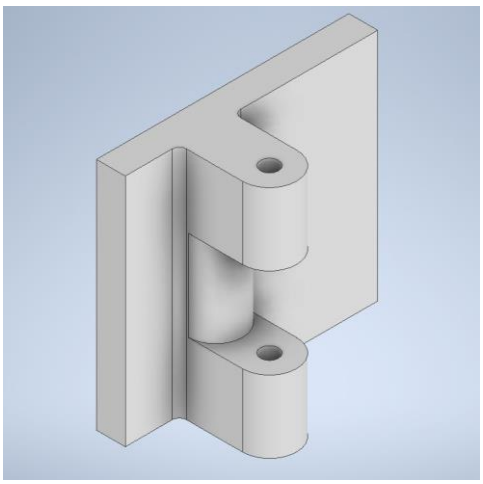


Figura 24. Soporte bisagra en formato CAD

En la figura 24 se observa la bisagra que va unida a otra pieza, la de la figura 25, a través de un eje para permitir la libre inclinación del motor y así se ajuste a la posición más cómoda en cada momento.

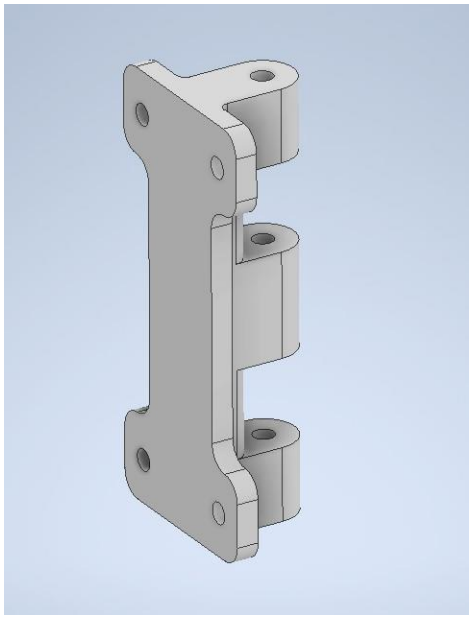


Figura 25. Bisagra 2 en formato CAD

En la figura 25 vemos la sujeción a la pared que tiene cuatro orificios para una sujeción resistente y en la parte contraria está el resto de la bisagra.

A continuación, observamos el ensamblaje completo del sistema en la figura 26

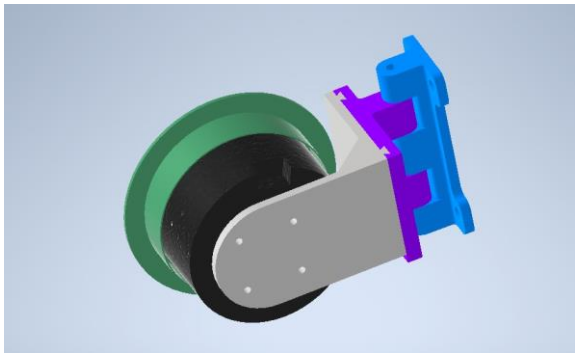


Figura 26. Ensamblaje en formato CAD

Podemos observar cómo va sujeto todo al motor, de esta forma el motor puede orientarse en función de la posición de la plataforma y se enrolla únicamente cable en el carrete.

Una vez sujeto el motor a la pared se ha diseñado una caja para colocar los componentes necesarios para el control del sistema, la caja es la que se puede observar en la figura 27.

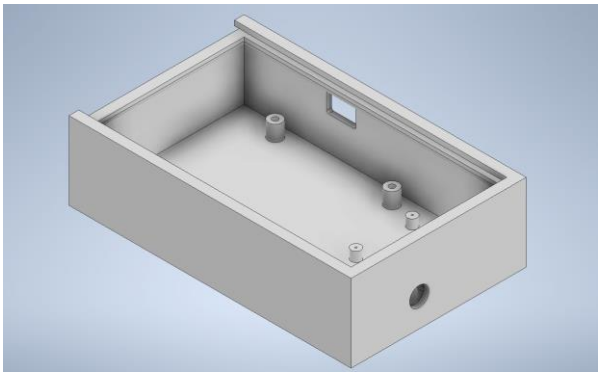


Figura 27. Caja en formato CAD

Es una caja lo más compacta posible, en el interior se encuentran los soportes donde se atornillarán los componentes electrónicos necesarios para el control del motor y tiene acceso para los cables de alimentación y los cables de señal del motor, además de un orificio para entrada del cable para poder reprogramar el esp32 si fuera necesario. En la parte superior tiene un rail para colocar la tapa de la caja y dejar todos los componentes protegidos.

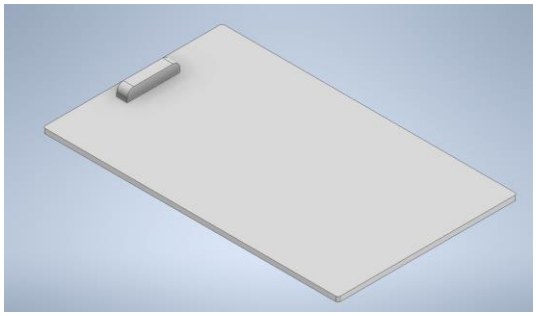


Figura 28. Tapa caja en formato CAD

Aquí vemos en la figura 28 la tapa utilizada en la caja.

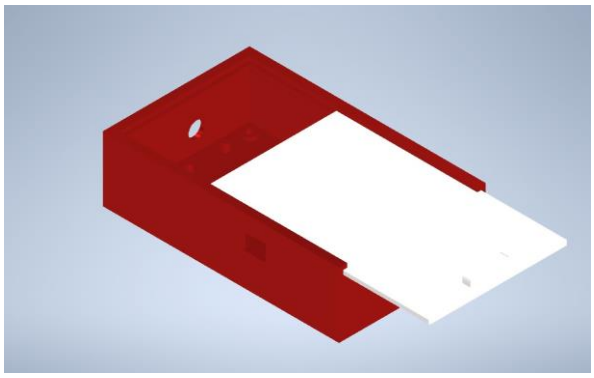


Figura 29. Ensamblaje caja en formato CAD

Aquí se observa toda la caja en su conjunto en la figura 29. Esta caja hay que sujetarla tambien a la pared por lo que se ha utilizado un sistema de rail para ello. En la parte posterior de la caja se sujeta la siguiente pieza mostrada en la figura 30.

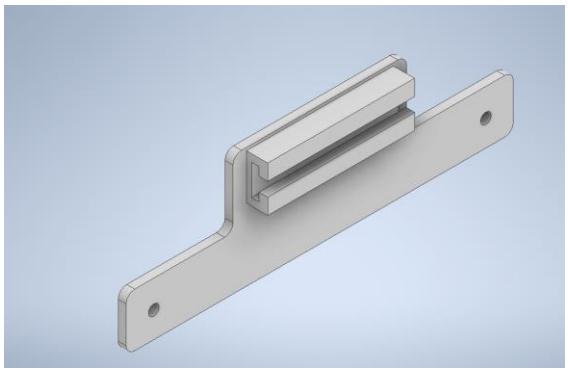


Figura 30. Rail caja en formato CAD

Esta pieza se atornilla a la caja para poder colgarla en la estructura del motor y de esta forma tener el montaje completo.

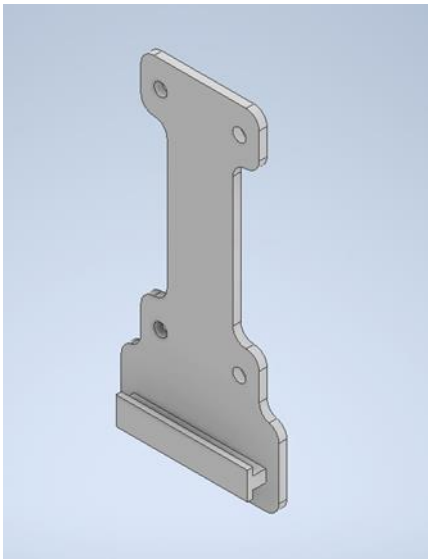


Figura 31. Sujeción caja en formato CAD

Esta última pieza representada en la imagen 31 se pega a la parte de atrás de la estructura de sujeción del motor, los orificios coinciden con los de la pieza de sujeción a la pared del motor por lo que se coloca ahí, de tal forma que cuando se sujete a la pared se sujete todo a la vez. En la parte inferior tiene el acople al rail de la figura anterior, de esta forma se puede fácilmente desacoplar la caja por si fuera necesario.

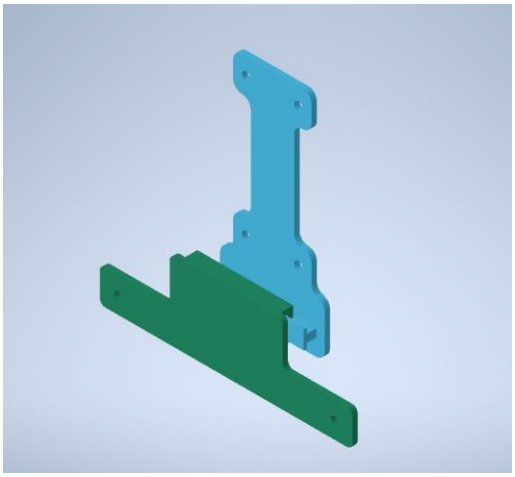


Figura 32. Ensamblaje sujeción caja en formato CAD

En la figura 32 se observa como quedaría montada la estructura del acople de la caja.

En la siguiente figura 33 se observa una muestra del montaje final ya instalado.

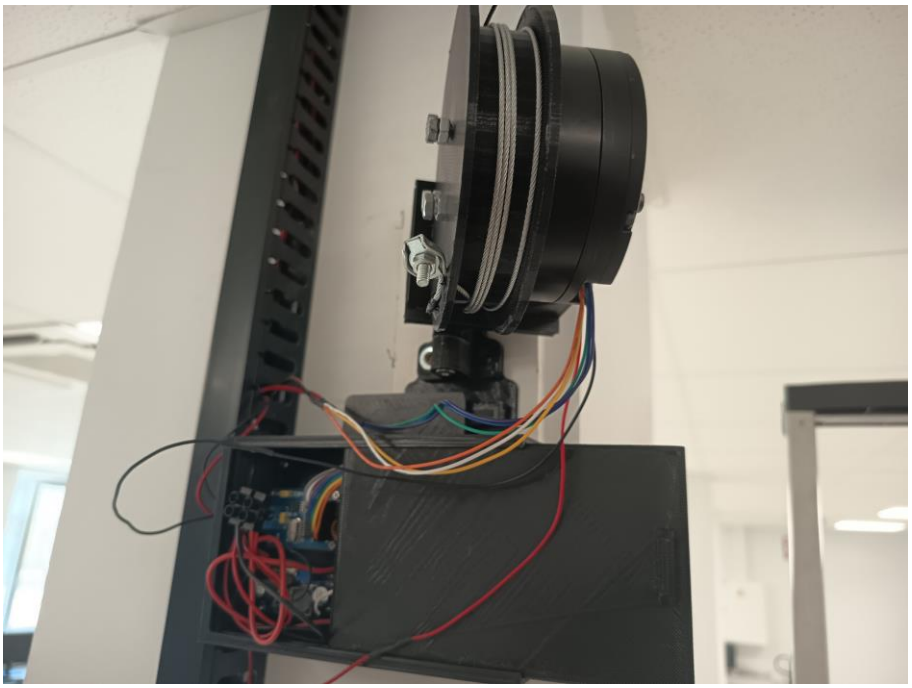


Figura 33. Montaje final

A continuación, se observa el diseño de la plataforma que al principio se optó por anclar a cada esquina de la plataforma un cable, pero después de variar pruebas se ha decidido colocar los cables más en el centro para mejor estabilidad.

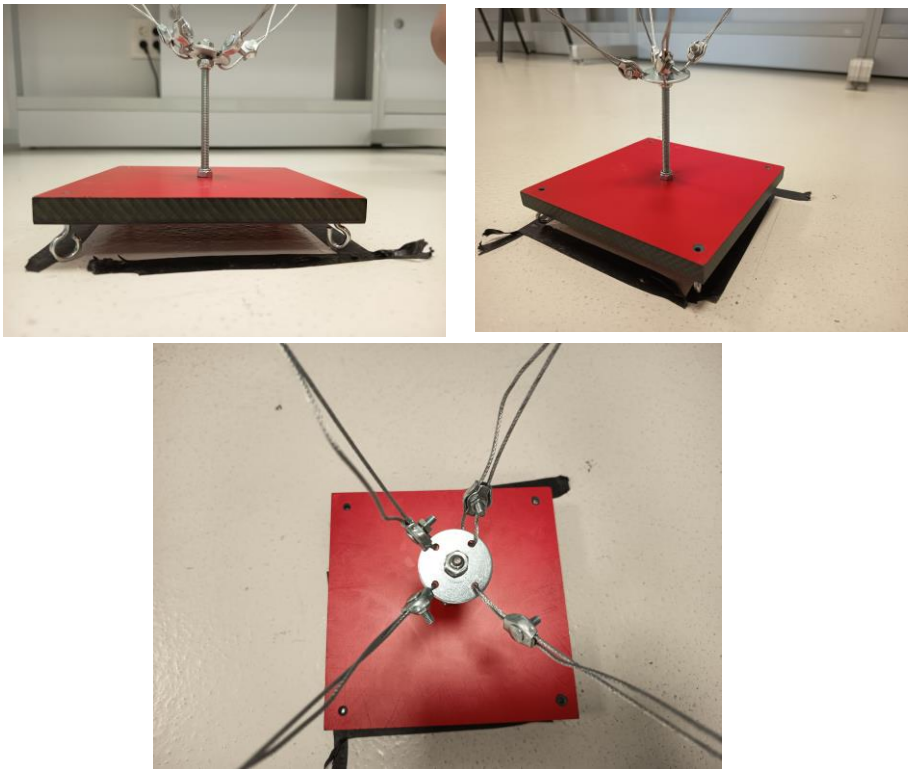


Figura 34. Plataforma utilizada para las pruebas

Figura 35. Montaje completo del sistema Spydercam

Por último en la figura 35 se puede observar el montaje final del sistema.

5.2.3 Validación del deseño físico

Una vez completado el montaje del sistema, se procedió a realizar una serie de pruebas mecánicas y de control con el objetivo de validar su funcionamiento y ajustar los parámetros de operación de los motores:

- **Prueba de carga**: Se suspendieron masas superiores a las previstas en condiciones de operación real, con el fin de verificar la resistencia estructural del sistema y su capacidad de carga.
- **Pruebas de desplazamiento**: movimientos repetidos en ejes X, Y y Z para evaluar la estabilidad estructural, su precisión y el juego mecánico.
- **Observación de vibraciones**: Se analizó visual y sensorialmente el comportamiento del sistema durante frenadas y cambios bruscos de dirección, prestando especial atención a posibles oscilaciones no deseadas.

Para optimizar el rendimiento dinámico del sistema, se utilizó el software RMD V2.0, una herramienta específica para la configuración avanzada de los motores RMD (para más detalles ir a los anexos). A través de este software, fue posible ajustar los parámetros del controlador PID de los motores, permitiendo una respuesta más suave y precisa del sistema. Las pruebas mencionadas anteriormente fueron fundamentales para identificar los valores óptimos de ganancia proporcional, integral y derivativa, ajustándolos hasta minimizar completamente las vibraciones y eliminar movimientos bruscos no deseados.

Además, el software RMD V2.0 permite controlar los motores mediante diferentes modos operativos, como control de velocidad, control de posición, o una combinación de ambos. Esta funcionalidad fue aprovechada durante la fase de validación para comprobar el comportamiento del sistema bajo distintos regímenes de operación, lo cual facilitó una puesta a punto más completa y adaptada a los requerimientos finales del proyecto.

5.3. Arquitectura del Sistema de Control Distribuido

Integración con el sistema embebido

Cada motor está conectado a un ESP32 esclavo, que actúa como nodo local de control. La comunicación entre el ESP32 y el motor se realiza mediante el protocolo CAN Bus, a través del controlador MCP2515.

El software embebido desarrollado para el ESP32 maestro se encarga de gestionar la posición y velocidad de cada motor enviando comandos específicos por el bus CAN. Los esclavos, a su vez, traducen estos comandos y se comunican con su motor correspondiente mediante instrucciones de bajo nivel.

- El motor dispone de múltiples modos de operación:
- Control de velocidad (RPM)
- Control de posición absoluta

La elección de estos modos se realiza mediante comandos específicos, y permite adaptar el sistema según las condiciones dinámicas o el tipo de movimiento deseado.

5.3.1 Asignación de funciones a cada nodo

El sistema Spydercam diseñado se basa en una arquitectura distribuida y jerárquica, donde múltiples nodos (microcontroladores ESP32) cooperan para el control y movimiento preciso de la plataforma suspendida mediante cables. Esta estructura se implementa bajo un modelo maestro-esclavo, en el que cada nodo tiene una función específica claramente definida. Esta organización favorece el modularidad, el aislamiento de errores, y la escalabilidad del sistema.

Nodo Maestro: Coordinador Central

El nodo maestro es el responsable de gestionar la lógica de control general del sistema y de tomar decisiones en función de los datos de entrada o comandos externos. Entre sus funciones principales se incluyen:

- Recepción de comandos desde la interfaz gráfica vía puerto serie.
- Procesamiento de trayectorias y cálculo de parámetros de movimiento (distancia, velocidad, sincronización).
- Generación de comandos de control (posición/velocidad) para cada nodo esclavo.

- Comunicación inalámbrica mediante ESP-NOW con los cuatro nodos esclavos.

Este nodo actúa como el centro de inteligencia del sistema, y su correcto funcionamiento garantiza la coordinación precisa entre todos los elementos de la plataforma.

Nodos Esclavos: Unidades de Control Local de Motor

Cada uno de los cuatro nodos esclavos se encarga del control de un motor individual. Están diseñados para funcionar como unidades autónomas de ejecución, que interpretan y aplican los comandos recibidos del maestro. Las responsabilidades asignadas a estos nodos incluyen:

- Recepción de mensajes ESP-NOW desde el maestro con los valores objetivo (velocidad, posición).
- Envío de comandos CAN al motor RMD-L-9025-35T mediante el controlador MCP2515.
- Recepción de realimentación desde el motor (velocidad actual, posición, temperatura, etc.).
- Reenvío de datos al maestro, si se requiere, para monitoreo o trazabilidad.

Cada nodo esclavo opera con una configuración de hardware específica:

- ESP32 como microcontrolador principal.
- Módulo MCP2515 para la interfaz CAN con el motor.
- Módulo reductor de tensión LM2596S para adaptar la alimentación del motor al voltaje operativo del ESP32 (de 12V a 5V).

Esta división funcional asegura que cada motor pueda operar de forma eficiente y sin interferencias, bajo la dirección sincronizada del maestro.

En conjunto, esta estructura distribuida permite mantener un alto nivel de determinismo, sincronización y fiabilidad, fundamentales para aplicaciones dinámicas como el movimiento de una cámara suspendida en tres dimensiones.

5.3.2 Montaje e integración Electrónica

El montaje e integración electrónica del sistema Spydercam representa una fase crítica en el desarrollo del proyecto, donde se materializa la arquitectura de hardware diseñada

previamente. Esta etapa consiste en interconectar correctamente todos los módulos electrónicos, ESP32, controladores CAN, fuentes de alimentación, y motores, asegurando un funcionamiento estable, seguro y mantenible. A continuación, se describen con detalle los distintos bloques de integración y consideraciones técnicas adoptadas.

Organización del sistema

Como se ha ido mencionando el sistema se compone de cinco nodos principales, distribuidos físicamente y conectados según una topología maestro-esclavo:

- 1 Nodo Maestro: ubicado generalmente en una posición central o de control externo.
- 4 Nodos Esclavos: montados físicamente junto a cada uno de los motores. Cada nodo esclavo tiene el control completo de un motor RMD-L-9025-35T, y está equipado con su propia electrónica, lo que reduce la dependencia de conexiones largas o centralizadas.

5.4 Implementación

Para la programación del sistema se ha usado el software de Arduino para programar los ESP32. Como se ha mencionado la idea es un sistema de robot de cables con un total de cuatro motores para controlar cada cable. El sistema se basa en cinco ESP32 que se comunican usando el protocolo ESP-NOW, uno comportándose como maestro y los otros cuatro como esclavos. La base del código es utilizar los cálculos de cinemática para realizar lo que requiera el usuario.

Se necesitan dos códigos, uno para la parte de los esclavos y otro para el maestro:

Código del esclavo

El presente apartado describe el funcionamiento del código desarrollado para los nodos esclavos del sistema de control distribuido del robot de cables. Cada uno de estos nodos está basado en un microcontrolador ESP32, cuya función principal es controlar un motor RMD-L-9025-35T a partir de los parámetros que recibe del nodo maestro a través del protocolo de comunicación inalámbrico ESP-NOW.

Cada ESP32 esclavo está conectado de forma individual a un motor, formando así una arquitectura modular en la que se utilizan un total de cuatro nodos esclavos para controlar los cuatro motores que conforman el sistema. Esta división del control en unidades independientes permite distribuir la carga computacional, facilitar el mantenimiento del sistema y aumentar la escalabilidad del conjunto. En la figura 36 se puede observar el diagrama de flujo del código.

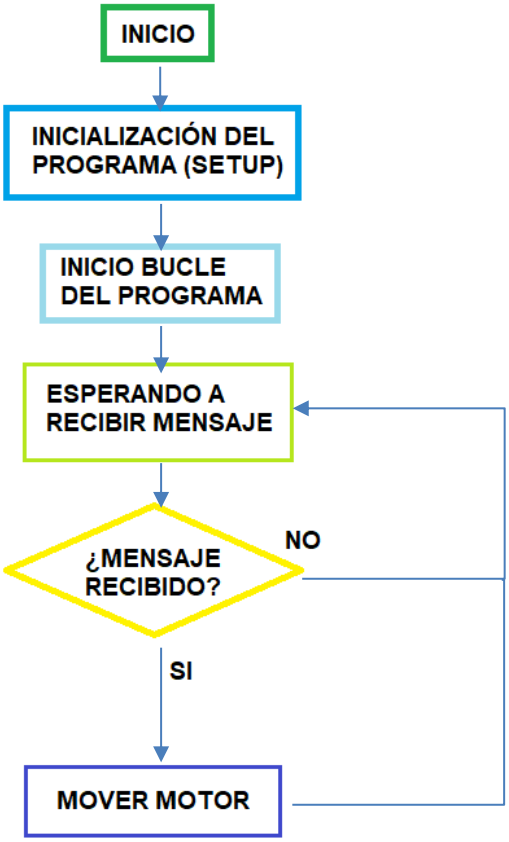


Figura 36. Diagrama flujo SLAVE

El código implementado en cada esclavo está orientado a recibir, interpretar y ejecutar las instrucciones enviadas por el maestro, las cuales contienen dos datos fundamentales: la velocidad (expresada en revoluciones por minuto) y la posición o ángulo objetivo (en grados) que el motor debe alcanzar para contribuir al movimiento coordinado de la plataforma móvil del robot.

El funcionamiento general del código se estructura en las siguientes fases:

- Inicialización del hardware y comunicación: En esta fase se configuran los periféricos necesarios para el control del motor. También se inicializa el protocolo ESP-NOW, registrando el nodo como receptor de mensajes provenientes del maestro.
- Recepción y procesamiento de datos: Una vez establecida la comunicación, el nodo esclavo se mantiene a la espera de mensajes entrantes. Al recibir una nueva instrucción, el código extrae los valores de velocidad y posición contenidos en el paquete y procede a traducir estos datos en comandos compatibles con el motor RMD.

- Control y supervisión del motor: A partir de los parámetros recibidos, el esclavo activa el motor, ajustando la velocidad de giro y el ángulo de rotación hasta alcanzar la longitud deseada del cable.

Este diseño permite que cada nodo funcione de forma autónoma en la ejecución de los movimientos, aunque siempre respondiendo a las decisiones centralizadas del nodo maestro, que se encarga de calcular la cinemática y de coordinar los movimientos de la plataforma. Esta separación de responsabilidades entre el procesamiento de alto nivel (maestro) y la ejecución de bajo nivel (esclavos) resulta especialmente adecuada para sistemas robóticos donde la precisión y la sincronización son críticas.

En resumen, el código del esclavo constituye un componente esencial del sistema, al encargarse de llevar a cabo el control directo de los actuadores del robot, transformando las órdenes abstractas en acciones físicas concretas de forma eficiente y sincronizada.

Código del maestro

El código desarrollado para el nodo maestro constituye el núcleo de control lógico y computacional del sistema robótico. Su principal responsabilidad es realizar los cálculos de cinemática necesarios para determinar las posiciones y velocidades objetivo de cada uno de los motores del robot de cables, en función de las órdenes introducidas por el usuario. A partir de estos cálculos, el maestro genera y transmite comandos personalizados a cada uno de los nodos esclavos mediante el protocolo ESP-NOW, permitiendo la ejecución coordinada de movimientos en la plataforma suspendida.

Este nodo centraliza la inteligencia del sistema y sirve como interfaz directa entre el operador y la estructura mecánica del robot. Está diseñado para interpretar una amplia gama de comandos que permiten controlar el sistema de forma precisa, flexible y eficiente, abarcando desde movimientos absolutos o relativos, hasta funciones de simulación, almacenamiento de posiciones y verificación del estado del sistema. En la figura 37 se puede observar el diagrama de flujo del código del maestro.

