UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Diseño, construcción y control de un brazo robótico SCARA de código abierto con capacidad de posicionamiento en cuatro puntos, como base para futuras aplicaciones en automatización

Trabajo de graduación presentado por Sergio Alejandro Vásquez Marroquín en modalidad de trabajo profesional para optar al grado académico de Licenciado en Ingeniería en Mecatrónica

Guatemala,
2025

UNIVERSIDAD DEL VALLE DE GUATEMALA

Facultad de Ingeniería

Diseño, construcción y control de un brazo robótico SCARA de código abierto con capacidad de posicionamiento en cuatro puntos, como base para futuras aplicaciones en automatización

Trabajo de graduación presentado por Sergio Alejandro Vásquez Marroquín en modalidad de trabajo profesional para optar al grado académico de Licenciado en Ingeniería en Mecatrónica

Guatemala,
2025

Vo. Bo.

Firma: _____

Ing. Dulce María Chacón Muñoz

Firma: _____

Ing. Hector Alejandro Klée González

Fecha de aprobación: 20 de noviembre del 2025

CONTENIDO

PREFACIO

Este trabajo representó el resultado de muchos meses de esfuerzo, aprendizaje y dedicación. Desde el inicio, la idea de diseñar este proyecto no fue solo un reto académico, sino también una gran oportunidad para aplicar todo lo aprendido durante mi formación como ingeniero mecatrónico. Cada etapa del proyecto reflejó valores como la perseverancia, el pensamiento crítico y el trabajo en equipo.

A lo largo de este proceso, se enfrentaron varios desafíos técnicos y personales que me ayudaron a crecer de manera profesional. Más allá de los resultados obtenidos, este proyecto simbolizó la culminación de una etapa muy esperada e importante en mi vida universitaria, ya que no representó el final, sino el inicio de algo más grande.

Agradezco profundamente a Dios, a mis asesores, profesores, familia, novia y todos mis amigos por apoyarme, creer en mí y motivarme incluso en los momentos más agotadores del proceso. Primeramente, doy gracias a Dios, porque Él habló a su corazón durante todo el desarrollo del proyecto; en los momentos de estrés y presión, me dio paz para seguir adelante y no rendirme. También agradezco a mi familia, ya que gracias a la forma en que me habían criado, aprendí a sobrellevar los problemas que se presentaron en el camino.

Además, agradezco a todos mis amigos de la universidad que fueron testigos del crecimiento de este proyecto y siempre me animaron a continuar, haciéndome recomendaciones, ayudándome y retroalimentando cualquier detalle que se me pasara por alto. Finalmente, agradezco profundamente a mi novia, porque sin su apoyo emocional en los momentos de desvelo no habría logrado avanzar con la misma fuerza. Me acompañó en cada etapa, desde el inicio hasta el final. Le deseo muchos éxitos en su futuro y espero poder seguir juntos para disfrutar de todo lo que Dios tenga preparado para ambos.

LISTA DE FIGURAS

LISTA DE CUADROS

RESUMEN

Este trabajo propuso el desarrollo de un brazo robótico tipo SCARA (*Selective Compliance Assembly Robot Arm*) como parte del proceso de formación académica en Ingeniería Mecatrónica. El objetivo principal fue diseñar, construir y programar un prototipo funcional que fuera capaz de posicionarse en cuatro ubicaciones predeterminadas. A diferencia de proyectos industriales completos, este trabajo no contempló la implementación total de una línea de automatización, sino que se enfocó en sentar las bases mecánicas, electrónicas y de control que permitieran su uso para fases futuras, como la integración con sistemas automatizados.

El desarrollo del prototipo inició con el diseño del sistema mecánico del brazo, tomando en cuenta parámetros como alcance, límites, materiales accesibles y facilidad de ensamblaje. Para ello, se emplearon herramientas de diseño asistido por computadora (CAD), específicamente Autodesk Inventor, a fin de modelar cada componente del sistema y asegurar la compatibilidad y funcionalidad entre las partes. Posteriormente, se procedió a la fabricación del prototipo utilizando técnicas de impresión 3D y mecanizado básico, con el objetivo de mantener bajos costos sin comprometer la calidad estructural ni la precisión mecánica del sistema.

En cuanto al sistema de control, se implementó en un microcontrolador Arduino Pro Portenta *machine control*, para aplicaciones industriales y sobre todo educativas. Este se encargó de recibir instrucciones y ejecutar los movimientos mediante motores paso a paso, controlados por drivers adecuados para garantizar exactitud en el posicionamiento. El algoritmo de control fue desarrollado en un lenguaje de programación de manera que permitiera la ejecución de trayectorias predefinidas hacia las cuatro posiciones establecidas y, sobre todo, que pudiera ser fácilmente editado para futuras modificaciones del proyecto.

Durante el proceso de validación del prototipo, se aplicaron pruebas de precisión y repetibilidad. Se documentaron posicionamientos fallidos y tiempo de respuesta de los actuadores, con el fin de evaluar la calidad del diseño y la robustez del sistema de control implementado. Aunque esta etapa no incluyó la interacción con procesos externos, como visión por computadora o bandas transportadoras, se dejaron recomendaciones técnicas para que estas integraciones pudieran realizarse en etapas posteriores del megaproyecto.

Este brazo robótico pudo utilizarse como plataforma de enseñanza, prototipo de investigación o base para el desarrollo de sistemas automatizados dentro del entorno universitario. El proyecto representó un ejercicio integral de diseño, manufactura y control que permitió consolidar habilidades clave del perfil profesional del ingeniero mecatrónico, y promovió el uso de tecnologías accesibles y replicables en entornos educativos.

ABSTRACT

This work proposes the development of a SCARA-type robotic arm (*Selective Compliance Assembly Robot Arm*) as part of the academic training process in Mechatronic Engineering. The main objective is to design, build, and program a functional prototype capable of positioning itself in four predetermined locations. Unlike full industrial projects, this work does not include the total implementation of an automation line but focuses on establishing the mechanical, electronic, and control foundations that will enable its use in future phases, such as integration with automated systems.

The development of the prototype begins with the design of the arm's mechanical system, taking into account parameters such as reach, limits, accessible materials, and ease of assembly. For this purpose, computer-aided design (CAD) tools will be used, specifically Autodesk Inventor, to model each system component and ensure compatibility and functionality between parts. Subsequently, the prototype will be manufactured using 3D printing and basic machining techniques, with the goal of keeping costs low without compromising the structural quality or mechanical precision of the system.

Regarding the control system, it will be implemented using an Arduino Portenta *machine control* microcontroller, intended for industrial and educational applications. This controller will be responsible for receiving instructions and executing movements through stepper motors, operated by appropriate drivers to ensure accurate positioning. The control algorithm will be developed in a programming language that allows the execution of predefined trajectories toward the four established positions and can be easily edited for future project modifications.

During the prototype validation process, precision and repeatability tests will be applied. Failed positionings and actuator response times will be documented in order to evaluate the design quality and robustness of the implemented control system. Although this stage will not include interaction with external processes, such as computer vision or conveyor belts, technical recommendations will be provided to enable these integrations in later stages of the megaproject.

This robotic arm may be used as a teaching platform, research prototype, or foundation for developing automated systems within the university environment. The project represents a comprehensive exercise in design, manufacturing, and control that consolidates key skills within the professional profile of a mechatronic engineer and promotes the use of accessible and replicable technologies in educational settings.

I. INTRODUCCIÓN

El avance constante de la tecnología impulsó la integración de sistemas automatizados en diversos entornos. Dentro de estos sistemas, los brazos robóticos jugaron un papel clave en tareas repetitivas, de precisión o de manipulación de objetos, contribuyendo a la eficiencia y calidad de los procesos. Uno de los tipos más utilizados en la industria moderna fue el brazo SCARA (*Selective Compliance Assembly Robot Arm*), por su diseño compacto, velocidad y capacidad de realizar desplazamientos en el plano horizontal con alta precisión. En este contexto, la formación académica en robótica aplicada se volvió fundamental para preparar a los futuros profesionales en el desarrollo e implementación de estas soluciones.

El trabajo de graduación se centró en el diseño, construcción y programación de un brazo robótico tipo SCARA con la capacidad de posicionarse en cuatro ubicaciones específicas. Este prototipo tuvo como propósito establecer una base sólida para fases posteriores de automatización dentro del Laboratorio de Diseño de Procesos de la Universidad del Valle de Guatemala, sin contemplar aún la integración con una línea de empaque o sensores industriales avanzados. Esta fase inicial buscó desarrollar una solución funcional y accesible que pudiera ser utilizada como herramienta de aprendizaje y como plataforma para futuras investigaciones o mejoras.

La utilidad principal del proyecto radicó en su enfoque formativo y replicable. Al emplear recursos y herramientas como Autodesk Inventor para el diseño CAD, Arduino Pro Portenta para el control, y procesos de fabricación interna como impresión 3D, se facilitó que otros estudiantes pudieran modificar, ampliar o adaptar el brazo robótico a sus propias necesidades académicas o proyectos. Adicionalmente, la documentación generada permitió que el diseño se mantuviera como una plataforma abierta y escalable.

El alcance del trabajo estuvo claramente delimitado: se desarrolló un prototipo funcional del brazo SCARA capaz de moverse con precisión hacia cuatro posiciones predeterminadas. Se contempló el modelado 3D del sistema, su fabricación física, la programación del sistema de control con Arduino y la validación de funcionamiento mediante pruebas básicas de precisión. No se abordó aún la integración con sistemas externos de automatización industrial, visión artificial ni control por retroalimentación avanzada. Estas áreas quedaron propuestas como fases futuras del megaproyecto.

La estructura del documento estuvo organizada en partes que abarcaron desde el planteamiento del problema y la justificación del proyecto, hasta la descripción de objetivos, marco teórico, metodología, resultados preliminares esperados y conclusiones. Cada sección fue redactada siguiendo los lineamientos establecidos por la Universidad del Valle de Guatemala para trabajos de graduación en la modalidad de megaproyecto.

II. OBJETIVOS

A. Objetivo general

- Diseñar, construir y controlar un brazo robótico SCARA de código abierto con capacidad de posicionamiento en cuatro puntos, como base para futuras aplicaciones en automatización.

B. Objetivos específicos

- Modelar en Autodesk Inventor la estructura mecánica del brazo SCARA, con el objetivo de garantizar estabilidad y funcionalidad mediante técnicas de diseño asistido por computadora (CAD).

- Fabricar físicamente el brazo robótico SCARA, para contar con una estructura que permita realizar pruebas de movimiento impresión 3D y ensamblaje de componentes.

- Programar el sistema de control del brazo SCARA, con el objetivo de lograr el posicionamiento preciso en cuatro ubicaciones específicas, mediante el uso del microcontrolador Arduino Pro Portenta.

- Evaluar la precisión y repetibilidad del brazo robótico, con el fin de validar el correcto funcionamiento del sistema en las posiciones definidas, mediante pruebas repetitivas.

- Documentar el proceso de diseño, implementación y pruebas, para facilitar la continuidad del proyecto en fases futuras, mediante la elaboración de informes técnicos, esquemas de conexión, y código fuente comentado.

III. JUSTIFICACIÓN

El desarrollo de sistemas robóticos ha cobrado una creciente relevancia en la industria moderna debido a su capacidad de automatizar tareas repetitivas, reducir errores humanos y aumentar la eficiencia de los procesos productivos. Sin embargo, en países en vías de desarrollo como Guatemala, el acceso a este tipo de tecnología es limitado, tanto por los altos costos de implementación como por la escasa disponibilidad de prototipos funcionales para fines académicos. En este contexto, el diseño y construcción de un brazo robótico tipo SCARA (*Selective Compliance Assembly Robot Arm*) se presenta como una solución viable para fomentar el aprendizaje práctico, al mismo tiempo que permite sentar las bases para futuros desarrollos en el ámbito de la automatización.

Este trabajo surge como respuesta a la necesidad de generar experiencias reales de diseño, manufactura y programación de sistemas, que integren los conocimientos adquiridos a lo largo de la carrera. A través de la implementación de un prototipo funcional, se busca fortalecer habilidades técnicas como el diseño asistido por computadora (CAD), la selección y ensamblaje de componentes mecánicos, la programación de microcontroladores y la validación de resultados. La elección del tipo SCARA responde a su relevancia en aplicaciones industriales por su precisión y velocidad en trayectorias horizontales, lo que lo convierte en un excelente modelo para fines educativos.

El principal beneficio de este proyecto es su enfoque pedagógico. Al tratarse de una construcción desde cero, el estudiante adquiere una comprensión profunda de cada etapa del desarrollo, lo cual es difícil de lograr únicamente con simulaciones o componentes prefabricados. Además, el prototipo resultante podrá ser utilizado por otros estudiantes como plataforma de pruebas, facilitando así la enseñanza de temas como control de movimiento, retroalimentación de sistemas, y diseño mecánico orientado a manufactura.

Adicionalmente, se promueve el uso de herramientas accesibles y de bajo costo, como el software Autodesk Inventor para el diseño y la placa Arduino Pro Portenta *machine control* para el control. Estas herramientas son económicas en comparación con otras alternativas industriales, lo que hace que el proyecto sea viable sin la necesidad de equipos especializados. Esta decisión estratégica permite replicar el proyecto en universidades o centros educativos con recursos limitados, facilitando el acceso a tecnologías de vanguardia a un costo reducido. La fabricación del brazo se llevará a cabo con tecnologías disponibles dentro de la universidad, como impresión 3D, corte láser y fresado básico, reforzando el uso de los recursos locales y fomentando la autosuficiencia tecnológica.

Otro beneficio esperado radica en la documentación generada a lo largo del proceso. Se pretende dejar un registro técnico detallado que incluya diseños, esquemas eléctricos, código fuente comentado, y resultados de pruebas, de forma que futuros estudiantes puedan

continuar con el desarrollo del sistema. Si bien este protocolo no contempla la integración directa del brazo en una línea de automatización, se diseña con esa posibilidad en mente, asegurando que la arquitectura sea modular y escalable.

Finalmente, este proyecto también puede contribuir a la visibilidad de la carrera de Ingeniería Mecatrónica dentro de la Universidad del Valle de Guatemala, al demostrar que es posible realizar desarrollos tecnológicos relevantes con recursos locales y conocimientos adquiridos durante la formación académica. El brazo robótico SCARA no solo representa un producto tangible, sino también una oportunidad de crecimiento profesional para el autor, y una herramienta valiosa para la comunidad educativa.

En resumen, el presente trabajo se justifica por su contribución a la formación integral del estudiante, por los beneficios académicos que ofrece a la institución y por su potencial como base para proyectos de automatización más complejos en fases posteriores. A través de este esfuerzo se busca consolidar una cultura de innovación práctica, accesible y replicable en el campo de la robótica y la mecatrónica.

IV. ALCANCE Y LIMITACIONES

A. Disponibilidad de recursos

El proyecto contó con los recursos necesarios proporcionados por los laboratorios de Ingeniería Mecatrónica de la Universidad del Valle de Guatemala. Entre estos recursos se incluyeron acceso a impresoras 3D, cortadora láser (si fuera necesario), estaciones de ensamblaje, software de diseño como Autodesk Inventor, así como plataformas de desarrollo y microcontroladores Arduino Pro Portenta, *drivers* y fuentes de voltaje. También se contó con el apoyo técnico de asesores y personal especializado.

B. Disponibilidad de tiempo

El proyecto fue planificado para ejecutarse dentro del ciclo académico establecido por la universidad. Se elaboró un cronograma semanal con la metodología planteada, el cual contempló tiempo para investigación, diseño, manufactura, programación, pruebas, validación y documentación. Se dispuso del tiempo suficiente para cubrir todas las etapas hasta la entrega del prototipo funcional.

C. Acceso a fuentes de información

El estudiante tuvo acceso a diversas fuentes de información, tanto primarias como secundarias, a través de la biblioteca virtual UVG, bases de datos académicas y repositorios de publicaciones científicas.

- Fuentes primarias: documentación técnica de fabricantes, temas de clases, manuales de controladores y motores, *datasheets* de componentes electrónicos, consultas a asesores académicos y simulaciones propias realizadas en MATLAB y Autodesk Inventor.
- Fuentes secundarias: libros especializados como los de (Siciliano, 2010; Corke, 2017; Niku, 2010), y artículos científicos obtenidos de IEEE Xplore, SpringerLink y MIT Press, entre otros.

D. Limitaciones del estudio

El alcance del proyecto se limitó al desarrollo de un prototipo funcional del brazo robótico tipo SCARA con la capacidad de posicionarse en cuatro puntos definidos. No se contempló la integración completa con una línea de automatización, visión artificial ni la ejecución de tareas con interacción adaptativa compleja. Las limitaciones técnicas incluyeron el uso de materiales de bajo costo, la precisión dependiente de componentes educativos y la restricción a un entorno controlado dentro del laboratorio.

E. Autorización para publicación

Dado que el proyecto no involucró información sensible ni confidencial, y estuvo basado en un diseño original y de código abierto, se contó con la libertad de publicar los resultados, diseños, planos, código fuente y documentación generada, bajo licencias que permitieran su replicación y mejora por otros estudiantes.

F. Áreas de excelencia que abarca

Este proyecto se enmarcó principalmente dentro de las siguientes áreas de excelencia definidas por la Universidad del Valle de Guatemala:

MECATRÓNICA – Área de automatización y robótica
El diseño e implementación del brazo robótico SCARA implicó la integración de varias disciplinas dentro del campo de la mecatrónica, tales como el diseño mecánico, la electrónica, la programación de sistemas embebidos y el control automático. En particular, se enfocó en la automatización de procesos mediante la construcción de un sistema robótico que permitió el posicionamiento preciso en puntos específicos. Este tipo de desarrollo contribuyó al avance de la robótica en aplicaciones industriales y educativas, y reforzó la capacidad de los estudiantes para aplicar teorías y herramientas de vanguardia en el diseño de sistemas automatizados. A través de esta integración de mecánica, electrónica y programación, el proyecto buscó fortalecer el conocimiento en el área de robótica, que se consideró una de las disciplinas clave dentro del campo de la mecatrónica.

V. MARCO TEÓRICO

A. Robótica

La robótica es una disciplina interdisciplinaria que combina principios de la ingeniería mecánica, electrónica, control y hasta ciencias computacionales para el diseño, construcción y operación de sistemas automáticos y semiautomáticos capaces de percibir su entorno, tomar decisiones y ejecutar acciones físicas. Uno de los tipos más representativos de robots en el ámbito industrial y académico son los brazos robóticos, los cuales imitan la estructura y funcionalidad del brazo humano para realizar tareas que requieren precisión, repetitividad y velocidad.

Dentro del campo de la robótica, los brazos robóticos han demostrado ser herramientas fundamentales en la automatización de procesos, especialmente en industrias como la manufactura, ensamblaje, empaquetado y medicina. Su estructura articulada permite realizar movimientos programados en uno o varios ejes, lo que los convierte en sistemas altamente adaptables a una amplia gama de tareas repetitivas o peligrosas para el ser humano (Siciliano, 2010; Lynch, 2017).

Desde el punto de vista del control, los brazos robóticos ofrecen un entorno ideal para la implementación de algoritmos de cinemática, dinámica, planificación de trayectorias y realimentación sensorial. Esto los convierte no solo en herramientas para la producción, sino también en plataformas educativas para el desarrollo de habilidades en ingeniería mecatrónica, electrónica, mecánica y hasta ciencias de la computación (Corke, 2017).

A nivel estructural, un brazo robótico está compuesto por una serie de eslabones rígidos conectados mediante articulaciones que pueden ser rotacionales o prismáticos. Cada articulación representa un grado de libertad (DOF, por sus siglas en inglés), y su cantidad y disposición definen la capacidad del robot para alcanzar diferentes posiciones y orientaciones en el espacio de trabajo. Estas configuraciones pueden ser clasificadas en varios tipos, incluyendo brazos cartesianos, cilíndricos, esféricos, antropomórficos y SCARA (*Selective Compliance Assembly Robot Arm*), (Correll, 2022).

En la actualidad, el diseño y programación de brazos robóticos también se ha facilitado gracias al acceso a plataformas de código abierto y software especializado, lo cual ha permitido su expansión en contextos educativos, makerspaces y laboratorios universitarios. Su versatilidad y aplicabilidad los convierten en un componente clave para la formación de profesionales en áreas STEM, ya que integran conocimientos de diseño mecánico, electrónica, sensores y programación (Siegwart, 2011).

B. Brazos robóticos SCARA

Los brazos robóticos SCARA (por sus siglas en inglés: *Selective Compliance Assembly Robot Arm*) constituyen una arquitectura cinemática especialmente diseñada para realizar movimientos rápidos y precisos en el plano horizontal, característica que los ha convertido en una solución ideal para procesos de ensamblaje y manipulación ligera. Este tipo de robot fue concebido a inicios de la década de 1980 por Hiroshi Makino, en la Universidad de Yamanashi, Japón, como una alternativa eficiente a los costosos y voluminosos brazos cartesianos utilizados en la industria electrónica (Makino, 1981; Siciliano, 2010).

