UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Actualización de los controladores de un robot antropomórfico para aplicaciones científicas e industriales

TESIS

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A N

Edgar Eloy de la Cruz Martínez

José Santiago Pérez Rizada

DIRECTOR DE TESIS

Dr. Roberto Giovanni Ramírez Chavarría

Ciudad Universitaria, Cd. Mx., 2026

Dedicatoria

A mi madre Isidra Martínez Vázquez por ser mi guía constante, por impulsarme en cada paso y motivarme a seguir adelante en el camino del estudio. Su apoyo incondicional y su ejemplo de fortaleza han sido fundamentales para construir este logro.

A mi hermano Gerardo G. De La Cruz Martínez porque sé que siempre puedo contar con él. Su respaldo y confianza me han dado la seguridad necesaria para continuar, recordándome que nunca estoy solo en este recorrido.

A mis amigos han sido parte fundamental de mi formación integral como persona. Su apoyo, compañía y confianza me permitieron ser auténtico en todo momento, convirtiéndose en mis confidentes y en un pilar que enriqueció este camino.

Edgar Eloy de la Cruz Martínez

Dedicatoria

A Dios, por la vida, la fortaleza y la sabiduría necesarias para culminar este proyecto, y por poner en mi camino a mi familia y a cada una de las personas que han sido apoyo fundamental en mi formación.

A mi madre, María Isabel Rizada Antel, por su amor y su confianza incondicional desde siempre. Gracias por cada consejo, por la fortaleza transmitida en los momentos difíciles y por ser el pilar fundamental que ha sostenido cada uno de mis logros.

A mi padre, José Pérez Rodríguez, por su apoyo constante y su ejemplo de esfuerzo y responsabilidad. Gracias por acompañar cada etapa de mi formación y por contribuir de manera significativa a la culminación de este logro.

A mi hermana Isabel y a mi hermano Isaac, por su compañía y fraternidad a lo largo de todos estos años.

A todos mis amigos, por su apoyo, motivación y compañía a lo largo de esta etapa universitaria.

A la memoria de mi tía, la ingeniera Uljana Rizada Antel, cuyo cariño y confianza en mi permanecen en mi mente.

José Santiago Pérez Rizada

Agradecimientos

Agradecemos al M.I. Daniel Martínez Gutiérrez, profesor, tutor y amigo, por su invaluable orientación académica, disposición constante y motivación permanente que fueron fundamentales para la culminación de nuestra carrera y en el desarrollo de esta tesis. Su acompañamiento profesional y humano deja una huella significativa en nuestra formación.

Agradecemos al Dr. Roberto Giovanni Ramírez Chavarría, por su valiosa labor como tutor de esta tesis. Su orientación y dedicación en la supervisión del presente trabajo contribuyeron de manera determinante a su consolidación.

Se extiende un distinguido agradecimiento al M.I. Antonio Salvá Calleja, cuyo compromiso, interés y apoyo técnico constante resultaron determinantes en el desarrollo del presente trabajo. Su disposición incondicional y acompañamiento oportuno representaron un apoyo esencial para la consolidación de esta tesis.

Agradecemos a la Universidad Nacional Autónoma de México, la Facultad de Ingeniería, al Instituto de Ingeniería y al Departamento de Control y Robótica, por la sólida formación académica recibida, el acceso a infraestructura especializada, el acompañamiento docente y el impulso constante al desarrollo científico y tecnológico.

Agradecemos al M.I. Eduardo Bernal Emmanuel Domínguez y Tulio Torres por su valiosa participación y el acompañamiento técnico brindado durante el desarrollo de esta tesis. Su apoyo y disposición contribuyeron de manera significativa a la resolución de aspectos clave del trabajo.

Esta tesis fue realizada gracias al apoyo del programa UNAM-DGAPA-PAPIME en el proyecto PE100725 y PE102626.

Índice general

Índice de figuras

Índice de tablas

Lista de abreviaturas

ADC – *Analog-to-Digital Converter*: Convertidor analógico–digital.

CAD – *Computer-Aided Design*: Diseño asistido por computadora.

CPU – *Central Processing Unit*: Unidad central de procesamiento.

CW – *Clockwise*: Sentido horario.

CCW – *Counter-Clockwise*: Sentido antihorario.

DAC – *Digital-to-Analog Converter*: Convertidor digital–analógico.

GPIO – *General Purpose Input/Output*: Entradas/salidas digitales de propósito general.

GUI – *Graphical User Interface*: Interfaz gráfica de usuario.

H-Bridge – Puente H: Etapa electrónica para el control bidireccional de motores.

HMI – *Human-Machine Interface*: Interfaz hombre–máquina.