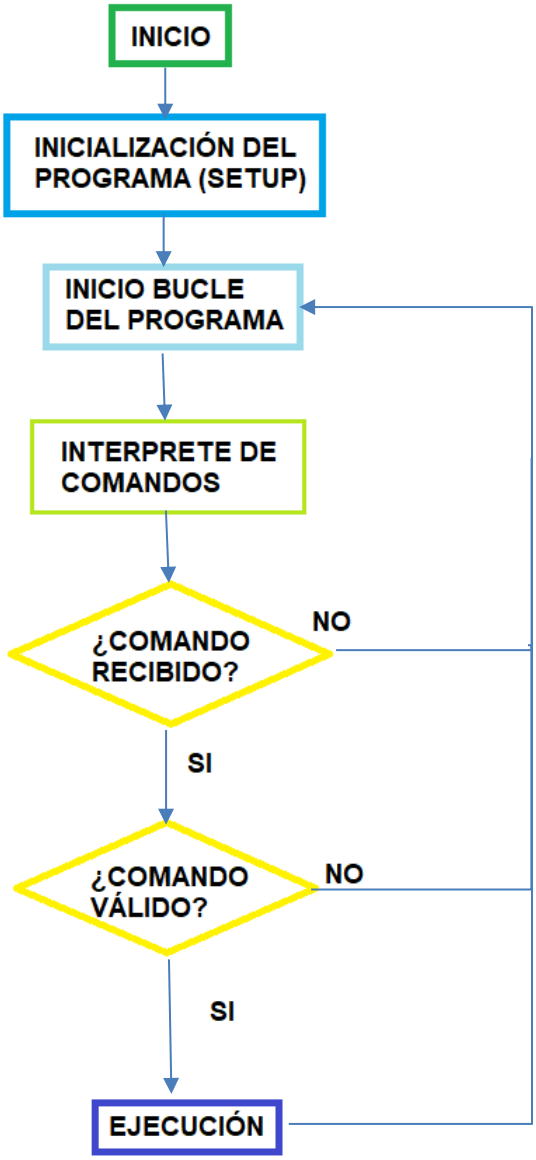


Figura 37. Diagrama flujo MASTER

El programa se basa en conectar con los otros nodos y a la esperar de recibir comandos y dependiendo del comando realiza el control de los motores para mandar a cada motor el movimiento correspondiente.

A continuación, se detallan los comandos disponibles:

- **move x y z**: Mueve la plataforma del robot suavemente hasta el punto indicado por las coordenadas (x, y, z). Este comando ejecuta los cálculos de cinemática

inversa necesarios para determinar los ángulos de los motores que logran dicha posición.

- **sim x y z**: Simula el movimiento hacia el punto (x, y, z) sin accionar físicamente los motores. Es útil para validar trayectorias o verificar que el punto sea alcanzable antes de ejecutar el movimiento real.
- **motorX vel pos**: Permite un control directo sobre un motor específico (donde X representa el número del motor, de 1 a 4). Se debe indicar la velocidad en RPM y la posición deseada en grados. Este modo es útil para pruebas individuales de motores o calibración manual.
- **delta dx dy dz**: Realiza un desplazamiento relativo desde la posición actual, sumando los valores (dx, dy, dz) a las coordenadas actuales. Es ideal para pequeños ajustes o movimientos incrementales.
- **moveL x y z**: Realiza un desplazamiento lineal desde el punto actual al punto indicado.
- **moveC x y ox oy**: Realiza una trayectoria circular en el plano xy desde el punto actual al punto indicado con centro de la circunferencia ox, oy. La altura se mantiene la de la posición actual.
- **moveAD x y z:** Realiza una trayectoria lineal con solo acelerando y desacelerando, muy útil para movimientos de distancias pequeñas, debido a que no tiene el tiempo suficiente para llegar a una velocidad constante.
- **moveAcel x y z t:** movimiento utilizando un perfil de velocidades trapezoidal donde se desplaza linealmente a la posición indicada. Se especifica con el parámetro t el tiempo de aceleración y desaceleración.
- **home**: Lleva la plataforma a la posición de origen (0, 0, 0), que normalmente representa la posición de referencia o inicio del sistema.
- **undo**: Revierte la última acción de movimiento, devolviendo la plataforma a su posición anterior. Esto resulta útil para corregir errores sin necesidad de recalcular manualmente la posición previa.
- **status**: Muestra el estado actual del sistema, incluyendo la posición de la plataforma, los ángulos de cada motor y otros parámetros relevantes. Es una herramienta importante para el monitoreo en tiempo real.
- **ping**: Verifica la conectividad y el estado del controlador maestro, confirmando que el sistema está activo y en condiciones de operar.
- **help**: Muestra un resumen de todos los comandos disponibles, junto con una breve descripción de su funcionalidad. Es útil para consulta rápida durante el uso del sistema.
- **save N**: Guarda la posición actual de la plataforma en una ranura de memoria identificada por el número N (de 0 a 9). Esto permite registrar ubicaciones frecuentes o críticas para fácil acceso posterior.
- **load N**: Recupera una posición previamente guardada en la ranura N y mueve la plataforma a esa ubicación.
- **listSave**: Lista todas las posiciones almacenadas en las ranuras de memoria, incluyendo sus coordenadas. Facilita la gestión de posiciones guardadas.
- **settime N**: Establece un nuevo tiempo (en milisegundos o segundos, según implementación) para ejecutar cada movimiento, permitiendo ajustar la velocidad global de las trayectorias.

Estos comandos han sido diseñados para abarcar de forma integral las operaciones requeridas en la manipulación de un robot de cables, brindando al usuario tanto precisión como flexibilidad. La interfaz de control busca ser intuitiva y eficiente, facilitando tanto tareas de prueba y calibración como la ejecución de movimientos complejos.

El código del maestro ha sido diseñado con un enfoque en la modularidad, legibilidad y facilidad de uso. Gracias a la separación clara entre los cálculos cinemáticos y el envío de comandos a los esclavos, el sistema permite tanto simulaciones como ejecuciones reales sin necesidad de alterar el flujo principal del programa.

Además, la implementación de una memoria interna para guardar posiciones facilita la programación de trayectorias repetitivas y mejora la experiencia del usuario en tareas de prueba o validación. La posibilidad de ejecutar comandos de forma incremental (como delta) o manual (motorX) también hace que el sistema sea versátil tanto para entornos educativos como industriales.

Capítulo 6: Desarrollo y evolución de las interfaces gráficas

En este capítulo se abordará el diseño y la evolución de las interfaces gráficas desarrolladas para el control y monitoreo del sistema Spydercam. Se describirán las herramientas y softwares utilizados, destacando cómo se han integrado diversas plataformas como Excel y Python para facilitar la gestión, el procesamiento y la visualización de datos esenciales del sistema. A lo largo del desarrollo, se buscará optimizar la interacción del usuario con el sistema, simplificando el manejo y mejorando la precisión en el cálculo y análisis de los parámetros operativos. Este enfoque ha permitido no solo automatizar procesos, sino también ofrecer una interfaz amigable que potencia la eficiencia y confiabilidad en el control de la Spydercam.

6.1 Introducción

La interfaz desarrollada tiene como objetivo principal facilitar la interacción entre el usuario y el nodo maestro del sistema robótico, el cual se encarga de realizar los cálculos de cinemática inversa y de coordinar los movimientos de la plataforma suspendida mediante la comunicación con los nodos esclavos a través del protocolo ESP-NOW.

Dada la complejidad y versatilidad de los comandos disponibles, que incluyen desde movimientos absolutos y relativos, hasta simulaciones, gestión de posiciones y monitoreo del estado del sistema, se consideró fundamental diseñar una interfaz gráfica que permitiera ejecutar estas funciones de forma intuitiva, ordenada y accesible para el usuario.

Este capítulo detalla cómo se ha abordado el diseño de la interfaz desde sus primeras etapas, incluyendo la elección de herramientas, la creación de prototipos, las iteraciones realizadas en base a pruebas y retroalimentación, y las funcionalidades finales implementadas. También se analizan las decisiones de diseño que permitieron optimizar la usabilidad, accesibilidad y adaptabilidad de la interfaz a distintos contextos de operación.

6.2 Fase inicial de diseño de la interfaz

El proceso de desarrollo de la interfaz gráfica comenzó con un enfoque experimental, utilizando herramientas accesibles y flexibles para validar los fundamentos del sistema antes de su implementación en hardware. En esta etapa inicial, se desarrolló una hoja de cálculo en Microsoft Excel (ver figura 38) que sirvió tanto como entorno de prueba para los cálculos de cinemática inversa como una primera versión rudimentaria de interfaz de usuario.

La hoja de cálculo permitía al usuario introducir coordenadas tridimensionales (x, y, z) correspondientes a la posición deseada de la plataforma suspendida. A partir de estas coordenadas, Excel calculaba automáticamente las longitudes de los cables y los ángulos necesarios para cada uno de los motores, replicando de forma simplificada el comportamiento esperado del sistema tipo spydercam. Este enfoque facilitó una validación temprana del modelo matemático y permitió identificar posibles errores o inconsistencias sin necesidad de realizar pruebas físicas.

1															
2															
3			Posicion final								Dimensiones (m)		radio	0,045	
4		x	0							W	2,8				
5		y	0							D	3		plataforma	0,15	
6		z	0							H	2,34				
7															
8															
9		Cable	Longitud		Motor	Grados				L0	3,04332877		offset	0,075	
10		L1	3,04332877		M1	0									
11		L2	3,04332877		M2	0									
12		L3	3,04332877		M3	0									
13		L4	3,04332877		M4	0									
14															

Figura 38. Excel de cálculos de cinemática

Además de los cálculos, ha incluido una pequeña interfaz visual con Python que tomaba los datos calculados en Excel y muestra la posición estimada de la plataforma en un esquema simplificado, junto con los valores de longitud y ángulo de cada motor (ver figura 39). Aunque básica, esta representación gráfica ofrecía una visión inmediata del efecto de los parámetros ingresados en el Excel y ha servido como primer intento de traducir comandos numéricos en una experiencia más comprensible y visual para el usuario.

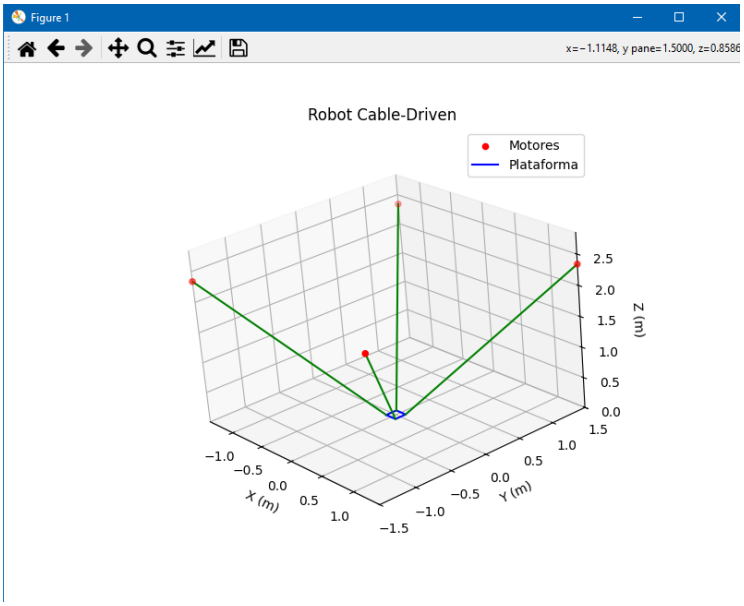


Figura 39. Simulación a partir de datos del Excel

Esta herramienta ha resultado fundamental en la toma de decisiones durante las fases iniciales del proyecto, ya que permitió ajustar parámetros críticos y afinar la precisión del modelo cinemático antes de su implementación definitiva en el microcontrolador ESP32. Una vez validados los cálculos en Excel, se procedió a enviar los valores de ángulo correspondientes a cada motor directamente al ESP32. Tras realizar algunas modificaciones en el código de control, se comprobó que la plataforma alcanzaba efectivamente la posición esperada, lo que confirmó la validez de los cálculos previos. Esta validación representó un punto de inflexión en el desarrollo, al demostrar que el modelo era funcional no solo en simulación, sino también en condiciones reales. Este fue el primer paso clave para la posterior integración completa de la cinemática inversa y la ejecución de trayectorias automáticas, que se han explicado en el capítulo anterior.

En resumen, el uso de Excel no solo ha sido útil como entorno de simulación, sino que también ha desempeñado un papel clave en la validación del modelo de cinemática inversa. Al permitir el cálculo directo de los ángulos de cada motor a partir de coordenadas espaciales, la herramienta facilitó la comprobación empírica del sistema: los ángulos generados en la hoja de cálculo se enviaron manualmente al microcontrolador ESP32, y tras algunas adaptaciones en el firmware, se pudo observar cómo la plataforma alcanzaba la posición esperada en el espacio. Este proceso permitió corroborar que los cálculos eran correctos y que el sistema físico respondía de forma coherente, incluso antes de integrar completamente la lógica de cinemática inversa en el propio código del microcontrolador. Por tanto, esta fase no solo sentó las bases en

términos de lógica y diseño visual, sino que ha representado un puente crucial entre la simulación teórica y la implementación real del sistema.

6.3 Desarrollo de la primera versión de la interfaz

La primera versión de la interfaz gráfica fue desarrollada utilizando Python, apoyándose en la biblioteca Tkinter para la construcción de los elementos visuales. Esta elección respondió principalmente a la necesidad de disponer de una herramienta sencilla, ligera y de rápida implementación, ideal para validar conceptos en una etapa temprana del proyecto. Tkinter permitió construir una interfaz básica sin dependencias externas, con los elementos necesarios para la interacción directa con el sistema. En esta fase, el objetivo principal era lograr una comunicación estable con el microcontrolador ESP32 y en facilitar la interacción básica con el sistema robótico mediante comandos manuales. Aunque se trataba de una versión inicial, permitió validar la estructura de comunicación y probar los primeros movimientos del robot en condiciones reales.

La interfaz (ver figura 40) consistía principalmente en un terminal de texto que mostraba en tiempo real los mensajes enviados y recibidos entre el ordenador y el ESP32, así como una barra de entrada para introducir comandos de forma manual. Esto replicaba el comportamiento de una consola serie, pero integrándolo dentro de un entorno gráfico más ordenado y accesible.

Figura 40. Interfaz Gráfica inicial

Se ha añadido también bloques funcionales específicos para el control directo de los motores. El usuario podía introducir manualmente la velocidad (en RPM) y la posición angular (en grados) para cada uno de los cuatro motores, utilizando los campos disponibles en la sección "Control de motores". Al presionar el botón "Enviar" correspondiente, se generaba y transmitía automáticamente el comando adecuado al ESP32, facilitando pruebas de funcionamiento y calibración individual de los motores.

Como se muestra en la imagen, la interfaz incluía también una sección para el control posicional en coordenadas espaciales, donde era posible ingresar valores X, Y y Z para mover la plataforma del robot mediante un comando de cinemática inversa. Aunque en esta etapa el sistema no realizaba visualizaciones avanzadas, sí mostraba actualizaciones constantes del estado del sistema. Cada vez que se ejecutaba una acción, la interfaz consultaba y mostraba en pantalla la posición actual de la plataforma, así como los valores de velocidad y posición de cada motor. Esta retroalimentación en tiempo real, aunque sencilla, ha resultado crucial para validar que el sistema respondía correctamente a las órdenes enviadas.

Finalmente, se ha incorporado un conjunto básico de botones para funciones complementarias como "Ping" (para verificar la conexión con el ESP32), "Ver" (para obtener el estado actual del sistema), "Limpiar" (para limpiar el terminal de mensajes), y "Conectar", que permitía seleccionar el puerto serie correcto y establecer la conexión con el nodo maestro.

Con la necesidad de añadir nuevas funciones, mejorar la experiencia del usuario y construir una interfaz más robusta y escalable, se ha optado en fases posteriores por migrar el desarrollo a PyQt6. Esta herramienta ofrece mayores capacidades a nivel visual y de arquitectura de software, lo que permitió abordar funcionalidades más complejas con una estructura modular y profesional.

En conjunto, esta primera versión sentó las bases de la arquitectura general de la interfaz: modular, funcional y centrada en el usuario. Si bien su diseño visual era simple, cumplía con el objetivo fundamental de ofrecer una capa de interacción cómoda y efectiva para validar tanto la lógica del sistema como su funcionamiento físico, y sirvió como punto de partida para el diseño de versiones posteriores más avanzadas.

6.4 Iteraciones y mejoras sucesivas

Tras la validación inicial del sistema mediante la primera interfaz en Tkinter, el desarrollo avanzó hacia una versión más completa y profesional, implementada en Python utilizando la biblioteca PyQt6. Este cambio respondió a la necesidad de disponer de una herramienta más potente y versátil, capaz de gestionar interfaces gráficas complejas de manera estructurada. PyQt6 ofrecía una arquitectura orientada a objetos más robusta, mayor control sobre el diseño visual y una integración más fluida de funcionalidades avanzadas gracias a su sistema de señales y slots. Esta nueva etapa tuvo como objetivo principal mejorar la experiencia de usuario, ampliar la funcionalidad y establecer una estructura modular que permitiera mantener y escalar fácilmente la aplicación a medida que evolucionaba el proyecto.

La aplicación se ha diseñado con la librería de PyQt6, una biblioteca de Python que permite crear aplicaciones con interfaces visuales profesionales y altamente personalizables.

El código está dividido de tal forma que todas las funciones correspondientes a cada ventana y pestaña están juntas para una mejor comprensión del código y futuras actualizaciones que se puedan realizar.

Características principales de PyQt6 utilizadas son:

- Widgets y Controles: PyQt6 proporciona una amplia variedad de widgets (controles visuales) listos para usar, como botones, etiquetas, campos de texto, menús, barras de herramientas y pestañas. En la aplicación se emplean estos widgets para construir la interfaz de manera modular y funcional.
- Barra de Menú (QMenuBar y QMenu): La aplicación incorpora una barra de menú principal, que es un componente esencial para la navegación y acceso rápido a las funciones. PyQt6 facilita la creación y gestión dinámica de menús y sus acciones (QAction).
- Ventanas y Contenedores (QMainWindow, QWidget, QTabWidget): PyQt6 permite crear ventanas principales y secundarias, así como paneles con pestañas para organizar contenido de forma clara y accesible. Esto mejora la experiencia del usuario al segmentar las funcionalidades en diferentes secciones.
- Sistema de Señales y Slots: PyQt6 utiliza un mecanismo de comunicación interno basado en señales y slots, que permite que los widgets y objetos interactúen entre sí de manera eficiente. En la aplicación, este sistema se ha empleado para

manejar eventos como clics de botón, selección de menú o cambios en las pestañas.

- Diseño y Layouts: La gestión del diseño dentro de las ventanas se realiza con layouts (QVBoxLayout, QHBoxLayout, etc.), que facilitan la organización dinámica y adaptativa de los widgets, haciendo que la interfaz sea responsiva y estética.
- Modularidad y Orientación a Objetos: PyQt6 se integra perfectamente con la programación orientada a objetos de Python, permitiendo crear clases que representan cada ventana o componente de la interfaz. Esto favorece la mantenibilidad y escalabilidad del código.

Ventajas de PyQt6 para el proyecto:

- Multiplataforma: Qt es compatible con Windows, macOS y Linux, lo que garantiza que la aplicación pueda ejecutarse en distintas plataformas sin modificar el código base.
- Madurez y Comunidad: PyQt6 está basado en Qt, que cuenta con años de desarrollo y una amplia comunidad, asegurando estabilidad y soporte.
- Flexibilidad para interfaces complejas: La gran variedad de widgets y opciones de personalización permiten adaptar la interfaz a las necesidades específicas del proyecto.

Una de las principales mejoras introducidas fue la incorporación de un sistema de pestañas y ventanas dentro de la propia interfaz. A través de una barra de menú superior, el usuario podía acceder rápidamente a diferentes secciones, lo que facilitaba la navegación y el uso eficiente del programa. Aunque en esta versión inicial con PyQt6 no se profundizó excesivamente en la personalización visual o en la lógica avanzada de interacción, sí se establecieron las bases para una interfaz mucho más organizada y usable.

La interfaz se dividía en varias pestañas, cada una centrada en una funcionalidad específica:

- Conexión: en esta pestaña el usuario podía seleccionar y gestionar el puerto serie, así como conectar o desconectar la comunicación con el ESP32. Se implementaron botones específicos para simplificar estas acciones.
- Control de motores: replicando la funcionalidad básica de la versión anterior, esta sección permitía controlar individualmente cada uno de los motores. El

usuario podía ingresar la posición angular (en grados) y la velocidad (en RPM), y enviar el comando correspondiente al microcontrolador.

- Comandos de usuario: en esta pestaña se organizaron distintos comandos de alto nivel como move, home, status o ping. En el caso del comando move, se incorporaron campos para ingresar las coordenadas espaciales X, Y y Z, los cuales se convertían internamente en comandos de cinemática inversa al ser enviados. La disposición en botones separados ayudaba a mejorar la claridad y reducir errores en el uso diario.

- Terminal: se incluyó una pestaña específica con dos terminales independientes, uno para mostrar los mensajes recibidos desde el ESP32 y otro para los mensajes generados internamente por Python, como logs o respuestas del sistema. En la parte inferior se mantenía una barra de entrada manual para que el usuario pudiera introducir directamente cualquier comando, replicando así la funcionalidad estilo consola de versiones anteriores, pero de forma más clara y separada.

Aunque esta versión aún no era definitiva ni completamente refinada en términos de diseño gráfico, representó un salto importante en la estructura y organización de la aplicación. El uso de PyQt6 permitió una mayor flexibilidad para gestionar eventos, separar la lógica de interfaz de la lógica de control y preparar la aplicación para futuras mejoras, como validaciones automáticas, historial de comandos o visualización gráfica en tiempo real.

Estas mejoras sucesivas sentaron las bases para una interfaz más robusta y adaptable, consolidando el camino hacia una herramienta final que no solo cumpliera con los requisitos técnicos del proyecto, sino que ofreciera una experiencia de usuario clara, intuitiva y funcional.

6.5 Versión final de la interfaz

La versión final de la interfaz gráfica representa la culminación del desarrollo del software de control del sistema robótico, integrando todas las funcionalidades necesarias para operar, programar y simular el comportamiento del robot de cables de manera eficaz y amigable para el usuario.

Esta versión, desarrollada íntegramente en Python utilizando PyQt6, está estructurada como un sistema de pestañas funcionales distribuidas en una única ventana principal, donde cada pestaña agrupa herramientas y controles relacionados con un área

específica del sistema: conexión serial, control de motores, comandos de usuario, programación, terminales y simulación.

A lo largo del desarrollo de la interfaz gráfica, se han mantenido algunos elementos clave de las versiones iniciales, como el terminal dual, el control individual de motores o la entrada manual de comandos, ya que resultaron eficaces desde las primeras pruebas. Sin embargo, muchos de estos componentes fueron mejorados o reorganizados para integrarse de forma más coherente dentro de un entorno visual unificado y profesional.

Por ejemplo, la funcionalidad de conexión al puerto serie, que en versiones anteriores estaba limitada a una sección aislada, fue completamente rediseñada con validación dinámica y control de acceso al resto de la aplicación. Del mismo modo, el control por comandos, que originalmente se realizaba a través de una consola simple, fue convertido en una interfaz gráfica con botones interactivos, lo que facilita su uso sin necesidad de conocimientos técnicos.

Otras funcionalidades que en versiones anteriores se encontraban de forma dispersa o poco desarrollada, como la programación de trayectorias o la simulación gráfica, han sido incorporadas como pestañas independientes, con interfaces dedicadas, soporte de validación, ayuda contextual y opciones de importación/exportación de datos. Esta evolución progresiva no solo ha incrementado el número de herramientas disponibles, sino que ha mejorado la coherencia, accesibilidad y escalabilidad del sistema. A continuación, se procede a explicar las mejoras en la interfaz gráfica.

En la parte superior de la interfaz se encuentra una barra de menú clásica, compuesta por las siguientes secciones:

- Archivo: para importar, exportar o guardar programas y configuraciones.
- Herramientas: ofrece accesos a funciones específicas como el estado de los motores o la ayuda de programación.
- Ajustes: permite modificar parámetros globales, como el modo oscuro.
- Ayuda: contiene el manual de usuario y la información sobre la aplicación.

Gracias a esta organización, el entorno se mantiene claro, intuitivo y accesible, incluso cuando se integran múltiples funcionalidades avanzadas. A continuación, se describen en detalle las distintas secciones y capacidades de esta versión final, cada uno de estos apartados representa una pestaña distinta dentro de la aplicación.