La principal particularidad de un SCARA radica en su *compliance* selectivo, lo que significa que es rígido en el eje vertical (Z), pero presenta cierta flexibilidad en el plano XY. Esto le permite realizar inserciones con tolerancias mínimas sin generar daños en las piezas ni requerir precisión absoluta de alineamiento. Su cinemática le otorga tres o cuatro grados de libertad, normalmente distribuidos entre dos rotaciones horizontales, una traslación vertical y, en algunos modelos, una rotación del efector final. Esta configuración le permite ejecutar tareas de "*pick and place*" con gran velocidad, eficiencia energética y bajo mantenimiento (Correll, 2022; Lynch, 2017).

La estructura típica de un SCARA incluye:

- Dos eslabones giratorios unidos por juntas rotacionales horizontales.
- Un eje vertical que permite movimientos ascendentes y descendentes (articulación prismática).
- Una base fija y, en algunos casos, una articulación adicional para rotar la herramienta o garra (efector final).

En comparación con otras configuraciones, como los robots cartesianos, cilíndricos o antropomórficos, los SCARA destacan por su alta velocidad de operación (hasta 120 ciclos/min en modelos industriales), bajo consumo energético y precisión de repetibilidad que puede llegar a ± 0.01 mm en modelos comerciales (Motoman, 2020; Festo, 2019).

Estas características han posicionado a los brazos SCARA como una solución dominante en líneas de ensamblaje electrónico, empaquetado, paletizado, etiquetado, y alimentación de máquinas. Además, su capacidad para manipular piezas livianas con rapidez y su estructura compacta los hace ideales para espacios reducidos y procesos donde la productividad es crítica (Wessling, The Robot Report, 2024; Bogue, 2012).

En contextos educativos y académicos, los SCARA también han ganado protagonismo gracias a la disponibilidad de versiones modulares, kits de código abierto y documentación técnica accesible. Plataformas como Annin Robotics, DOBOT, Innova Mecatrónica y múltiples proyectos comunitarios permiten replicar estos brazos utilizando impresión 3D, microcontroladores como Arduino, y sistemas de bajo costo, facilitando su integración en

laboratorios universitarios o *makerspaces* (Annin Robotics, 2019; Mircescu, vojjta, & Tsogoo, 2021).

Desde la perspectiva del aprendizaje en ingeniería mecatrónica, los brazos SCARA ofrecen una valiosa oportunidad para aplicar conocimientos en diseño mecánico, cinemática directa e inversa, control en lazo abierto y cerrado, y programación de trayectorias, consolidando así una formación integral en robótica aplicada.

C. Motores paso a paso NEMA 17

Los motores paso a paso NEMA 17 son motores eléctricos que giran en "pasos" muy pequeños y controlados. El término NEMA 17 no describe el modelo exacto del motor, sino el tamaño del marco: 42 × 42 mm en la cara frontal (Motion Control Products, s. f.).

En la robótica y en impresoras 3D se usan mucho porque:

- Permiten controlar la posición del eje sin necesidad de un sensor externo.
- Cada "paso" del motor corresponde a un ángulo fijo, normalmente 1.8° por paso, lo que significa 200 pasos para dar una vuelta completa (Robocraze, 2022).
- Son motores bipolares de dos fases en la mayoría de aplicaciones, lo que facilita el control con *drivers* específicos para *stepper*.

Un NEMA 17 típico puede tener:

- Ángulo de paso: 1.8°
- Par de sujeción (*holding torque*): desde ~0.13 N·m hasta ~0.7 N·m, dependiendo del modelo.
- Corriente por fase: alrededor de 0.4 a 1 A, según la versión (Stepperonline, s. f.).

En proyectos de brazos robóticos tipo SCARA, estos motores se usan para los eslabones principales porque:

- Ofrecen buen compromiso entre tamaño, torque y precisión.
- Son relativamente económicos y fáciles de conseguir.
- Trabajan bien con controladores estándar (por ejemplo, *drivers* tipo DM542T, A4988, etc.), lo que simplifica la integración con microcontroladores.

D. Servo MG90S

El MG90S es un microservo muy usado en robótica pequeña, drones y proyectos con Arduino. Es un servo compacto, de unos 22–23 mm de largo, con un peso cercano a 13.4 g, equipado con engranajes metálicos, lo que le da mayor resistencia que los servos plásticos (Electronicoscaldas, s. f.).

Sus características principales son:

- Voltaje de operación: 4.8 – 6.0 V (normalmente se usa a 5 V) (Components101, 2019).
- Torque en paro: alrededor de 1.8 kg · cm a 4.8 V y hasta 2.2–2.8 kg · cm a 6 V, según el fabricante.
- Velocidad típica: ~0.1 s/60° a 4.8 V (es decir, gira 60 grados en una décima de segundo sin carga).
- Rango de giro: aproximadamente 180° (90° hacia cada lado).
- Tipo de engranajes: metálicos, lo que mejora la durabilidad frente a cargas repetitivas.

En un brazo robótico SCARA como el tuyo, el MG90S es especialmente útil en la garra o efector final, porque:

- Permite abrir y cerrar la pinza con precisión.
- Puede rotar la garra para orientar el objeto sin necesidad de añadir otro motor grande.
- Su tamaño pequeño y su bajo peso ayudan a no sobrecargar el segundo eslabón del brazo.

Además, al ser un servo controlado por señal PWM, su posición se puede manejar fácilmente desde un microcontrolador (como Arduino Pro Portenta), enviando un pulso cuyo ancho indica el ángulo que debe tomar el eje del servo.

E. Ventajas de los brazos SCARA

Los brazos robóticos tipo SCARA ofrecen una combinación de velocidad, precisión y simplicidad que los posiciona como una de las arquitecturas más eficientes para tareas repetitivas en el plano horizontal. Su diseño cinemático optimizado para movimientos en XY y una estructura rígida en el eje Z les permite destacar en procesos industriales que requieren alta productividad sin comprometer la precisión. A continuación, se detallan las principales ventajas que justifican su uso extendido en líneas de producción y su adopción en entornos académicos:

1. ***Alta velocidad operativa***

Gracias a su bajo peso móvil, sus articulaciones rotacionales y su estructura simplificada, los SCARA pueden alcanzar velocidades de ciclo que superan los 100 movimientos por minuto. Esta rapidez los hace ideales para operaciones de *pick-and-place*, ensamblaje de componentes electrónicos y transferencias entre estaciones de trabajo (Motoman, 2020; Wessling, The Robot Report, 2024).

2. ***Excelente repetibilidad y precisión***

Los SCARA logran niveles de repetibilidad entre ±0.01 mm y ±0.05 mm en modelos industriales, lo que los hace aptos para procesos donde el margen de error debe ser mínimo. Su estructura rígida en el eje vertical evita desviaciones durante la inserción de piezas, lo que es clave para operaciones de montaje delicado (Festo, 2019; Siciliano, 2010).

3. ***Compliance selectivo***

Una de las características distintivas del SCARA es su *compliance* selectivo: ofrece cierta flexibilidad en el plano horizontal (XY) pero mantiene rigidez vertical. Esto permite que, ante pequeñas desalineaciones durante tareas de inserción, el robot se adapte ligeramente sin causar daños, una ventaja sobre robots completamente rígidos (Makino, 1981; Lynch, 2017).

4. ***Diseño compacto y ahorro de espacio***

La configuración en plano horizontal y su base fija reducen significativamente el espacio ocupado por un SCARA en comparación con brazos antropomórficos o cartesianos. Esta ventaja es especialmente valiosa en entornos donde el área de trabajo es limitada o donde se requiere una alta densidad de estaciones automatizadas (Correll, 2022).

5. ***Menor complejidad de control***

Al tener una cinemática más simple que la de robots antropomórficos, el SCARA requiere menor carga computacional para resolver problemas de cinemática directa e inversa, facilitando su programación y calibración. Esta característica es especialmente beneficiosa para aplicaciones educativas, donde se busca enseñar fundamentos de robótica sin complejidad innecesaria (Corke, 2017; Bogue, 2012).

6. Versatilidad y adaptabilidad

Si bien su principal fortaleza está en el plano XY, el SCARA puede ser adaptado a tareas más complejas mediante la incorporación de un cuarto eje rotacional en el efector final, o con sistemas de visión artificial y retroalimentación sensorial. Además, su estructura abierta permite su construcción con tecnologías accesibles como impresión 3D o mecanizado CNC, favoreciendo su uso en prototipado o proyectos *open source* (Annin Robotics, 2019; Mircescu, vojjta, & Tsogoo, 2021).

F. Consideraciones y proceso para el diseño de un brazo robótico SCARA

El diseño de un brazo robótico tipo SCARA requiere una aproximación integral que combine fundamentos de mecánica, cinemática, dinámica, electrónica, control y programación. A diferencia de otros robots industriales, los brazos SCARA presentan una estructura altamente optimizada para movimientos rápidos y repetitivos en el plano horizontal, por lo que su diseño debe considerar tanto las exigencias funcionales como las limitaciones estructurales del entorno de aplicación.

1. Objetivos del diseño

El primer paso en el proceso de diseño es establecer claramente los objetivos funcionales del sistema. En el caso de un brazo SCARA, los objetivos típicos pueden incluir tareas como:

- Alcanzar y posicionarse con precisión en puntos específicos del espacio de trabajo (planos XY).
- Ejecutar acciones repetitivas con alta velocidad y baja variación.
- Manipular piezas de bajo peso, típicas en líneas de ensamblaje o empaque.
- Ser fácilmente replicable o editable, especialmente en entornos educativos.

Definir estos objetivos permite establecer los criterios de éxito del diseño, como la precisión esperada, el tiempo de ciclo, la carga útil máxima y los límites del espacio de trabajo. En el contexto académico, estos objetivos también se amplían a aspectos como la facilidad de construcción, disponibilidad de materiales, uso de herramientas de software accesibles y compatibilidad con microcontroladores ampliamente utilizados (Arduino, ESP32, etc.) (Craig, 2005; Correll, 2022).

G. Ecuaciones de referencia para el diseño y control de brazos robóticos SCARA

El diseño de un brazo robótico SCARA no se limita únicamente a aspectos estructurales o estéticos; requiere de una base matemática que permita asegurar que el sistema cumpla con los requisitos funcionales de precisión, velocidad, capacidad de carga y estabilidad. A continuación, se presentan los principales tipos de cálculos que sustentan este tipo de diseño, los cuales están estrechamente relacionados con la cinemática, dinámica y control del robot.

1. Cinemática directa e inversa

La cinemática directa permite determinar la posición del efector final del robot (la garra o herramienta) en el espacio, con base en los ángulos conocidos de las articulaciones. En un SCARA típico de dos grados de libertad rotacionales, esta relación se expresa mediante funciones trigonométricas:

$$x = L_1 \cos(\theta_1) + L_2 \cos(\theta_1 + \theta_2)$$
$$y = L_1 \sin(\theta_1) + L_2 \sin(\theta_1 + \theta_2) \quad (1)$$

Donde:

- L_1 y L_2 son las longitudes de los eslabones,
- θ_1 y θ_2 son los ángulos de rotación de las juntas.

Por otro lado, la cinemática inversa permite calcular los ángulos requeridos para que el efector alcance una posición deseada en el espacio. Resolver esta cinemática es fundamental para el desarrollo de algoritmos de control y programación del robot (Craig, 2005; Corke, 2017).

2. Análisis del espacio de trabajo

El espacio de trabajo (*workspace*) de un brazo SCARA es el conjunto de todas las posiciones que el efector puede alcanzar. Este espacio tiene forma de anillo o corona circular, y está delimitado por la suma y resta de las longitudes de los eslabones:

$$R_{max} = L_1 + L_2 \,, \qquad R_{min} = |L_1 - L_2| \quad (2)$$

Comprender este espacio es crucial para la disposición de estaciones, piezas y sensores en la zona operativa del robot (Siciliano, 2010; Correll, 2022).

3. Estimación del tiempo de ciclo

En procesos automatizados, el tiempo que toma el robot en completar una acción es crítico. Este se puede estimar con base en la velocidad angular máxima de los motores y la trayectoria a recorrer. Aunque no se exige un modelo dinámico avanzado en proyectos académicos, estimaciones prácticas ayudan a determinar la eficiencia general del diseño:

$$t = \frac{\Delta\theta}{\omega} \qquad (3)$$

Donde:

- $\Delta\theta$ es el ángulo total de rotación,
- ω es la velocidad angular del motor.

En líneas de producción reales, los brazos SCARA pueden alcanzar hasta 120 ciclos/minuto (Correll, 2022; Motoman, 2020).

4. Frecuencia de paso

En los brazos robóticos SCARA con motores a pasos, la velocidad de movimiento de cada articulación depende de la señal de pulsos que recibe el *driver* del motor. Esta señal suele representarse como una onda cuadrada, donde cada flanco (cambio de nivel lógico) se asocia a un avance de un paso. A partir del semiperiodo de esta señal es posible obtener la frecuencia de paso del motor:

$$f = \frac{1}{2 \cdot semiperiodo} \qquad (4)$$

Donde:

- f es la frecuencia de paso del motor $[Hz]$, es decir, el número de pasos por segundo.
- semiperiodo es el tiempo entre dos flancos consecutivos de la señal de paso $[s]$. (Ni, 2025)

5. Velocidad angular

Conocida la frecuencia de paso, puede determinarse la velocidad angular del eje del motor en función del número de pasos necesarios para completar una vuelta completa:

$$\omega = \frac{360\ f}{Paso\ por\ revolución} \tag{5}$$

Donde:

- ω es la velocidad angular del motor [°/s].
- f es la frecuencia de paso $[Hz]$.
- Pasos por revolución es el número total de pasos que requiere el motor para girar $360°$.

Estas relaciones permiten vincular la señal de control (frecuencia de pulsos) con la velocidad real de las articulaciones del brazo SCARA, y se integran con ecuaciones como $t = \frac{\Delta\theta}{\omega}$ para estimar tiempos de ciclo y evaluar el desempeño del robot (Open Stax College Physics).

VI. METODOLOGÍA

La metodología adoptada para este trabajo siguió una estructura por objetivos y se desarrolló en varias fases secuenciales, enfocadas en el diseño, validación e implementación parcial de un brazo robótico tipo SCARA. El proceso estuvo orientado a cumplir con los objetivos generales y específicos del proyecto, garantizando una integración adecuada entre el diseño mecánico, la selección de componentes y el sistema de control.

A. Tipo de investigación y justificación

Este trabajo se enmarcó en un enfoque descriptivo y experimental. Fue descriptivo porque se analizaron y compararon distintos diseños previos de brazos SCARA, tanto de código abierto como comerciales, para identificar características funcionales para el prototipo a implementar. El proceso fue también experimental porque implicó la validación práctica de un diseño propio, mediante pruebas de montaje, operación y precisión, permitiendo realizar ajustes conforme se identificaron oportunidades de mejora en el diseño físico o electrónico.

B. Fases del desarrollo y recopilación de datos

En la primera fase se realizó una revisión de proyectos existentes disponibles en plataformas web y repositorios de diseño, para seleccionar una base sólida que pudiera ser fácilmente modificada y reutilizada por futuros estudiantes. Se analizaron tres diseños de código abierto; dos de ellos fueron descartados debido a que usaban mecanismos con poleas y correas específicas difíciles de encontrar en Guatemala, lo cual dificultaba su modificación para alcanzar mayores áreas de trabajo. Finalmente, se seleccionó un diseño de la página Innova Mecatrónica que ofrecía mayor flexibilidad de adaptación. A pesar de también utilizar poleas de tamaños específicos, utilizaba medidas comerciales fáciles de encontrar y, a su vez, implementarlas si fuera necesario.

En la segunda fase, se llevó a cabo la modificación del diseño original utilizando la herramienta de diseño asistido por computadora Autodesk Inventor. Se ajustaron las dimensiones de los eslabones del brazo para adaptarlos al área de trabajo del laboratorio y se diseñó un ensamblaje completo con todos los componentes requeridos para la construcción del prototipo.

De manera paralela, se trabajó en el diseño de la garra o actuador final. A través de la investigación de modelos industriales y comerciales, se determinó que una pinza de dos puntos de contacto sería la opción más adecuada, priorizando el bajo peso y la capacidad de rotación de 360°. La referencia principal fue un diseño adaptado a partir del modelo de

Cults3D por el autor GG_CADWorks Gripper “SG90 (2DOF)” (GG_CADworks, 2025), que sirvió como base para desarrollar una pinza funcional con rotación incluida.

C. Herramientas e instrumentos utilizados

- Autodesk Inventor: modelado 3D de las piezas y ensamblajes del brazo.
- Arduino Pro Portenta *machine control*: para el sistema de control del brazo.
- Motores paso a paso y *drivers* específicos: para el movimiento de los eslabones y la garra.
- Impresora 3D y cortadora láser: fabricación de componentes mecánicos y posible base.
- Componentes electrónicos: para las conexiones con el Microcontrolador y los motores.
- Cinta métrica, vernier y herramientas de prototipado: para validación de dimensiones físicas.

D. Análisis de datos y validación

Los datos recolectados durante las pruebas del prototipo incluyeron el número de errores en posicionamiento, tiempos de respuesta y repetibilidad de movimientos. Estos datos se analizaron de forma comparativa y cuantitativa para determinar el nivel de precisión alcanzado y realizar mejoras si fue necesario. Se utilizaron herramientas básicas de análisis estadístico y gráficas para representar los resultados obtenidos.

E. Consideraciones éticas

Todo el desarrollo se realizó respetando principios de accesibilidad tecnológica y ética académica. Se utilizaron diseños de código abierto, citando debidamente sus fuentes. Además, se priorizó el uso de materiales y recursos disponibles dentro de la universidad para reducir el impacto económico del proyecto. El diseño final fue documentado y compartido bajo licencia abierta para que otros estudiantes pudieran replicarlo y mejorarlo.

VII. RESULTADOS Y DISCUSIÓN

A. Procesos de construcción

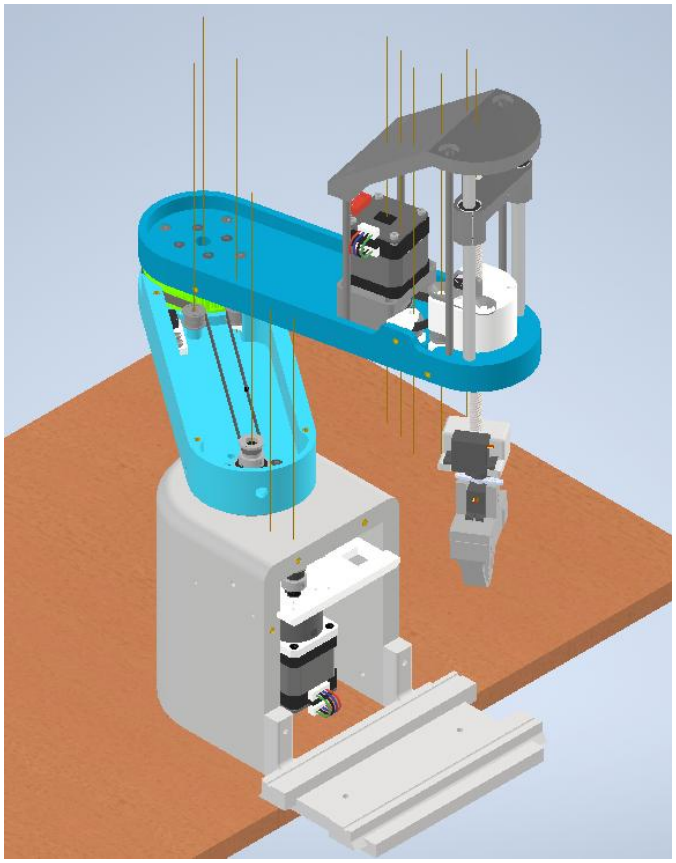


Figura 1. Brazo SCARA modelado en Inventor

Durante la etapa preliminar del diseño, se evaluó el tamaño y las dimensiones de los eslabones del brazo. Se identificó que la longitud inicial no era suficiente para cubrir el ancho de la banda transportadora destinada al prototipo. Por lo tanto, editando los diseños disponibles se elaboró un prototipo mediante impresión 3D. A partir de este análisis se concluyó que, al extender cada brazo 10 cm adicionales, el sistema podía abarcar una gran área de la superficie de la banda. Además, se tomó en cuenta el hecho de mantener un espacio adecuado para depositar los productos en la mesa de trabajo.