IDE – *Integrated Development Environment*: Entorno de desarrollo integrado.

IFR – *International Federation of Robotics*: Federación Internacional de Robótica.

Ki – *Integral Gain*: Ganancia integral del controlador PID.

Kd – *Derivative Gain*: Ganancia derivativa del controlador PID.

Kp – *Proportional Gain*: Ganancia proporcional del controlador PID.

LDO – *Low Dropout Regulator*: Regulador lineal de baja caída.

LED – *Light Emitting Diode*: Diodo emisor de luz.

MCU – *Microcontroller Unit*: Microcontrolador.

MINICON_STF4B® – Tarjeta de desarrollo de la FI–UNAM.

PID – *Proportional–Integral–Derivative Controller*: Controlador proporcional–integral–derivativo.

PWM – *Pulse Width Modulation*: Modulación por ancho de pulso.

ROS – *Robot Operating System*.

SP – *Setpoint*: Valor de referencia.

SYSCLK – *System Clock*: Reloj del sistema.

UART – *Universal Asynchronous Receiver–Transmitter*.

URDF – *Universal Robot Description Format*.

USB – *Universal Serial Bus.*

VREF – *Voltage Reference*: Voltaje de referencia del ADC.

Resumen

El presente trabajo desarrolla una solución integral para recuperar, modernizar y extender la funcionalidad del robot antropomórfico MENTOR, equipo perteneciente al Laboratorio de Control y Robótica de la Facultad de Ingeniería de la UNAM. Debido a la obsolescencia total de su controlador propietario, su incompatibilidad con el software actual y la ausencia de soporte técnico del fabricante original, el robot se encontraba inoperante desde hace varios años, afectando directamente la ejecución de prácticas de la materia de Robótica Industrial. Para resolver esta problemática se diseñó e implementó un entorno de operación y programación basado en: la actualización de los componentes electrónicos, establecer una comunicación estandarizada y desarrollar herramientas actuales de simulación y visualización. Inicialmente se revisaron los componentes mecánicos evaluando su estado y las posibles restricciones de operación. Se determinaron las características eléctricas para la fuente alimentación, se revisó el buen estado de los sensores de posición y la distribución del cableado. Se selecciono un MCU que permitió, a nivel de firmware, la programación en lenguaje C bajo una arquitectura modular y configurar parámetros de control en tiempo real desde MATLAB. Paralelamente, se construyó una representación virtual del robot a partir de su modelo CAD, generando un archivo URDF que describe su estructura mecánica y cinemática. Con este modelo se obtuvieron las transformaciones homogéneas, la cinemática directa y el espacio de trabajo, validando que la representación virtual coincida con el robot físico. Como resultado esta representación virtual permite diseñar, simular y ejecutar trayectorias en el robot. Finalmente, se desarrollaron aplicaciones gráficas con MATLAB App Designer para la operación manual, monitoreo cinemático en tiempo real y seguimiento de trayectorias. El trabajo concluye con la integración de todos los elementos en un sistema funcional, robusto y completamente documentado, que devuelve al robot MENTOR su operatividad y habilita la creación de nuevas prácticas de laboratorio alineadas con el plan de estudios vigente.

Abstract

This work develops a comprehensive solution to recover, modernize, and extend the functionality of the anthropomorphic MENTOR robot, a system belonging to the Robotics Laboratory of the School of Engineering at UNAM. Due to the complete obsolescence of its proprietary controller, its incompatibility with current software, and the absence of technical support from the original manufacturer, the robot had been inoperative for several years, directly affecting the execution of laboratory practices for the Industrial Robotics course. To address this issue, a completely new control system was designed and implemented, based on modern hardware, standardized communication, and current simulation and visualization tools.

The development began with an exhaustive exploration of the existing hardware, evaluating mechanical constraints, electrical characteristics, position sensors, and cable distribution. Based on these findings, a new electronic circuit was implemented to fully replace the original controller. At the firmware level, C-language code was developed under a modular architecture, enabling real-time configuration of control parameters from MATLAB.

In parallel, a digital twin of the robot was constructed from its CAD model, generating a URDF file that describes its kinematic structure. Using this model, homogeneous transformations, forward kinematics, and the workspace were obtained, validating that the virtual representation matches the physical robot. The digital twin also made it possible to design and simulate trajectories before executing them on the real system. Finally, graphical applications were developed in MATLAB App Designer for manual control, real-time kinematic monitoring, and trajectory tracking.

This work concludes with the integration of all components into a functional, robust, and fully documented system that restores the MENTOR robot's operability and enables the creation of new laboratory practices aligned with the current study plan.