Conexión Serial

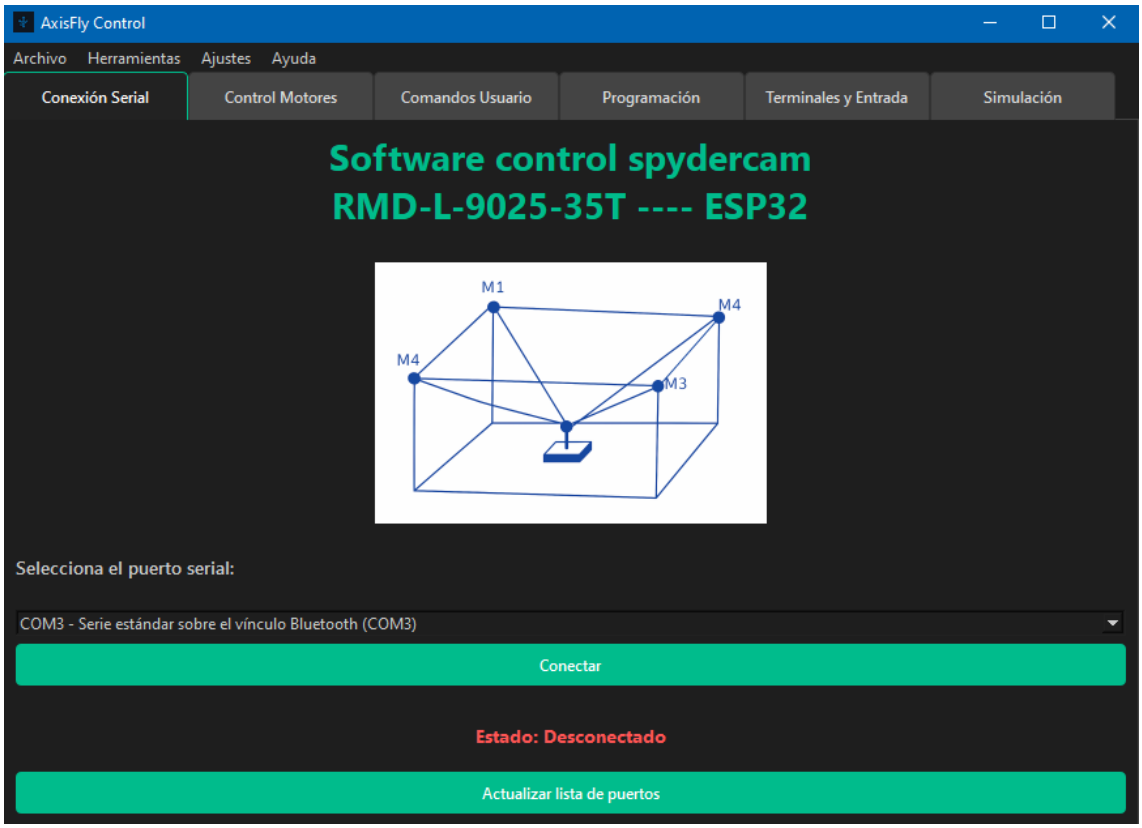


Figura 41. Aplicación final – Conexión serial

En la versión final, la pestaña Conexión Serial (ver figura 41) ha sido completamente rediseñada para ofrecer una experiencia más intuitiva, robusta y funcional en comparación con las versiones anteriores. En los prototipos iniciales, la conexión con el ESP32 se realizaba mediante una simple barra desplegable y un botón de conexión sin validación avanzada ni retroalimentación clara sobre el estado del enlace. Esta solución, aunque funcional, era limitada y poco informativa para el usuario.

En esta versión definitiva, se ha implementado un sistema dinámico de gestión de puertos, que detecta automáticamente los puertos serie disponibles en el sistema y permite actualizarlos manualmente mediante un botón de “Actualizar Puertos”, útil si el ESP32 se conecta tras el arranque de la aplicación. Una vez seleccionado el puerto deseado, el usuario puede establecer o cerrar la conexión mediante un botón claro de “Conectar / Desconectar”.

Como mejora clave respecto a versiones anteriores, se ha incluido una validación del estado de conexión: solo al establecer correctamente la comunicación con el ESP32 se habilitan el resto de las pestañas de la interfaz (como control de motores, comandos de usuario, etc.). Esta restricción evita errores accidentales o el envío de comandos sin conexión activa.

Además, la interfaz incluye un esquema visual del montaje del robot, donde se indican las posiciones y numeración de los motores, proporcionando una referencia rápida que ayuda tanto en la conexión física como en la interpretación de los controles posteriores.

Control de motores

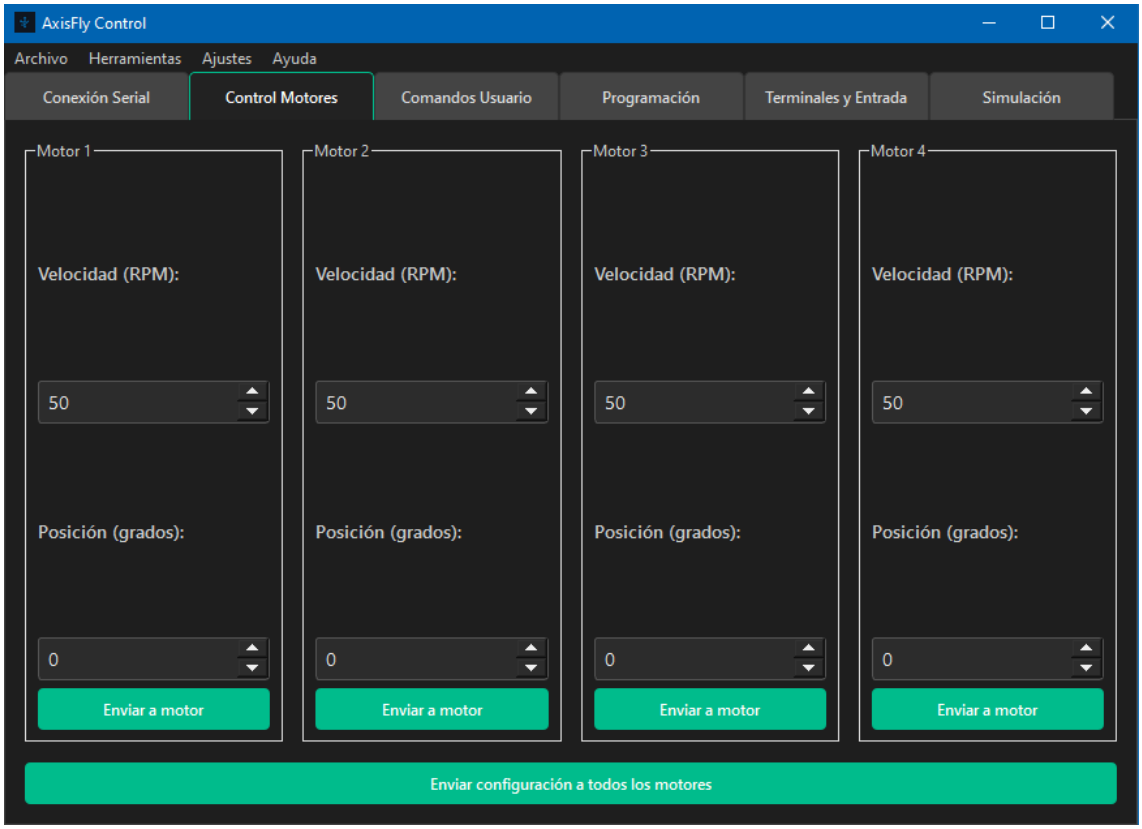


Figura 42. Aplicación final – Control de motores

La pestaña Control de motores ha evolucionado notablemente respecto a las primeras versiones del software, pasando de una interfaz muy básica a una herramienta avanzada y flexible para el control individual y simultáneo de los motores del robot. Se puede observar su resultado en la Figura 42.

En las primeras implementaciones con Tkinter, esta funcionalidad se limitaba a unos pocos campos numéricos donde el usuario introducía manualmente la velocidad (RPM) y la posición angular (grados) de cada motor, sin validación ni confirmación visual de lo enviado. Aunque era funcional para pruebas básicas y calibraciones iniciales, su diseño era estático, poco intuitivo y visualmente limitado.

En la versión final desarrollada con PyQt6, esta pestaña se ha reorganizado por completo, proporcionando una interfaz mucho más estructurada y visualmente clara. El usuario dispone ahora de:

- Campos independientes para cada motor (1 al 4), donde puede establecer velocidad y posición.
- Opción para enviar comandos de forma individual por motor o de manera conjunta, aplicando todos los parámetros definidos a la vez, lo que resulta especialmente útil para posicionamientos precisos.
- Botones de acción claros para cada motor y para el envío global.
- Sistema de validación y mensajes de retroalimentación que confirman los datos enviados.

Una de las mejoras más significativas ha sido la inclusión de funciones para guardar, importar o exportar configuraciones de estado de los motores en formato. Esto permite reutilizar perfiles de calibración, restaurar configuraciones previas o compartir ajustes fácilmente.

Este rediseño ha convertido la pestaña de control de motores en una herramienta versátil, adecuada tanto para pruebas unitarias como para ajustes precisos en entornos reales, facilitando enormemente las tareas de calibración y posicionamiento.

Comandos de usuario

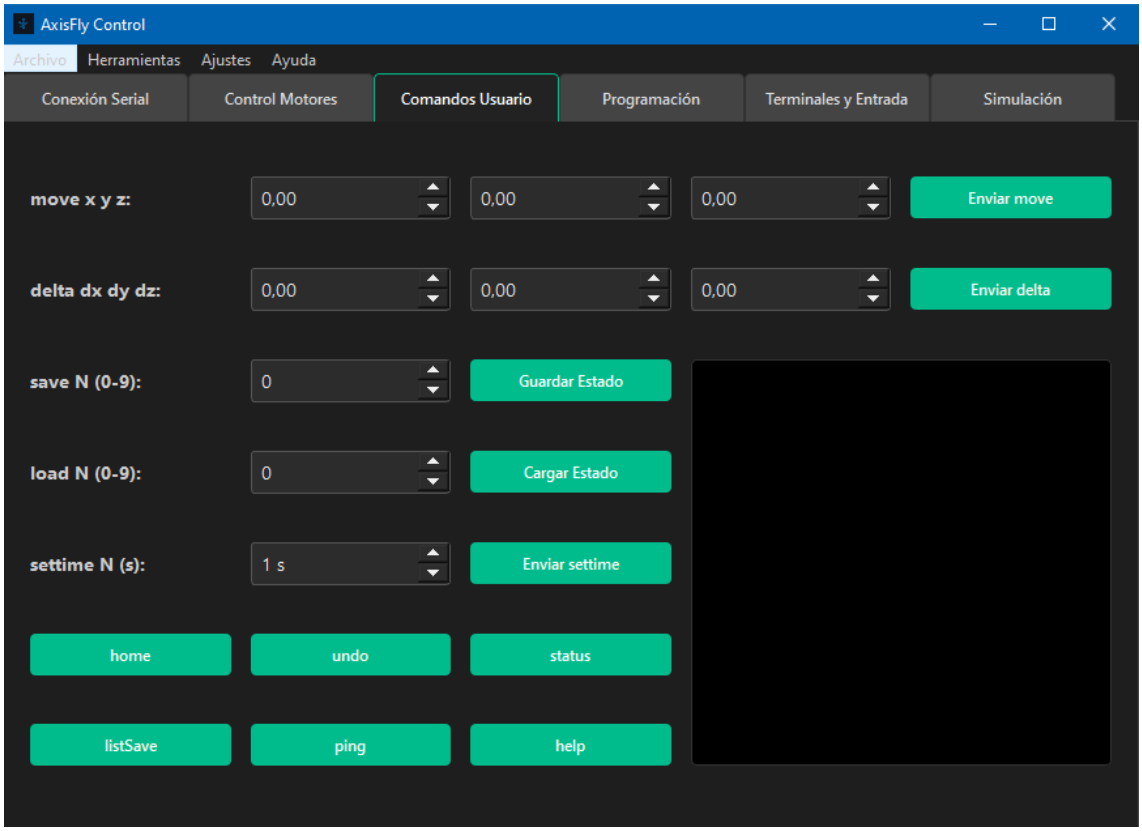


Figura 43. Aplicación final – Comandos de usuario

La pestaña Comandos de usuario fue concebida para facilitar el envío de instrucciones al robot sin necesidad de recordar la sintaxis exacta de los comandos (ver Figura 43). En versiones anteriores del software, esta funcionalidad se realizaba manualmente a través de una barra de texto, replicando el comportamiento de una consola serie. Aunque esta aproximación ofrecía flexibilidad para usuarios con experiencia técnica, no era adecuada para entornos donde se necesitara rapidez, precisión o facilidad de uso.

En la versión final, esta funcionalidad ha sido totalmente renovada e integrada en una pestaña específica. Ahora, el usuario puede acceder a todos los comandos implementados en el nodo maestro (como move, home, status, save, load, etc.) mediante una serie de botones interactivos agrupados por funcionalidad. Cada uno de estos botones ejecuta directamente el comando correspondiente con los valores introducidos en los campos relacionados.

Además, se ha añadido una ventana de retroalimentación en el lateral inferior derecho, donde se muestra el comando que ha sido enviado al ESP32. Esto permite al usuario verificar rápidamente qué instrucción se ha transmitido, ayudando a evitar errores y a mantener un seguimiento claro de las acciones realizadas.

Gracias a esta mejora, los comandos más utilizados del sistema pueden ejecutarse de forma rápida, clara y sin margen de error, contribuyendo a una experiencia de usuario mucho más fluida.

Programación

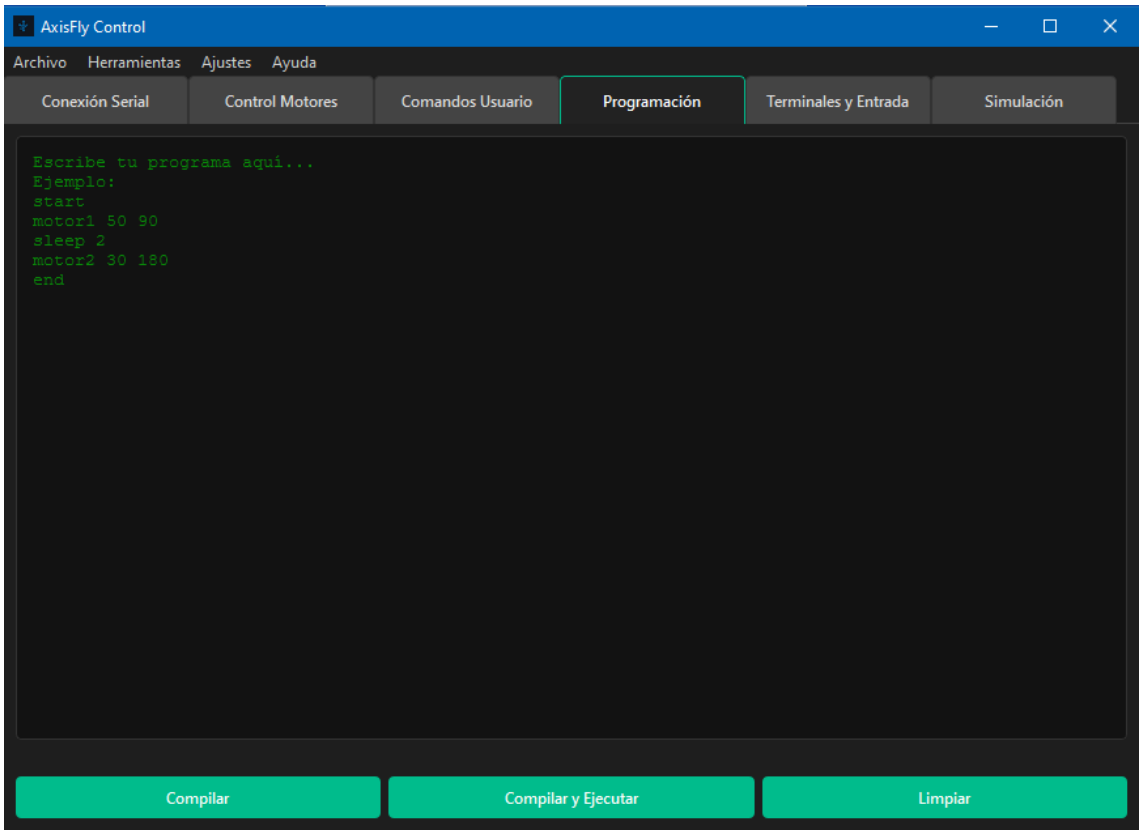


Figura 44. Aplicación final – Programación

La pestaña Programación es una de las incorporaciones más potentes en la versión final de la interfaz, ya que permite al usuario diseñar, depurar y ejecutar trayectorias complejas de forma estructurada (ver figura 44). Esta funcionalidad no existía en las primeras versiones de la aplicación, que estaban limitadas al envío manual de comandos individuales y carecían de cualquier capacidad de automatización.

En esta versión final, se ha desarrollado una interfaz de edición tipo editor de texto, donde el usuario puede escribir un pequeño programa compuesto por una secuencia de comandos del sistema, tal como se explicó en el capítulo correspondiente al nodo maestro. Este entorno simula un pequeño lenguaje propio para planificar el comportamiento del robot.

Principales características:

- Área de edición con soporte para múltiples líneas de código.
- Compilador integrado que analiza sintácticamente el código antes de ejecutarlo.

- Botones de acción para: Compilar, verificar que el código sea válido. Compilar y ejecutar, ejecuta el programa si no hay errores. Limpiar, borra el contenido del editor.
- Funcionalidad de importar, exportar y guardar programas.

Una característica destacable es la ventana de ayuda al programador, accesible desde Herramientas, que ofrece una explicación detallada de todos los comandos aceptados.

Con esta herramienta, la interfaz deja de ser solo un entorno de control para convertirse también en una plataforma de desarrollo ligera, ideal para ensayar secuencias, validar comportamientos y ajustar rutinas de funcionamiento del robot.

Terminales y entrada de comandos

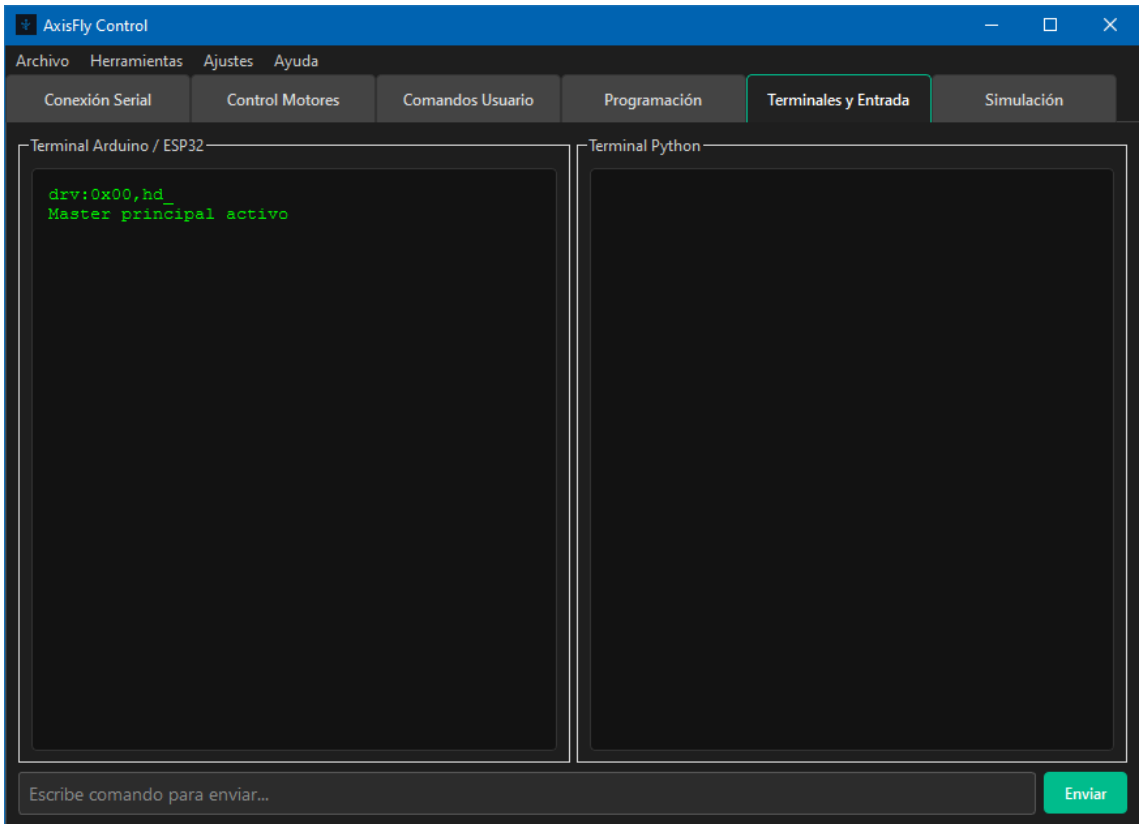


Figura 45. Aplicación final – Terminales y entrada de comandos

La pestaña Terminales y entrada de comandos cumple una doble función: por un lado, permite visualizar en tiempo real la comunicación entre la aplicación y el microcontrolador ESP32, y por otro, ofrece una forma alternativa y directa de enviar comandos manuales al sistema (ver figura 45). Aunque ya existía una versión

simplificada de esta funcionalidad en las primeras etapas del proyecto, en esta versión final se ha refinado y ampliado significativamente.

En la primera versión, el terminal era un único cuadro de texto que mostraba los mensajes entrantes, y una barra de entrada permitía enviar instrucciones. Esto funcionaba, pero carecía de estructura y no distinguía claramente entre mensajes generados por Python y respuestas del ESP32, lo que dificultaba el diagnóstico y la trazabilidad.

En la versión final, esta funcionalidad se ha trasladado a una pestaña dedicada que incluye dos terminales separados, uno muestra los mensajes generados por el código Python (por ejemplo, confirmaciones internas o mensajes de depuración) y el otro muestra exclusivamente las respuestas y datos recibidos del ESP32. Estos terminales muestran los mensajes resultantes de forma inmediata, permitiendo comprobar visualmente que el comando ha sido interpretado correctamente.

Se sigue conservando la barra de entrada de comandos en la parte inferior de la pestaña, donde el usuario puede escribir manualmente cualquier comando compatible con el sistema, como si se estuviera en la interfaz del propio software de Arduino.

Esta estructura proporciona una vista clara y separada del flujo de información, mejorando el control, la transparencia y la capacidad de depuración del sistema. La pestaña sigue cumpliendo una función clave durante el desarrollo y pruebas del sistema, permitiendo al usuario técnico mantener un control directo y supervisar con precisión el comportamiento interno del robot.

Simulación

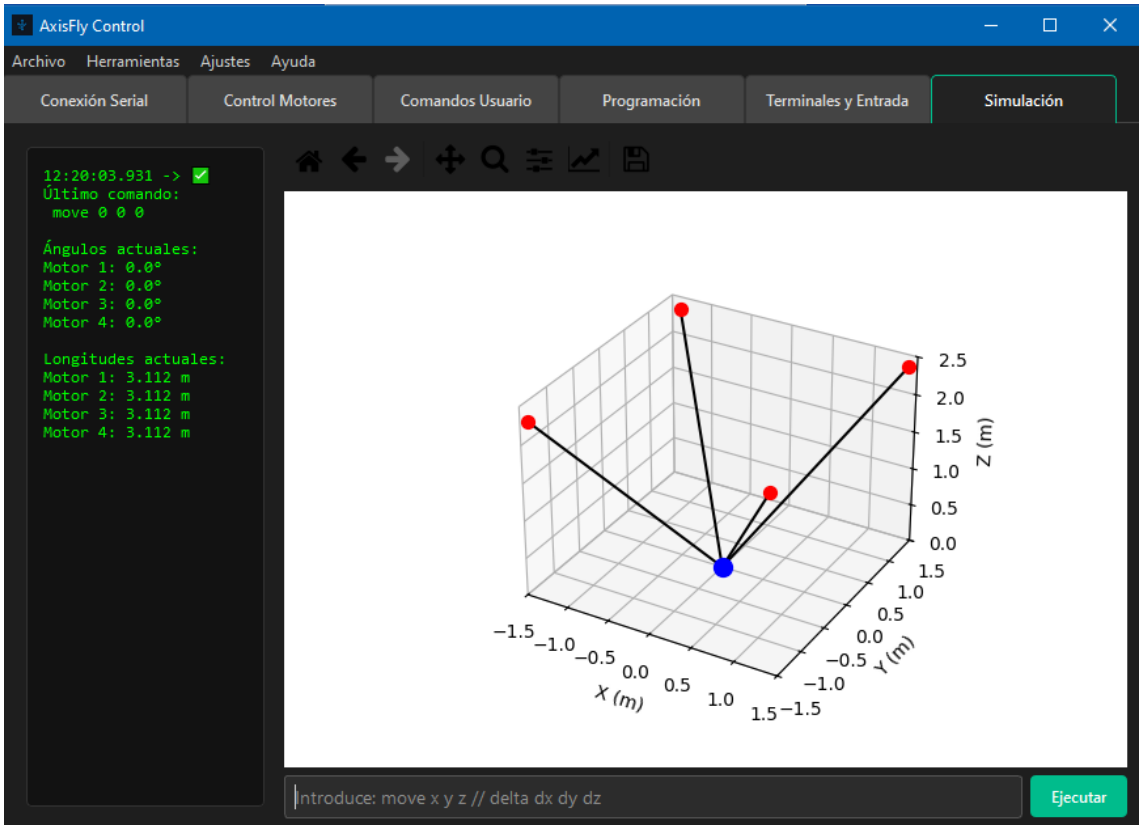


Figura 46. Aplicación final – Simulación

La pestaña Simulación representa una evolución directa del entorno de pruebas desarrollado inicialmente en Excel, pero ahora completamente integrado en la interfaz gráfica (ver figura 46). Su objetivo principal es permitir al usuario visualizar de forma interactiva la posición de la plataforma (spydercam) a partir de coordenadas espaciales, mostrando también los ángulos de los motores y las longitudes necesarias de los cables.

A diferencia de las versiones anteriores, donde solo se mostraban valores numéricos o resultados de cinemática en tablas, esta versión cuenta con un gráfico 3D interactivo que proporciona una representación espacial mucho más intuitiva. Aunque esta simulación no muestra trayectorias dinámicas en tiempo real, sí permite validar si una posición deseada es alcanzable y cómo se comportan los motores para llegar a ella.

Gracias a esta pestaña, el usuario puede experimentar con posiciones, validar configuraciones y entender mejor cómo interactúan los motores sin mover físicamente la plataforma, lo que aporta seguridad, eficiencia y un enfoque más didáctico al uso del sistema.

Para facilitar la instalación y uso de la aplicación en cualquier ordenador con sistema operativo Windows, se procedió a la creación de un archivo ejecutable independiente (.exe). Esto permite que el usuario final pueda ejecutar la aplicación sin necesidad de instalar Python ni las dependencias manualmente. Para ejecutarla no es necesario instalar nada simplemente ejecuta el .exe y se abrirá la interfaz para que se pueda empezar a trabajar.

La versión final de la interfaz gráfica representa la culminación de un proceso iterativo de mejora continua, en el que cada etapa del desarrollo anterior, desde las pruebas iniciales en Excel hasta las primeras aplicaciones en Tkinter, aportó conocimientos valiosos sobre las necesidades reales del sistema y del usuario.