No obstante, con dichas dimensiones existía el riesgo de pandeo estructural, además de la imposibilidad de imprimir los componentes en las impresoras 3D disponibles debido a su longitud. Por ello, se optó por fabricar un modelo reducido de 70 mm, equivalente a 7 cm de altura, compatible con una impresora Creality Ender 3, que fue la que se utilizó para toda la impresión de piezas en 3D. Con este ajuste, cada brazo alcanzó una longitud de 270.5 mm, generando un diámetro operativo de 541 mm (54.1 cm). Esta área de trabajo resultó suficiente

para cubrir gran parte de la banda transportadora, aunque no en su totalidad, lo que motivó la búsqueda de soluciones complementarias.

Tras la revisión de configuraciones industriales de brazos SCARA, se observó que muchas líneas de producción incorporan guías laterales o desviadores que redirigen los productos hacia áreas específicas de manipulación. En consecuencia, se propuso la implementación de desviadores impresos en 3D que cumplieran con dicha función, facilitando la operación del brazo SCARA en tareas repetitivas y optimizando su desempeño en el sistema planteado.

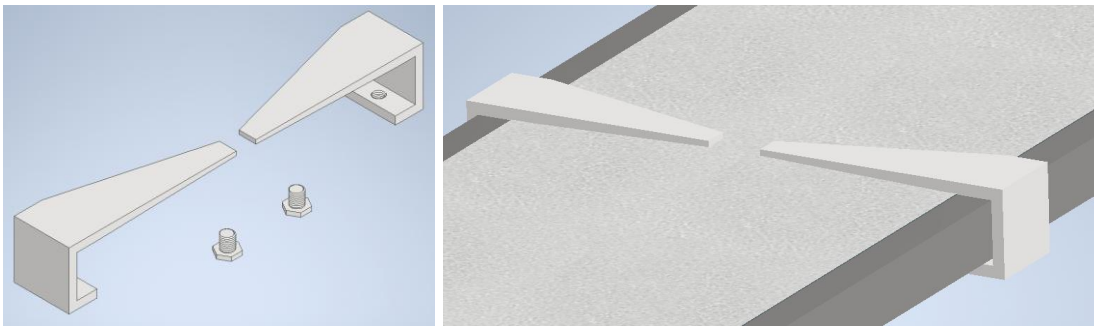


Figura 2. Diseño para desviador de productos

La Figura 2 se propuso un desviador pasivo para guiar los productos hacia una zona fija de recolección. Este desviador se compone como un par de guías inclinadas atornilladas al marco de la banda. Su función es "empujar" suavemente el producto desde el centro o desde un lateral hacia la línea de *pick*. Este enfoque es compatible con el modo de operación ya programado: el brazo mantiene su *HOME*, espera el arribo del objeto a la posición de toma y ejecuta la secuencia de *pick-and-place* de forma repetitiva.

En resumen, el desviador permite adaptar la banda al brazo (y no al revés), concentrando el trabajo del SCARA en su zona más estable, precisa y dejando abierta la posibilidad de automatización gradual si el proceso lo requiere (Innova Mecatronica, 2025).

Figura 3. Primeras piezas impresas en 3D

La impresión inicial de piezas en 3D permitió obtener un modelo físico manipulable del brazo, lo cual facilitó el inicio de su ensamblaje. Durante esta fase, se presentaron dificultades al acoplar los componentes mecánicos, específicamente rodamientos y tornillos, ya que no encajaban adecuadamente en sus respectivos alojamientos debido a discrepancias dimensionales.

Figura 4. Tolerancias de piezas

En un primer momento, se atribuyeron estas diferencias a las tolerancias propias de la impresión 3D, por lo que se realizaron pruebas de lijado en las piezas afectadas. Sin embargo,

este procedimiento resultó poco eficiente, ya que requería un trabajo exhaustivo en todas las superficies de contacto y algunas zonas eran difíciles de tratar con papel de lija convencional. Como alternativa, se consideró el uso de una herramienta tipo Dremel, aunque esta opción exigía precaución, dado que a altas velocidades podía generar calor excesivo y provocar quemaduras en las piezas, comprometiendo su integridad estructural.

Mientras se realizaba el ensamblaje, también se identificaron inconvenientes relacionados con los agujeros destinados a tornillos e insertos. En el diseño original elaborado en Inventor, varios de los agujeros fueron pensados para colocar insertos roscados, fijándolos mediante calor con un cautín. No obstante, surgieron problemas durante este proceso: los insertos solicitados eran más grandes de lo necesario y no encajaban adecuadamente, incluso al derretir el plástico. Además, la soldadura de insertos requirió una alta precisión para que quedaran perfectamente alineados; si se torcían durante la instalación, el agujero quedaba inutilizable y comprometía el diseño completo. Como solución, se decidió cambiar los tamaños y largos de algunos tornillos, además de emplear tuercas externas, de modo que pudieran fijarse firmemente al brazo sin depender de los insertos.

Ante estas limitaciones, antes de proceder con un rediseño general de tolerancias se optó por modificar el diámetro de la boquilla de la impresora mediante el software Cura. Este ajuste permitió mejorar significativamente el encaje de las piezas con los elementos mecánicos.

Como resultado, se determinó que las piezas con incompatibilidad dimensional debían ser reimpresas en función de su uso. En algunos casos, el problema podía resolverse únicamente mediante lijado controlado, mientras que en otros se requería la reducción del tamaño de la boquilla, pasando de 0.28 mm a 0.12 mm. Esta configuración permitió obtener piezas con mayor precisión dimensional, garantizando una correcta alineación y compatibilidad con los componentes mecánicos del brazo SCARA.

Figura 5. Primer brazo pandeado

Durante la primera impresión del brazo se presentó un problema de deformación por pandeo, similar al observado previamente en otras piezas. Este inconveniente se originó principalmente por la descalibración de la cama de la impresora en las esquinas.

Adicionalmente, en esta prueba inicial se utilizó un *infill* del 100%, lo que generó un brazo con excesivo peso, aunque con alta resistencia mecánica. Sin embargo, esta condición habría ocasionado un pandeo aún más pronunciado al momento de ensamblar el segundo eslabón.

Para solventar esta situación, en la siguiente impresión se redujo el porcentaje de *infill* al 10%, manteniendo una configuración recta de capas superpuestas con el fin de conservar la resistencia en sentido perpendicular. Esta modificación permitió disminuir considerablemente el peso de la pieza sin comprometer su rigidez estructural.

Asimismo, durante esta segunda impresión se efectuó una calibración precisa de la cama de la impresora y se aplicó un aerosol adhesivo en la superficie de impresión. Este procedimiento favoreció la adherencia de las primeras capas al entrar en contacto con la cama, reduciendo el riesgo de desprendimiento incluso en presencia de pequeñas discrepancias de nivelación.

Figura 6. Primer brazo ensamblado del brazo SCARA

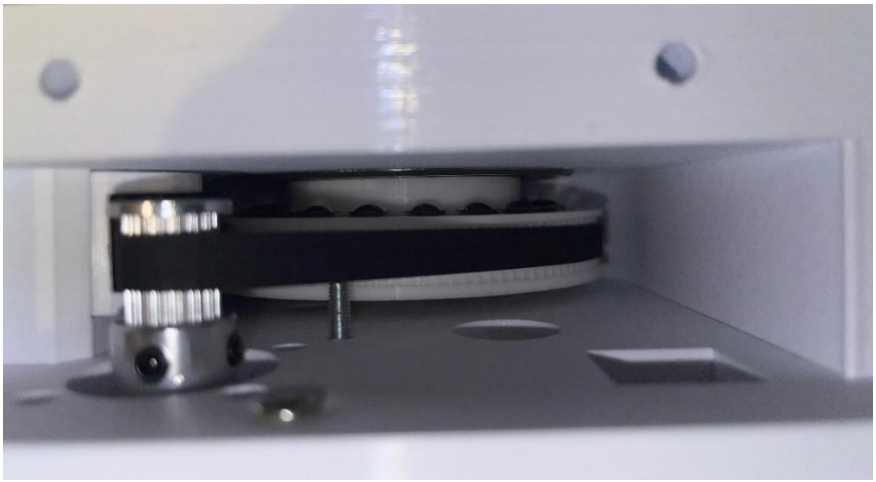


Figura 7. Primeras pruebas de engranaje y correas del brazo SCARA

Durante el proceso de ensamblaje del diseño físico se trabajó con los dos motores paso a paso encargados de accionar el primer y segundo eslabón del brazo. Antes de continuar con la integración completa, fue necesario realizar pruebas de posicionamiento de la faja que transmitía el movimiento al primer brazo, ya que no se tenía certeza de que la ubicación inicial del motor paso a paso garantizara un desplazamiento fluido y sin inconvenientes. El diseño había sido planteado tanto para motores Nema 17 con reductores de revoluciones, a fin de proporcionar mayor torque, como para motores paso a paso convencionales, lo que permitía cierta flexibilidad en la implementación.

Previo al ensamble completo del brazo y antes de lubricar los rodamientos, fue necesario aplicar calor a la base donde se encontraba ensamblado el primer eslabón. Este procedimiento se ejecutó para la introducción de los rodamientos fijos en la base y se aplicó tanto en el rodamiento inferior, correspondiente al engranaje, como en el superior, encargado de garantizar la exactitud del movimiento. La base había sido impresa con una boquilla de 0.26 mm para reducir el tiempo de fabricación, dado que se trataba de una de las piezas más grandes del prototipo. No obstante, ello provocó discrepancias en las tolerancias de los orificios respecto a las dimensiones esperadas. Para resolverlo, se aplicó calor controlado con aire caliente, de manera que el plástico se suavizara lo suficiente para permitir la inserción de los rodamientos, y posteriormente se enfrió con aire a temperatura ambiente utilizando una secadora de cabello para todo este procedimiento. Al finalizar, los rodamientos quedaron correctamente posicionados sin comprometer la integridad de la base.

En esta etapa, el primer brazo quedó ensamblado en su base y se acoplaron las fajas encargadas de su movimiento. Durante las pruebas de resistencia se detectaron fuerzas de fricción inusuales, ocasionadas por la manipulación repetida de los rodamientos, la cual provocó la pérdida parcial de lubricante. Para resolverlo, se procedió a desensamblar los rodamientos, limpiarlos de cualquier suciedad o polvo y aplicarles lubricación con aceite 3 en 1. De manera paralela, mientras se continuaba con la impresión y ensamblaje de las demás

piezas, se solicitó la elaboración de un manual de mantenimiento que permitiera garantizar el cuidado preventivo y correctivo del brazo SCARA a largo plazo, el cual sería entregado de manera externa y aparte a los objetivos de este megaproyecto.

Tras la lubricación de los rodamientos, las resistencias disminuyeron notablemente. Las leves fricciones restantes se atribuyeron a la precisión dimensional de los engranajes impresos en 3D, aunque no representaron un inconveniente significativo, ya que se presentaban únicamente en un ángulo muy específico del movimiento del brazo. Además, se estableció que esta posición no sería utilizada durante las tareas repetitivas previstas sobre la banda transportadora, por lo que no afectó el desempeño funcional del sistema.

Figura 8. Primeros motores ensamblados del brazo SCARA

Figura 9. Pruebas de eje para segundo brazo

Una vez verificados los rodamientos y el movimiento correcto del primer brazo, se identificó un posible inconveniente durante la instalación del segundo motor encargado de accionar el segundo eslabón. Como se aprecia en la Figura 8, ambos motores quedaban colocados en línea, uno detrás del otro. Esta disposición generaba el riesgo de que los cables del segundo motor, ubicado en la parte posterior, presentaran dificultades al momento de conectarse con su respectivo *driver*, ya fuera por falta de espacio o por la posibilidad de que los cables se doblaran excesivamente.

Para solucionar este problema se diseñó una pieza adicional (Figura 9), ubicada sobre el segundo motor, que permitió colocar este componente a un nivel ligeramente inferior al del primero. De esta manera, se aseguró el espacio necesario para realizar la conexión de los cables sin interferencias, garantizando así un ensamblaje más práctico y funcional.

En paralelo, se continuó con el montaje del sistema de transmisión del segundo brazo, el cual incluyó un acople rígido, una varilla roscada de 8 mm y una polea destinada a movilizar la faja correspondiente. Durante esta etapa se presentó la dificultad de encontrar varillas lisas o tubos de 8 mm en ferreterías locales, por lo que se optó inicialmente por fabricar varillas impresas en 3D. Aunque estas funcionaron en pruebas preliminares, se consideró que, bajo la aplicación de torque elevado, podían romperse o desgastarse con el paso del tiempo. Por tal motivo, se planteó como alternativa el uso de una varilla roscada de 8 mm, capaz de ofrecer la resistencia necesaria para soportar los esfuerzos mecánicos del segundo brazo durante su operación.

Figura 10. Fabricación de una faja de 570 mm

Durante la etapa de diseño en Inventor se realizaron simulaciones del sistema de transmisión por fajas con el propósito de visualizar la posición de estas y estimar la longitud necesaria de la faja para el mecanismo del primer eslabón, que fue el único que requirió una faja de largo no comercial. Sin embargo, este simulador únicamente permitió conocer la disposición

geométrica y la trayectoria de la faja, sin proporcionar un valor exacto de su longitud. Por esta razón, se procedió a calcular manualmente la medida requerida, segmento por segmento.

El cálculo teórico realizado en digital difería ligeramente del resultado físico, debido principalmente al estiramiento y a la disposición diagonal que adquiría la faja una vez instalada, en contraste con la simulación que consideraba un trazo completamente recto. Finalmente, se determinó que la longitud óptima para el brazo era de 565 mm - 570 mm. No obstante, este valor no correspondía a una medida comercial disponible, incluso considerando la tolerancia de más de 5 mm que ofrecía el sistema de tensado implementado en el prototipo.

Ante esta limitación, se optó por fabricar manualmente una faja de la longitud requerida a partir de una faja comercial más larga. Para ello, se peló un extremo eliminando la zona dentada y, en el otro extremo, la parte lisa, lo que permitió unir ambas secciones mediante adhesivo instantáneo en un área de aproximadamente 10 mm. El resultado fue una faja completamente funcional, aunque con una apariencia menos estética respecto a una pieza industrial.

Se recomendó que, en futuras mejoras del diseño, se consideren dos alternativas: modificar el largo del brazo para ajustarse a una medida de faja comercial o mandar a fabricar una faja de entre 565 mm - 570 mm a nivel industrial, ya sea en el extranjero o, de ser posible, en talleres especializados en Guatemala.

Figura 11. Tercer bazo con acople para la garra

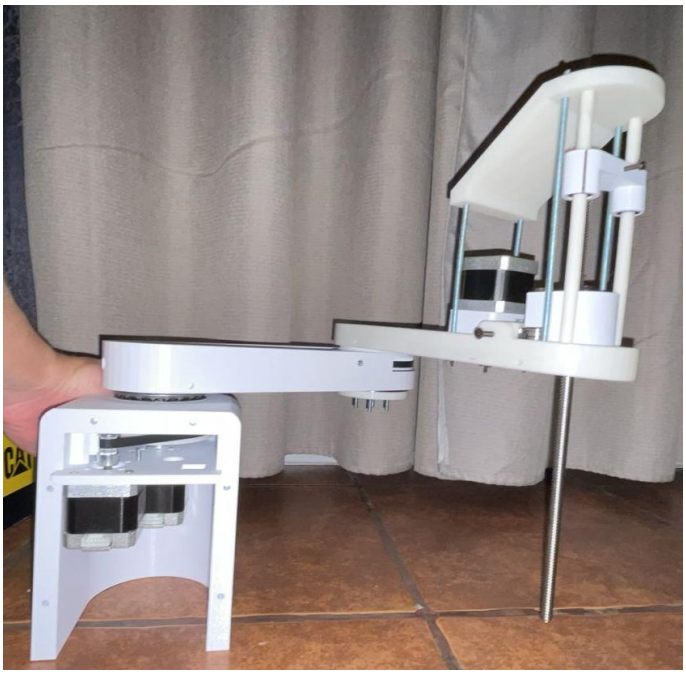


Figura 12. Brazo completo

En el segundo brazo también se requirieron tubos de 8 mm, aunque en este caso únicamente fueron necesarios para funcionar como guías del movimiento del tornillo sin fin encargado de desplazar la garra en sentido vertical. Para este propósito se optó por fabricar los tubos mediante impresión 3D, dado que las exigencias mecánicas eran menores en comparación con las del primer brazo.

Una vez instaladas las guías, se evaluó la longitud del tornillo sin fin más adecuada para cumplir con la función de descenso de la garra a la mesa y a la banda transportadora. Se identificó que el brazo SCARA se encontraba a una altura mayor respecto a la banda transportadora, por lo que era necesario un tornillo sin fin lo suficientemente largo para tomar un objeto, pero no tan extenso como para colisionar con la mesa. Inicialmente se había considerado un tornillo sin fin de 200 mm, pero tras el análisis se determinó que no era suficiente, optándose en su lugar por uno de 300 mm.

No se seleccionó un tornillo de 400 mm ya que la base de uno de los lados de este componente no debía llegar justo hasta la mesa, debido a que en su extremo inferior se acoplaría el mecanismo de la garra. Esta disposición requirió dejar un espacio adicional para el montaje del efector final y para garantizar la manipulación de productos sin interferencias durante la operación.

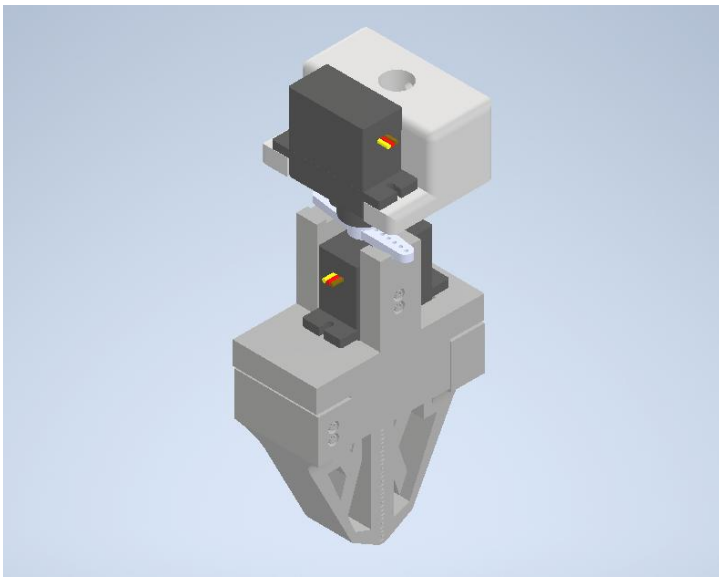


Figura 13. Diseño de la garra ensamblado

Para la garra se partió de un diseño público del mecanismo de pinza (GG_CADworks, 2025) y se adaptó al prototipo SCARA y a su eje vertical con el tornillo sin fin. Las principales modificaciones fueron:

- Rediseño del soporte principal para fijarse al tornillo sin fin.
- Cambio de servos previstos (de SG90 a MG90S) y ajuste de sus dimensiones.
- Alineación de agujeros de sujeción para asegurar el ensamblaje con el resto de las piezas.

El fichero original se convirtió de .stl a .ipt con Fusion para editarlo en Inventor. Sobre esa base se alargaron los perfiles que sostienen el segundo servo (para que cupiera el MG90S), se reubicaron los orificios de anclaje de todos los servos y se incluyó un mecanismo tipo tornillo prisionero para apretar con firmeza el eje del tornillo sin fin. Además, se dejó un agujero adicional para poder variar la altura de la garra en un futuro si fuera necesario.

En un principio se tuvieron complicaciones con el engranaje que se encargaba del movimiento de apertura y cierre de la garra; esto se resolvió atornillando el engranaje al servo de manera fija por completo. Por otro lado, en ocasiones el dentado de las pinzas no hacía contacto con el movimiento del engranaje y esto era debido a un error de orientación al momento de la impresión de estas piezas. Por lo tanto, se reimprimieron por separado las piezas de las pinzas de la garra con los dientes que hacían contacto con el engranaje orientados hacia arriba.

Con estas correcciones, la garra ofreció fuerza de agarre suficiente para las piezas previstas. Aún así, se recomendó no sobreapretar los dientes de la garra (ya que fueron impresos en plástico) y ajustar el ángulo del segundo servo según el tamaño del objeto.

Cuadro 1. Componentes electrónicos

Componente	Fabricante	Modelo	Descripción	Cantidad
Motor Stepper Nema17	Casun®	17HD40005H-22B	Motor bipolar de 2 fases con ángulo de paso de 1.8°, voltaje nominal de 2.2 V, corriente de 1.66 A y torque de 3.22 N·m. Dimensiones 42×42×40 mm y eje de 5 mm. Diámetro de eje de 5mm. Encargados del movimiento de los eslabones del brazo SCARA.	3
Digital Stepper *Driver* DMT542T	STEPPERONLINE	DMT542T (V4.0)	*Driver* digital para motores paso a paso con algoritmo DSP avanzado que garantiza movimiento suave, bajo ruido y mínima vibración. Utilizados para la autoidentificación de los motores y configuración automática de parámetros. Compatible con motores NEMA 17 y NEMA 23.	3
Arduino® Portenta *machine control*	Arduino	AKX00032	Controlador industrial basado en el microcontrolador Portenta H7, con procesador dual-core STM32H747 (Cortex-M7 a 480 MHz y Cortex-M4 a 240 MHz). Permite múltiples opciones de conectividad (USB, Ethernet, Wi-Fi/Bluetooth LE, RS485). Compatible	1

			con sensores y actuadores mediante entradas y salidas digitales, analógicas aisladas, canales de temperatura configurables y conector I2C. Diseñado para operar en rangos industriales de -40 °C a +85 °C sin necesidad de enfriamiento externo. Utilizado para el control de los motores Stepper.	
Arduino Uno	Arduino	R3	Placa basada en el microcontrolador ATmega328P. Utilizada para el control de la garra utilizando servos.	1
Servo MG90S	Servo	MG90S	Servo de 5 V con recorrido de 180°, velocidad de 0.1 s/60° y torque de 3.0 kgf·cm. Con engranajes metálicos, alta precisión (banda muerta de 5 µs) utilizado para el movimiento de la garra.	2
Módulo MB102 alimentación para *protoboard*	Gtronica	MB102	Fuente de poder para *protoboard*, con entrada de 6.5–12 V DC y salidas seleccionables de 3.3 V o 5 V DC. Utilizada para la alimentación de los Servos MG90S.	1
Fuente AC-DC		Fuente AC-DC de 110V – 220V y 12A – 10A	Fuente con entrada de 110–220 V AC ±15% y salida de 12 V DC / 10 A, con potencia de	

			120 W. Compacta (160 × 100 × 40 mm), utilizada para la alimentación de los drivers y el Arduino Portenta *machine control*.	
PushButton		PCB NA de 4 pines	Interruptor momentáneo NA de 4 pines y 4 mm de longitud, con voltaje nominal de 30 V y corriente de 50 mA. Activador de la secuencia de movimiento de la garra.	1
Adaptador AC-DC		N.A. 5V 2A	Fuente de alimentación con salida de 5 V DC y corriente máxima de 2 A, equipada con conector DC de 2.5 mm. Utilizado para alimentar el modulo MB102 que alimentan los servos MG90S.	1
Bornera		3 pines 250V a 8A	Conector para PCB con 250 V y 8 A nominales; admite conductores 22–14 AWG. Compacta y fiable para asegurar conexiones de alimentación y señal.	2
Cables		24AWG	Alambre estañado para conexiones. Con 6 diferentes colores (amarillo, azul, blanco, negro, rojo y verde).	1

Cuadro 2. Componentes mecánicos

Componente	Fabricante	Modelo	Descripción	Cantidad
Poleas	Gtronica	Polea GT2	Polea especial para aplicaciones de movimiento lineal. Con estas poleas se obtiene una buena precisión en el movimiento, sin el inconveniente del juego por *Backslash*. Estas poleas están diseñadas para trabajar con motores paso a paso Nema17 con un diámetro de eje de 5 mm.	3
Riel Din		Riel Din 35MM X 1 MTS	Riel Din, de 35 mm x 1 m. Este se recortó ya que solo se utiliza para el soporte de los *Drivers* DM542T	1
Perfiles roscados		Varilla Roscada de 3/16” y 5/16”	Dos medidas distintas de varillas roscada, 8 mm y 5 mm de ancho, ambas utilizadas como líneas guías.	5
Rodamiento lineal	Gtronica	LM8UU	Rodamiento lineal utilizado como guía para los ejes encargados de sostener el movimiento del tornillo sin fin de forma perpendicular al brazo 2.	2
Rodamiento para correa dentada	TGHCHHCE	F624zz	Rodamiento pequeño divido en 2 partes para acoplar a una correa dentada, en	8

			este caso de gt2 de 6 mm de ancho.	
Rodamiento 51109	TransLink	51109	Rodamiento dividido en 3 partes utilizado para acoplar el brazo uno con la base y el brazo 2 con el brazo 1.	4
Rodamiento 608zz		608zz	Rodamiento habitual utilizado para el movimiento lineal del tornillo sin fin y para el motor que moviliza el brazo 2.	2
Tornillo sin fin		TORT840	Tornillo sin fin usualmente utilizado para impresora 3D 8 mm x 400 mm con paso de 2 mm, utilizado para que suba y baje el efector final.	1
Tuerca trapezoidal		T8 *antibacklash*	Tuerca en forma trapezoidal con resorte para evitar *backlash* utilizada para evitar el retroceso del tornillo sin fin.	1
Correa dentada	Xianglaa	gt2 de 6 mm de anchwdwddwwDWWo	Correa dentada antideslizante con bucle cerrado, utilizada para el movimiento de los codos de cada uno de los eslabones.	3
Tornillos métricos		• Allen cilíndricos • Allen cabeza pache tipo Roldana • Cabeza plana	Tornillos de fijación de distintas medidas y modelos, avellanados,	30

			planos, estrella, etc.	
Roldanas		Roldana natural	Roldanas de distintas medidas para protección de la pieza a atornillar y distribuir la presión del tornillo que la acompaña.	7
Tuercas		• Tuerca 2H • Tuerca artillera • Tuerca de seguridad	Tuercas de distintas medias para el ajuste como sustituto de los insertos en los agujeros del diseño.	12

Como se observó en los Cuadros 1 y 2, se mostraron los componentes mecánicos y electrónicos que se utilizaron en el proyecto, tanto para el control de los motores Stepper como para los servos de la garra. Esta información permitió tener una documentación clara y ordenada de todo lo que formó parte del brazo SCARA, tanto en su parte física como en el sistema de control.

Los componentes mecánicos se escogieron buscando precisión y estabilidad en los movimientos, mientras que los componentes electrónicos permitieron un control confiable de los motores y servos. En conjunto, estos elementos hicieron posible que el brazo robótico trabajara de forma eficiente y que pudiera considerarse una base sólida para futuras mejoras e integraciones.

B. Procesos de toma de resultados

1. Análisis del espacio de trabajo

Tomando en cuenta el largo de los brazos que conforman al SCARA, su espacio de trabajo se empleó la *Ecuación (2):*

$$R_{max} = 190.542\ mm + 200\ mm\ , \qquad R_{min} = |190.542\ mm - 200\ mm|$$

$$\boldsymbol{R_{max} = 390.542\ mm,} \qquad \boldsymbol{R_{min} = 9.458\ mm}$$

En el modelo ideal de un SCARA, el espacio de trabajo queda descrito movimiento circular en forma de anillo con radios $R_{\max} = L_1 + L_2$ y $R_{\min} = | L_1 - L_2 |$. Con las longitudes medidas de los eslabones del prototipo ($L_1 = 190.542$ mm y $L_2 = 200.000$ mm) se obtuvo $R_{\max} = 390.542$ mm y $R_{\min} = 9.458$ mm. Estos valores un alcance real de manera teórica. Se considera que el movimiento no es por completo en forma circular, sino en un ángulo de 150° aproximadamente desde el punto *HOME* lo que es suficiente para la tarea de *pick-and-place*.

No obstante, el espacio de trabajo se ve reducido por condicionantes reales:

- La herramienta (garra) y un margen de seguridad para evitar colisiones con la base y con elementos del entorno (controladores y componentes eléctricos).
- Los límites articulares impuestos por topes mecánicos y software.
- La presencia de la banda transportador.

Como se mencionó anteriormente el brazo opera en un área de trabajo de aproximadamente 150° con respecto del punto *HOME*, esto simplifica la cantidad de trayectorias a realizar que reduce tiempos de ciclo y mejora su repetitividad. De igual manera el brazo presenta la capacidad de un alcance radial amplio, únicamente que por el hecho de que momentáneamente no se tiene una forma de fijar al brazo a la mesa, en este proceso de estirar por completo los eslabones a un ángulo amplio el centro de masa cambia y el brazo tiene una fuerza mayor en el efector final (la garra).

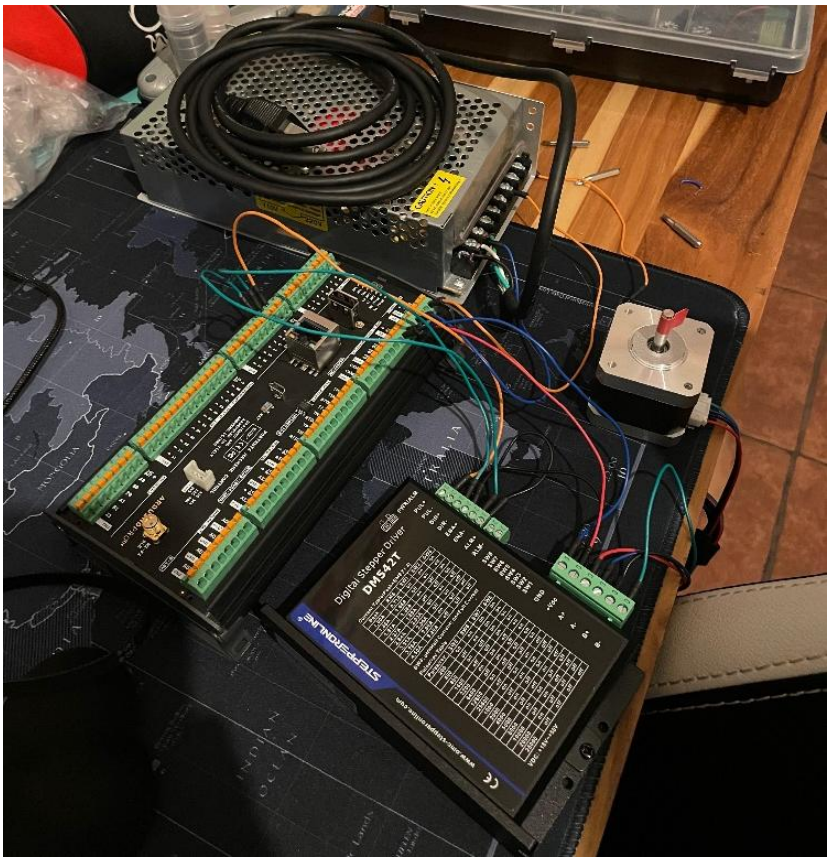
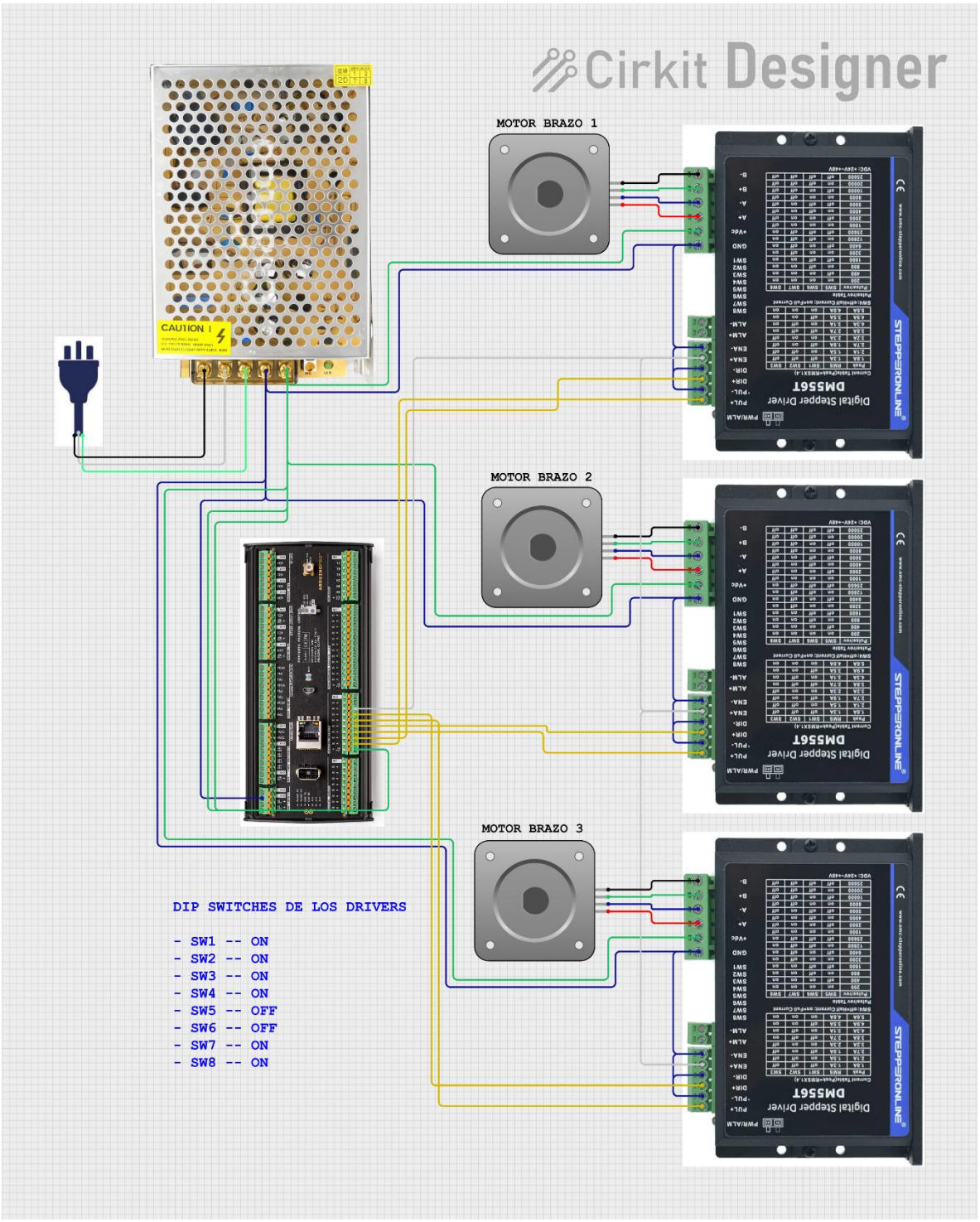


Figura 14. Pruebas físicas de programación

Figura 15. Esquema eléctrico de motores paso



Fuente: Vasquez, 2025.

Durante las pruebas iniciales de programación se utilizó únicamente un motor y se identificó que el motor mostraba un sobrecalentamiento al realizar movimientos repetitivos por un breve periodo de tiempo. El hecho que el motor mostrara sobrecalentamiento no era gran problema ya que en su hoja de datos muestra que su temperatura máxima a la que puede permanecer en un estado de trabajo es de 80° centígrados. El comportamiento anómalo a esta situación era que el motor se calentaba rápido sin realizar una gran cantidad de trabajo y a la vez las conexiones, lo cual no debía ser así. Por lo que se reemplazo de *jumpers* delgados por cables de 24 AWG lo que se redujo la elevación térmica en conectores a nula. Este ajuste, evidenció la sensibilidad del sistema a la calidad de las, pero a pesar de que se solucionó el inconveniente de las conexiones, el motor continuaba mostrando un sobrecalentamiento acelerado.

Se profundizó en conocer los usos de los DIP *switches* del DM542T, comprendiendo su función de tres maneras:

- Corriente nominal de fase (cantidad de corriente permitida a pasar hacia el motor).
- Reducción de corriente en reposo (para limitar la temperatura cuando el motor mantiene posición y se mantiene energizado).
- Micropaso (suaviza el movimiento y mitiga resonancias mecánicas).

La reducción de corriente en reposo con el motor energizado resultó de manera efectiva la disminución de la temperatura sin comprometer las condiciones de fuerza del motor; por otro lado, la mayor razón al comportamiento del sobrecalentamiento acelerado era por mantener una corriente nominal excesiva por lo que se cambió la configuración de DIP *switches* para que el motor recibiera una corriente acomodada al trabajo que fueran a realizar.

Como resultado, se adoptó la configuración SW1, SW2, SW3, SW4, SW7 y SW8 en *ON*, con SW5 y SW6 en *OFF* (según la tabla del modelo utilizado DM542T), lo que equilibró el par disponible, suavidad de movimiento y disipación térmica. Con esta combinación y el redimensionamiento del cableado, el sistema dejó de presentar un sobrecalentamiento anómalo en el motor y en las conexiones, se redujo el riesgo de pérdida de pasos por caídas de tensión por calentamiento y se obtuvo una operación más estable y segura para el diseño.

El ajustar que los motores no presentaran un calentamiento anómalo no era solo pensando en la vida útil de estos, sino que a la vez por el hecho de que cada uno de los motores se encuentra en contacto con una pieza impresa en 3D. Si el motor hubiera permanecido a su temperatura máxima de 80° centígrados las piezas impresas en 3D en contacto con el motor hubiesen presentado mal formaciones por su uso prolongado.

2. Estimación del tiempo de ciclo

Se obtuvieron las incógnitas mostradas a continuación para la resolución de la estimación del tiempo de ciclo empleando la *Ecuación (3):*

Pasos por revolución (según la programación) : $1600 = 1/8\ Microstep$

Semiperíodo de pulso: $\text{M1 y M2} = 500\ \mu s\ , \text{M3} = 200\ \mu s$

Se emplearon las *Ecuaciones (4) y (5)* para los análisis de frecuencia de paso y velocidad angular necesarias para la obtención de las siguientes incógnitas:

Velocidades resultantes

$$\text{M1 y M2: } f = 1000\ \frac{pasos}{s} \Rightarrow \omega = 225\ \text{deg/s}$$

$$\text{M3: } f = 2500\ \frac{pasos}{s} \Rightarrow \omega = 562.5\ \text{deg/s}$$

Barridos máximos

$$M1\text{: } \Delta\theta_1 = 600°$$

$$M2\text{: } \Delta\theta_2 = 850°$$

$$M3\text{: } \Delta\theta_3 = 1700 - (-6800) = 8500°$$

Tiempos de movimiento puro $(t = \Delta\theta/\omega)$

$$t_1 = \frac{600}{225} = 2.67\ s$$

$$t2 = \frac{850}{225} = 3.78\ s$$

$$t3 = \frac{8500}{562.5} = 15.11\ s$$

Pausas programadas

$$PRE_{M3_{DELAY_{MS}}} = 3\ s,$$

$$M3_{HOLD_{MS}} = 6\ s,$$

$$REP\_PAUSE\_MS = 3\ s \implies \mathbf{12\ s}\ en\ total.$$

Tiempo de ciclo (dos escenarios operativos)

- *Movimientos secuenciales (conservador)*

$$T_{ciclo} = t1 + t2 + t3 + 12 = 2.67 + 3.78 + 15.11 + 12 = \mathbf{33.56}\ \boldsymbol{s}$$

$$\boldsymbol{Ciclos/min} \approx \frac{\mathbf{60}}{\mathbf{33.56}} = \mathbf{1.79}\ \boldsymbol{cpm}.$$

- *M1 y M2 en paralelo; luego M3 (habitual en SCARA):*

$$T_{ciclo}\ par = max(t1, t2) + t3 + 12 = 3.78 + 15.11 + 12 = \mathbf{30.89}\ \boldsymbol{s}$$

$$\boldsymbol{Ciclos/min} \approx \frac{\mathbf{60}}{\mathbf{30.89}} = \mathbf{1.94}\ \boldsymbol{cpm}.$$

En la estimación del tiempo de ciclo se empleó la ecuación $t = \Delta\theta/\omega$ con los parámetros de operación del prototipo (micro-paso 1600 steps/rev, semiperíodos de pulso de 500 μs para Motor1–Motor2 y 200 μs para Motor3). A partir de esta información se obtuvieron las velocidades angulares de 225 deg/s(M1–M2) y 562.5 deg/s(M3), y tiempos de movimiento para los barridos máximos de los eslabones: $t_1 = 2.67$ s(600°), $t_2 = 3.78$ s(850°) y $t_3 = 15.11$ s(−6800° a 1700°, 8500° en total). Considerando las pausas programadas como *delays* (3 s + 6 s + 3 s = 12 s), el tiempo de ciclo resulta **33.56s** si los ejes se mueven en forma de secuencia y **30.89s** cuando M1–M2 operan en paralelo y luego el M3, esto equivale a aproximadamente 1.79 – 1.94 ciclos/min. Además, M3 cuenta con un 45 % aproximadamente del ciclo debido a las pausas con un aproximado de 36 %, lo que evidencia que el eje vertical y sus *delays* dominan en el rendimiento temporal debido a su función de espera para evitar vibraciones y para la función del efector final de *pick-and-place*.