A través de la implementación en PyQt6, la aplicación ha alcanzado un grado de madurez funcional y visual que permite un control preciso, seguro y eficiente del robot de cables. La integración de pestañas especializadas, junto con una barra de menú estructurada, proporciona una experiencia de usuario clara, profesional y organizada.

En conjunto, esta interfaz no solo cumple con los objetivos funcionales del proyecto, sino que también lo dota de una herramienta intuitiva y profesional, adecuada tanto para entornos de desarrollo como para demostraciones, educación o futuras extensiones. Se trata, por tanto, de una base sólida para el control completo del sistema robótico, con un diseño modular preparado para crecer y adaptarse a nuevas funcionalidades.

Capítulo 7: Resultados

El desarrollo progresivo de las interfaces gráficas, desde sus versiones más básicas hasta la aplicación final implementada en PyQt6, ha sido clave para validar, controlar y visualizar el funcionamiento del sistema robótico basado en cables. Este capítulo ha servido no solo para documentar su evolución técnica, sino también para demostrar de forma práctica la utilidad del software desarrollado.

La versión final de la interfaz se ha ejecutado en un entorno Windows con conexión directa vía puerto serie al microcontrolador ESP32. El sistema físico del robot de cables se encontraba completamente operativo, permitiendo realizar pruebas de movimiento, calibración, simulación y ejecución de trayectorias completas.

Se establecieron distintos escenarios de prueba para verificar el comportamiento del sistema bajo diferentes condiciones de uso:

- Control individual de motores.
- Envío de comandos manuales (move, delta, home, etc.).
- Carga y ejecución de trayectorias programadas.
- Validación de posiciones a través de la simulación 3D.
- Análisis en tiempo real de la comunicación y el estado del robot.
- Descripción de los experimentos realizados
- Pruebas de conexión y respuesta del sistema

Se realizaron múltiples ciclos de conexión y desconexión entre la aplicación y el ESP32, utilizando diferentes puertos y simulando errores típicos (como ESP32 no conectado, o cambio de puerto tras el arranque). La interfaz respondió correctamente, reconociendo dispositivos activos y mostrando el estado de conexión.

Para probar el control manual de los motores se introdujeron diferentes combinaciones de velocidad y posición para cada motor, verificando su respuesta tanto de forma individual como simultánea. Se observó una respuesta inmediata y precisa en los motores, lo que demostró la fiabilidad del envío de comandos desde la interfaz.

Se probaron todos los comandos del sistema utilizando los botones interactivos de la pestaña correspondiente. Cada uno de ellos fue registrado y mostrado en pantalla. Comandos como *save, load* o *undo* permitieron validar no solo la movilidad sino también la lógica de control interno del sistema.

Mediante la pestaña de simulación, se enviaron coordenadas espaciales para calcular ángulos y longitudes. Los resultados visuales en el entorno 3D coincidieron con las posiciones esperadas, lo que validó la exactitud del modelo cinemático implementado.

Se escribieron programas sencillos que combinaban movimientos absolutos y relativos, así como pausas y estados intermedios. La interfaz compiló y ejecutó correctamente estos scripts, mostrando en tiempo real qué línea se ejecutaba y cómo evolucionaba el sistema.

Se introdujeron intencionadamente comandos incorrectos o programas con errores de sintaxis. El compilador resaltó adecuadamente las líneas erróneas y evitó la ejecución, garantizando la integridad del sistema y la seguridad operativa.

Durante todas las pruebas que se han mencionado se obtuvieron resultados satisfactorios de las respuestas tanto del hardware como el software y se ha logrado diseñar un sistema económico tipo spydercam. También todas estas pruebas sirvieron para confirmar que el montaje mecánico funcionaba correctamente.

Se ha comprobado que el sistema tiene una buena estabilidad, la conexión se mantuvo estable durante sesiones prolongadas, incluso al alternar entre tareas (simulación, control manual, programación). La precisión fue muy buena, las órdenes enviadas desde la interfaz se tradujeron en movimientos precisos, con errores de posicionamiento mínimos, atribuibles únicamente a las limitaciones físicas del sistema.

Con todas las pruebas realizadas se validó la cinemática inversa ya que los datos generados coincidieron con los cálculos previos realizados en Excel y los valores teóricos esperados, tanto en ángulos como en longitudes.

Los resultados obtenidos confirman que la interfaz gráfica desarrollada cumple con todos los objetivos planteados: permite interactuar con el sistema de forma intuitiva, precisa y robusta. La progresiva evolución desde versiones simples a una herramienta avanzada fue clave para ir ajustando tanto la lógica del microcontrolador como la estructura visual y funcional de la interfaz.

Además, la introducción de herramientas como la simulación 3D o el editor de programación no solo mejora la usabilidad, sino que transforma la interfaz en una plataforma educativa, de pruebas y desarrollo. También abre la puerta a futuras ampliaciones, como la conexión inalámbrica, control remoto, o incluso interfaces multiplataforma.

Durante el desarrollo de las pruebas, se logró validar satisfactoriamente el diseño físico y estructural de la Spydercam. Se comprobó que la estructura respondía adecuadamente ante las exigencias operativas, desplazándose con precisión y estabilidad a lo largo de todas las trayectorias programadas. La fase inicial de validación se centró en pruebas de carga, con el fin de verificar la resistencia mecánica del sistema y asegurar que pudiera soportar adecuadamente el peso de los componentes sin

presentar deformaciones ni fallos estructurales. Una vez superada esta etapa, se realizaron pruebas dinámicas de desplazamiento, aplicando principios de cinemática para evaluar el movimiento controlado de la plataforma en los tres ejes del espacio. Estas pruebas permitieron observar el comportamiento del sistema en condiciones reales de funcionamiento, validando no solo la rigidez y estabilidad de la estructura, sino también su capacidad para mantener un control preciso y seguro durante el desplazamiento. En conjunto, los resultados obtenidos fueron altamente positivos, confirmando que la estructura de la Spydercam cumple con los requisitos técnicos de diseño y funcionamiento previstos en el proyecto.

ENSAYOS DE REPETIBILIDAD Y PRECISION:

Se ha hecho los ensayos de repetibilidad y precisión y se han ido observando, la tabla 3 se representa la posición ideal y en las pruebas las posiciones reales obtenidas y de esta forma observamos la precisión y repetibilidad del prototipo, todos los valores están en metros. Esta tabla se ha realizado probando movimientos desde el punto (0, 0, 0) a la posición indicada en la primera columna y se ha repetido tres veces y se han tomado los valores en metros de la posición a la que a llegado la plataforma en la realidad.

	POSICION DESTINO	ENSAYO 1	ENSAYO 2	ENSAYO 3
1	X0 Y0 Z1	x = 0 m y = 0 m z = 1.02 m	x = 0 m y = 0 m z = 0.99 m	x = 0 m y = 0 m z = 1.01 m
2	X1 Y0 Z1.75	x = 1.02 m y = 0 m z = 1.75 m	x = 1 m y = 0 m z = 1.73 m	x = 1 m y = 0 m z = 1.76 m
3	X0 Y1 Z1	x = 0 m y = 1 m z = 1.02 m	x = 0 m y = 1.01 m z = 1.02 m	x = 0 m y = 1 m z = 0.98 m
4	X0.8 Y0.2 Z0	x = 0.81 m y = 0.2 m z = 0 m	x = 0.79 m y = 0.2 m z = 0.01 m	x = 0.8 m y = 0.2 m z = 0.02 m
5	X0 Y0 Z1.5	x = 0 m y = 0 m z = 1.51 m	x = 0 m y = 0 m z = 1.49 m	x = 0 m y = 0.01 m z = 1.53 m

Tabla 3. Pruebas de precisión y repetibilidad con punto inicial (0, 0, 0)

En la tabla 3 podemos observar las variaciones que ha habido en la medida real, por ejemplo, en la primera prueba se ha mandado a la spydercam ir a la posición (0, 0, 1) desde la posición (0, 0, 0) y se ha obtenido que se ha desviado en z 0.02m, 0.01m y 0.01m de la posición real $(1 - 1.02 = -0.02, \quad 1 - 0.99 = 0.01 \quad y \quad 1 - 1.01 = 0.01)$.

Al realizar lo mismo con todos los ensayos de cada prueba se obtiene para cada posición realizada un error máximo obtenido.

Para calcular el error máximo en cada movimiento se va a utilizar la siguiente formula:

$$E_{rr,max} = \frac{E_{abs}}{Valor\ real} \cdot 100 \qquad (7.1)$$

- Prueba 1 movimiento de (0, 0, 0) a (0, 0, 1):

$$E_{rr,maxZ} = \frac{0.02}{1} \cdot 100 = 2\%$$

- Prueba 2 movimiento de (0,0,0) a (1, 0, 1.75):

$$E_{rr,maxX} = \frac{0.02}{1} \cdot 100 = 2\%$$

$$E_{rr,maxZ} = \frac{0.01}{1.5} \cdot 100 = 0.66\%$$

- Prueba 3 movimiento de (0,0,0) a (0, 1, 1):

$$E_{rr,maxY} = \frac{0.01}{1} \cdot 100 = 1\%$$

$$E_{rr,maxZ} = \frac{0.02}{1} \cdot 100 = 2\%$$

- Prueba 4 movimiento de (0,0,0) a (0.8, 0.2, 0):

$$E_{rr,max} = \frac{0.01}{0.8} \cdot 100 = 1.25\%$$

- Prueba 5 movimiento de (0,0,0) a (0, 1, 1.5):

$$E_{rr,max} = \frac{0.03}{1.5} \cdot 100 = 2\%$$

Analizando los valores desde el punto de vista de la repetibilidad se puede observar que todos oscilan en torno a ±2cm de la medida real. Por lo que para los objetivos puestos en el proyecto se cumplen.

Una vez realizadas las pruebas experimentales correspondientes, se ha podido constatar que el prototipo presenta un desempeño satisfactorio en términos de precisión y repetibilidad, parámetros fundamentales para evaluar la fiabilidad de cualquier sistema de medición.

En primer lugar, en lo que respecta a la precisión, los resultados obtenidos muestran que el error máximo registrado fue del orden del 2%. Este valor se presentó únicamente en las mediciones correspondientes a las distancias más pequeñas, donde es habitual que las variaciones relativas tengan un mayor impacto. Cabe señalar que, a medida que se incrementa la magnitud de la distancia medida, el error relativo disminuye, lo cual es consistente con el comportamiento esperado de este tipo de sistemas. Dado que el prototipo está diseñado principalmente para trabajar en rangos de medición del orden de metros, una variación absoluta de entre 1 y 2 cm entra dentro de los márgenes aceptables para la aplicación prevista. Esta tolerancia no compromete la operatividad ni la utilidad del sistema, especialmente considerando el contexto de uso final.

Por otro lado, la repetibilidad de las mediciones también ha sido evaluada y los resultados indican un comportamiento altamente consistente. En todas las pruebas realizadas, los valores obtenidos se mantuvieron dentro del margen de error previamente estipulado, mostrando una dispersión mínima entre ensayos sucesivos. Esta característica es especialmente relevante cuando se requiere garantizar la estabilidad del sistema ante mediciones repetidas en condiciones similares. La baja variabilidad observada demuestra que el prototipo es capaz de reproducir resultados con un alto grado de fiabilidad, lo cual es un indicativo claro de un buen diseño tanto en términos del hardware como del procesamiento de señales.

En resumen, los ensayos realizados permiten concluir que el prototipo cumple adecuadamente con los criterios de precisión y repetibilidad requeridos para su propósito. Las desviaciones detectadas se encuentran dentro de los márgenes estipulados en un inicio, y no afectan significativamente la calidad ni la funcionalidad del sistema. Estos resultados respaldan la viabilidad del prototipo como una herramienta de medición robusta y confiable dentro del rango de operación previsto.

ENSAYOS DE CURVAS ACELERACIÓN Y PERFIL TRAPEZOIDAL:

Se comprobaron los movimientos del sistema al seguir estos movimientos y a continuación se puede ver el resultado obtenido en comparación con lo ideal.

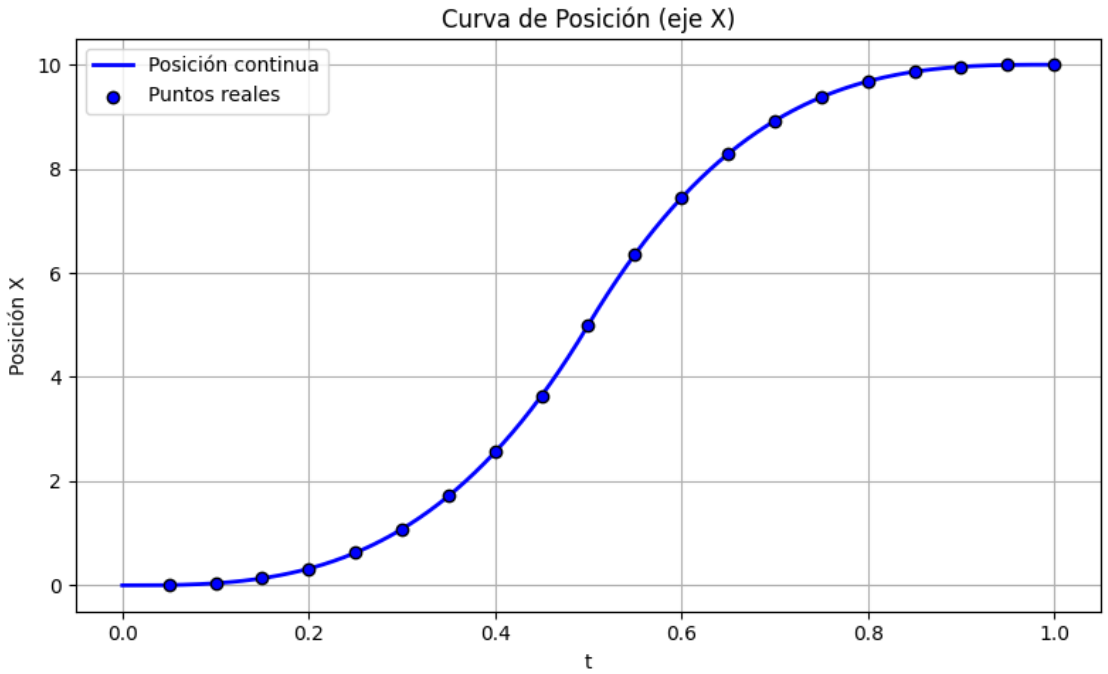


Figura 47. Pruebas de la ejecución de aceleración y desaceleración - posición

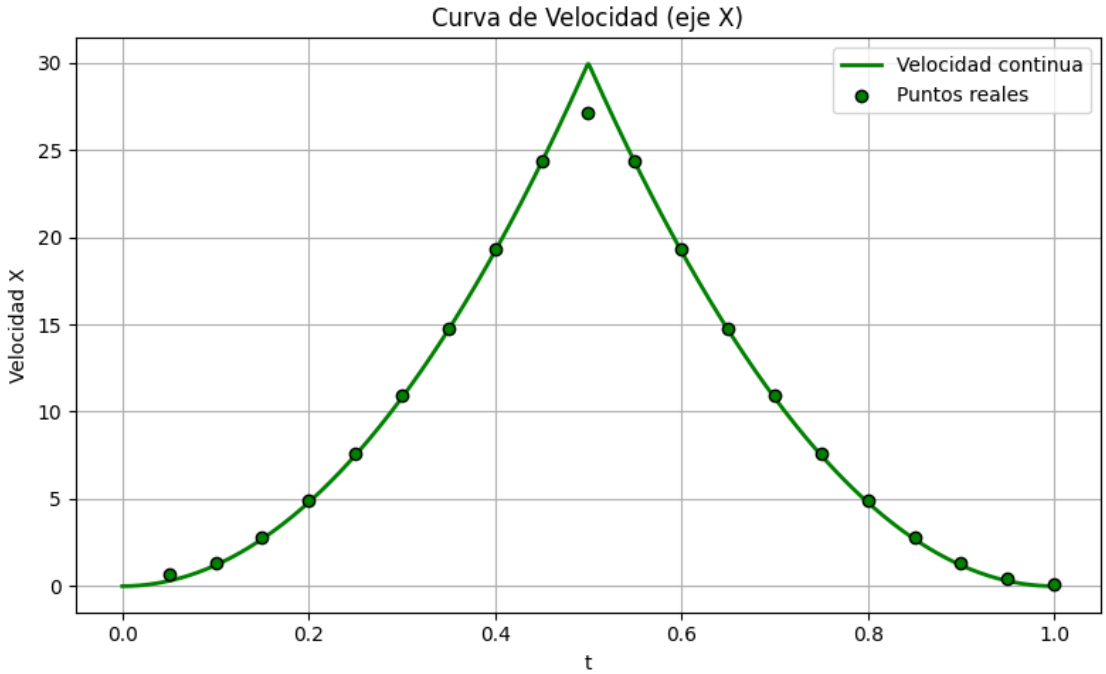


Figura 48. Pruebas de la ejecución del perfil de aceleración y desaceleración - velocidad

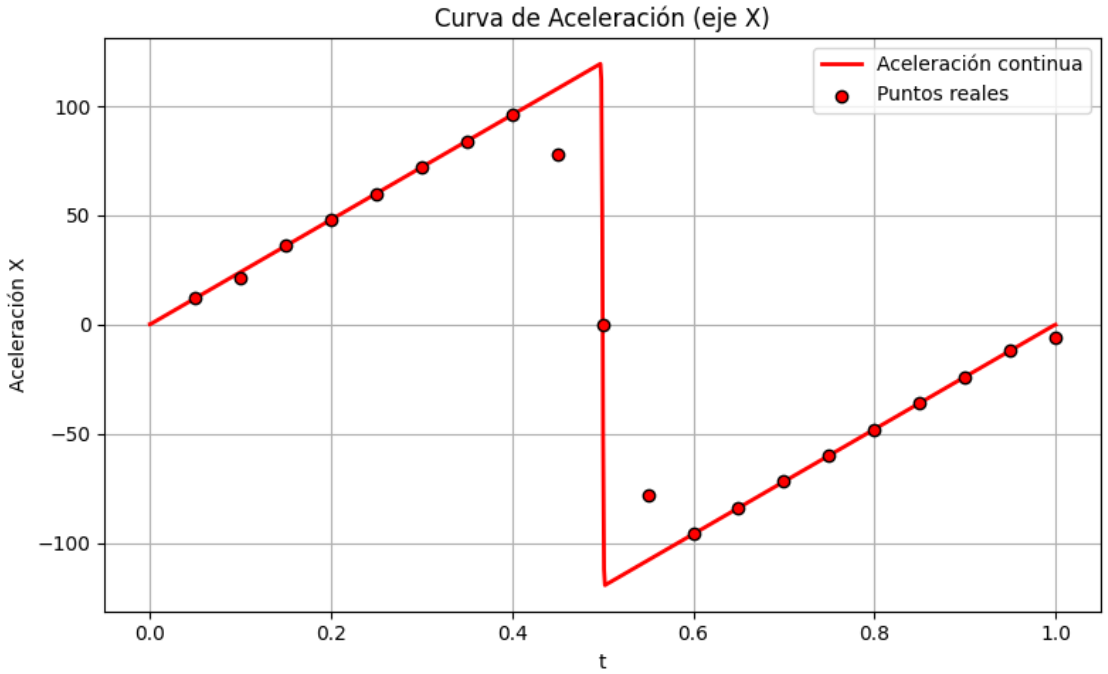


Figura 49. Pruebas de la ejecución del perfil de aceleración y desaceleración - aceleración

En las pruebas realizadas se ha observado que el comportamiento del sistema se ajusta de forma satisfactoria al perfil de aceleración-desaceleración esperado. El movimiento generado es suave y continuo en todo momento, sin presentar transiciones bruscas ni discontinuidades, lo cual confirma la correcta implementación del perfil de velocidad planificada.

Uno de los aspectos más relevantes observados es que, al aumentar el número de pasos utilizados para discretizar el perfil, la curva obtenida se aproxima con mayor precisión al perfil teórico ideal. Este efecto se debe a que una mayor cantidad de pasos permite representar de forma más detallada las variaciones graduales en la aceleración, especialmente durante las fases de arranque y frenado. Sin embargo, se ha determinado que incluso con un total de 20 pasos (valor utilizado en las pruebas representadas en las Figuras 47, 48 y 49) se consigue un resultado suficientemente preciso, manteniendo un desplazamiento suave y estable.

Este número de pasos representa un buen compromiso entre la fidelidad de la curva y la eficiencia del sistema, ya que permite lograr un comportamiento realista sin requerir un procesamiento excesivo. En definitiva, el perfil de aceleración-desaceleración implementado proporciona un control de movimiento más natural y menos agresivo, ideal para aplicaciones que requieren evitar vibraciones o impactos mecánicos durante el desplazamiento.

A continuación, se realizaron pruebas con el perfil trapezoidal para movimientos con mayor distancia, en las siguientes Figuras se muestra el ensayo con 20 pasos y 40 pasos.

Con 20 pasos:

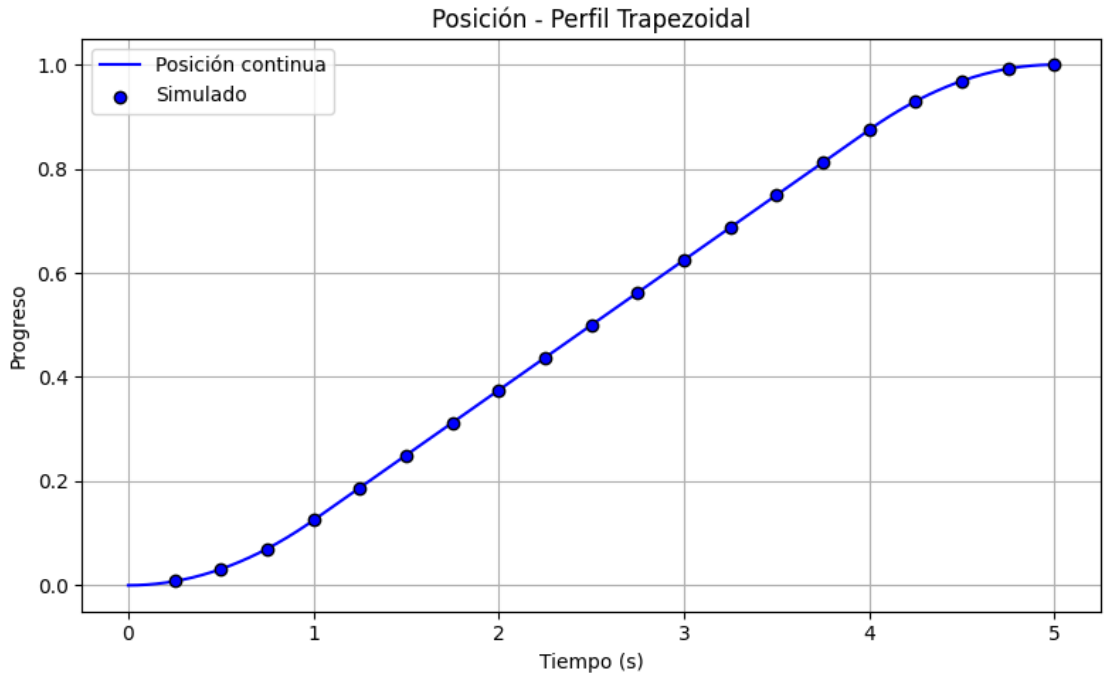


Figura 50. Pruebas de la ejecución del perfil de velocidad trapezoidal - posición

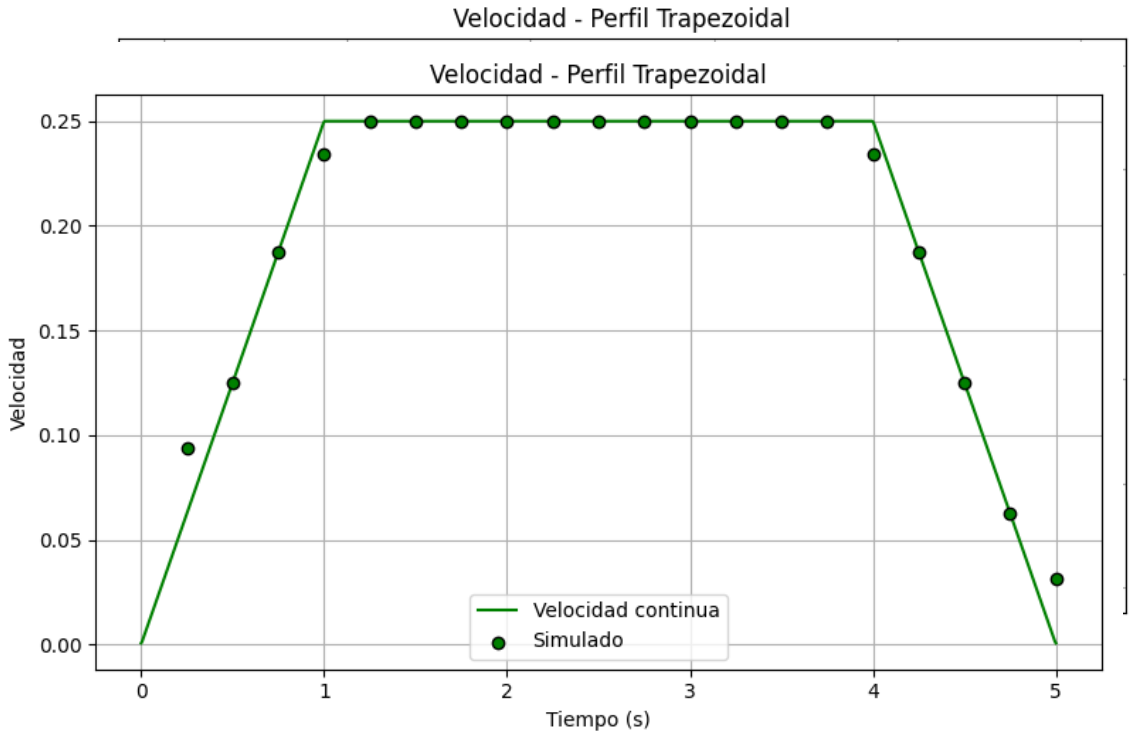


Figura 51. Pruebas de la ejecución del perfil de velocidad trapezoidal - velocidad

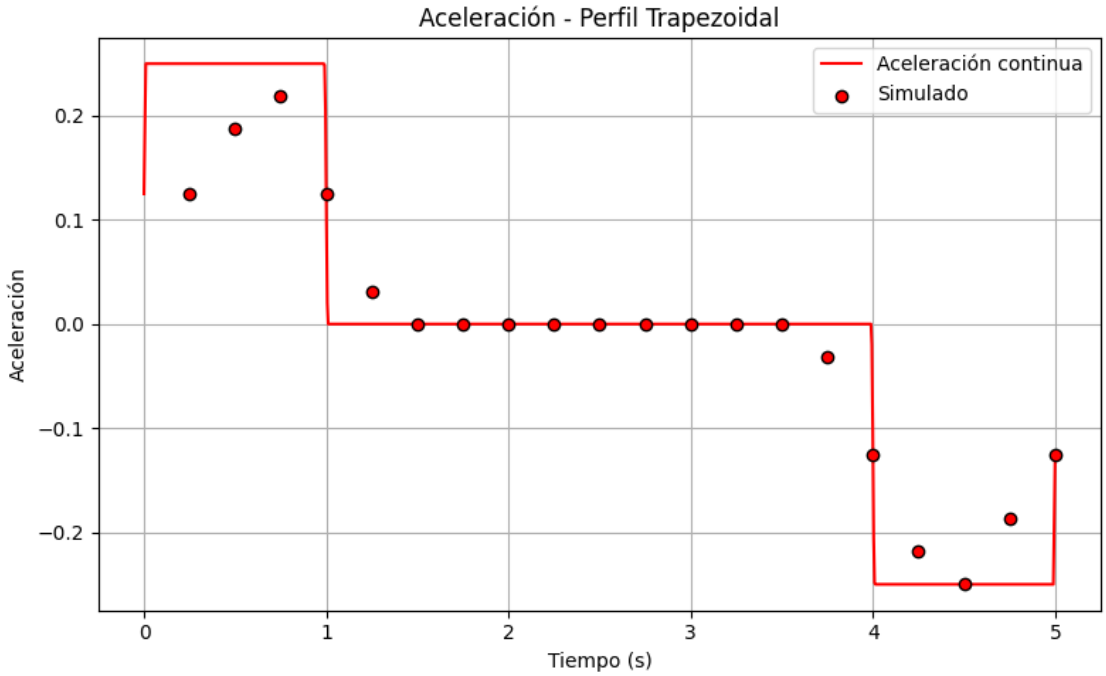


Figura 52. Pruebas de la ejecución del perfil de velocidad trapezoidal - aceleración

Con 40 pasos:

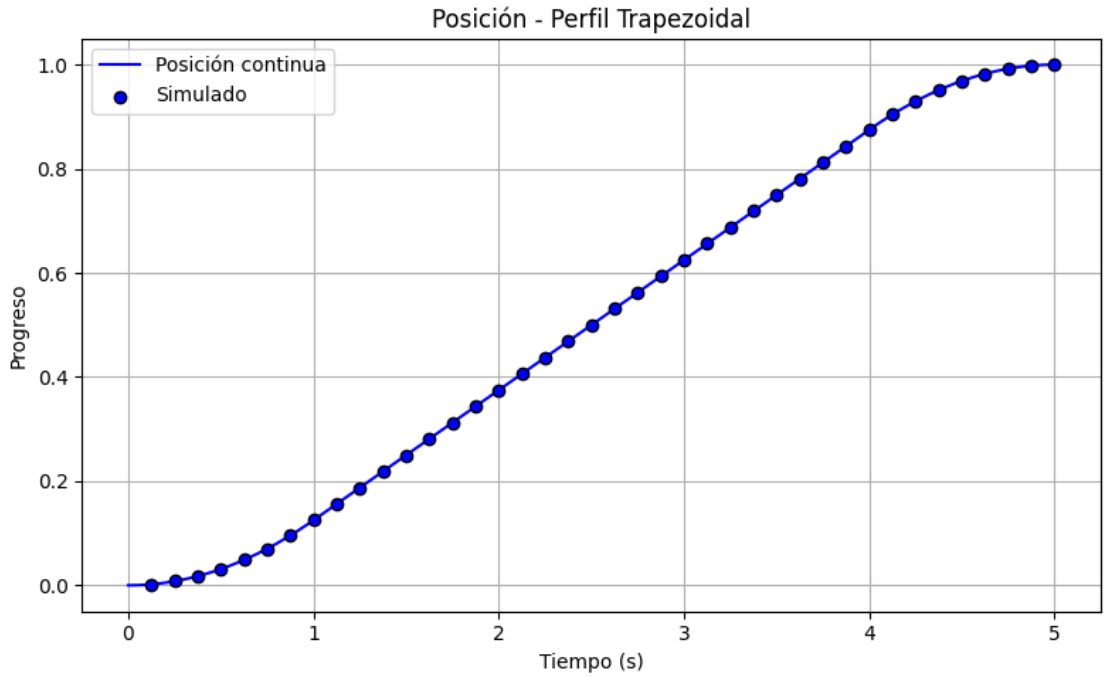


Figura 53. Pruebas de la ejecución del perfil de velocidad trapezoidal - posición

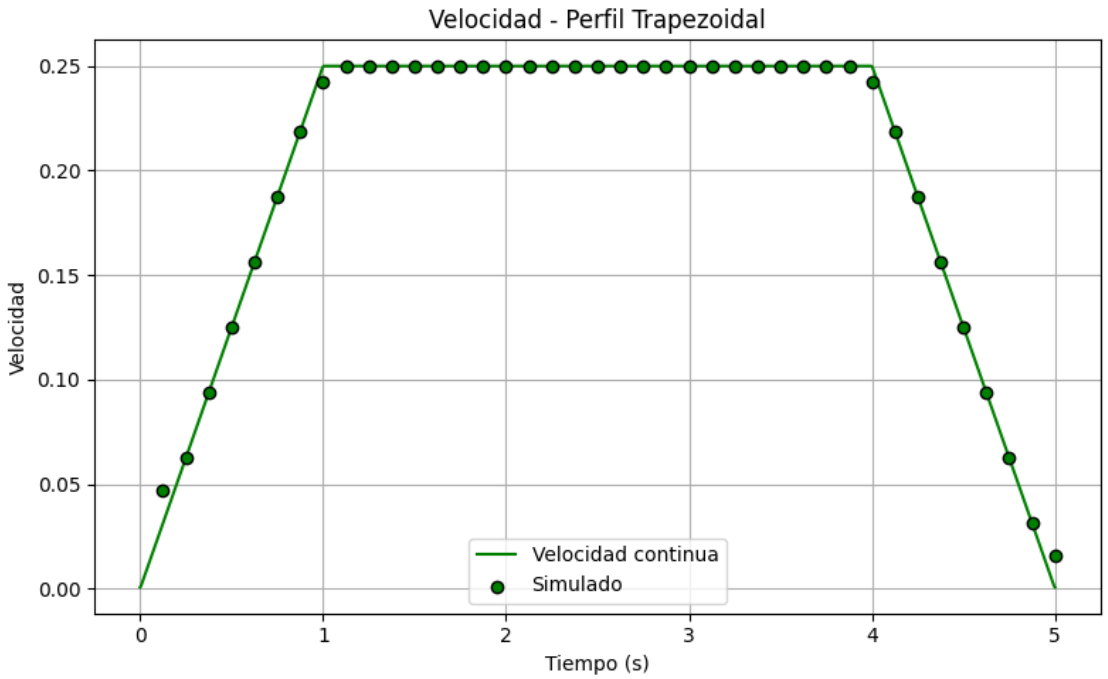


Figura 54. Pruebas de la ejecución del perfil de velocidad trapezoidal - velocidad

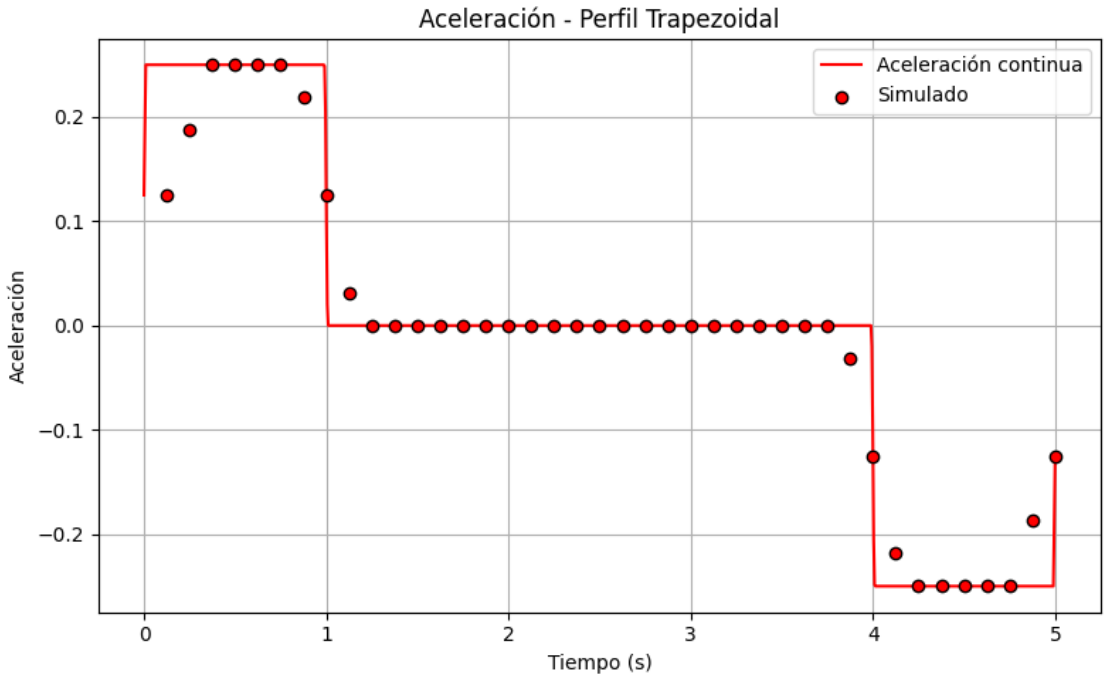


Figura 55. Pruebas de la ejecución del perfil de velocidad trapezoidal – aceleración

En las pruebas realizadas se ha observado una buena correspondencia entre el comportamiento del sistema y el perfil de velocidad trapezoidal teórico esperado. El movimiento generado presenta transiciones suaves en cada una de sus fases aceleración, velocidad constante y deceleración, lo cual indica una correcta implementación del perfil de movimiento.

Además, se ha comprobado que, al incrementar el número de pasos utilizados para discretizar el perfil, la aproximación a la curva ideal mejora de forma significativa. Esto se debe a que una mayor discretización permite representar con mayor fidelidad las transiciones entre las diferentes fases del movimiento. Aun así, con una configuración de 20 pasos (utilizada en las pruebas representadas en las Figuras 50, 51 y 52) se obtiene un resultado lo suficientemente cercano al perfil teórico como para garantizar un desplazamiento suave y funcional.

No obstante, en el caso particular de la fase de aceleración, se ha evidenciado que el uso de 40 pasos permite una aproximación aún más precisa al perfil ideal, como se muestra en la Figura 55. Esta mejora es especialmente relevante cuando se busca optimizar la calidad del movimiento en aplicaciones que requieren una aceleración progresiva y sin variaciones abruptas.

En conjunto, este nivel de precisión alcanzado con un número moderado de pasos representa un buen equilibrio entre fidelidad del perfil y eficiencia computacional. Esto resulta especialmente ventajoso en sistemas con recursos limitados o en aquellos donde es importante reducir el tiempo de cálculo sin comprometer la calidad del movimiento.

En resumen, a lo largo del desarrollo del proyecto sean llevado a cabo numerosas pruebas con el objetivo de evaluar el comportamiento dinámico del sistema, prestando especial atención a la precisión de las trayectorias, la repetibilidad de los movimientos y la correcta implementación de los perfiles de velocidad. Los ensayos de repetibilidad y precisión han demostrado que el sistema presenta un margen de error bajo, con desviaciones que oscilan entre 1 y 2 cm, lo cual resulta totalmente aceptable considerando que las trayectorias analizadas se desarrollan en rangos de metros. Este nivel de error confirma que el sistema es fiable para aplicaciones donde se requiere un posicionamiento razonablemente exacto sin necesidad de una resolución submilimétrica.

Por otro lado, las trayectorias ejecutadas durante las pruebas mostraron una buena coherencia con respecto a las planificadas, reflejando tanto estabilidad como precisión en el movimiento. Esto se evidenció en diferentes perfiles de control utilizados, entre ellos el perfil de aceleración-desaceleración y el perfil trapezoidal. En ambos casos, se observó que el sistema responde correctamente a las variaciones de velocidad, manteniendo transiciones suaves en las fases de aceleración y frenado, sin generar oscilaciones ni discontinuidades. Además, se comprobó que el número de pasos empleados para discretizar estos perfiles tiene un impacto directo en la fidelidad del movimiento: si bien con 20 pasos se logra una aproximación aceptable, al aumentar a 40 pasos, especialmente en la fase de aceleración.

En conjunto, los resultados obtenidos permiten concluir que el sistema desarrollado cumple con los requisitos de precisión, estabilidad y comportamiento dinámico necesarios para su aplicación prevista. Las pruebas realizadas validan tanto el diseño del control de movimiento como la implementación de los perfiles de velocidad, ofreciendo una base sólida para futuras optimizaciones o ampliaciones del sistema.

Capítulo 8: Conclusiones y líneas futuras

El proyecto desarrollado ha consistido en el diseño e implementación de un sistema de robot de cables tipo spydercam, el sistema se basa en la comunicación de un nodo maestro basado en un ESP32, y gestionado a través de una interfaz gráfica propia programada en Python. Este nodo maestro se comunica con otros cuatro nodos esclavos a través del protocolo ESP-NOW para mandarles la información necesaria para el control de cada motor, estos nodos esclavos se sitúan en las esquinas superiores de un prisma de base rectangular donde cada uno controla su motor correspondiente. Como sea visto en el desarrollo a lo largo de la memoria la idea es mover la plataforma a cualquier punto del espacio dentro de los límites. A lo largo del desarrollo se abordaron aspectos claves como la comunicación entre dispositivos mediante ESP-NOW, la implementación de cálculos de la cinemática inversa, el control individual de motores, programación de trayectorias, y la simulación de movimientos en el entorno tridimensional.

Uno de los principales retos del proyecto ha sido la implementación de cálculos de cinemática inversa, los cuales permiten traducir las coordenadas deseadas en el espacio tridimensional en longitudes de cable que deben ajustar los motores. Este algoritmo fue optimizado para operar en tiempo real, permitiendo el control dinámico de trayectorias con alta precisión.

Paralelamente al desarrollo físico, ha diseñado una interfaz gráfica completamente funcional utilizando PyQt6, la cual ha evolucionado a partir de prototipos iniciales en Excel y Tkinter. Esta interfaz no solo facilita la interacción del usuario con el robot, sino que también permite la programación, simulación y depuración de trayectorias en un entorno virtual tridimensional. Se han integrado múltiples herramientas como vistas gráficas, controles manuales, monitoreo de estados y carga de trayectorias predefinidas, todo ello dentro de una estructura modular y navegable.

El proyecto ha abarcado desde el diseño de las piezas del montaje hasta la programación del hardware y software, por lo que ha sido un proyecto muy productivo para el ámbito de la educación ya que se aplican conocimientos de diversas áreas (física, mecánica, informática y robótica). En conjunto, el proyecto ha requerido una fuerte integración entre hardware y software, un enfoque riguroso en la resolución de problemas de control y movimiento, así como una metodología iterativa de diseño que permitió avanzar desde conceptos iniciales hasta una solución operativa y funcional.

A lo largo del desarrollo del proyecto se han cumplido satisfactoriamente los objetivos propuestos, entre los cuales destacaba la creación de un prototipo funcional de un sistema tipo Spydercam, capaz de ejecutar movimientos precisos y controlados en un espacio tridimensional. El sistema ha demostrado un comportamiento estable, versátil y técnicamente sólido, tanto en términos de control de trayectoria como en la

implementación de perfiles de movimiento como el trapezoidal y el de aceleración-desaceleración.

Las pruebas realizadas han permitido validar el correcto funcionamiento del prototipo. Se ha comprobado una buena precisión y repetibilidad en las trayectorias, con errores mínimos (del orden de 1 a 2 cm) que resultan totalmente admisibles para distancias del orden de metros. Además, el sistema ha respondido correctamente ante diferentes configuraciones de control, manteniendo una transición suave entre fases de aceleración, velocidad constante y frenado, lo que garantiza un movimiento continuo y sin oscilaciones.

El prototipo desarrollado cumple con los requisitos técnicos esperados, ofreciendo una base robusta sobre la que se pueden construir futuras mejoras tanto en el ámbito físico como en el software, consolidando así su utilidad para aplicaciones experimentales, educativas o de automatización.

Si bien el sistema actual presenta un rendimiento satisfactorio, su diseño modular y escalable abre la puerta a múltiples posibilidades de mejora y ampliación. Una de las líneas más prometedoras consiste en la incorporación de funcionalidades adicionales en la aplicación de control, como la programación y simulación en tiempo real de trayectorias, lo cual permitiría visualizar y ajustar los movimientos antes de ejecutarlos físicamente.

Asimismo, se podrían integrar sensores adicionales al sistema físico, como encoders, sensores inerciales o de proximidad, para mejorar la retroalimentación y la precisión del control en lazo cerrado. Otra línea de evolución interesante sería el aprovechamiento de la cámara incorporada al sistema, permitiendo desarrollar funciones de visión artificial o seguimiento en tiempo real de objetos en movimiento, ampliando así el espectro de aplicaciones posibles del prototipo.

Estas y otras mejoras potenciales representan oportunidades claras para aumentar la robustez, inteligencia y adaptabilidad del sistema, consolidando su desarrollo como una plataforma tecnológica flexible y en constante evolución.

En definitiva, este proyecto representa no solo la realización de un sistema robótico funcional y complejo, sino también una base sólida sobre la cual pueden construirse futuras mejoras e innovaciones. El diseño integral de esta spydercam, que combina precisión física, control distribuido e interacción intuitiva a través de una interfaz gráfica avanzada, demuestra que es posible abordar desafíos multidisciplinarios con soluciones creativas y eficientes.

A partir de los cimientos establecidos, tanto a nivel de hardware como de software, el sistema está preparado para evolucionar, incorporar nuevas funcionalidades y adaptarse a escenarios más exigentes, reafirmando así el valor del trabajo realizado y su proyección hacia el futuro.

Bibliografía

[1] Audio Visuales Fader. (s.f). *Toda la información sobre el Spidercam*: https://audiovisualesfader.com/uncategorized/spidercam/ *(F.Acceso 21/06/2025)*

[2] Objetivos de desarrollo sostenible. (s.f). *17 objetivos para trasformar nuestro mundo*: https://www.un.org/sustainabledevelopment/es/ (F. Acceso 05/09/2025)

[3] NIST. (s.f) *NIST: https://www.nist.gov (F.Acceso 18/08/2025)*

[4] Sergio Santamaria, Eduardo Zalama, Raúl Gomez, Pablo Muñoz y Jaime Gómez-García-Bermejo. (2023) Robot de cables para limpieza de fachadas. *Revista Iberoamericana de Automática e Informática Industrial.*

[5] Espressif. (s.f.). *Espressif*: https://www.espressif.com/ *(F.Acceso 21/06/2025)*

[6] Daniel Carrasco. (2021, mayo 14). *ESP-Now conecta dos o más ESP32/ESP8266*: https://www.electrosoftcloud.com/esp-now-conecta-dos-o-mas-esp32-esp8266/ *(F.Acceso 21/06/2025)*

[7] My Actuator.(s.f). *My Actuator:* https://www.myactuator.com/ (F.Acceso 03/06/2025)

[8] Python GUIs. (2025, mayo 19). *PyQt6 tutorial*: https://www.pythonguis.com/pyqt6-tutorial/ *(F.Acceso 21/06/2025)*

[9] Arduino. (s.f.). *Arduino documentation*: https://docs.arduino.cc *(F.Acceso 21/06/2025)*

[10] Daniel Carrasco. (2021, mayo 14). *ESP-Now conecta dos o más ESP32/ESP8266*: https://www.electrosoftcloud.com/esp-now-conecta-dos-o-mas-esp32-esp8266/ *(F.Acceso 21/06/2025)*

[11] Python. (s.f). *Python 3.13.5*: https://docs.python.org/es/ *(F.Acceso 21/06/2025)*

[12] My Actuator.(s.f). *L-9025 Details:* https://www.myactuator.com/l-9025-details *(F. Acceso 21/06/2025)*

[13] Panorama audiovisual.com (s.f). *Noticias de audiovisual:* https://www.panoramaaudiovisual.com/ *(F. Acceso 18/08/2025)*

[14] EFFRA. (s.f). *EFRA:* https://portal.effra.eu/ *(F. Acceso 18/08/2025)*

[15] Rugby League. (s.f) *Spidercam: https://abejleaquetech.weebly.com/spider-cam.html (F. Acceso 18/08/2025)*

[16] Andreas Pott. (2018) *Cable-Driven Parallel Robots.* (Eds) Springer, ISBN 978-3-319-76137-4

[17] Sen Qian, Bin Zi, Wei-Wei Shang & Qing-Song Xu. (2018). A Review on Cable-driven Parallel Robts.*Chinese. Journal of Machanical Engineering, 12:* https://doi.org/10.1186/s10033-018-0267-9 *(F. Acceso 4/09/2025)*

[18] Joy Mechanix. (s.f). *Reign Above Every Set 1D, 2D, 3D cable cam systems tailored for your vision:* https://joymechanix.com/cable-cam-systems#!/tab/770331766-1. *(F. Acceso 11/09/2025)*

[19] Basque digital innovation hub.(s.f) *COGIRO: Robot de cables montado en estructura fija:* https://bdih.spri.eus/es/cogiro-robot-de-cables-montado-en-estructura-fija/ *(F. Acceso 7/09/2025)*

[20] NI. (2025, mayo 30). *Descripción general del protocolo de la red de controladores de área (CAN):* https://www.ni.com/es/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/controller-area-network--can--overview.html?srsltid=AfmBOorYnucRPmHPIPY1gKK_6OZ7_5NX0aJl6d4SHw2kd9VRTt4jLgGx *(F. Acceso 10/09/2025)*

[21] Naylamp mechatronics. (s.f) *Módulo CAN MCP2515:* https://naylampmechatronics.com/alambrico/253-modulo-can-mcp2515.html *(F. Acceso 11/09/2025)*

[22] 3DExperience Make. (s.f). *Impresión 3D:* https://www.3ds.com/es/make/guide/process/3d-printing *(F.Acceso 11/09/2025)*

Anejos

Esquema eléctrico

Pinout y conexiones del ESP32 – MCP2515 CAN

En este anejo se detallan las conexiones utilizadas del microcontrolador ESP32 y el módulo MCP2515, que se utiliza para la comunicación CAN con los motores.

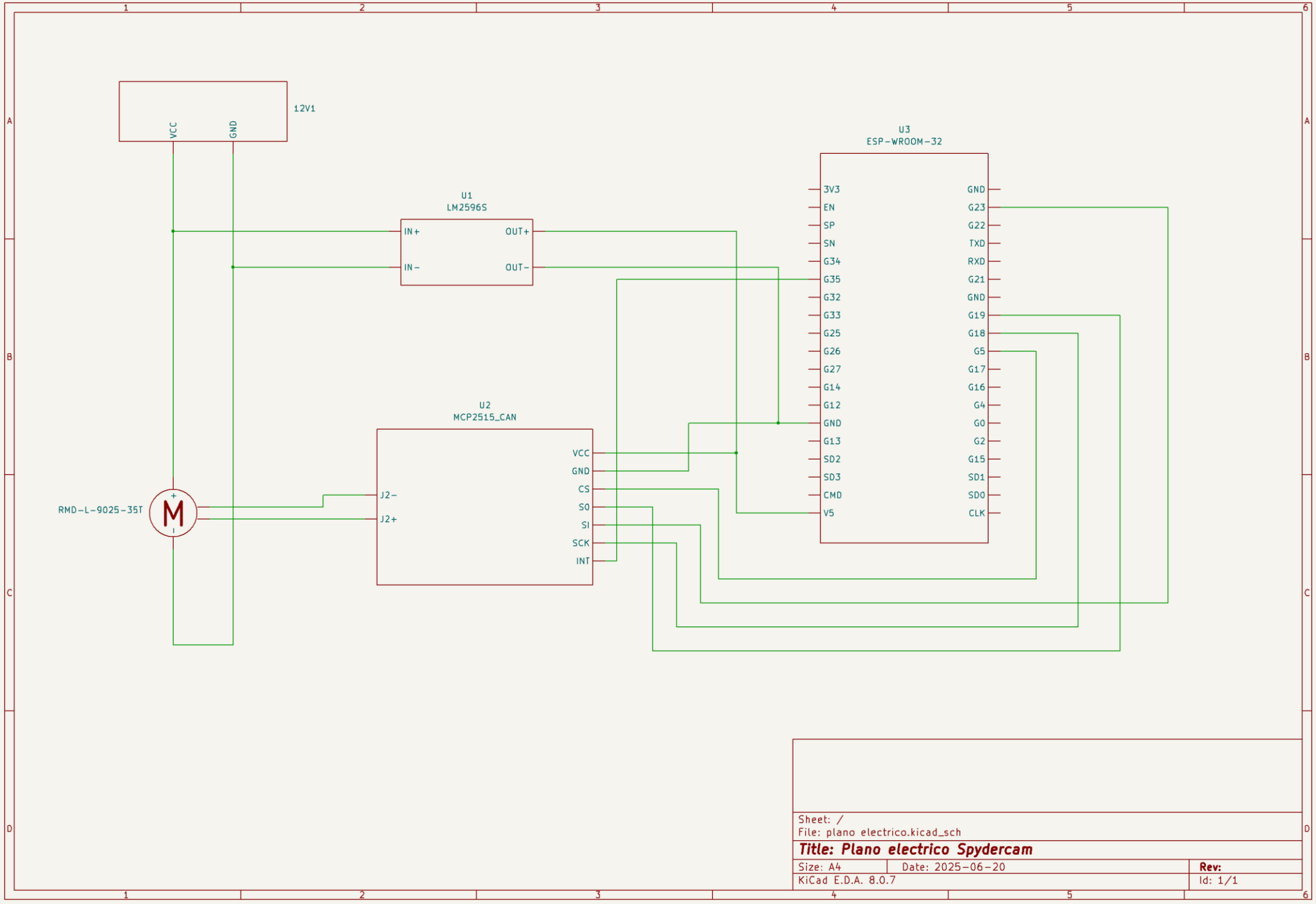
- GPIO 2 VCC
- GPIO 18 SCK (CLK)
- GPIO 23 SI (MOSI)
- GPIO 19 SO (MISO)
- GPIO 5 CS
- GPIO 35 INT
- GPIO X GND

LM2596S – Módulo regulador de voltaje DC-DC

Este módulo se conecta a la fuente de alimentación de 12V y se regula su potenciómetro para dar en la salida 5V, la salida de este módulo se conecta a la entrada de 5V del ESP32 y al pin de tierra (GND).