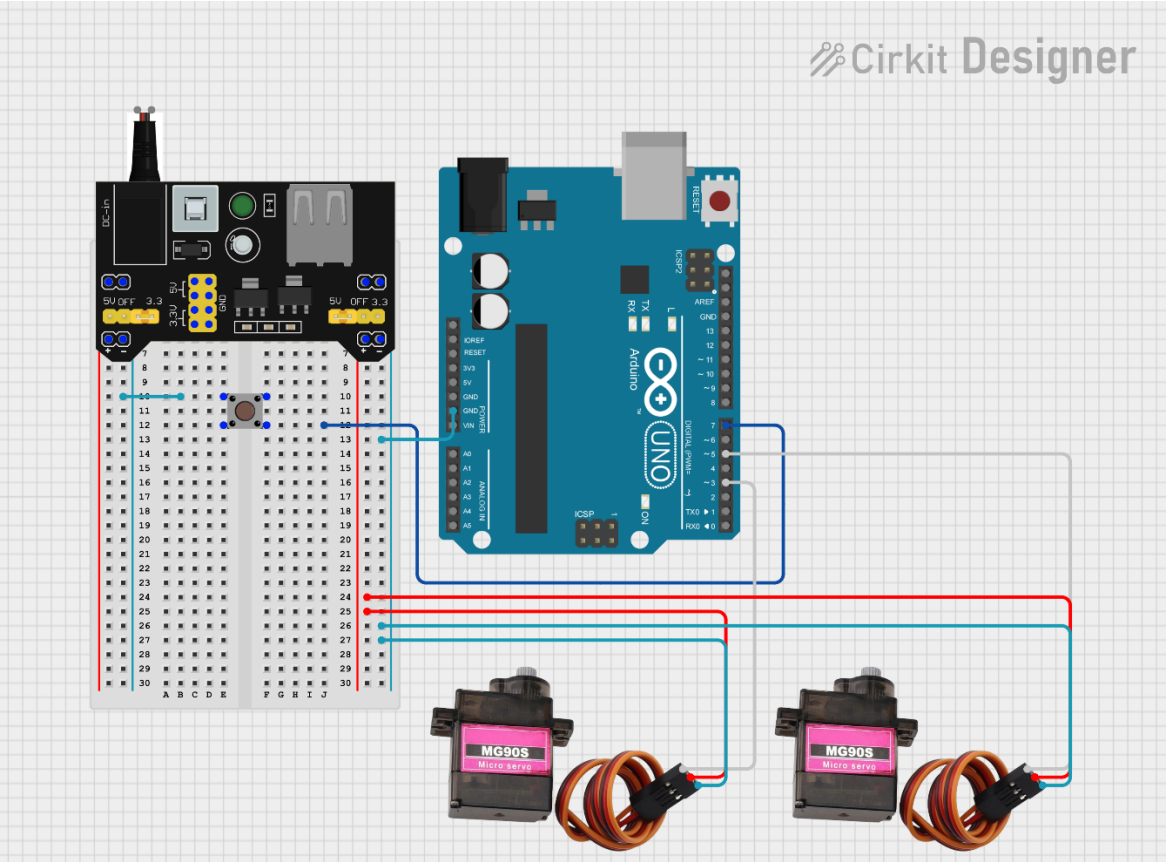
Es importante tomar en cuenta que estos cálculos demuestran una velocidad moderada elegida para el prototipo: si bien los motores paso a paso permiten aumentar su velocidad, se prefirió un movimiento leve para mostrar con claridad el funcionamiento y a la vez la precisión en la operación. Esta decisión es debido a condicionantes del montaje actual:

- La base aún no está completamente anclada a la mesa de trabajo, por lo que velocidades mayores podrían inducir vibraciones y efectos inerciales indeseados que concluyeran en que el brazo se caiga.
- Existen conexiones expuestas que, a altas velocidades, podrían enredarse con elementos mecánicos. Operar a velocidad moderada reduce estos riesgos y mejora la repetitividad mientras se implementa una forma de optimizar las líneas de conexión.

Adicionalmente, los *delays* incorporados en el programa se definieron debido a que la garra opera de forma manual. Al llegar M3 al punto de recolección, el operador acciona el botón para abrir/cerrar la pinza por medio de un ArduinoUno como controlador, su función es de mantener el agarre durante la elevación y el retorno, y finalmente se libera la pieza sobre la

mesa. Estos *delays* garantizan sincronía y seguridad con el uso manual, pero aumentan el tiempo de ciclo.

Figura 16. Esquema eléctrico de la garra



Fuente: Vasquez, 2025.

La elección de los servos MG90S (de engranajes metálicos) frente a SG90 (de engranajes plásticos) es debido al par y durabilidad ante esfuerzos repetitivos de cierre de la pinza, reduciendo el riesgo de desgaste o "pelado" de dientes en los engranajes. Para esta garra los servos utilizados son una versión posicional del MG90S de giro completo (aproximadamente 360°), para fijar ángulos de apertura y cierre. Al energizar el sistema, el servo toma la posición *HOME* porque el programa envía inmediatamente el pulso de referencia (*attach* + *setpoint* inicial), evitando procedimientos de calibración adicionales.

Se tuvo un ligero desfase al montar el segundo servo debido que era imposible de quedar exactamente perpendicular al segundo eslabón del brazo por el paso de los dientes de los

engranajes del servo, esto es un comportamiento esperable que se puede ajustar cambando el ángulo de sujeción de la garra al tornillo sin fin del efector final.

Por otro lado, como se observa en la Figura 16 se utilizaron borneras para las conexiones de los servos a la *protoboard*, el uso de estas borneras de 3 entradas cumple un doble propósito:

- Retención mecánica de los conductores para evitar desconexiones por vibraciones o tirones durante el movimiento de los servos.
- Distribución limpia de Vcc y GND hacia cada servo, facilitando mantenimiento y orden del cableado.

La alimentación de los servos se realiza desde el MB102 (5 V), con un adaptador de 12V /2A a la entrada. Esta topología evita sobrecargar el regulador de 5 V del ArduinoUno y reduce reinicios por caídas de tensión.

La lógica de control como se observa en el Anexo A, la garra opera de manera manual con un botón, al detectar pulsación (*INPUT_PULLUP*), el primer servo orienta la garra hacia la banda (p. ej., 90°), el segundo servo cierra las pinzas, se mantiene un tiempo de espera para el traslado y, ya sobre la mesa, el primer servo regresa a 0° y el segundo abre para liberar el objeto recolectado. Esta secuencia simple permitió sincronizar la intervención del operario con el SCARA y sirvió como base para las pruebas de tiempo de ciclo.

Figura 17. Muestra de recopilación de resultados de análisis de precisión y repetitividad

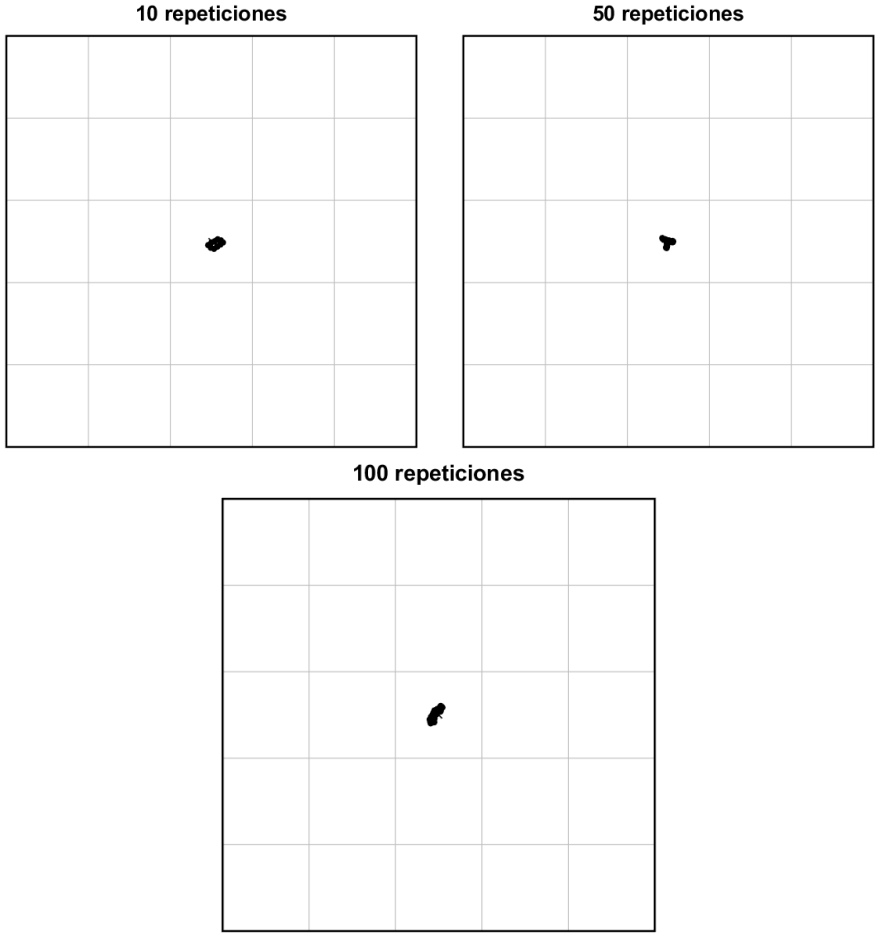


Figura 18. Resultado físico de análisis de precisión y repetitividad

Con el prototipo completamente ensamblado se evaluó la precisión y la repetitividad mediante un ensayo manual. Como se muestra en la Figura 17, se fijó un marcador de punta fina en la garra (rapidógrafo *edding 1880 drawliner 0.2*) y se ejecutaron series de 10, 50 y 100 repeticiones hacia una misma posición. Durante la prueba se utilizaron todos los motores paso del brazo; la garra permaneció cerrada para eliminar variaciones por apertura y cierre.

Los resultados (Figura 18) muestran una muy mínima dispersión entre los distintos puntos, estos tuvieron una dispersión máxima de aproximadamente 2 mm. Este desvío entre algunos puntos es debido a vibraciones de los eslabones y a paradas bruscas del movimiento al llegar las posiciones finales, más que a una pérdida de exactitud de los motores. Aún bajo estas condiciones, la nube de marcas se concentró en un área considerablemente aceptable, lo que indica un comportamiento consistente del sistema en un trabajo de muchas repeticiones.

Se identificó, además, un factor de error asociado al brazo 3 (eje vertical con tornillo sin fin). Tras repetir varios ciclos, el punto *HOME* presentaba una posición distinta (deriva). El análisis apuntó a fricción en las guías impresas en 3D por donde se desplaza el carro. Se realizaron acciones de lijado y lubricación, que mejoraron el deslizamiento, pero no eliminaron totalmente la deriva. Como medida temporal, se implementó una compensación

por software aplicando un ajuste de −45° al finalizar cada ciclo, con esto se estabilizó la posición de *HOME* durante las pruebas.

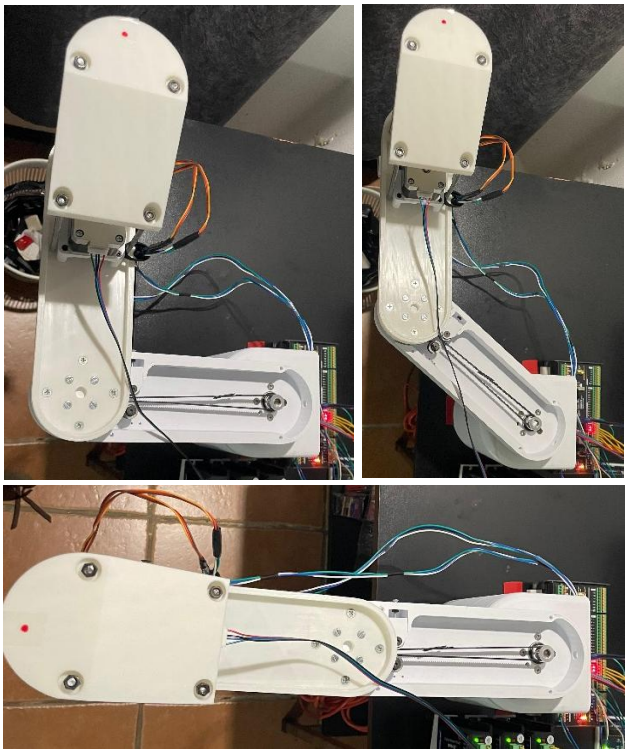


Figura 19. Brazo y pruebas finales de toma de 4 posiciones

El Anexo B muestra el programa final que conforma todo el sistema de control con el Portenta *machine control* y los tres *drivers* DM542T. El código se organizó en tres modos de operación que se pueden acceder por el monitor serial, lo que permitió calibrar, ensayar y demostrar el objetivo final sin recargar firmware:

- Modo 1 (*single*). Ejecuta un ciclo único con los tres motores con los ángulos actuales y editables de A1, A2 y A3. Este modo se usó para calibrar posiciones y probar cambios rápidos de posiciones.
- Modo 2 (repeticiones). Ejecuta "N" repeticiones de un mismo ciclo y permite ajustar, desde consola, velocidades (V12/V3), retardos (D3, RP), tiempo de espera en A3 (H3) y una compensación por repetición en M3 (AC). Con este modo se realizaron las pruebas de precisión y repetitividad.
- Modo 3 (puntos P1..P4). Ejecuta la secuencia de cuatro posiciones (P1→0, P2→0, P3→0, P4→0) con una pausa configurable entre puntos. Fue el modo usado para demostrar el objetivo general: el brazo alcanza cuatro posiciones definidas de forma estable y repetible.

La interfaz por comandos es simple (tecla “H” muestra ayuda, “?” el estado). Esto facilitó el trabajo del operario: cambiar ángulos, velocidades y tiempos, revisar poses P1..P4 y habilitar o deshabilitar los *drivers* (E/E0/E1). Además, se añadió un paro de emergencia por consola (S) que corta la señal ENA (activa en bajo en el DM542T), desenergizando inmediatamente los motores.

En cuanto a la forma del movimiento, el ciclo se programó en seis pasos para asegurar repetibilidad:

1. M1 y M2 se mueven sincronizados hasta A1/A2.
2. Un *delay* antes de mover M3 (PRE_M3_DELAY_MS), para estabilizar.
3. M3 baja/sube hasta A3 (*moveFixedSingle*).
4. Espera en A3 (M3_HOLD_MS) para realizar la acción (p. ej., *pick-and-place* con la garra).
5. M3 regresa a 0.
6. M1 y M2 regresan a 0 en sincronía.

Esta secuencia demuestra un flujo real del *pick-and-place* y se alineó con la estimación de tiempo de ciclo comentado antes (el *delay* previo, el *hold* en A3 y la pausa entre puntos son los tres tiempos muertos que más pesan y que pueden optimizarse a futuro).

Para proteger el hardware, los ángulos se “delimitaron” a rangos seguros con *clampAngleForMotor*(), (M1: 0 a 600°, M2: 0 a 850°, M3: −6800° a 1700°). También se respeta un tiempo mínimo tras cambiar de dirección (DIR_SETUP_US) antes del primer pulso, y se cuenta con la habilitación común de los *drivers* (ENA) desde el programa. Las velocidades se ajustan con el semiperíodo de pulso (g_PULSE12_US para M1/M2 y g_PULSE3_US para M3), separando las velocidades ya que al ser un tornillo sin fin cuenta con una mayor cantidad de pasos para su movilización ya que la cantidad de micropasos seleccionada en los DIP (STEPS_PER_REV) es de 1600 en estas pruebas.

En la Figura 19 se observa el montaje completo que operó durante las pruebas de cuatro posiciones: Portenta, drivers, fuente y la garra. Con este programa se consiguió:

- Calibrar y mover articulaciones de forma individual (Modo 1).
- Medir repetitividad y ajustar parámetros en línea (Modo 2).
- Ejecutar con fiabilidad P1..P4 (Modo 3) demostrando el objetivo general del proyecto.

Cuadro 3. Variables de código

Nombre de Variable	Tipo de Señal	Tipo de Dato	Valores	Descripción
`servo1`	Salida PWM	Servo	0–180° (pin D3)	Servo 1 (garra).
`servo2`	Salida PWM	Servo	0–180° (pin D5)	Servo 2 (mecanismo de pinzas).
`buttonPin`	Entrada digital (pull-up)	const int	D7 (INPUT_PULLUP)	Pin del *pushbutton*.
`buttonState`	Lectura digital	int	0/1 (LOW/HIGH)	Estado leído del *pushbutton*.
`M3_HOLD_MS`	Parámetro de tiempo (software)	uint16_t	6000 ms (ajustable)	Espera en posición A3 antes de regresar.
`PRE_M3_DELAY_MS`	Parámetro de tiempo (software)	uint16_t	3000 ms (ajustable)	*Delay* tras mover M1+M2 antes de iniciar M3.
`REP_PAUSE_MS`	Parámetro de tiempo (software)	uint16_t	3000 ms (ajustable)	Pausa entre repeticiones y entre puntos P1 a P4.
`STEP1_CH`	Salida digital (STEP M1)	const uint8_t	CH0 (DigitalOutputs)	Pulso STEP hacia DM542T para motor 1.
`DIR1_CH`	Salida digital (DIR M1)	const uint8_t	CH1 (DigitalOutputs)	Dirección para motor 1.
`STEP2_CH`	Salida digital (STEP M2)	const uint8_t	CH2 (DigitalOutputs)	Pulso STEP hacia DM542T para motor 2.
`DIR2_CH`	Salida digital (DIR M2)	const uint8_t	CH3 (DigitalOutputs)	Dirección para motor 2.
`STEP3_CH`	Salida digital (STEP M3)	const uint8_t	CH4 (DigitalOutputs)	Pulso STEP hacia DM542T para motor 3.
`DIR3_CH`	Salida digital (DIR M3)	const uint8_t	CH5 (DigitalOutputs)	Dirección para motor 3.
`ENA_CH`	Salida digital (ENA común)	const uint8_t	CH6 (activo-bajo)	Habilitación común para *drivers* DM542T.

ENA_LEVEL_ENABLE	Lógica de control ENA	const PinStatus	LOW	Nivel lógico para habilitar drivers.
ENA_LEVEL_DISABLE	Lógica de control ENA	const PinStatus	HIGH	Nivel lógico para deshabilitar *drivers*.
DIR1_LEVEL_NEG	Lógica de control DIR M1	const PinStatus	LOW	Define sentido negativo de M1.
DIR2_LEVEL_NEG	Lógica de control DIR M2	const PinStatus	LOW	Define sentido negativo de M2.
DIR3_LEVEL_NEG	Lógica de control DIR M3	const PinStatus	LOW	Define sentido negativo de M3.
g_stop	Estado/flag de sistema	volatile bool	true/false	Paro de emergencia (comando 'S' por serial).
g_ena	Estado de salida ENA	PinStatus	LOW/HIGH	Último nivel escrito a ENA (LOW = habilitado).
g_A1	Setpoint articulación M1	float	default 400°; rango 0–600°	Ángulo objetivo de A1 (se ajusta con el serial).
g_A2	Setpoint articulación M2	float	default 300°; rango 0–850°	Ángulo objetivo de A2 (se ajusta con el serial).
g_A3	Setpoint articulación M3	float	default −100°; rango −6800°..1700°	Ángulo objetivo de A3 (se ajusta con el serial).
g_R1	Parámetro de calibración	float	1.0 (>0)	Grados de motor por 1° de articulación en M1.
g_R2	Parámetro de calibración	float	1.0 (>0)	Grados de motor por 1° de articulación en M2.
g_R3	Parámetro de calibración	float	1.0 (>0)	Grados de motor por 1° de articulación en M3.
g_M3CompDeg	Parámetro de compensación	float	1.0 °/ciclo	Incremento acumulativo aplicado a A3 por repetición.
STEPS_PER_REV	Parámetro del sistema	constexpr long	1600 pasos/rev	Resolución de micro-paso configurada en DM542T.

g_PULSE12_US	Temporización STEP M1/M2	unsigned int	500 µs (semiperiodo)	Ajusta la velocidad de M1 y M2.
g_PULSE3_US	Temporización STEP M3	unsigned int	200 µs (semiperiodo)	Ajusta la velocidad de M3.
DIR_SETUP_US	Temporización DIR	constexpr unsigned int	30 µs	Delay mínimo tras cambiar DIR antes del STEP.
g_mode	Estado de modo	Mode (enum)	MODE_SINGLE/REPEAT/POINTS	Modo de operación actual.
g_repStage	Estado de repetición	RepStage (enum)	RS_IDLE (inicial)	Etapa del las variables en el Modo 2.
g_repN	Parámetro de conteo	uint32_t	0 (ajustable)	Número de repeticiones en Modo 2.
g_repA1	Setpoint base repetición	float	0° (ajustable)	Ángulo base A1 para Modo 2.
g_repA2	Setpoint base repetición	float	0° (ajustable)	Ángulo base A2 para Modo 2.
g_repA3	Setpoint base repetición	float	0° (ajustable)	Ángulo base A3 para Modo 2.
g_pStage	Estado de puntos	PStage (enum)	PS_IDLE (inicial)	Etapa de la secuencia P1..P4 (Modo 3).

En el Cuadro 3 se presentaron las variables del código que se utilizaron en el proyecto, tanto para el control de los motores Stepper como para los servos de la garra. Estas variables facilitaron el ajuste y la coordinación del funcionamiento del sistema, haciendo posible que el brazo se moviera correctamente hacia las posiciones establecidas.