Esquema eléctrico

El esquema eléctrico ha sido dibujado con el software de diseño KiCad, se puede observar en la siguiente página.



Sheet: /
File: plano electrico.kicad_sch

Title: Plano electrico Spydercam

Size: A4	Date: 2025-06-20	**Rev:**
KiCad E.D.A. 8.0.7		Id: 1/1

Explicación código ESP32 maestro

Con el protocolo ESP-NOW se trasmiten los datos de velocidad y ángulo que reciben los dispositivos ESP32 esclavos, utilizando la siguiente estructura:

```
typedef struct {
  float velocity;
  float angle;
} comando_master;
```

El código incluye una función loop(), en la cual se puede observar la funcionalidad de cada comando y las llamadas a las distintas funciones para mejorar la claridad del programa. Esta función loop() se basa en esperar la entrada de un comando y, según el comando recibido, ejecuta la acción correspondiente.

```
void loop() {
  if (Serial.available()) {
    String entrada = Serial.readStringUntil('\n');
    entrada.trim();

    if (entrada.startsWith("motor")) {
        procesarComandoMotor(entrada);

    }else if (entrada.startsWith("moveL")) {
      procesarComandoGoto(entrada);
    }else if (entrada.startsWith("moveAD")) {
      procesarComandoAcel(entrada);
    }else if (entrada.startsWith("moveAcel")) {
      procesarComandoTrap(entrada);
    }else if (entrada.startsWith("moveC")) {
      procesarComandoMoveCircular(entrada);
    }
    else if (entrada.startsWith("move")) {
      procesarComandoMove(entrada);
    }else if (entrada.startsWith("sim")) {
      procesarComandoSim(entrada);
    } else if (entrada == "ping") {
      Serial.println("pong");
    } else if (entrada == "help") {
      mostrarAyuda();
    }else if (entrada.startsWith("save ")) {
      int slot = entrada.substring(5).toInt();
      guardarPosicion(slot, posicionActual);
    } else if (entrada.startsWith("load ")) {
      int slot = entrada.substring(5).toInt();
      Vector3 pos = cargarPosicion(slot);
      destino = pos;
```

```
      calcularLongitudes(pos.x, pos.y, pos.z);
      convertirAGrados();
      moverTodosLosMotores(5.0);
    } else if (entrada == "listSave") {
      listaPosicionesGuardadas();
    } else if (entrada == "status") {
      mostrarStatus();
    } else if (entrada == "home") {
      irAOrigen();
    } else if (entrada == "undo") {
      comandoUndo();
    } else if (entrada.startsWith("delta")) {
      procesarComandoDelta(entrada);
    }else if (entrada.startsWith("settime")) {
      procesarComandoSetTime(entrada);
    }else {
      Serial.println("Comando no reconocido. Usa 'help' para ver
opciones");
    }
  }
}
```

A continuación, se presentan todos los comandos que admite el código, junto con una breve descripción de su función:
Lista de comandos disponibles:

- **move x y z** → Mueve la plataforma al punto (x,y,z) suavemente.
- **sim x y z** → Simula el movimiento al punto sin activar los motores.
- **motorX vel pos** → Control directo del motor X (1-4), con velocidad (RPM) y ángulo (grados).
- **delta dx dy dz** → Desplaza la plataforma desde posición actual sumando (dx,dy,dz).
- **moveL x y z** → Realiza un desplazamiento lineal desde el punto actual al punto indicado.
- **moveC x y ox oy** → Realiza una trayectoria circular en el plano xy desde el punto actual al punto indicado con centro de la circunferencia ox, oy. La altura se mantiene la de la posición actual.
- **moveAD x y z** → Realiza una trayectoria lineal con solo acelerando y desacelerando, muy útil para movimientos de distancias pequeñas, debido a que no tiene el tiepo suficiente para llegar a una velocidad constante.
- **moveAcel x y z t** → movimiento utilizando un perfil de velocidades trapezoidal donde se desplaza linealmente a la posición indicada. Se especifica con el parámetro t el tiempo de aceleración y desaceleración.
- **home** → Regresa la plataforma a la posición (0,0,0).
- **undo** → Retorna la posición anterior.
- **status** → Muestra estado actual y ángulos de motores.
- **ping** → Verifica si el dispositivo maestro está activo; responde con un ‘pong’.

- **help** → Muestra esta lista de comandos disponibles.
- **save N** → Guarda la posición actual en ranura N (0-9).
- **load N** → Mueve la plataforma a la posición guardada en ranura N (0-9).
- **listSave** → Muestra todas las posiciones guardadas.
- **settime N** → Actualiza en nuevo tiempo de cada movimiento al valor N.

Las funciones del código son:

```
void moverMotor(int motor, int vel, int pos)
```
Función encargada de mandar el mensaje por ESP-NOW al motor especificado, indicando su posición y velocidad.

```
void moverTodosLosMotoresDirectos()
```
Función encargada de mover todos los motores a las posiciones indicadas, llamando a la función moverMotor para cada uno, con la velocidad especificada. Este movimiento se realiza sin trayectoria ni desplazamiento suave, y está diseñado para verificar que la cinemática se ha calculado correctamente.

```
void moverTodosLosMotores(float tiempoMovimiento)
```
Función encargada de mover todos los motores con un movimiento suave, evitando cambios bruscos en el movimiento.

```
void calcularLongitudes(float x, float y, float z)
```
Función encargada de calcular la cinemática inversa, determina las longitudes de los cables a partir de la posición en el espacio de la plataforma dada.

```
void convertirAGrados()
```
Función encargada de trasformar las longitudes calculadas con calcularLongitudes a grados para que posteriormente se envíen a los motores.

```
void interpolarMovimientoLineal(Vector3 inicio, Vector3 fin, int pasos)
```
Función que realiza la interpolación lineal, dado los puntos de inicio y fin además de especificar el número de pasos.

```
void interpolarMovimientoCurvo(Vector3 inicio, Vector3 fin, int pasos)
```
Función que realiza la interpolación circular en un plano, dado los puntos de inicio y fin además de especificar el número de pasos.

```
float perfilTrapezoidal(float t, float tiempoTotal, float
tiempoAceleracion)
```
Encargada de realizar los cálculos del perfil trapezoidal de velocidad.

```
void procesarComandoMotor(String entrada)
```
Función encargada de procesar el comando para mover uno solo de los motores.

```
void interpolarMovimientoTrapezoidal(Vector3 inicio, Vector3 fin, int pasos, float tiempoAceleracion)
```
Es la encargada de calcular la interpolación de las posiciones que indica el perfil trapezoidal.

```
void interpolarCircular2D(Vector3 inicio, Vector3 fin, Vector3 centro, int pasos)
```
Es la encargada de calcular los puntos intermedios para realizar la trayectoria curva.

```
void procesarComandoMove(String entrada)
```
Función encargada de procesar el comando para desplazar la plataforma a un punto del espacio especificado.

```
void procesarComandoGoto(String entrada)
```
Es la encargada de procesar el comando moveL.

```
void procesarComandoAcel(String entrada)
```
Es la encargada de procesar el comando moveAD para calcular las aceleraciones y desaceleraciones para el movimiento.

```
void procesarComandoTrap(String entrada)
```
Procesa el comando moveAcel donde se encarga en gestionar el cálculo del perfil de velocidades y realizar el movimiento.

```
void procesarComandoMoveCircular(String entrada)
```
Es la encargada de procesar el comando moveC para gestionar el cálculo y el movimiento en la trayectoria circular.

```
void procesarComandoDelta(String entrada)
```
Función encargada de procesar el comando para desplazar la plataforma desde la posición actual a la suma de otras coordenadas.

```
void procesarComandoSim(String entrada)
```
Función encargada de el comando para simular el siguiente movimiento a una posición en el espacio especificada.

```
void guardarPosicion(int slot, Vector3 pos)
```
Función encargada de guardar la última posición en una ranura de guardado especificada.

```
Vector3 cargarPosicion(int slot)
```
Función encargada de cargar el ultimo guardado en la ranura de guardado especificado.

```
void listaPosicionesGuardadas()
```
Función encargada de mostrar todo el contenido de las ranuras de guardado.

```
void irAOrigen()
```

Función encargada de llevar la plataforma al origen de coordenadas.

```
void mostrarStatus()
```
Función encargada de mostrar el estado actual de la plataforma y todos los motores.

```
void procesarComandoSetTime(String entrada)
```
Función encargada de procesar el comando settime e imponer el tiempo de duración de todos los movimientos.

```
void mostrarAyuda()
```
Función encargada de mostrar por pantalla todos los comandos para que se puedan consultar en cualquier momento.

Explicación código ESP32 esclavo

Este programa recibe la información que le envía el master por el protocolo ESP-NOW siguiendo la siguiente estructura:

```
typedef struct comando_master {
  float velocity;
  float angle;
};
comando_master dhtData;
```

Luego inicializa el motor para que hasta que no le llegue nueva información este en la posición inicial cero, que es como se ha planteado la cinemática.
Los valores que recibe se actualizan en las variables Veloc y Pos y se llama a la función encargada de mover al motor.

```
void loop() {
  unsigned long now = millis();
  if (now - lastCommandTime >= commandInterval) {
    lastCommandTime = now;
    uint16_t velocidadRPM = (uint16_t)Veloc;
    int32_t anguloGrados = (int32_t)Pos;
    MotorDerecho.PositionClosedLoop2(velocidadRPM, anguloGrados);
  }
}
```

Función encargada de mover el motor:

```
void RMD::PositionClosedLoop2(uint16_t speedRPM, int32_t angleDeg)

{
  for (int i = 0; i < 8; i++)
    dataSend[i] = 0;

  dataSend[0] = _PositionClosedLoop2;

  // Convertir RPM a °/s
  uint16_t speedDPS = speedRPM * 6;

  // Insertar velocidad en dataSend[2..3]
  dataSend[2] = speedDPS & 0xFF;
  dataSend[3] = (speedDPS >> 8) & 0xFF;

  // Convertir ángulo a LSBs (0.01°/LSB)
  int32_t angleLSB = (int32_t)(angleDeg / 0.01f);
```

```
  // Insertar posición en dataSend[4..7]
  dataSend[4] = angleLSB & 0xFF;
  dataSend[5] = (angleLSB >> 8) & 0xFF;
  dataSend[6] = (angleLSB >> 16) & 0xFF;
  dataSend[7] = (angleLSB >> 24) & 0xFF;

  sendData();
}
```

Hay más funciones dentro de RMD.cpp para funcionalidad de los motores, donde puedes realizar paradas, modificar el PID pero en este caso se decidió regular el PID a través de la propia aplicación para los motores RMD.

Software RMD V2.0

RMD Motor Assistant V2.0 es el software propio de los motores RMD para poder regular los valores del PID además de poder realizar pruebas con los motores para que funcionen correctamente. En la siguiente imagen se puede observar la interfaz del software mencionado (ver figura 56).

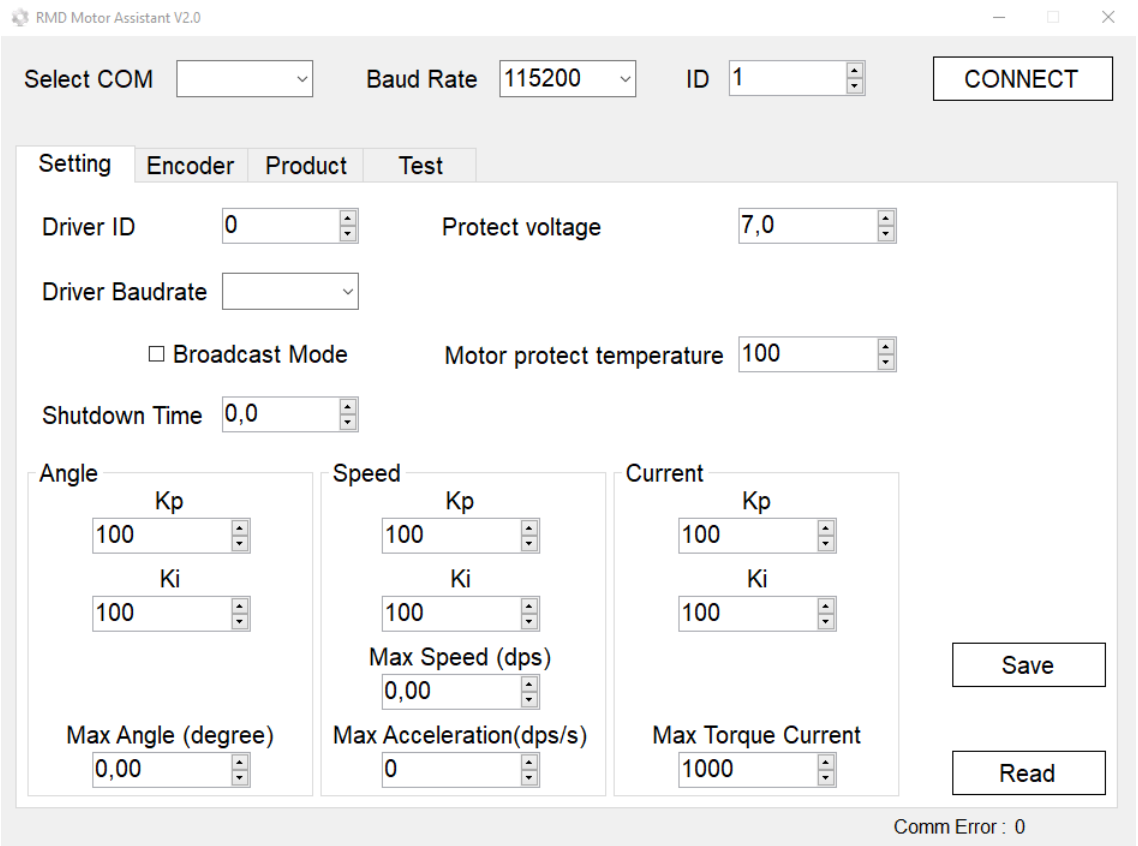


Figura 56. Software RMD Motor Assistant

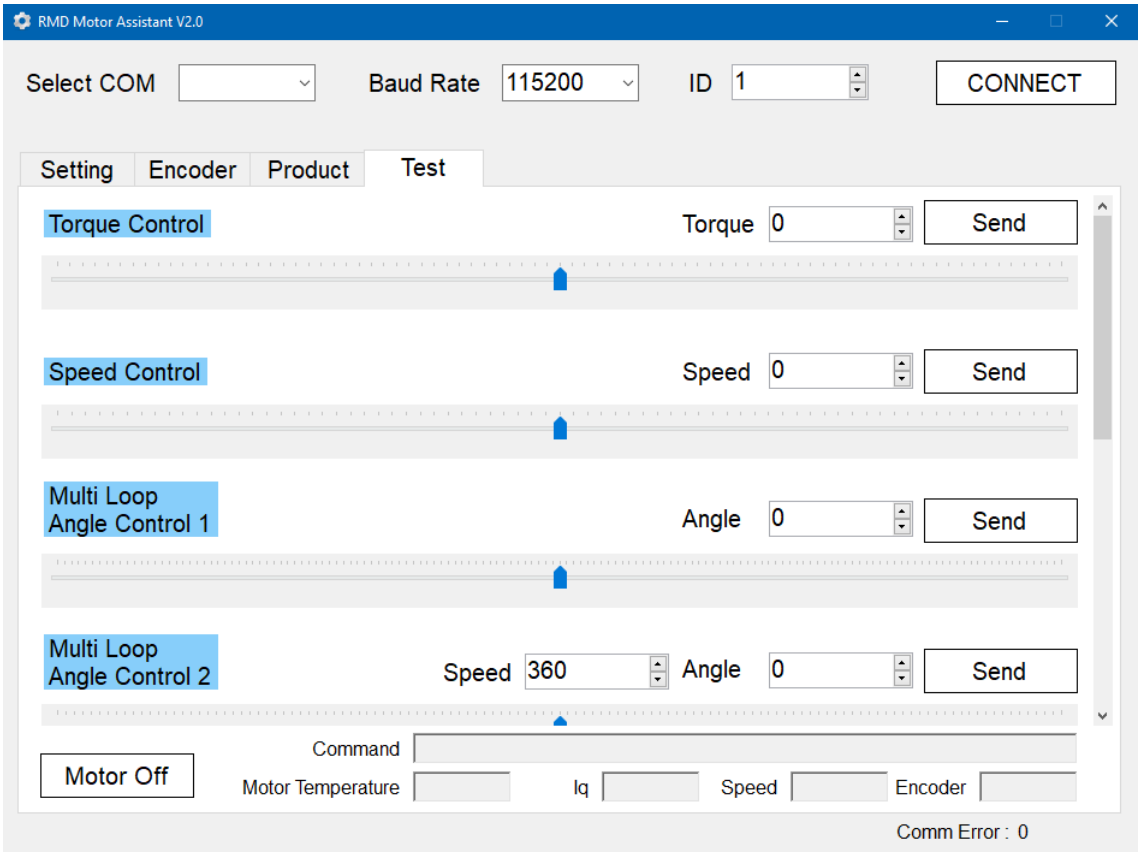


Figura 57. Software RMD Motor Assistant

En la Figura 57 se muestra la interfaz principal del software. En la parte superior, se puede seleccionar el puerto de comunicación y conectar con el motor correspondiente.

La pestaña "Setting", que se observa en la imagen, permite regular los parámetros PID para optimizar el comportamiento del motor. Además, el software cuenta con una pestaña "Test", que se muestra en la imagen siguiente, donde es posible probar el motor con la configuración PID establecida. Esto facilita modificar y ajustar los parámetros de forma iterativa hasta encontrar la configuración más adecuada para el montaje específico.
En el caso de este proyecto después de varias pruebas y analizar los resultados se optó por la siguiente configuración, se indicó en el programa y se mandó a cada motor dando al botón “Save”.

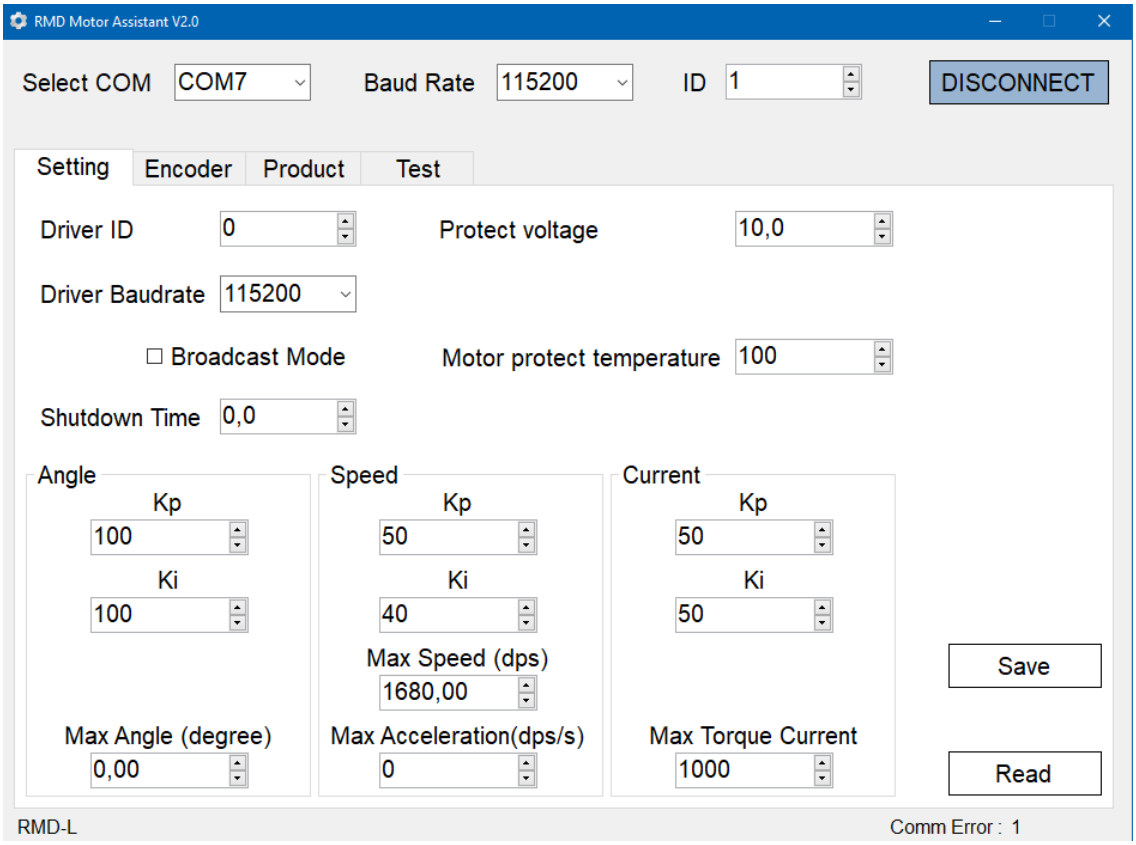


Figura 58. Software RMD Motor Assistant

En el caso de este proyecto, tras varias pruebas y análisis de resultados, se seleccionaron los siguientes parámetros PID (mostrados en la figura 58), que se ingresaron en el software y se enviaron a cada motor presionando el botón "Save":

Se utilizaron los siguientes parámetros PID en el controlador del motor RMD-L-9025-35T: Kp=Ki=100 para el control de posición, Kp=50 y Ki=40 para el lazo de velocidad, y Kp=Ki=50 para el lazo de corriente. Estos valores fueron seleccionados empíricamente para lograr una respuesta rápida pero estable, con buena precisión en posición y sin oscilaciones ni sobreimpulsos en velocidad y par. La velocidad máxima fue configurada en 1680 rpm como valor de referencia interna, para no limitar artificialmente el desempeño del sistema y permitir uso compartido de configuración con versiones de mayor velocidad.

Excel de simulación de la cinemática

Para probar las primeras veces la cinemática se realizó el cálculo en una hoja de Excell y según los valores obtenidos se mandaban a cada motor respectivo para comprobar su precisión y exactitud.

La tabla utilizada en Excel se puede observar en la figura 58.

		Posicion final							Dimensiones (m)		radio	0,045		
	x	0					W	2,8						
	y	1					D	3			plataforma	0,15		
	z	1					H	2,34						
	Cable	Longitud		Motor	Grados				L0	3,112169661	offset	0		
	L1	3,163162974		M1	64,92670238									
	L2	3,163162974		M2	64,92670238									
	L3	2,00139951		M3	-1414,276481									
	L4	2,00139951		M4	-1414,276481									

Figura 58. Programación en Excel

Se introdujeron las fórmulas deducidas de la cinemática directa e inversa y se procedieron a hacer unas simulaciones de los resultados y se observaron que fueran coherentes y se procedió a mandar esos valores a cada motor correspondiente y se observó un resultado satisfactorio por lo que se procedió a incluir las fórmulas en el código de Arduino del maestro.

Funcionamiento APP de control del sistema

En este apartado se va a explicar el funcionamiento del Software diseñado para el control del proyecto.

Para abrir la aplicación simplemente ejecuta el *AxisFly.exe* correspondiente y espera unos momentos a que se abra correctamente.