La definición adecuada de estas variables permitió organizar toda la lógica del programa y controlar el comportamiento del brazo SCARA según las necesidades del proyecto. Gracias a ello, el sistema de control pudo configurarse y cuenta con una forma de modificarse de manera más sencilla a futuro dejando una base programada que podrá seguir ajustándose y mejorándose.

VIII. CONCLUSIONES

- Se elaboró un diseño de un brazo SCARA con todos sus componentes por medio de Autodesk Inventor y Fusion, se verificó que la configuración final de eslabones proporciona un espacio de trabajo radial teórico de 390.542 mm y aproximadamente 150° respecto al punto *HOME*. Este alcance fue suficiente para cubrir la zona útil de la banda transportadora, manteniendo estabilidad y evitando problemas de pandeo gracias a los ajustes en longitud, calibre de boquilla e *infill* de las piezas impresas.

- Se fabricó el prototipo físico del brazo SCARA con todos sus componentes impresos en 3D, comprobando que la estructura soporta adecuadamente el peso de los motores, la garra y componentes mecánicos. Las pruebas de ensamblaje y funcionamiento mostraron que, tras corregir tolerancias de impresión, recalibrar la cama y ajustar el *infill*, el brazo operó sin deformaciones críticas y con un nivel de rigidez adecuado para tareas repetitivas de *pick-and-place*.

- Se logró implementar un control con interfaz intuitiva mediante el Arduino Pro Portenta *machine control*, validando tres modos de operación (*Single*, repeticiones y puntos P1–P4) que permitieron calibrar, ensayar y ejecutar la secuencia de cuatro posiciones. La configuración de los *drivers* DM542T redujeron el sobrecalentamiento de los motores, garantizaron un movimiento suave y estable, con tiempos de ciclo en el rango de 30.89 s a 33.56 s por secuencia completa.

- Se evaluó de manera física la precisión y repetitividad del brazo SCARA (Figuras 17 y 18) mediante series de 10, 50 y 100 repeticiones hacia un mismo punto, utilizando un marcador fijado en la garra. Los resultados mostraron una dispersión máxima cercana a 2 mm, atribuida principalmente a vibraciones de los eslabones y paradas bruscas, sin evidencia de pérdida de pasos en los motores. Este comportamiento se considera aceptable para un prototipo académico impreso en 3D y demuestra que el sistema es capaz de repetir posiciones con un error mínimo.

- Se logró documentar de forma detallada el proceso de diseño, implementación y pruebas del brazo robótico, generando modelos en Inventor (Figura 1), listas de componentes (Cuadros 1 y 2), esquemas de conexión (Figuras 15 y 16), código fuente comentado (Anexos A y B) y registro de resultados. Esta documentación deja una base abierta y replicable que facilitará la continuidad del megaproyecto a futuro y eventualmente integrar el brazo SCARA a una línea de automatización completa.

IX. RECOMENDACIONES

- Limpiar de cualquier polvo previo a engrasar y/o aceitar los rodamientos y guías cada 6 meses o cada 200 h de uso (lo que ocurra primero) con lubricante ligero (PTFE o litio). Evitar solventes que arrastren la grasa.
- Ejecutar ciclos cortos semanales para distribuir lubricante así evitar gripado en engranes y guías.
- Evaluar *drivers* de 2-3A RMS más pequeños para M1/M2/M3 si el par requerido lo permite (evaluar si pueden ser programables por Arduino Pro Portenta *machine control*). Mantener micropaso y corriente nominal/reducción en reposo bien ajustados.
- Pasos para desarrollos futuros:
    - Integrar motores con caja reductora para aumentar el par de los movimientos de las poleas encargadas de la movilidad de cada eslabón.
    - Sustituir guías impresas del eje vertical por tubos lisos de Ø8 mm (aluminio o acero; importante que sean ligeros) y casquillos adecuados; tolerancia recomendada Ø8 H7 en el soporte.
    - Si a un futuro el problema es la garra, se puede cambiar fácilmente por una bomba de succión de un punto de contacto fácil de programar con Arduino (solo es necesario su *driver* de succión ya incluido habitualmente).
    - Si el calentamiento de los motores llega a ser problema para el futuro, probar proteger las piezas impresas en 3D por capaz de aluminio o algún material que disipe el calor. También se puede probar cambiar las piezas en contacto por una impresión 3D con un alto nivel de resistencia al calor.
    - Añadir aceleración y deceleración (trapezoidal o S-curve) para reducir impactos, vibraciones y dispersión en las posiciones finales.
    - Sustituir el *HOME* manual e instalar finales de carrera/sensor de referencia para un *HOME* calculable sin manipulación humana y límites seguros.
    - Dos alternativas:
        - (a) Adaptar el brazo al ancho de una faja convencional acortando el primer eslabón.
        - (b) Mandar a fabricar una faja de entre 565 mm y 570 mm solo si el *layout* actual es el definitivo.

X. BIBLIOGRAFIA

@innovamecatronica (Dirección). (2025). *Scara Robot DIY* [Película].

Annin Robotics. (2019). *Annin Robotics*. Obtenido de AR2 and AR3 SCARA-style Open Source Robot Arms: https://www.anninrobotics.com

Arduino. (2023). *Arduino*. Obtenido de Portenta Pro Series: https://www.arduino.cc/pro/hardware/product/portenta-h7

Arduino. (02 de Marzo de 2023). *Arduino.* Obtenido de Pinout Portenta Machine Control: https://docs.arduino.cc/resources/pinouts/AKX00032-full-pinout.pdf?_gl=1*1ui6gnj*_up*MQ..*_ga*OTM4MzY3NDAwLjE3NDY3NTg0NDU.*_ga_NEXN8H46L5*czE3NDY3NTg0NDQkbzEkZzAkdDE3NDY3NTg0NDQkajAkbDAkaDc3MDU3MzM3MQ..

Arduino. (11 de Noviembre de 2025). *Arduino.* Obtenido de Arduino® Portenta Machine Control: https://docs.arduino.cc/resources/datasheets/AKX00032-datasheet.pdf

Autodesk. (s.f.). *Autodesk*. Obtenido de Inventor Product Page: https://www.autodesk.com/products/inventor

Bogue, R. (2012). Robots in manufacturing: a review of recent developments. *Industrial Robot: An International Journal*, 219–223.

Components101. (30 de Marzo de 2019). *Components101*. Obtenido de MG90S – Metal Gear Micro Servo Motor: https://components101.com/motors/mg90s-metal-gear-servo-motor?

Corke, P. (2017). *Robotics, Vision and Control: Fundamental Algorithms in MATLAB (2nd ed.).* Springer.

Correll, N. e. (2022). *Introduction to Autonomous Robots: Mechanisms, Sensors, Actuators and Algorithms.* The MIT Press.

Craig, J. J. (2005). *Introduction to Robotics: Mechanics and Control.* Pearson Prentice Hall: 3ra Ed.

Electronicoscaldas. (s.f.). *Electronicoscaldas*. Obtenido de Micro Servo Motor MG90S - Tower Pro: https://www.electronicoscaldas.com/datasheet/MG90S_Tower-Pro.pdf?srsltid=AfmBOoqze5xgHxp91761Rwa-1wK82cwd0fDCHR1_Et_Sx1PIsbJNttT3&

EVST. (s.f.). *EVST*. Obtenido de https://www.evsint.com/es

Festo. (2019). *Festo*. Obtenido de Industrial Robots Technical Guide: https://www.festo.com/us/en/e/solutions/industries/life-science/precision-and-reliability-in-automated-liquid-handling-id_1641635/

GG_CADworks. (25 de Junio de 2025). *https://cults3d.com*. Obtenido de Gripper SG90 (2DOF): https://cults3d.com/en/3d-model/various/gripper-sg90-2dof

Govani, H. (1 de Enero de 2025). *GrabCad*. Obtenido de TowerPro SG90 Mini Servo-SunRobotics_in: https://grabcad.com/library/4145-towerpro-sg90-mini-servo-sunrobotics_in-1

Hap. (s.f.). *CirkitDesigner*. Obtenido de Hao: https://app.cirkitdesigner.com/project/200f7a86-ebf5-4356-8122-ac951ffccf44

HITBOT. (s.f.). *HITBOT*. Obtenido de Brazo robótico de 4 ejes: https://www.hitbotrobot.com/es/product/2442-robotic-arm

Innova Mecatrónica. (7 de Octubre de 2024). *Patreon*. Obtenido de Innova Mecatrónica: https://www.patreon.com/posts/113540145?collection=51275

Innova Mecatronica. (2 de Marzo de 2025). *Youtube*. Obtenido de https://www.youtube.com/shorts/uZfH0YKdiwk

jjRobots. (2019). *Sketfab*. Obtenido de SCARA Robotic Arm by (jjRobots): https://sketchfab.com/3d-models/scara-robotic-arm-by-jjrobots-6db379cb0c064c2e8d0e7fd01469bf44

Lynch, K. M. (2017). *Modern Robotics: Mechanics, Planning, and Control.* Cambridge University Press.

Makino, H. (1981). *Selective Compliance Assembly Robot Arm: A new concept of robot for assembly operations.* University of Yamanashi.

Mircescu, A., vojjta, & Tsogoo. (19 de Septiembre de 2021). *Github*. Obtenido de x-scara: https://github.com/madl3x/x-scara/tree/master

Motion Control Products. (s.f.). *Motion Control Products*. Obtenido de NEMA 17 Stepper Motors (42mm): https://motioncontrolproducts.com/electric-motors/stepper-motors/hybrid-stepper-motors/nema-17.html

Motoman. (2020). *Yaskawa*. Obtenido de SCARA Robots for High-Speed Assembly: https://www.motoman.com/es-mx

Ni. (2 de Abril de 2025). *Ni Adquisición de datos (DAQ)*. Obtenido de Guía de medidas de frecuencia - ¿Cómo se mide la frecuencia?: https://www.ni.com/es/shop/data-acquisition/measurement-fundamentals/frequency-measurements-how-to-

guide.html?srsltid=AfmBOopcftoBX872mvMBViL_QRTIOzUboAU-y2fDF6LCalc6tkWiNfy9

Niku, S. B. (2010). *Introduction to Robotics: Analysis, Control, Applications.* Wiley: 2da Ed.

Open Stax College Physics. (s.f.). 9.1 Ángulo de rotación y velocidad angular. En O. S. Physics, *Física 1107 del Douglas College.* Obtenido de https://pressbooks-bccampus-ca.translate.goog/douglasphys1107/chapter/9-1-rotation-angle-and-angular-velocity/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=sge

OpenAI. (2024). *OpenAI*. Obtenido de ChatGPT (versión del 13 de diciembre) [Modelo de lenguaje de gran tamaño]: https://chat.openai.com/

Park, K. M. (2017). *MODERN ROBOTICS MECHANICS, PLANNING, AND CONTROL.* Cambridge : Cambridge University Press.

Robocraze. (19 de Diciembre de 2022). *Robocraze*. Obtenido de What is NEMA17?: robocraze.com/blogs/post/nema-17-the-high-torque-stepper-motor-working-principle

Siciliano, B. S. (2010). *Robotics: Modelling, Planning and Control.* Springer.

Siegwart, R. N. (2011). *Introduction to Autonomous Mobile Robots (2nd ed.).* The MIT Press.

STEPPERONLINE. (Octubre de 2020). *STEPPERONLINE.* Obtenido de User Manual DM542T(V4.0): https://www.omc-stepperonline.com/download/DM542T_V4.0.pdf?srsltid=AfmBOoqbzyC0CdVR1vSNIPjwkpFeMwbts_u45wRLfX7-N20Hqk-vax8T

STEPPERONLINE. (31 de Octubre de 2023). *STEPPERONLINE*. Obtenido de Learning & FAQ: Can You Send Me A Schematic That How to Wire The Stepper Driver to An Arduino?: https://help.stepperonline.com/en/article/can-you-send-me-a-schematic-that-how-to-wire-the-stepper-driver-to-an-arduino-d5ot82

Stepperonline. (s.f.). *Stepperonline*. Obtenido de Nema 17 Stepper Motor: https://www.omc-stepperonline.com/nema-17-stepper-motor

Stuggi. (29 de Marzo de 2017). *Thingiverse*. Obtenido de DIN 35 Rail Feet: https://www.thingiverse.com/thing:2211951

Thingiverse. (1 de Noviembre de 2021). Obtenido de DIN RAIL DM542T STEPPER DRIVER: https://www.thingiverse.com/thing:5078955

Vasquez, S. (Octubre de 2025). *CirkitDesigner*. Obtenido de Brazo Scara: https://app.cirkitdesigner.com/project/99657ccf-235c-4e36-877b-6006cac446f2

Vasquez, S. (Octubre de 2025). *CirkitDesigner*. Obtenido de Garra de Brazo Scara: https://app.cirkitdesigner.com/project/6bdbc835-a05b-4505-bcb9-8bead94dd35d

Wessling, B. (10 de Enero de 2024). *The Robot Report.* Obtenido de IFR: World sets record for operational robots in 2022: https://www.therobotreport.com/ifr-world-sets-record-for-operational-robots-in-2022/#:~:text=The%20world%20set%20a%20record,information%20about%20robot%20density%20worldwide.

Wessling, B. (10 de Enero de 2024). *The Robot Report*. Obtenido de IFR: World sets record for operational robots in 2022: https://www.therobotreport.com/ifr-world-sets-record-for-operational-robots-in-2022/#:~:text=The%20world%20set%20a%20record,information%20about%20robot%20density%20worldwide.

XI. ANEXOS

Prueba_Control_Servos.ino

```ino
#include <Servo.h>

Servo servo1;  // Servo 1 (garra)
Servo servo2;  // Servo 2 (mecanismo de pinzas)

const int buttonPin = 7;  // Pin para el botón
int buttonState = 0;  // Variable para leer el estado del botón

void setup() {
  // Inicializar los servos
  servo1.attach(3);  // Servo 1 en pin D3
  servo2.attach(5);  // Servo 2 en pin D5

  // Inicializar el botón
  pinMode(buttonPin, INPUT_PULLUP);  // Configuramos el pin del botón como entrada con resistencia pull-up
}

void loop() {
  // Leer el estado del botón
  buttonState = digitalRead(buttonPin);

  // Si el botón es presionado (el pin será LOW porque está en INPUT_PULLUP)
  if (buttonState == LOW) {
    // Mover el primer servo a 90 grados
    servo1.write(90);
    delay(1000);  // Esperar 1 segundo

    // Mover el segundo servo a 40 grados (cerrar la garra)
    servo2.write(150);
    delay(1000);  // Esperar 6 segundos a que regrese a la mesa

    // Regresar el primer servo a 0 grados
    servo1.write(0);
    delay(6000);  // Esperar 1 segundo

    // Regresar el segundo servo a 0 grados (abre la garra soltando producto)
    servo2.write(0);
    delay(1000);  // Esperar 1 segundo
  }
}

```

Anexo A. Código de la garra

```ino
#include <Arduino.h>
#include <Arduino_PortentaMachineControl.h>

/* ============================================================
   SCARA Sync v6+
   Control de 3 motores paso a paso (DM542T) con Portenta
   Modos:
     - Modo 1 (Single): Un ciclo con A1/A2/A3 actuales.
     - Modo 2 (Repeat): Asistente para N repeticiones con compensación en M3.
     - Modo 3 (Points): Secuencia P1..P4 -> retorno a 0 con pausa entre puntos.
   Interacción por monitor serial (115200 bps). Paro de emergencia: 'S'.
   ============================================================ */

/* ==================== ADVERTENCIA MECÁNICA ====================
   Mantener una separación mínima de ~4 cm entre el brazo y la mesa
   para evitar colisiones durante movimientos de prueba.
   ============================================================ */

/* ==================== Parámetros rápidos ====================
   Variables de tiempos en milisegundos que pueden ajustarse por comandos:
   - M3_HOLD_MS : espera en posición objetivo de M3 antes de regresar.
   - PRE_M3_DELAY_MS : retardo tras mover M1+M2 y antes de iniciar M3.
   - REP_PAUSE_MS : pausa entre repeticiones (Modo 2) y entre P1..P4 (Modo 3).
   ============================================================ */
uint16_t           M3_HOLD_MS       = 6000;   // Espera en A3 antes de regresar (ajustable con H3)
uint16_t           PRE_M3_DELAY_MS  = 3000;   // Retardo previo a mover M3 (ajustable con D3)
uint16_t           REP_PAUSE_MS     = 3000;   // Pausa entre ciclos/puntos (ajustable con RP)

/* ==================== Asignación de pines (Portenta MC: J6) ====================
   Cada motor usa un canal STEP (pulso) y DIR (dirección).
   ENA_CH es común para los tres drivers DM542T (entrada ENA+).
   Nota: el DM542T usa entradas optoacopladas y ENA es activo en nivel bajo.
   ============================================================================== */
// M1
const uint8_t STEP1_CH = 0;  // CH0 -> PUL+ M1
const uint8_t DIR1_CH  = 1;  // CH1 -> DIR+ M1
// M2
const uint8_t STEP2_CH = 2;  // CH2 -> PUL+ M2
const uint8_t DIR2_CH  = 3;  // CH3 -> DIR+ M2
// M3
const uint8_t STEP3_CH = 4;  // CH4 -> PUL+ M3
const uint8_t DIR3_CH  = 5;  // CH5 -> DIR+ M3
// ENA común (DM542T activo-bajo)
const uint8_t ENA_CH   = 6;  // CH6 -> ENA+

/* ============== Resolución y temporización de pasos ==============
   - STEPS_PER_REV: pasos por revolución del motor a la resolución de micro-paso
     configurada por DIP (debe coincidir con el driver, p.ej. 1600 = 1/8 micro-paso).
   - g_PULSE*_US: semiperiodo del pulso STEP en microsegundos. Periodo = 2*valor.
     Valores mayores -> menor velocidad (más lento y más torque disponible).
   - DIR_SETUP_US: tiempo mínimo tras cambiar DIR antes del primer pulso STEP.
   ============================================================ */
constexpr long   STEPS_PER_REV   = 1600;  // Debe corresponder al micro-paso DIP del DM542T
unsigned int     g_PULSE12_US    = 500;   // M1 & M2: semiperiodo de STEP en us
unsigned int     g_PULSE3_US     = 200;   // M3: semiperiodo de STEP en us
constexpr unsigned int DIR_SETUP_US = 30; // Guard time tras cambiar DIR (us)

/* ============== Lógica de habilitación (ENA activo-bajo) ==============
   - ENA_LEVEL_ENABLE : nivel lógico para habilitar drivers (LOW en DM542T).
   - ENA_LEVEL_DISABLE: nivel lógico para deshabilitar (HIGH).
   ============================================================ */
const PinStatus ENA_LEVEL_ENABLE  = LOW;   // 0 = habilitado
const PinStatus ENA_LEVEL_DISABLE = HIGH;  // 1 = deshabilitado
```

```
/* ============== Convención de dirección por motor ==============
   DIR*_LEVEL_NEG define el nivel de DIR que se considera "negativo".
   Si un eje invierte su sentido real vs. esperado, ajustar estos niveles.
   ============================================================== */
const PinStatus DIR1_LEVEL_NEG = LOW;
const PinStatus DIR2_LEVEL_NEG = LOW;
const PinStatus DIR3_LEVEL_NEG = LOW;

/* ============== Estado global ==============
   - g_stop: bandera de paro de emergencia (forzada con comando 'S').
   - g_ena : último estado escrito a la salida ENA (habilitado/deshabilitado).
   ============================================================== */
volatile bool g_stop = false;
PinStatus     g_ena  = ENA_LEVEL_DISABLE;

/* ============== Objetivos de articulación (grados) y límites ==============
   A1/A2/A3 son ángulos de articulación (no del motor). Sus rangos son
   validados con clampAngleForMotor() antes de mover. R1/R2/R3 son las
   reducciones (grados de motor necesarios por cada grado de articulación).
   ============================================================== */
float g_A1 = 400.0f;  // M1: 0..600
float g_A2 = 300.0f;  // M2: 0..850
float g_A3 = -100.0f; // M3: -6800..1700

float g_R1 = 1.0f;    // Grados de motor por 1° de articulación en M1
float g_R2 = 1.0f;    // "" M2
float g_R3 = 1.0f;    // "" M3

/* ============== Compensación en Modo 2 ==============
   g_M3CompDeg: incremento acumulativo aplicado a A3 por repetición,
   útil para compensar deriva mecánica/sag en ciclos repetidos.
   ============================================================== */
float g_M3CompDeg = 1.0f;

/* ============== Buffer de línea para comandos por Serial ============== */
char   buf[96];
size_t blen = 0;

/* ============== Selección de modo de trabajo ==============
   MODE_SINGLE  (1): ciclo único con A1/A2/A3.
   MODE_REPEAT  (2): asistente para N repeticiones.
   MODE_POINTS  (3): ejecuta P1..P4 con retorno y pausa.
   ============================================================== */
enum Mode : uint8_t { MODE_SINGLE=1, MODE_REPEAT=2, MODE_POINTS=3 };
Mode g_mode = MODE_SINGLE;

/* ============== Estado del asistente del Modo 2 ==============
   RS_NEED_N: espera el número de repeticiones.
   RS_NEED_ANGLES: espera A1 A2 A3.
   RS_READY: preparado para ejecutar (Enter).
   ============================================================== */
enum RepStage : uint8_t { RS_IDLE=0, RS_NEED_N, RS_NEED_ANGLES, RS_READY };
RepStage g_repStage = RS_IDLE;
uint32_t g_repN = 0;
float    g_repA1=0, g_repA2=0, g_repA3=0;

/* ============== Modo 3 (cuatro puntos P1..P4) ==============
   - Estructura Pose con A1/A2/A3.
   - g_P[] contiene los cuatro puntos. g_P_set[] indica cuál está definido.
   - g_pStage controla la carga guiada de P1..P4 desde el serial.
   ============================================================== */
struct Pose { float A1, A2, A3; };
Pose g_P[4] = {
  {  0.f,   500.f, -6800.f },
  {  200.f, 400.f, -6800.f },   // Valores iniciales de ejemplo
  {  200.f, 850.f, -6800.f },
  {  400.f, 250.f, -6800.f }
};
bool  g_P_set[4] = { true, true, true, true };
enum PStage : uint8_t { PS_IDLE=0, PS_NEED_P1, PS_NEED_P2, PS_NEED_P3, PS_NEED_P4, PS_READY };
PStage g_pStage = PS_IDLE;
```

```cpp
/* ================== Utilitarios numéricos ==================
   - motorDegToSteps: convierte grados de motor a pasos (según STEPS_PER_REV).
   - jointDegToSteps: convierte grados de articulación a pasos via relación R*.
   - setENA/toggleENA: controlan el pin ENA común y reportan estado por serial.
   - stepHigh/stepLow: helpers para emitir flanco en canal STEP.
   - quickSerialPoll: lee carácter 'S' para PARO inmediato y deshabilita ENA.
   - delayWithPoll: espera no bloqueante (sondeo de 'S' durante la espera).
   ============================================================ */
inline long motorDegToSteps(double motorDeg) {
  return (long) lround((motorDeg / 360.0) * (double)STEPS_PER_REV);
}
inline long jointDegToSteps(double jointDeg, double ratioMotorPerJoint) {
  return motorDegToSteps(jointDeg * ratioMotorPerJoint);
}
inline void setENA(PinStatus level) {
  g_ena = level;
  MachineControl_DigitalOutputs.write(ENA_CH, g_ena);
  if (g_ena == ENA_LEVEL_ENABLE) g_stop = false; // Al habilitar, limpiar paro.
  Serial.print(F("ENA = "));
  Serial.println((g_ena == ENA_LEVEL_ENABLE) ? F("0 (HABILITADO)") : F("1 (DESHABILITADO)"));
}
inline void toggleENA() { setENA((g_ena==ENA_LEVEL_ENABLE)?ENA_LEVEL_DISABLE:ENA_LEVEL_ENABLE); }

inline void stepHigh(uint8_t stepCh) { MachineControl_DigitalOutputs.write(stepCh, HIGH); }
inline void stepLow (uint8_t stepCh) { MachineControl_DigitalOutputs.write(stepCh, LOW);  }

inline void quickSerialPoll() {
  while (Serial.available()) {
    char c = Serial.read();
    if (c=='S' || c=='s') {              // Paro de emergencia por consola
      g_stop = true;
      setENA(ENA_LEVEL_DISABLE);         // Desenergizar inmediatamente
      Serial.println(F(">>> PARO: ENA=1"));
    }
  }
}

/* ================== Espera con sondeo (no bloqueante) ==================
   Mantiene lazo de espera en ms pero monitorea 'S' y buffer serial.
   Retorna false si se activó g_stop durante la espera.
   ====================================================================== */
bool delayWithPoll(uint32_t ms) {
  uint32_t start = millis();
  while ((millis() - start) < ms) {
    if (g_stop) return false;
    quickSerialPoll();
    delay(1);
  }
  return true;
}

/* ================== Movimiento de un solo eje ==================
   Genera 'steps' pulsos en stepCh, con DIR en dirLevel y semiperiodo pulseUs.
   Respeta g_stop y realiza sondeo de seguridad durante el lazo.
   Retorna la cantidad de pasos efectivamente emitidos.
   ================================================================ */
long moveFixedSingle(uint8_t stepCh, uint8_t dirCh, PinStatus dirLevel, long steps, unsigned int pulseUs)
{
  if (steps <= 0) return 0;
  MachineControl_DigitalOutputs.write(stepCh, LOW);
  delayMicroseconds(5);
  MachineControl_DigitalOutputs.write(dirCh, dirLevel);
  delayMicroseconds(DIR_SETUP_US);

  long done = 0;
  for (long i = 0; i < steps; i++) {
    if (g_stop) break;
    stepHigh(stepCh);
    delayMicroseconds(pulseUs);
    stepLow(stepCh);
    delayMicroseconds(pulseUs);
    done++;
    quickSerialPoll();
  }
  return done;
}
```

```
/* ================== Movimiento sincronizado de dos ejes (M1 y M2) ==================
   Implementación tipo DDA/Bresenham:
   - Sincroniza la emisión de pasos para que ambos ejes terminen al mismo tiempo.
   - stepsA/B: número total de pasos requeridos en cada eje.
   - dirLevA/B: niveles de dirección.
   - pulseUs: semiperiodo de STEP compartido.
   - doneA/doneB: pasos efectivos ejecutados (salida).
   ================================================================================ */
void movePairSync(uint8_t stepA, uint8_t dirA, PinStatus dirLevA, long stepsA,
                  uint8_t stepB, uint8_t dirB, PinStatus dirLevB, long stepsB,
                  long &doneA, long &doneB, unsigned int pulseUs)
{
  doneA = 0; doneB = 0;
  if (stepsA <= 0 && stepsB <= 0) return;

  MachineControl_DigitalOutputs.write(stepA, LOW);
  MachineControl_DigitalOutputs.write(stepB, LOW);
  delayMicroseconds(5);
  MachineControl_DigitalOutputs.write(dirA, dirLevA);
  MachineControl_DigitalOutputs.write(dirB, dirLevB);
  delayMicroseconds(DIR_SETUP_US);

  long ticks = max(stepsA, stepsB);
  long accA = 0, accB = 0;

  for (long i = 0; i < ticks; i++) {
    if (g_stop) break;
    bool doA=false, doB=false;

    accA += stepsA; if (accA >= ticks) { doA = true; accA -= ticks; }
    accB += stepsB; if (accB >= ticks) { doB = true; accB -= ticks; }

    if (doA || doB) {
      if (doA) stepHigh(stepA);
      if (doB) stepHigh(stepB);
      delayMicroseconds(pulseUs);
      if (doA) { stepLow(stepA); doneA++; }
      if (doB) { stepLow(stepB); doneB++; }
      delayMicroseconds(pulseUs);
    } else {
      // Si ningún eje requiere paso en este tick, mantener timing.
      delayMicroseconds(pulseUs*2);
    }
    quickSerialPoll();
  }
}

/* ================== Validación de límites por motor ==================
   Limita el valor angular de articulación 'v' a los rangos seguros por eje.
   ================================================================== */
float clampAngleForMotor(uint8_t id, float v) {
  if      (id == 1) { if (v < 0.f) v = 0.f;     if (v > 600.f)  v = 600.f;  }
  else if (id == 2) { if (v < 0.f) v = 0.f;     if (v > 850.f)  v = 850.f;  }
  else if (id == 3) { if (v < -6800.f) v=-6800.f; if (v > 1700.f) v = 1700.f; }
  return v;
}
```

```
/* ================== Ciclo completo con ángulos dados ==================
   Secuencia:
     1) M1 y M2 en sincronía: 0 -> A1/A2
     2) Retardo PRE_M3_DELAY_MS
     3) M3: 0 -> A3
     4) Espera M3_HOLD_MS en la posición de A3
     5) M3: A3 -> 0
     6) M1 y M2 sincronizados: A1/A2 -> 0
   Condiciones:
     - Requiere ENA habilitado (E0).
     - Respeta paro de emergencia en todo momento.
   ====================================================================== */
void runCycleSyncAngles(float A1, float A2, float A3) {
  if (g_ena != ENA_LEVEL_ENABLE) {
    Serial.println(F("No puedo mover: ENA=1 (OFF). Usa 'E' o 'E0' para habilitar."));
    return;
  }
  g_stop = false;

  // Clampeo de seguridad a rangos por eje
  A1 = clampAngleForMotor(1, A1);
  A2 = clampAngleForMotor(2, A2);
  A3 = clampAngleForMotor(3, A3);

  // Conversión a pasos según reducción
  long s1 = labs(jointDegToSteps((double)A1, (double)g_R1));
  long s2 = labs(jointDegToSteps((double)A2, (double)g_R2));
  long s3 = labs(jointDegToSteps((double)A3, (double)g_R3));

  // Determinar niveles DIR de ida y regreso
  PinStatus dir1_go = (A1 < 0.0f) ? DIR1_LEVEL_NEG : (DIR1_LEVEL_NEG==HIGH?LOW:HIGH);
  PinStatus dir2_go = (A2 < 0.0f) ? DIR2_LEVEL_NEG : (DIR2_LEVEL_NEG==HIGH?LOW:HIGH);
  PinStatus dir3_go = (A3 < 0.0f) ? DIR3_LEVEL_NEG : (DIR3_LEVEL_NEG==HIGH?LOW:HIGH);

  PinStatus dir1_back = (dir1_go==HIGH?LOW:HIGH);
  PinStatus dir2_back = (dir2_go==HIGH?LOW:HIGH);
  PinStatus dir3_back = (dir3_go==HIGH?LOW:HIGH);

  // 1) M1 y M2 sincronizados hasta A1/A2
  long out1=0, out2=0;
  movePairSync(STEP1_CH, DIR1_CH, dir1_go, s1,
               STEP2_CH, DIR2_CH, dir2_go, s2,
               out1, out2, g_PULSE12_US);
  if (g_stop) { Serial.println(F("Abortado en etapa 1.")); return; }

  // 2) Espera antes de mover M3
  if (!delayWithPoll(PRE_M3_DELAY_MS)) { Serial.println(F("Abortado en espera pre-M3.")); return; }

  // 3) M3 hacia A3
  long out3 = moveFixedSingle(STEP3_CH, DIR3_CH, dir3_go, s3, g_PULSE3_US);
  if (g_stop) { Serial.println(F("Abortado en etapa 2.")); return; }

  // 4) Hold en A3
  if (!delayWithPoll(M3_HOLD_MS)) { Serial.println(F("Abortado en hold.")); return; }

  // 5) Retorno M3 a 0
  long back3 = moveFixedSingle(STEP3_CH, DIR3_CH, dir3_back, out3, g_PULSE3_US);
  if (g_stop) { Serial.println(F("Abortado en etapa 4.")); return; }

  // 6) Retorno M1 y M2 a 0 en sincronía
  long b1=0, b2=0;
  movePairSync(STEP1_CH, DIR1_CH, dir1_back, out1,
               STEP2_CH, DIR2_CH, dir2_back, out2,
               b1, b2, g_PULSE12_US);
  if (g_stop) { Serial.println(F("Abortado en etapa 5.")); return; }

  Serial.println(F("Ciclo OK."));
}
```

```
/* ================== Modo 1 (Single) ==================
   Ejecuta un ciclo usando los valores actuales g_A1, g_A2, g_A3.
   ======================================================== */
void runCycleSync() { runCycleSyncAngles(g_A1, g_A2, g_A3); }

/* ================== Parseo de argumentos por Serial ==================
   - parseFloatAfter : lee un float tras espacios/igual.
   - parseUIntAfter_u32/u16 : lee enteros sin signo (32/16 bits).
   - parse3floats : lee tres floats separados por espacios (A1 A2 A3).
   ====================================================================== */
bool parseFloatAfter(const char* s, float &val) {
  while (*s==' '||*s=='\t'||*s=='=') s++;
  char* endp=nullptr;
  val = strtof(s,&endp);
  return (endp!=s);
}
bool parseUIntAfter_u32(const char* s, uint32_t &val) {
  while (*s==' '||*s=='\t'||*s=='=') s++;
  char* endp=nullptr;
  long tmp = strtol(s,&endp,10);
  if (endp==s || tmp<=0) return false;
  val = (uint32_t)tmp;
  return true;
}
bool parseUIntAfter_u16(const char* s, uint16_t &val) {
  uint32_t tmp;
  if (!parseUIntAfter_u32(s, tmp)) return false;
  if (tmp > 65535) tmp = 65535;
  val = (uint16_t)tmp;
  return true;
}
bool parse3floats(const char* s, float &a, float &b, float &c) {
  char* endp=nullptr;
  while (*s==' '||*s=='\t') s++;
  a = strtof(s,&endp); if (endp==s) return false; s=endp;
  while (*s==' '||*s=='\t') s++;
  b = strtof(s,&endp); if (endp==s) return false; s=endp;
  while (*s==' '||*s=='\t') s++;
  c = strtof(s,&endp); if (endp==s) return false;
  return true;
}
```

```
/* ================== Estado & Ayuda por Serial ==================
   - printPositions : imprime P1..P4 en formato A1/A2/A3.
   - printStatus : imprime modo, ENA, A*, R*, velocidades, tiempos, etc.
   - help : lista de comandos disponibles y breve guía de modos.
   ================================================================ */
void printPositions() {
  for (int i=0;i<4;i++){
    Serial.print(F("P")); Serial.print(i+1); Serial.print(F(": "));
    Serial.print(g_P[i].A1); Serial.print(F("/"));
    Serial.print(g_P[i].A2); Serial.print(F("/"));
    Serial.println(g_P[i].A3);
  }
}
void printStatus() {
  Serial.println(F("=== Estado ==="));
  Serial.print(F("Modo: "));
  if      (g_mode==MODE_SINGLE)  Serial.println(F("1 (Single)"));
  else if (g_mode==MODE_REPEAT)  Serial.println(F("2 (Repeticiones)"));
  else                           Serial.println(F("3 (Puntos)"));
  Serial.print(F("ENA: ")); Serial.println((g_ena==ENA_LEVEL_ENABLE)?F("ON(0)"):F("OFF(1)"));
  Serial.print(F("A1=")); Serial.print(g_A1); Serial.print(F("°  R1=")); Serial.println(g_R1,4);
  Serial.print(F("A2=")); Serial.print(g_A2); Serial.print(F("°  R2=")); Serial.println(g_R2,4);
  Serial.print(F("A3=")); Serial.print(g_A3); Serial.print(F("°  R3=")); Serial.println(g_R3,4);
  Serial.print(F("V12(us)=")); Serial.print(g_PULSE12_US);
  Serial.print(F("  V3(us)=")); Serial.println(g_PULSE3_US);
  Serial.print(F("HoldM3(ms)=")); Serial.print(M3_HOLD_MS);
  Serial.print(F("  PreM3(ms)=")); Serial.print(PRE_M3_DELAY_MS);
  Serial.print(F("  RepPause(ms)=")); Serial.println(REP_PAUSE_MS);
  Serial.print(F("CompM3(deg/rep)=")); Serial.println(g_M3CompDeg,4);
  if (g_mode==MODE_REPEAT) {
    Serial.print(F("RepStage=")); Serial.print((int)g_repStage);
    Serial.print(F("  N=")); Serial.print(g_repN);
    Serial.print(F("  A1/A2/A3=")); Serial.print(g_repA1); Serial.print('/');
    Serial.print(g_repA2); Serial.print('/'); Serial.println(g_repA3);
  }
  if (g_mode==MODE_POINTS) {
    Serial.println(F("Posiciones P1..P4 (A1/A2/A3):"));
    printPositions();
    Serial.print(F("PStage=")); Serial.println((int)g_pStage);
  }
}
void help() {
  Serial.println(F("Comandos:"));
  Serial.println(F("  Enter     : Modo 1 -> un ciclo;  Modo 2 -> según asistente;  Modo 3 -> ejecuta
P1..P4 si READY"));
  Serial.println(F("  M1 / M2 / M3 : Cambiar a Modo 1 (Single) / 2 (Repeticiones) / 3 (Puntos)"));
  Serial.println(F("  A1/A2/A3 <deg> : (Modo 1) Cambia ángulo (M1:0..600, M2:0..850, M3:-6800..1700)"));
  Serial.println(F("  R1/R2/R3 <ratio> : Grados motor / 1° articulación (>0)"));
  Serial.println(F("  V12 <us>  : Velocidad M1/M2 (mitad período en us)"));
  Serial.println(F("  V3  <us>  : Velocidad M3 (mitad período en us)"));
  Serial.println(F("  D3  <ms>  : Delay antes de mover M3 (default 3000 ms)"));
  Serial.println(F("  H3  <ms>  : Hold sobre A3 antes de regresar (default 600 ms)"));
  Serial.println(F("  RP  <ms>  : Pausa entre repeticiones (M2) y entre puntos (M3)"));
  Serial.println(F("  AC  <deg> : Compensación M3 por repetición en Modo 2 (deg/rep).  AC? muestra
valor"));
  Serial.println(F("  P1/P2/P3/P4 <A1> <A2> <A3>  : Define una pose;  P1? / P? para ver"));
  Serial.println(F("  E / E0 / E1 : ENA toggle / ON / OFF"));
  Serial.println(F("  S        : PARO de emergencia (desenergiza ENA)"));
  Serial.println(F("  ? / h    : Estado / Ayuda"));
  Serial.println(F(""));
  Serial.println(F("Modo 2: al entrar, ingresa N y luego A1 A2 A3; Enter ejecuta."));
  Serial.println(F("Modo 3: al entrar, ingresa P1..P4 (si hace falta); Enter ejecuta P1->0, P2->0, P3->0,
P4->0 con pausa RP."));
}
```

```
/* ================== Modo 2 (Repeticiones) ==================
   Flujo:
     - RS_NEED_N      : ingresar N.
     - RS_NEED_ANGLES : ingresar A1 A2 A3.
     - RS_READY       : Enter ejecuta N ciclos, con A3 compensado por AC*k.
   ============================================================ */
void enterMode2() {
  g_mode = MODE_REPEAT;
  g_repStage = RS_NEED_N;
  g_repN = 0;
  g_repA1 = g_repA2 = g_repA3 = 0.0f;
  Serial.println(F("MODO 2: Repeticiones."));
  Serial.println(F("Ingresa N (repeticiones) y presiona Enter."));
}
void runCycleSyncAngles(float A1, float A2, float A3); // forward declaration

void handleMode2Line() {
  if (g_repStage == RS_NEED_N) {
    uint32_t n;
    if (parseUIntAfter_u32(buf, n)) {
      g_repN = n;
      g_repStage = RS_NEED_ANGLES;
      Serial.print(F("N=")); Serial.println(g_repN);
      Serial.println(F("Ingresa A1 A2 A3 (grados, separados por espacio) y presiona Enter."));
    } else {
      Serial.println(F("Valor inválido. Ingresa un entero > 0 para N."));
    }
    return;
  }
  if (g_repStage == RS_NEED_ANGLES) {
    float a1,a2,a3;
    if (parse3floats(buf, a1,a2,a3)) {
      g_repA1 = clampAngleForMotor(1, a1);
      g_repA2 = clampAngleForMotor(2, a2);
      g_repA3 = clampAngleForMotor(3, a3);
      g_repStage = RS_READY;
      Serial.print(F("A1/A2/A3 = ")); Serial.print(g_repA1); Serial.print('/');
      Serial.print(g_repA2); Serial.print('/'); Serial.println(g_repA3);
      Serial.println(F("READY: Enter ejecuta; 'N' reingresa N; 'A' reingresa ángulos."));
    } else {
      Serial.println(F("Formato inválido. Ej: 20 35 -15"));
    }
    return;
  }
  if (g_repStage == RS_READY) {
    if (blen==0) {
      if (g_ena != ENA_LEVEL_ENABLE) { Serial.println(F("ENA=1 (OFF). Usa 'E0' para habilitar."));
return; }
      Serial.print(F("Ejecutando ")); Serial.print(g_repN); Serial.println(F(" repetición(es)..."));
      for (uint32_t k=1; k<=g_repN; ++k) {
        float A3k = clampAngleForMotor(3, g_repA3 + (float)k * g_M3CompDeg);
        Serial.print(F("  Repetición ")); Serial.print(k); Serial.print(F("/")); Serial.print(g_repN);
        Serial.print(F("  | A3k=")); Serial.println(A3k);
        runCycleSyncAngles(g_repA1, g_repA2, A3k);
        if (g_stop) { Serial.println(F(">>> PARO durante repeticiones.")); break; }
        if (k < g_repN) {
          if (!delayWithPoll(REP_PAUSE_MS)) { Serial.println(F("Abortado en pausa entre repeticiones."));
break; }
        }
      }
      Serial.println(F("Fin de repeticiones. (Enter=otra vez; 'N'/'A' para cambiar)"));
      return;
    } else if ((blen==1) && (buf[0]=='N' || buf[0]=='n')) {
      g_repStage = RS_NEED_N;
      Serial.println(F("Ingresa N (repeticiones) y presiona Enter."));
      return;
    } else if ((blen==1) && (buf[0]=='A' || buf[0]=='a')) {
      g_repStage = RS_NEED_ANGLES;
      Serial.println(F("Ingresa A1 A2 A3 (grados) y presiona Enter."));
      return;
    } else {
      Serial.println(F("READY: Enter ejecuta, 'N' cambia N, 'A' cambia ángulos."));
      return;
    }
  }
  Serial.println(F("Modo 2: estado inválido. Escribe M2 para reiniciar el asistente."));
}
```

```
/* ================== Modo 3 (Puntos: P1..P4) ==================
   - enterMode3: si P1..P4 están definidos, queda READY, de lo contrario
     solicita cada punto secuencialmente.
   - handleMode3Line: en READY, Enter ejecuta P1->0, P2->0, P3->0, P4->0
     con pausa REP_PAUSE_MS entre puntos.
   ============================================================ */
void enterMode3() {
  g_mode = MODE_POINTS;
  bool allset = g_P_set[0] && g_P_set[1] && g_P_set[2] && g_P_set[3];
  if (allset) {
    g_pStage = PS_READY;
    Serial.println(F("MODO 3: Puntos (P1..P4 listos). Enter ejecuta secuencia. Usa P1/P2/P3/P4 <A1 A2 A3>
para cambiar."));
  } else {
    g_pStage = PS_NEED_P1;
    Serial.println(F("MODO 3: Define P1 <A1 A2 A3> y presiona Enter."));
  }
}
void handleMode3Line() {
  // Función lambda para parsear una línea como Pn <A1 A2 A3>
  auto parsePoseLine = [&](int idx)->bool{
    float a1,a2,a3;
    if (!parse3floats(buf,a1,a2,a3)) return false;
    g_P[idx].A1 = clampAngleForMotor(1,a1);
    g_P[idx].A2 = clampAngleForMotor(2,a2);
    g_P[idx].A3 = clampAngleForMotor(3,a3);
    g_P_set[idx] = true;
    return true;
  };

  // Carga guiada de P1..P4
  if (g_pStage == PS_NEED_P1) {
    if (parsePoseLine(0)) { Serial.println(F("P1 ok. Ahora P2 <A1 A2 A3> y Enter.")); g_pStage =
PS_NEED_P2; }
    else Serial.println(F("Formato inválido. Ej: 20 10 -5"));
    return;
  }
  if (g_pStage == PS_NEED_P2) {
    if (parsePoseLine(1)) { Serial.println(F("P2 ok. Ahora P3 <A1 A2 A3> y Enter.")); g_pStage =
PS_NEED_P3; }
    else Serial.println(F("Formato inválido. Ej: 30 12 -8"));
    return;
  }
  if (g_pStage == PS_NEED_P3) {
    if (parsePoseLine(2)) { Serial.println(F("P3 ok. Ahora P4 <A1 A2 A3> y Enter.")); g_pStage =
PS_NEED_P4; }
    else Serial.println(F("Formato inválido. Ej: 40 15 -10"));
    return;
  }
  if (g_pStage == PS_NEED_P4) {
    if (parsePoseLine(3)) {
      g_pStage = PS_READY;
      Serial.println(F("P4 ok. READY. Enter ejecuta P1->0, P2->0, P3->0, P4->0 (pausa RP)."));
    } else Serial.println(F("Formato inválido. Ej: 50 18 -12"));
    return;
  }
```

```
  // Ejecución en READY con Enter
  if (g_pStage == PS_READY) {
    if (blen==0) {
      if (g_ena != ENA_LEVEL_ENABLE) { Serial.println(F("ENA=1 (OFF). Usa 'E0' para habilitar."));
return; }
      Serial.println(F("Ejecutando secuencia P1..P4:"));
      for (int i=0;i<4;i++){
        Serial.print(F("  P")); Serial.print(i+1); Serial.print(F(" -> 0  | A1/A2/A3="));
        Serial.print(g_P[i].A1); Serial.print(F("/"));
        Serial.print(g_P[i].A2); Serial.print(F("/"));
        Serial.println(g_P[i].A3);
        runCycleSyncAngles(g_P[i].A1, g_P[i].A2, g_P[i].A3);
        if (g_stop) { Serial.println(F(">>> PARO durante Modo 3.")); return; }
        if (i<3) {
          if (!delayWithPoll(REP_PAUSE_MS)) { Serial.println(F("Abortado en pausa entre puntos."));
return; }
        }
      }
      Serial.println(F("Fin secuencia Modo 3. (Enter=otra vez; usa P1..P4 para cambiar)"));
      return;
    } else {
      Serial.println(F("READY: Enter ejecuta. Usa P1..P4 <A1 A2 A3> para actualizar o P? para ver
todo."));
      return;
    }
  }

  Serial.println(F("Modo 3: estado inválido. Escribe M3 para reiniciar el asistente."));
}

/* ================== Comandos P1..P4 (globales) ==================
   Permiten:
     - 'P?'        : listar P1..P4.
     - 'Pn?'       : ver pose individual.
     - 'Pn a b c'  : definir pose (A1=a, A2=b, A3=c).
   Pueden usarse en cualquier modo.
   ============================================================== */
bool handlePCommandInline() {
  if (blen==1 && (buf[0]=='P'||buf[0]=='p')) { // Ayuda corta
    Serial.println(F("Uso: P1/P2/P3/P4 <A1 A2 A3>  |  P1?  |  P?"));
    return true;
  }
  if ((blen==2) && (buf[0]=='P'||buf[0]=='p') && buf[1]=='?') {
    printPositions();
    return true;
  }
  if ((blen==3) && (buf[0]=='P'||buf[0]=='p') && (buf[1]>='1'&&buf[1]<='4') && buf[2]=='?') {
    int idx = buf[1]-'1';
    Serial.print(F("P")); Serial.print(idx+1); Serial.print(F(" = "));
    Serial.print(g_P[idx].A1); Serial.print('/');
    Serial.print(g_P[idx].A2); Serial.print('/');
    Serial.println(g_P[idx].A3);
    return true;
  }
  if ((blen>=2) && (buf[0]=='P'||buf[0]=='p') && (buf[1]>='1'&&buf[1]<='4')) {
    float a1,a2,a3;
    if (parse3floats(buf+2, a1,a2,a3)) {
      int idx = buf[1]-'1';
      g_P[idx].A1 = clampAngleForMotor(1,a1);
      g_P[idx].A2 = clampAngleForMotor(2,a2);
      g_P[idx].A3 = clampAngleForMotor(3,a3);
      g_P_set[idx] = true;
      Serial.print(F("P")); Serial.print(idx+1); Serial.print(F(" actualizado a "));
      Serial.print(g_P[idx].A1); Serial.print('/');
      Serial.print(g_P[idx].A2); Serial.print('/');
      Serial.println(g_P[idx].A3);
    } else {
      Serial.println(F("Uso: Pn <A1 A2 A3>   (ej. P1 20 10 -5)"));
    }
    return true;
  }
  return false;
}
```

```
/* ================== Procesamiento de línea de comandos ==================
   Orden de evaluación:
     - Paro ('S'), cambio de modo ('M1/M2/M3'), ENA ('E/E0/E1').
     - Velocidad ('V12', 'V3'), tiempos ('D3', 'H3', 'RP'), compensación ('AC').
     - Ángulos (A1/A2/A3) en Modo 1, reducciones (R1/R2/R3).
     - P1..P4 (en cualquier modo).
     - Estado ('?') y ayuda ('h').
     - Enter: ejecuta según modo (Single/Repeat/Points).
   Si el comando no coincide, muestra lista resumida.
   ====================================================================== */
void processLine() {
  buf[blen] = '\0';

  // Paro de emergencia
  if ((blen==1) && (buf[0]=='S'||buf[0]=='s')) {
    g_stop = true; setENA(ENA_LEVEL_DISABLE);
    Serial.println(F(">>> PARO: ENA=1, movimiento abortado."));
    blen = 0; return;
  }

  // Cambiar modo
  if ((blen==2) && (buf[0]=='M'||buf[0]=='m') && (buf[1]=='1')) {
    g_mode = MODE_SINGLE; g_repStage = RS_IDLE; g_pStage = PS_IDLE;
    Serial.println(F("Modo 1 (Single). Enter ejecuta un ciclo."));
    blen=0; return;
  }
  if ((blen==2) && (buf[0]=='M'||buf[0]=='m') && (buf[1]=='2')) { enterMode2(); blen=0; return; }
  if ((blen==2) && (buf[0]=='M'||buf[0]=='m') && (buf[1]=='3')) { enterMode3(); blen=0; return; }

  // ENA (toggle/ON/OFF)
  if ((blen==1) && (buf[0]=='E'||buf[0]=='e')) { toggleENA(); blen=0; return; }
  if ((blen==2) && (buf[0]=='E'||buf[0]=='e') && (buf[1]=='0'||buf[1]=='1')) {
    setENA((buf[1]=='0') ? ENA_LEVEL_ENABLE : ENA_LEVEL_DISABLE); blen=0; return;
  }

  // Velocidades (V12/V3)
  if ((blen>=4) && (buf[0]=='V'||buf[0]=='v')) {
    if ((buf[1]=='1' && buf[2]=='2') || buf[1]=='3') {
      uint32_t v32;
      const char* p = (buf[1]=='3') ? buf+2 : buf+3;
      if (parseUIntAfter_u32(p, v32)) {
        if (v32 < 50) v32 = 50; if (v32 > 10000) v32 = 10000;
        if (buf[1]=='3') { g_PULSE3_US  = (unsigned int)v32;  Serial.print(F("Vel M3 (V3) = "));
 Serial.print(g_PULSE3_US);  Serial.println(F(" us")); }
        else             { g_PULSE12_US = (unsigned int)v32;  Serial.print(F("Vel M1/M2 (V12) = "));
Serial.print(g_PULSE12_US); Serial.println(F(" us")); }
      } else {
        Serial.println(F("Uso: V12 <us>  |  V3 <us>"));
      }
      blen=0; return;
    }
  }

  // Tiempos (D3, RP)
  if ((blen>=2) && (buf[0]=='D'||buf[0]=='d') && (buf[1]=='3')) {
    uint16_t ms;
    if (parseUIntAfter_u16(buf+2, ms)) {
      PRE_M3_DELAY_MS = ms;
      Serial.print(F("PreM3 delay = ")); Serial.print(PRE_M3_DELAY_MS); Serial.println(F(" ms"));
    } else {
      Serial.println(F("Uso: D3 <ms>"));
    }
    blen=0; return;
  }
  if ((blen>=2) && (buf[0]=='R'||buf[0]=='r') && (buf[1]=='P')) {
    uint16_t ms;
    if (parseUIntAfter_u16(buf+2, ms)) {
      REP_PAUSE_MS = ms;
      Serial.print(F("Pausa (RP) = ")); Serial.print(REP_PAUSE_MS); Serial.println(F(" ms"));
    } else {
      Serial.println(F("Uso: RP <ms>"));
    }
    blen=0; return;
  }
```

```
  // Hold M3 (H3)
  if ((blen>=2) && (buf[0]=='H'||buf[0]=='h') && (buf[1]=='3')) {
    if (blen==3 && buf[2]=='?') {
      Serial.print(F("HoldM3(ms) = ")); Serial.println(M3_HOLD_MS);
      blen=0; return;
    }
    uint16_t ms;
    if (parseUIntAfter_u16(buf+2, ms)) {
      M3_HOLD_MS = ms;
      Serial.print(F("HoldM3 = ")); Serial.print(M3_HOLD_MS); Serial.println(F(" ms"));
    } else {
      Serial.println(F("Uso: H3 <ms>   (ej. H3 1200)   o H3?"));
    }
    blen=0; return;
  }

  // Compensación M3 (AC)
  if ((blen>=2) && (buf[0]=='A' || buf[0]=='a') && (buf[1]=='C' || buf[1]=='c')) {
    if (blen==2) {
      Serial.print(F("CompM3(deg/rep) actual = ")); Serial.println(g_M3CompDeg,4);
      Serial.println(F("Uso: AC <deg>   (ej. AC 1.0)   o AC?"));
      blen=0; return;
    }
    if (blen==3 && buf[2]=='?') {
      Serial.print(F("CompM3(deg/rep) = ")); Serial.println(g_M3CompDeg,4);
      blen=0; return;
    }
    float v;
    if (parseFloatAfter(buf+2, v)) {
      g_M3CompDeg = v;
      Serial.print(F("CompM3(deg/rep) actualizado a ")); Serial.println(g_M3CompDeg,4);
    } else {
      Serial.println(F("Uso: AC <deg>   (ej. AC 1.0)   o AC?"));
    }
    blen=0; return;
  }

  // Ángulos (A1/A2/A3) sólo en Modo 1
  if ((blen>=2) && (buf[0]=='A'||buf[0]=='a')) {
    if (g_mode!=MODE_SINGLE) { Serial.println(F("Estás en Modo 2/3. Usa el asistente o cambia a M1."));
blen=0; return; }
    if (buf[1]>='1' && buf[1]<='3') {
      float v;
      if (parseFloatAfter(buf+2, v)) {
        uint8_t id = buf[1]-'0';
        if      (id==1){ g_A1 = clampAngleForMotor(1,v); Serial.print(F("M1: A1=")); Serial.print(g_A1);
Serial.println(F("° (0..600)")); }
        else if (id==2){ g_A2 = clampAngleForMotor(2,v); Serial.print(F("M2: A2=")); Serial.print(g_A2);
Serial.println(F("° (0..850)")); }
        else           { g_A3 = clampAngleForMotor(3,v); Serial.print(F("M3: A3=")); Serial.print(g_A3);
Serial.println(F("° (-6800..1700)")); }
      } else {
        Serial.println(F("Uso: A1/A2/A3 <deg>"));
      }
      blen=0; return;
    }
  }

  // Reducciones (R1/R2/R3)
  if ((blen>=2) && (buf[0]=='R'||buf[0]=='r')) {
    if (buf[1]>='1' && buf[1]<='3') {
      float v;
      if (parseFloatAfter(buf+2, v) && v > 0.f) {
        uint8_t id = buf[1]-'0';
        if      (id==1){ g_R1 = v; Serial.print(F("M1: R1=")); Serial.println(g_R1,6); }
        else if (id==2){ g_R2 = v; Serial.print(F("M2: R2=")); Serial.println(g_R2,6); }
        else           { g_R3 = v; Serial.print(F("M3: R3=")); Serial.println(g_R3,6); }
      } else {
        Serial.println(F("Uso: R1/R2/R3 <ratio> (>0)"));
      }
      blen=0; return;
    }
  }
```

```
  // P1..P4 (siempre disponible)
  if (handlePCommandInline()) { blen=0; return; }

  // Estado / Ayuda
  if ((blen==1) && (buf[0]=='?' ))             { printStatus(); blen=0; return; }
  if ((blen==1) && (buf[0]=='H'||buf[0]=='h'))  { help();        blen=0; return; }

  // Enter: ejecuta según modo
  if (blen==0) {
    if (g_mode==MODE_SINGLE)      { runCycleSync(); return; }
    else if (g_mode==MODE_REPEAT) { handleMode2Line(); return; }
    else if (g_mode==MODE_POINTS) { handleMode3Line(); return; }
  }

  // Si nada coincidió, ayuda resumida
  if (g_mode==MODE_REPEAT) { handleMode2Line(); blen=0; return; }
  if (g_mode==MODE_POINTS) { handleMode3Line(); blen=0; return; }

  Serial.println(F("Cmd inválido. Usa M1/M2/M3, Enter, A1/A2/A3, R1/R2/R3, V12/V3, D3, H3, RP, AC,
P1..P4, E/E0/E1, S, ?, h"));
  blen = 0;
}

/* ================== Setup ==================
   - Inicializa Serial a 115200 bps.
   - Configura salidas digitales del Portenta MC (J6) en LOW.
   - Deshabilita ENA por seguridad (drivers OFF).
   - Imprime banner, ayuda y estado inicial.
   ========================================== */
void setup() {
  Serial.begin(115200);
  while (!Serial) {}

  MachineControl_DigitalOutputs.begin(true);
  MachineControl_DigitalOutputs.writeAll(0);

  MachineControl_DigitalOutputs.write(STEP1_CH, LOW);
  MachineControl_DigitalOutputs.write(DIR1_CH,  LOW);
  MachineControl_DigitalOutputs.write(STEP2_CH, LOW);
  MachineControl_DigitalOutputs.write(DIR2_CH,  LOW);
  MachineControl_DigitalOutputs.write(STEP3_CH, LOW);
  MachineControl_DigitalOutputs.write(DIR3_CH,  LOW);

  setENA(ENA_LEVEL_DISABLE);  // Inicio seguro (drivers deshabilitados)

  Serial.println(F("SCARA Sync v6+: modos 1/2/3, P1..P4 con RP, pre-M3 (D3), hold M3 (H3), AC Modo2, paro
S, ENA por comandos."));
  help();
  printStatus();
}

/* ================== Loop principal ==================
   Lector de consola:
     - Acumula caracteres en 'buf' hasta CR/LF.
     - Al recibir CR/LF, llama a processLine() y limpia el buffer.
   ================================================== */
void loop() {
  while (Serial.available()) {
    char c = Serial.read();
    if (c=='\r' || c=='\n') processLine();
    else if (blen < sizeof(buf)-1) buf[blen++] = c;
  }
}
```