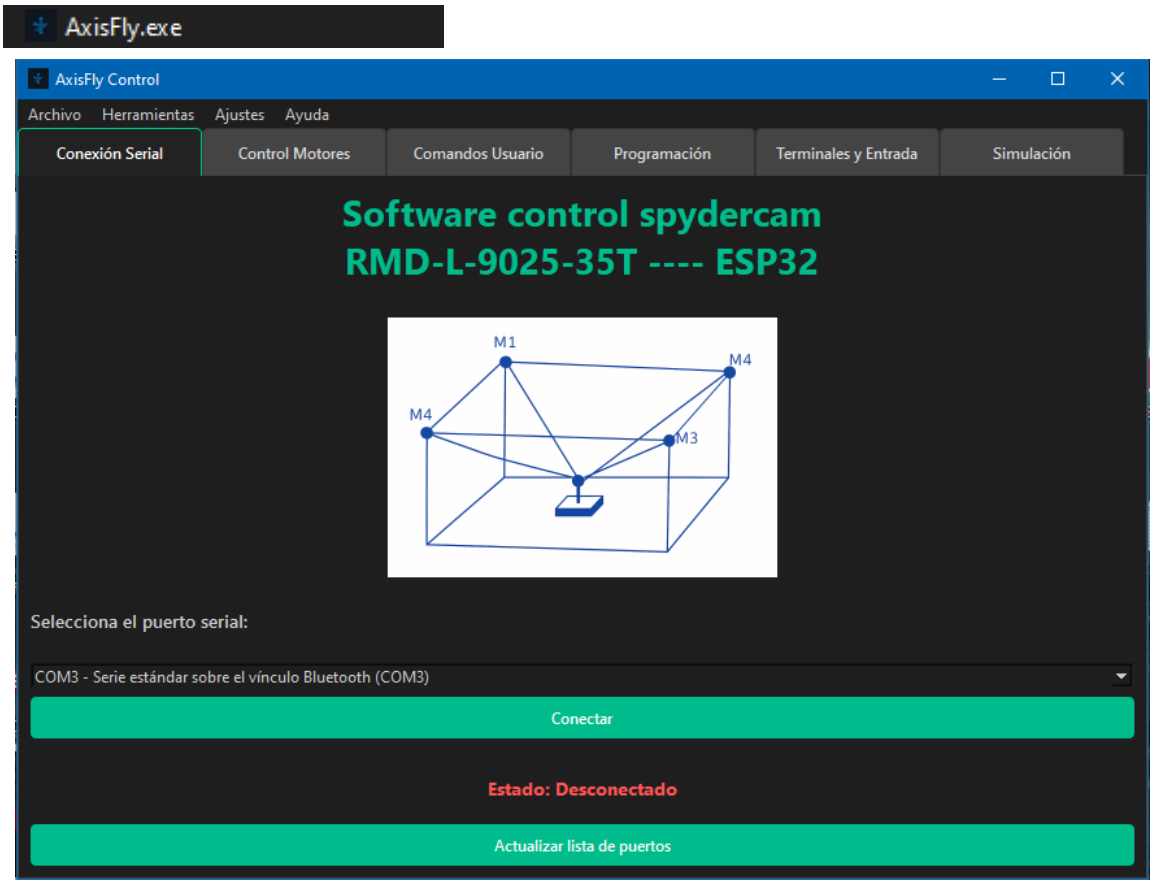


Figura 59. Aplicación diseñada - Conexión Serial

Al ejecutar el *AxisFly.exe* y se abra la aplicación se verá la pantalla anterior. En ella se pueden observar un menú de herramientas superior y debajo unas ventanas con el contenido en la aplicación.

Conexión Serial

En la primera pestaña (ver figura 59) se observa un esquema del montaje con la numeración de los motores y una pestaña extensible con la elección de puerto además de un botón para conectar/desconectar y otro de actualización de puertos (por si el ESP32 se conectó después de abrir el software).

Por defecto solo se puede acceder a las ventanas de Conexión Serial, Programación y Simulación sin conectar el puerto serie. Una vez conectado se habilitan todas las

pestañas de la aplicación. Una vez que conectes la aplicación te lo indicará como se muestra en la figura 60.

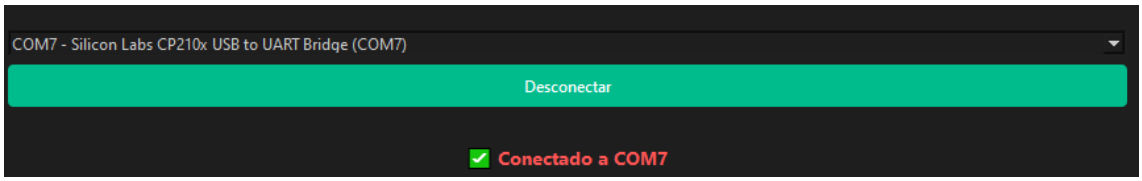


Figura 60. Aplicación diseñada - Conexión Serial

Control de motores

La ventana de control de motores es una interfaz para mover cada motor por separado indicando la velocidad y posición y después o se pueden indicar todas y luego mandar todas a la vez, por si se quiere aplicar directamente todos los grados de los motores para que vaya a una posición específica (ver figura 61). Para ello simplemente escribe los valores correspondientes en la casilla de cada motor, si solo quieres mover uno pulsa el botón correspondiente de "Enviar a motor", en caso de querer enviar toda la configuración de todos los motores a la vez pulsa "enviar configuración a todos los motores".

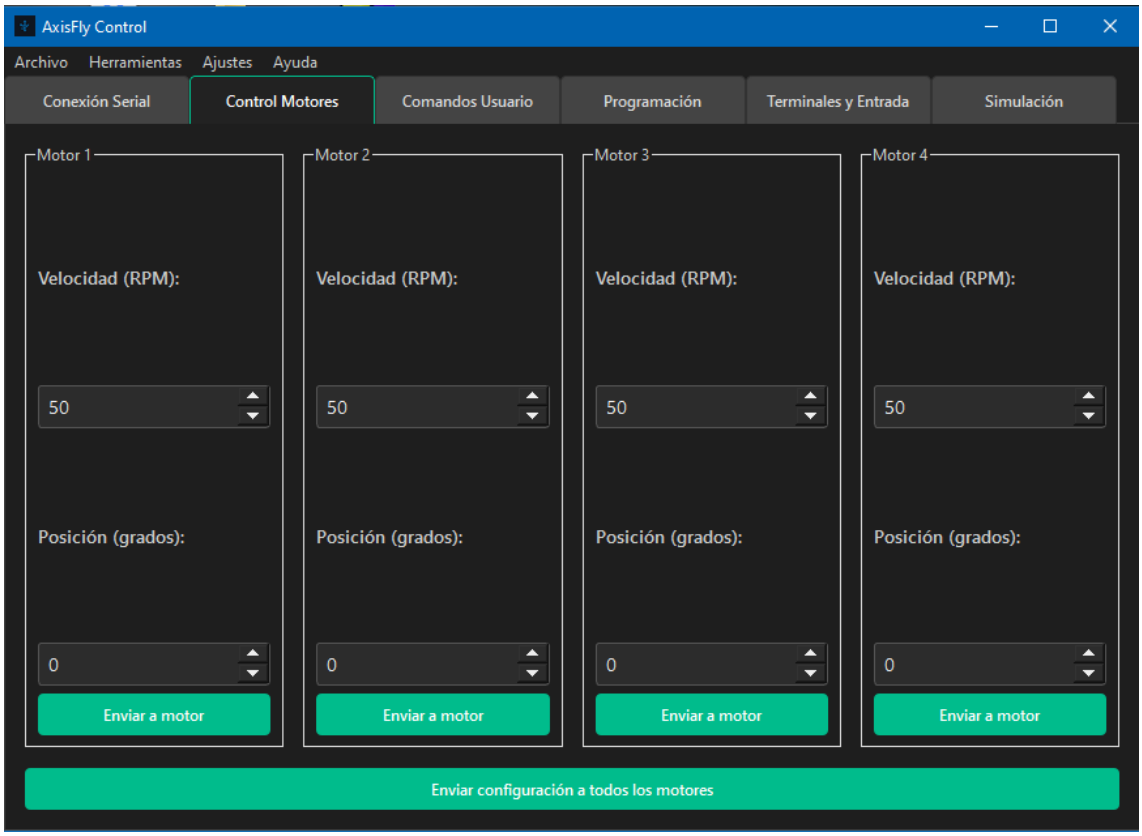


Figura 61. Aplicación diseñada – Control de motores

Puedes guardar la configuración de los motores que tienes escrita en ese momento o importar otra configuración guardada anteriormente. Para ello ve al menú superior Herramientas --> Estado control de motores --> Guardar, exportar o importar estado.

Te guarda el archivo en un .json para su posterior uso.

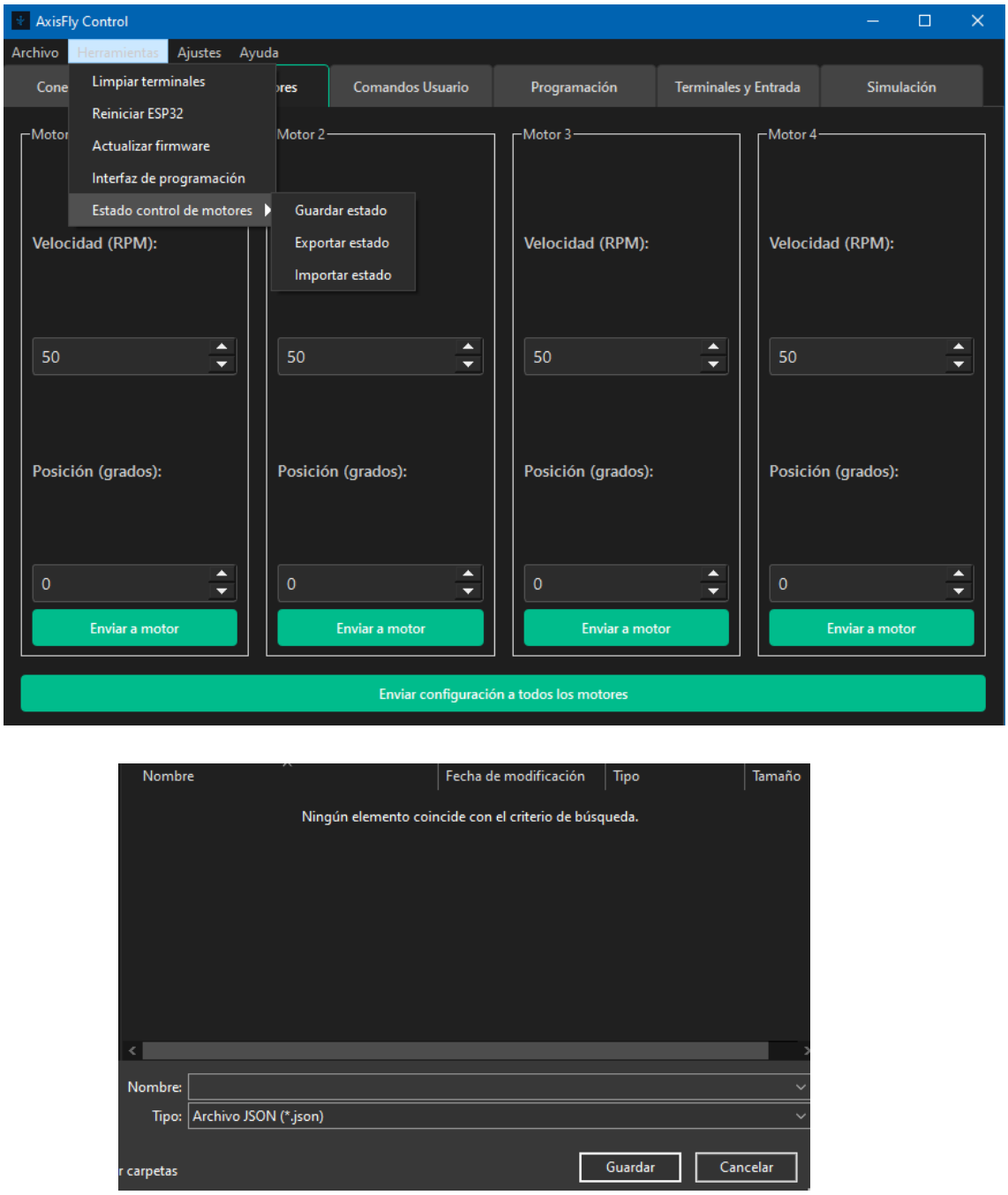


Figura 62. Aplicación diseñada – Control de motores

Para guardarlo tendrás que poner un nombre al archivo (ver figura 62).

Comandos Usuario

En esta ventana se encuentra el acceso rápido a todos los comandos del sistema de la spydercam que se han explicado anteriormente en el código del maestro. En esta ventana esta realizada para mandar de forma interactiva los comandos al robot sin necesidad de saberse los comandos.

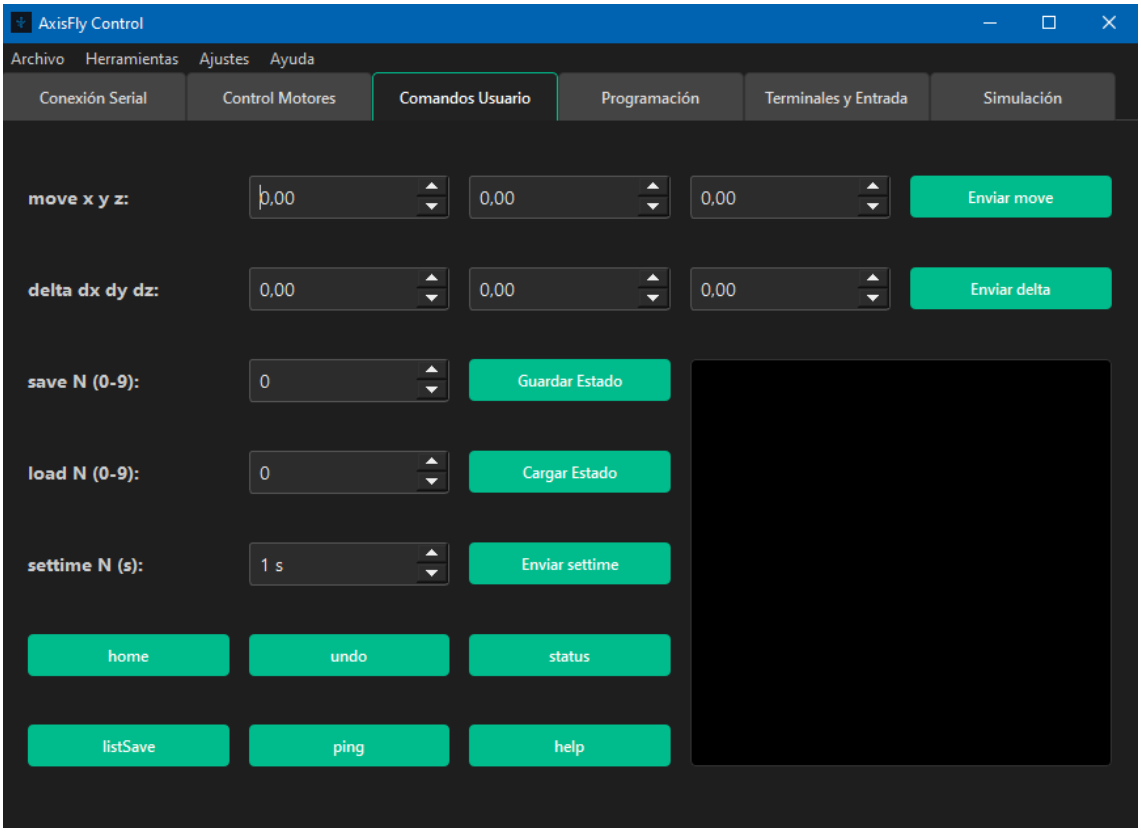


Figura 63. Aplicación diseñada – Comandos Usuario

En el lateral inferior derecho se muestra un panel de texto donde indica el comando que has enviado para que se vea que se ha mandado correctamente, te dará un mensaje de que "*Para más información, consulta la pestaña de Terminales y Entrada*", ya que en esa pestaña es donde el esp32 maestro manda los mensajes a tiempo real por el puerto serie de lo que se está ejecutando y los datos que se envían a los esp32 esclavos que controlan los motores (ver figura 63 y 64).

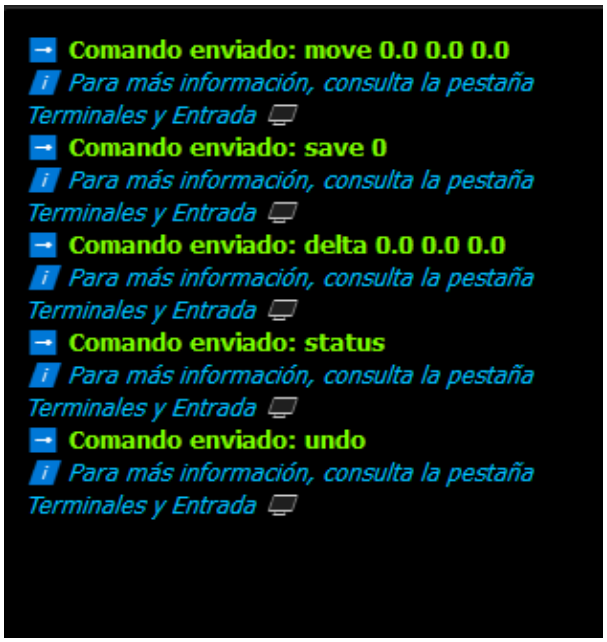


Figura 64. Aplicación diseñada – Comandos Usuario

Programación

En esta ventana se ha creado una interfaz de programación para que se puedan programar las trayectorias de la spydercam como se crea más conveniente (ver figura 65 y 66).

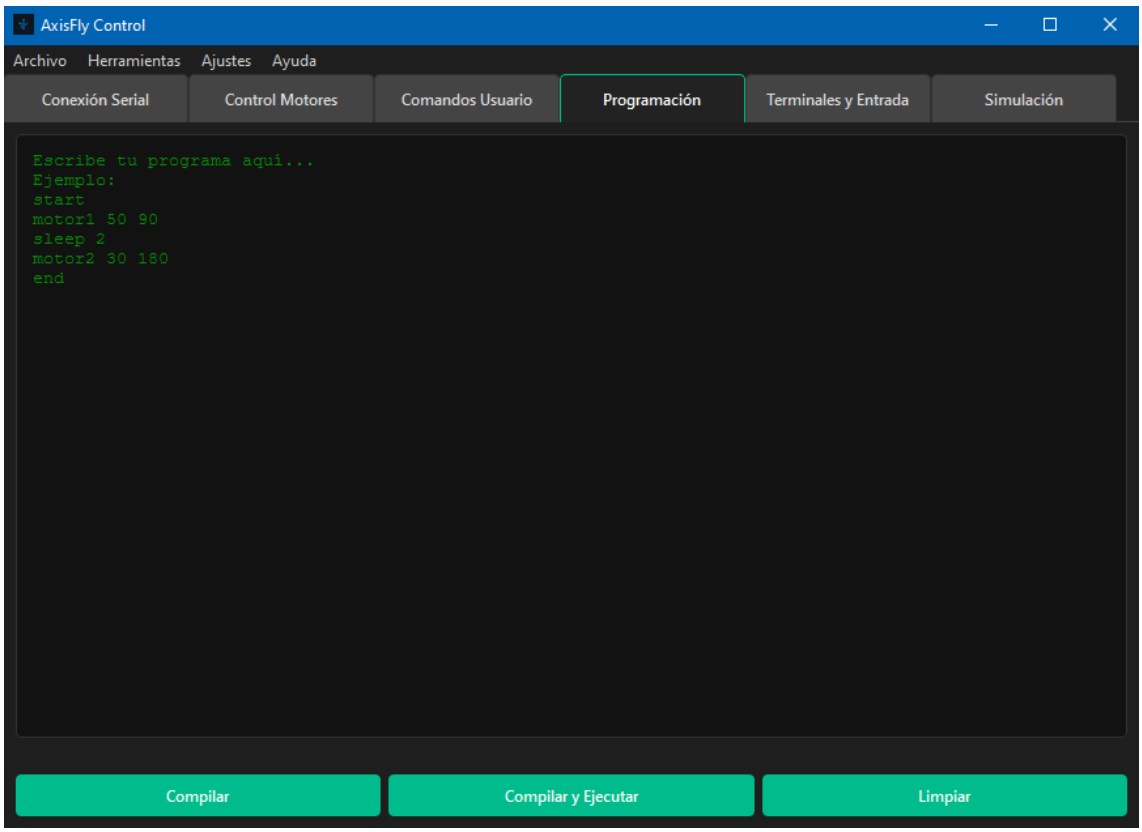


Figura 65. Aplicación diseñada – Programación

Donde pone *"Escribe tu programa aquí…"* puedes comenzar a escribir el código.

```
start
sleep 2
move 0 0 1
sleep 5.5
move 0 0 1.5
sleep 5.5
move 0 0.5 1.5
sleep 5.5
move 0 -0.5 1.5
sleep 5.5
move 0 0 1.5
sleep 5.5
move 1 0 1
sleep 5.5
move 0 0 0.5
sleep 5.5
home
end
```

Figura 66. Aplicación diseñada - Programación

Como ayuda se ha diseñado una ventana de explicación de todos los comandos y su uso. Para acceder a la pequeña ayuda hay que seguir la siguiente ruta:

Herramientas --> Interfaz de programación (ver figura 67).

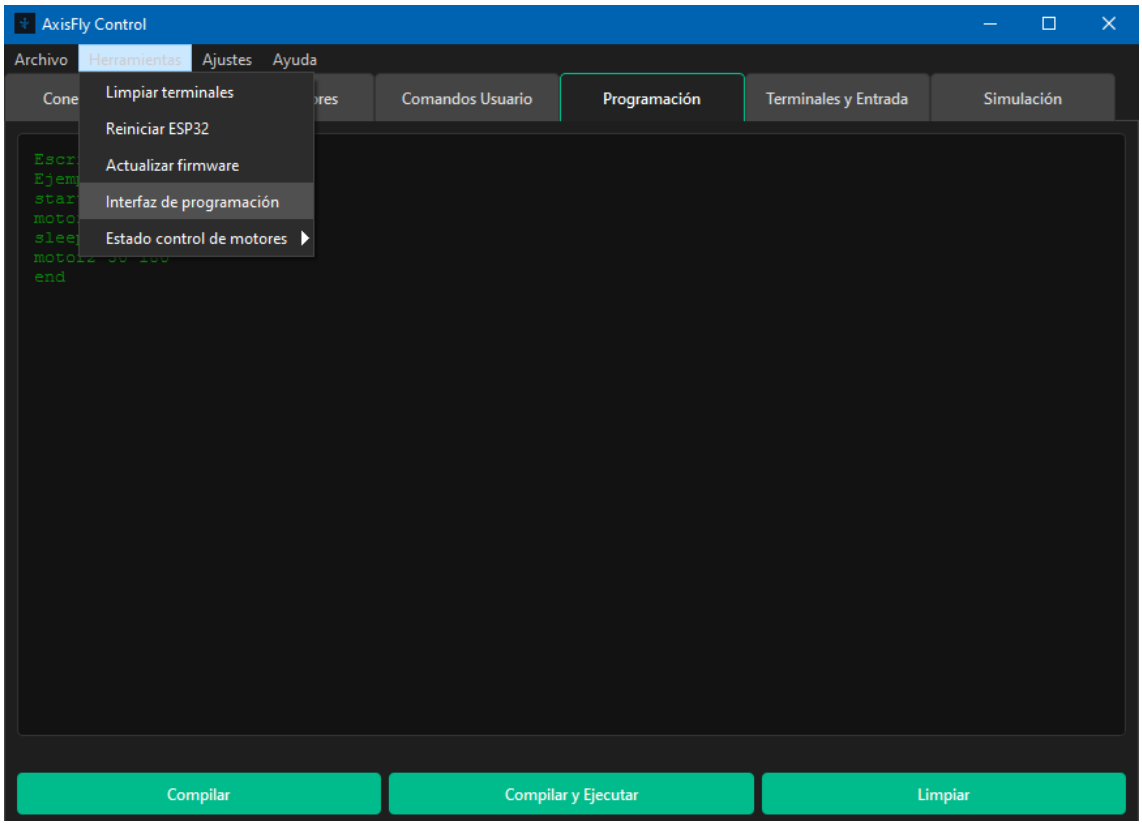


Figura 67. Aplicación diseñada - Programación

Aparecerá una ventana emergente con la explicación de todos los comandos que acepta el compilador como se muestra en la figura 68. En la columna de la izquierda elijes el comando y a la derecha te aparecerá su explicación y un ejemplo de uso en el código.

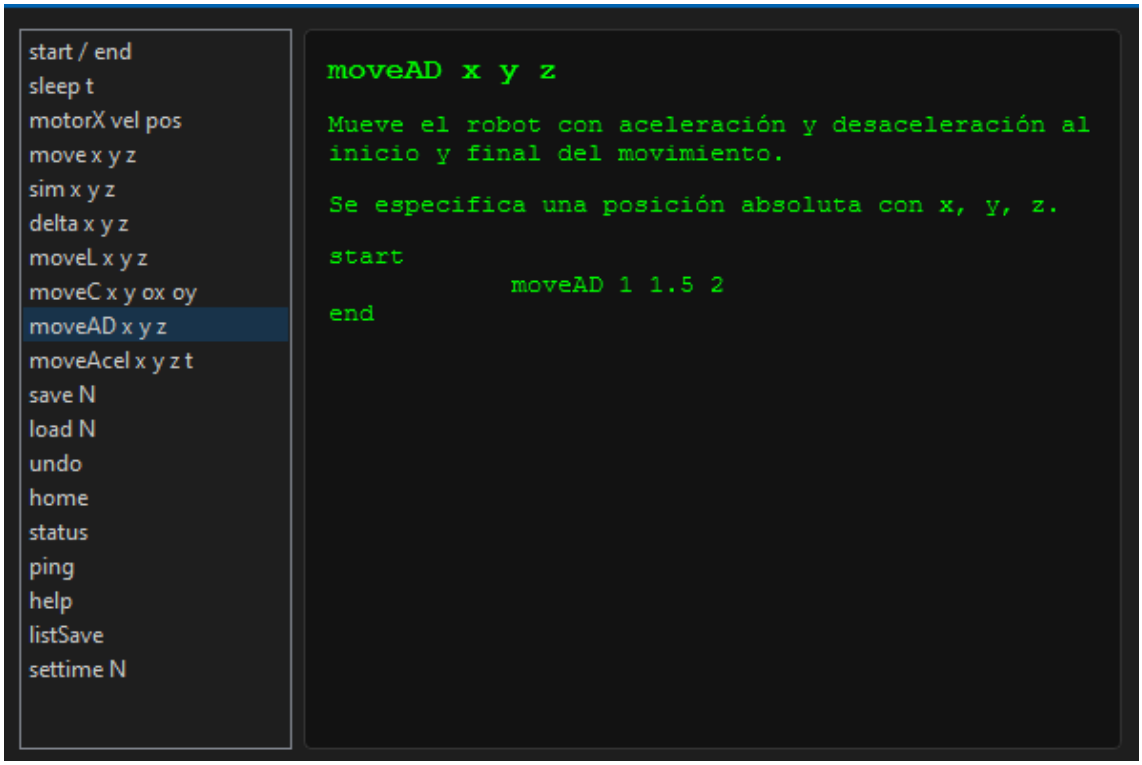


Figura 68. Aplicación diseñada – Ayuda comandos

En la parte inferior de la ventana están unos botones para activar distintas funciones, como se puede observar en la figura 69.