Anexo B. Código del brazo SCARA

XII. GLOSARIO

- **Acople rígido:** dispositivo mecánico que conecta dos ejes coaxiales transmitiendo movimiento sin permitir flexión o desalineación.
- **Arduino:** plataforma de hardware y software abierto para prototipado electrónico, basada en microcontroladores programables.
- **Arduino Portenta pro *machine control*:** placa de desarrollo de la familia Arduino Portenta, diseñada para proyectos avanzados de IoT, inteligencia artificial y aplicaciones industriales.
- **Autodesk Inventor:** software CAD para modelado 3D paramétrico, simulación y documentación de diseños mecánicos.
- **Brazo SCARA:** tipo de robot de 4 grados de libertad, con movimiento *Selective Compliance Assembly Robot Arm*, utilizado en ensamblaje y automatización.
- **CAD:** (*Computer-Aided Design*), herramienta digital para diseñar, modelar y documentar piezas en 2D y 3D.
- **Cura:** software *slicer* que convierte modelos 3D en instrucciones G-code para impresoras 3D.
- **Delay:** pausa programada antes/entre acciones. En tu código hay bloqueo (*delay*) y espera no bloqueante (*delayWithPoll*).
- **Deriva:** desplazamiento lento e indeseado de la posición *HOME* por fricción, holguras o flexión; se compensa con un *offset*.
- **Ender 3:** impresora 3D de bajo costo y código abierto, ampliamente usada en prototipado y fabricación aditiva.
- **Estilo Bresenham/DDA:** algoritmo con acumulador entero que reparte pasos proporcionalmente entre ejes para un movimiento sincronizado.
- ***Horn* (servo):** brazo que se atornilla al *spline* para transmitir el giro.
- ***Infill*:** porcentaje de material con el que se rellena el interior de una pieza impresa en 3D. Define la estructura interna (patrones como rejilla, panal, etc.) y afecta el peso, la rigidez y el tiempo de impresión.
- ***INPUT_PULLUP*:** modo de pin con resistencia interna a 5 V; el pin está en *HIGH* y al pulsar a GND lee *LOW* (evita estados flotantes).
- **MATLAB:** software de cálculo numérico y programación especializado en simulación, análisis de datos y modelado matemático.
- ***moveFixedSingle*:** mueve un eje una cantidad fija de pasos a una velocidad dada, vigilando el paro y la consola.
- ***movePairSync*:** mueve M1 y M2 coordinados para que terminen juntos, calculando cuándo dar cada paso.

- **Motor servo:** motor con sistema de control integrado que permite posicionamiento angular preciso dentro de un rango determinado.
- **Motor Stepper:** motor eléctrico que se mueve en pasos discretos y precisos, ideal para aplicaciones de control de posición.
- ***Offset*:** desplazamiento o diferencia establecida entre una posición, valor o punto de referencia y su valor real, utilizada para corregir o ajustar mediciones, coordenadas o movimientos en un sistema.
- **Par (torque):** capacidad de giro ($\tau = F \cdot r$); más par = más fuerza en la garra. Unidades: N·m o kg·cm.
- ***Spline* (servo):** eje dentado del servo.