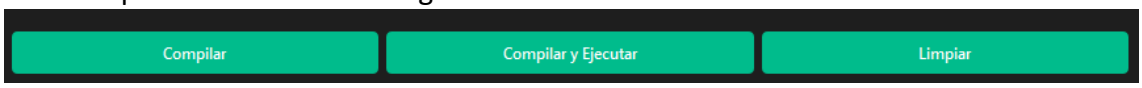


Figura 69. Aplicación diseñada - Programación

- **Compilar**: para comprobar que tu código es correcto, una vez compilado te pondrá en verde las líneas correctas y en rojo las incorrectas aparte de mostrar un aviso. start y end no se marcan (ver Figura 70).

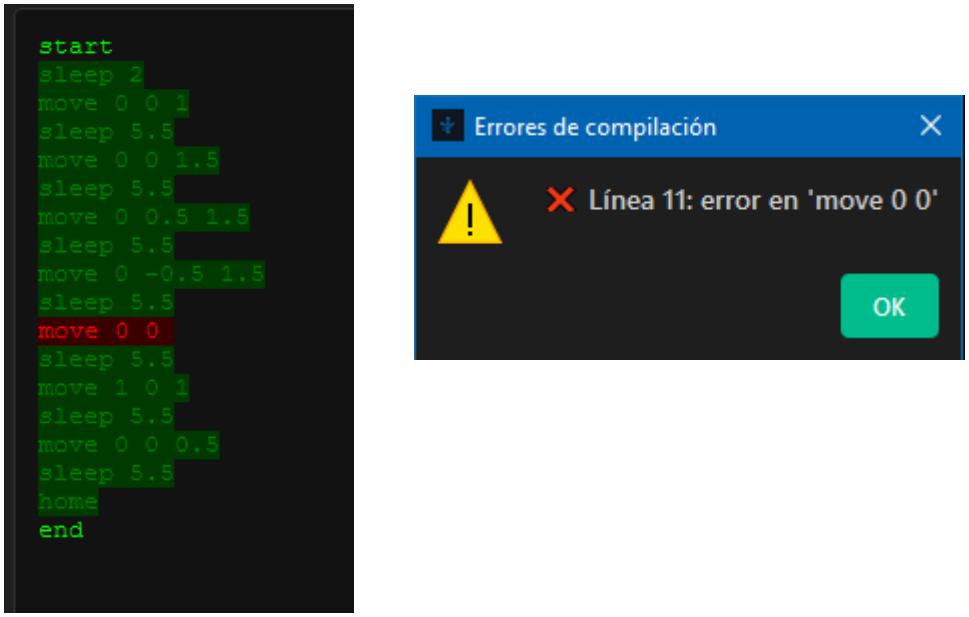


Figura 70. Aplicación diseñada - Compilador

El compilador diseñado es muy básico ya que solo acepta los comandos especificados, pero funciona perfectamente para el diseño de trayectorias que es para lo que se ha usado para el desarrollo de este proyecto.

- **Compilar y Ejecutar**: comprueba la sintaxis del código y si es correcta procede a ejecutarlo. Debajo del espacio para programar justo encima de los motores te dirá que línea se está ejecutando en cada momento (ver Figura 71).

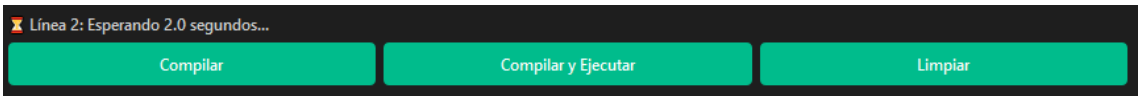


Figura 71. Aplicación diseñada - Programación

También lo puedes ver en la pestaña Terminales y Entrada que se explicará a continuación, puedes observar que se está ejecutando en cada momento, como se puede observar en la Figura 72.

Figura 72. Aplicación diseñada - Terminal

- **Limpiar**: para eliminar todo el contenido de la pantalla de programación.

Como en el caso del control de motores tambien se pude importar, exportar o guardar el código. Para ello ve a la siguiente ruta:

Archivo --> Importar programa (.txt), Exportar programa (.txt) y Guardar archivo. (ver Figura 73)

Cómo en el caso anterior tendrás que dar un nombre al archivo y elegir la ruta donde quieres guardarlo.

Se ha elegido un documento de texto ya que de esta forma para programar no es necesario abrir la aplicación puedes escribirlo en la aplicación de notas de tu propio ordenador.

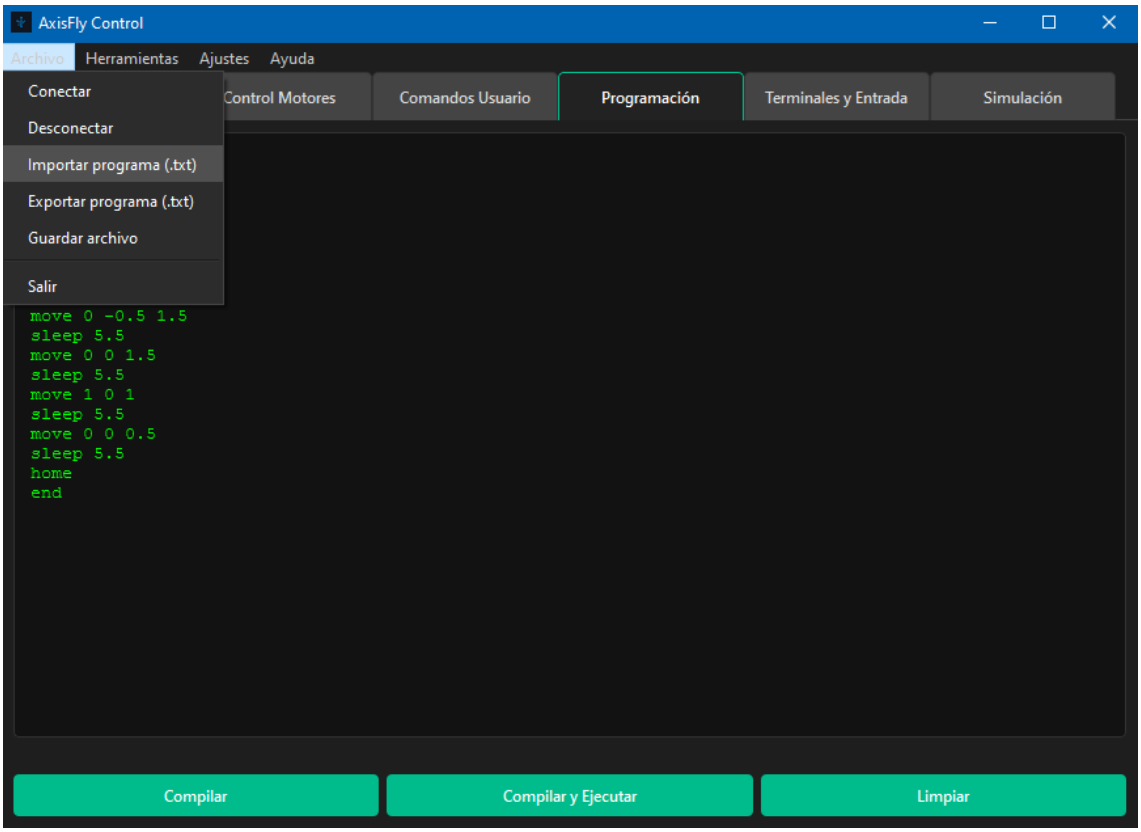


Figura 73. Aplicación diseñada – Importar/Exportar

Terminales y Entrada

En esta ventana se puede observar los terminales del ESP32 y de Python, para Seguir en tiempo real lo que está ocurriendo con cada comando que se ejecute (ver figura 74). En el lado izquierdo está el terminal de ESP32 donde podrás observar la retroalimentación del esp32 maestro y observar que valores se están enviando en cada momento. Los comandos de help, status y listSave provocan una respuesta del maestro por pantalla, para ver la información tendrás que ir al terminal para observar la información. Estos comandos también están de forma rápida en la pestaña de Comandos Usuario.

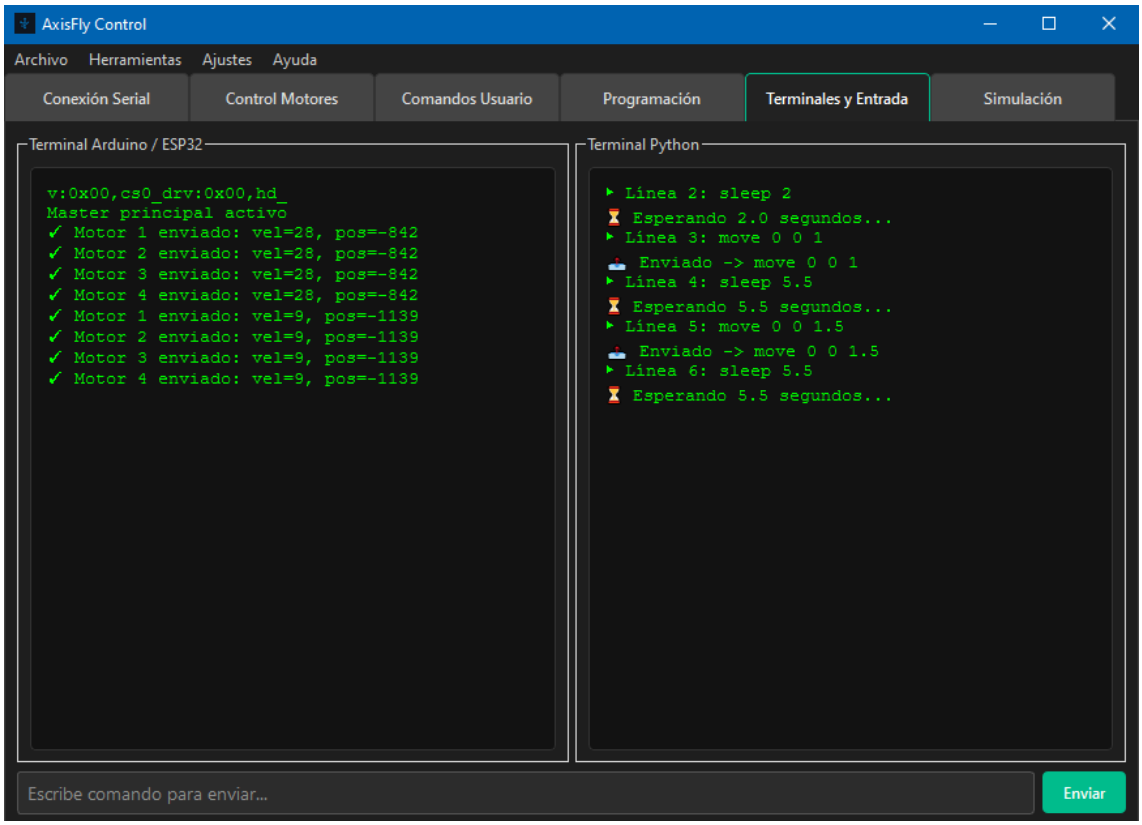


Figura 74. Aplicación diseñada – Terminales y Entrada

Además de la interfaz de acceso rápido y programación que se ha visto antes tambien se puede enviar comandos directamente desde la entrada de comandos de la parte inferior de esta ventana como si estuvieras en la interfaz de Arduino. Una vez que este escrito pulsa intro o da a enviar. Para ayuda se puede mandar el comando help o simplemente ir a la ayuda de programación ya que son los mismos comandos (ver figura 75)

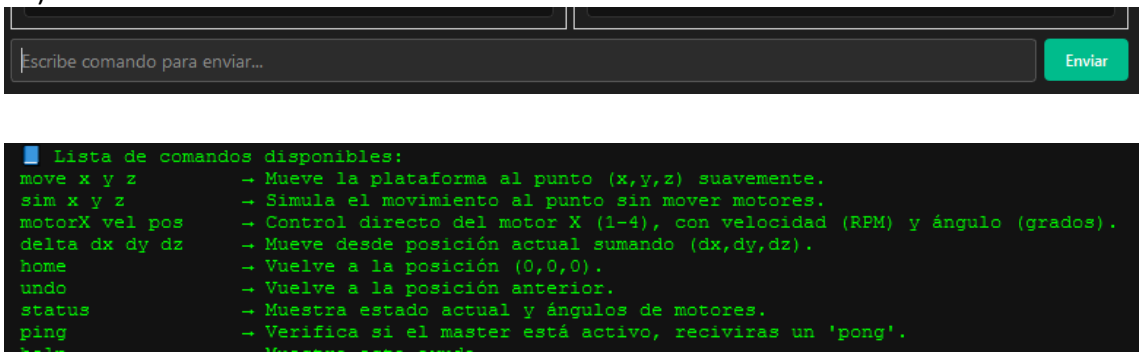


Figura 75. Aplicación diseñada – Terminales y Entrada

Simulación

Esta ventana es una pequeña interfaz de simulación para comprobar la cinemática y observar en que posición se coloca la plataforma, además de indicar a que ángulos y longitudes corresponde esta posición (ver figura 76).

En la parte de la derecha se muestra un cuadro de texto que mostrara los ángulos y longitudes de los cables necesarios para estar en la posición solicitada. Es una interfaz más interactiva que el Excel que se usó inicialmente.

En la parte de la derecha se pude ver un gráfico de tres dimensiones interactivo, encima de este se encuentran las herramientas del gráfico (zoom, desplazar, ajustes, guardar imagen, …), este grafico solo muestra la posición con los cables no simula ninguna trayectoria, para actualizar el gráfico una vez mandado un comando hay que clicar sobre el propio gráfico.

En la parte inferior esta la entrada de comandos donde solo se pueden escribir dos comandos para simular la posición y el estado del sistema, el comando move x y z y el comando delta dx dy dz visto anteriormente.

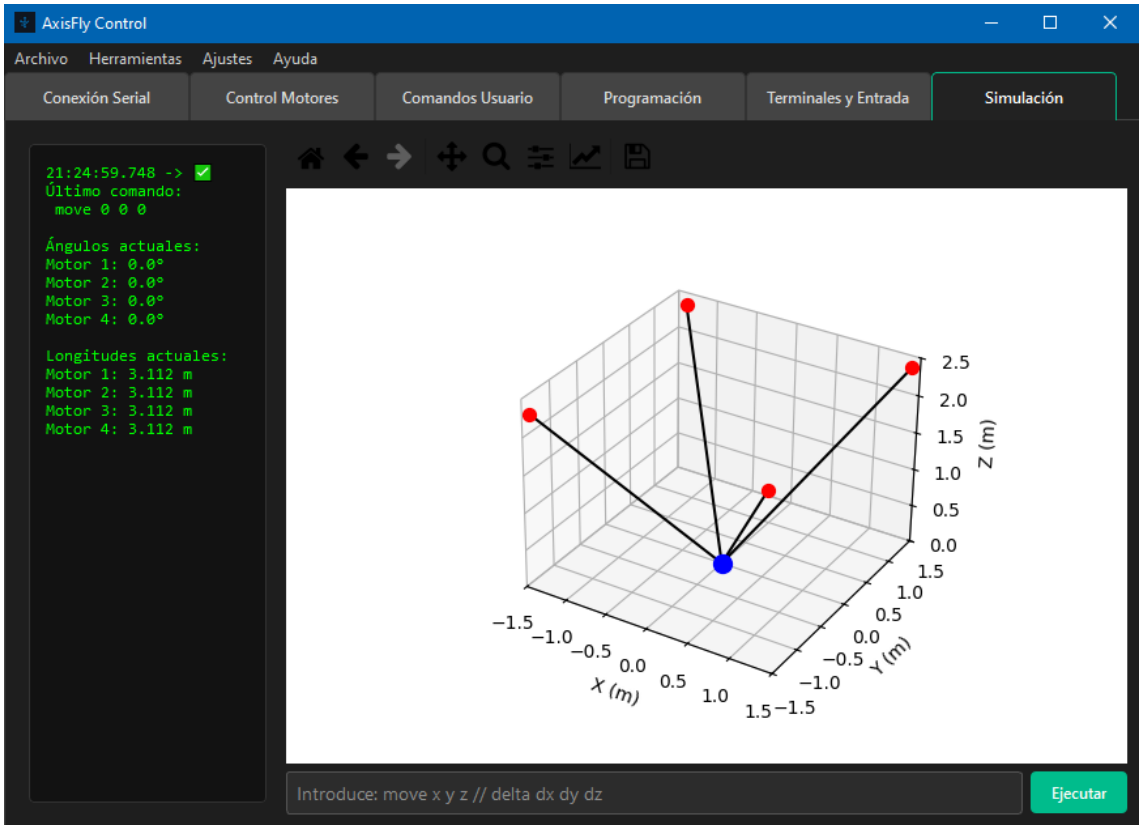
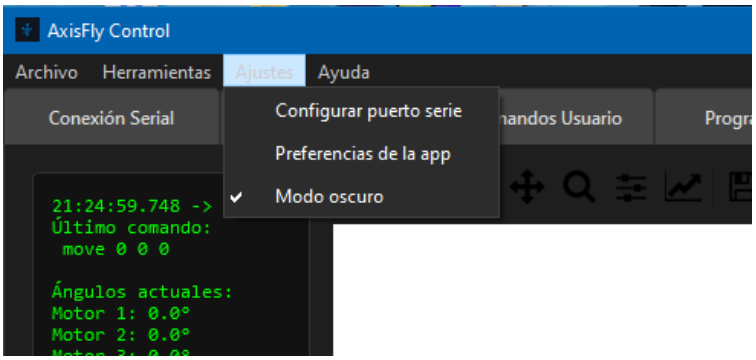


Figura 76. Aplicación diseñada – Simulación

Otras herramientas

Además de todas las herramientas que se han ido mostrando tambien se pude cambiar el tema de la aplicación que por defecto está en modo oscuro, este se pude desactivar en Ajustes --> modo oscuro, una vez desactivado la aplicación cambia (ver figura 77).



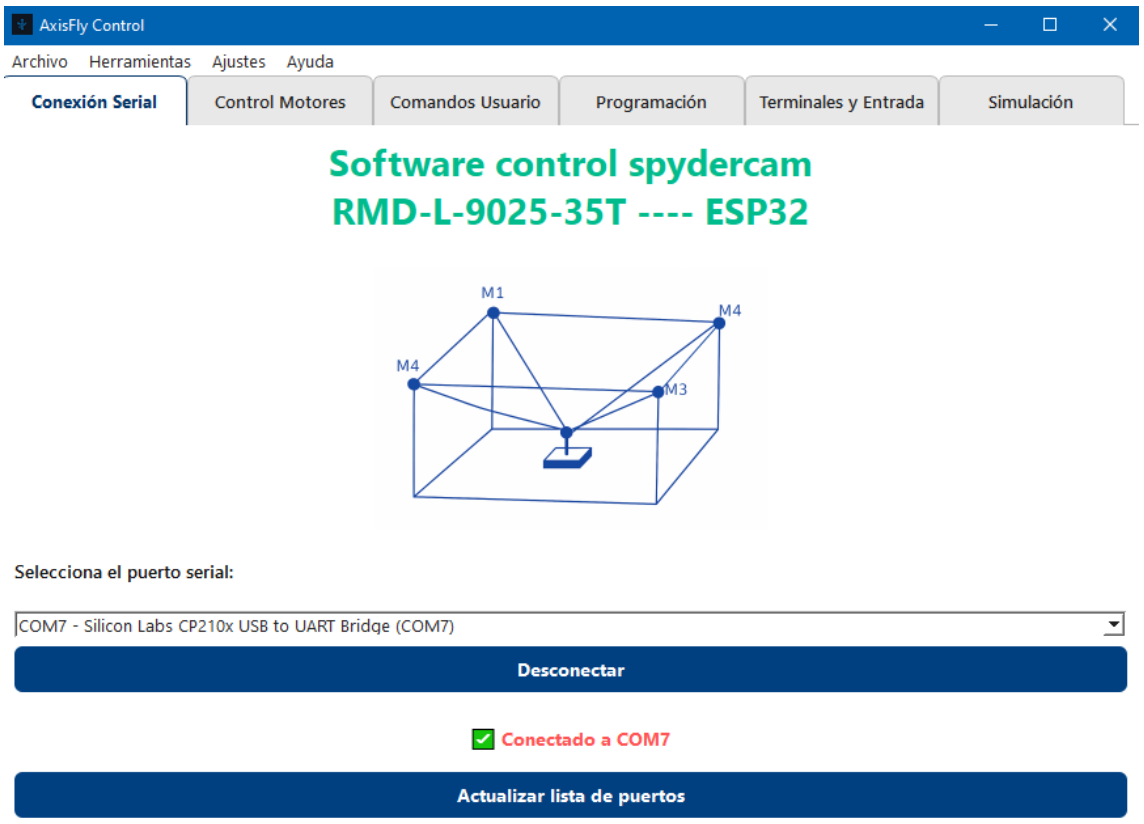


Figura 77. Aplicación diseñada – Modo claro/oscuro

Para cualquier duda del funcionamiento del software, se ha diseñado dentro un manual, donde explica en detalle el uso de la aplicación, además de una información de la aplicación, se pude encontrar en el menú en la pestaña Ayuda (ver figura 78).

Figura 78. Aplicación diseñada – Ayuda

Al dar al manual de usuario aparecerá una ventana emergente con dicho manual, donde se puede encontrar la información del uso del programa y contiene una explicación de la aplicación, como se observa en la figura 79.

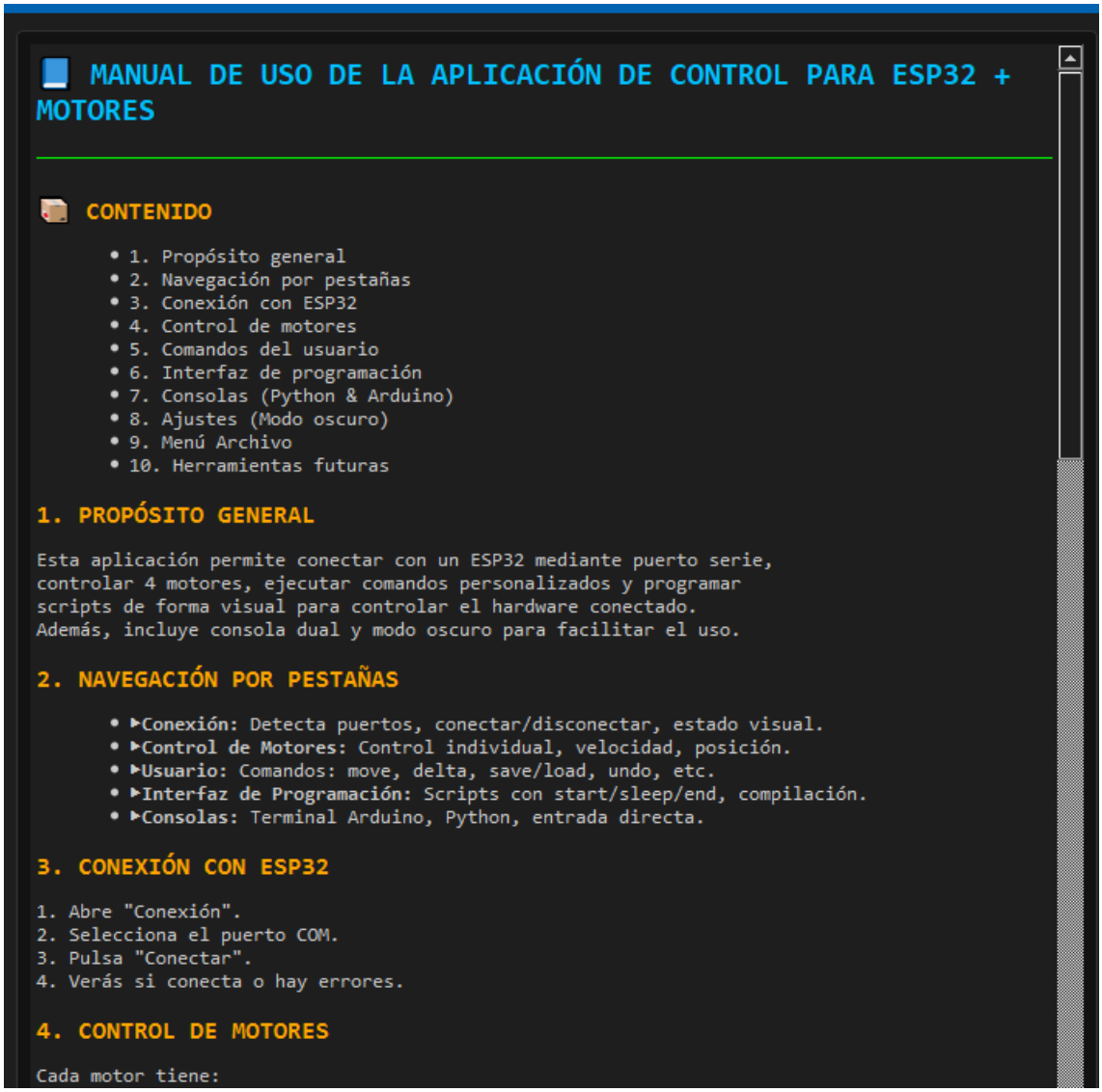


Figura 79. Aplicación diseñada – Manual