1. Introducción

La decisión de desarrollar esta tesis estuvo motivada por plasmar el aprendizaje teórico y materializar los conocimientos adquiridos en el área de robótica industrial para desarrollo de actividades de enseñanza en la Facultad de Ingenieria. Esto debeido a un creciente interes en el área impulsado por el avance de la electrónica embebida, los sistemas de control digital, las herramientas de simulación y los entornos de desarrollo de software. En el ámbito universitario, los robots manipuladores constituyen una herramienta fundamental para la enseñanza de asignaturas relacionadas con control, robótica, y electrónica, ya que permiten vincular de manera directa la teoría con la experimentación práctica.

Sin embargo, muchos de los sistemas robóticos disponibles en laboratorios académicos fueron desarrollados bajo arquitecturas de hardware y software que hoy se encuentran obsoletas, con controladores propietarios, interfaces incompatibles con sistemas operativos actuales y una limitada capacidad de expansión. Esta situación reduce significativamente su aprovechamiento didáctico, en muchos casos, conduce al abandono de equipos que aún poseen un alto valor mecánico y estructural.

El robot antropomórfico MENTOR, perteneciente al Laboratorio de Control y Robótica de la Facultad de Ingeniería de la UNAM, representa un ejemplo claro de esta problemática. A pesar de contar con una estructura mecánica funcional y adecuada para fines académicos, su controlador original dejó de ser operativo debido a la obsolescencia tecnológica, la falta de soporte técnico y la imposibilidad de integrarlo con MATLAB. Como consecuencia, el robot permaneció fuera de servicio durante varios años, imposibilitando el desarrollo de prácticas en asignaturas relacionadas con robótica industrial y control.

Ante este escenario, la presente tesis propone la actualización de la estrategia de control para cada articulación del robot MENTOR mediante la integración de un MCU encargado del control y comunicación, que permite la implementación arquitectura modular. El proyecto contempla la sustitución total del controlador electrónico original por un sistema embebido, conocido como MINICON_STF4B desarrollado en el Departamento de Control

y Robótica (basado en el microcontrolador STM32F446), en que se implementaron los algoritmos de control necesarios, y la programación de una interfaz que contiene herramientas de supervisión y operación a través de MATLAB.

Adicionalmente, se desarrollo una representación virtual del robot mediante un modelo URDF, el cual permite validar la cinemática del sistema, simular trayectorias y analizar su espacio de trabajo, fortaleciendo la relación entre el sistema físico y su representación virtual. Esta integración hardware–software posibilita no solo la recuperación operativa del robot, sino también su adaptación a nuevos esquemas de enseñanza acordes con el plan de estudios vigente.

El trabajo presentado no se limita a la restauración funcional del robot, sino que busca sentar las bases para su uso continuo como plataforma didáctica y experimental, promoviendo la reutilización de infraestructura existente, el uso de tecnología desarrollada en la propia Facultad de Ingeniería.

1.1. Objetivos

1.1.1. Objetivo general

Diseñar, implementar y validar un sistema de control electrónico, de comunicación y de software para el robot antropomórfico MENTOR, empleando la tarjeta de desarrollo MINICON_STF4B y MATLAB, con el fin de restaurar su operatividad y habilitar su uso para actividades de enseñanza practica.

1.1.2. Objetivos específicos

1. Diseñar e implementar un controlador electrónico que sustituya al controlador electrónico original del robot.

2. Implementar un sistema de control de posición basado en PID para cada articulación del robot.

3. Realizar la sintonización de cada controlador PID validando su desempeño mediante pruebas experimentales.

4. Construir una representación virtual del robot MENTOR mediante un modelo URDF, validar su cinemática directa con transformaciones homogéneas y emplearlo para determinar su espacio de trabajo y simular movimientos.

5. Elaborar una propuesta de práctica de laboratorio para la asignatura de Robótica Industrial, en concordancia con el contenido del temario de la asignatura para validar su uso con estudiantes.

La estructura de la tesis consta de once capítulos organizados de la siguiente manera: Introducción donde se planteó el contexto del problema, los objetivos y el alcance del trabajo, del Capítulo 2 al Capítulo 8 se desarrolla la investigación, incluyendo el marco teórico, así como las etapas de diseño, implementación y desarrollo del sistema propuesto; los Capítulos 9 y 10 presentan los resultados obtenidos y su análisis, evaluando el desempeño y la validación de las estrategias implementadas; finalmente, el Capítulo 11 expone las conclusiones del estudio, resaltando los aportes, limitaciones y posibles líneas de trabajo futuro.

2. Marco teórico

En este capítulo se presentan definiciones y conceptos básicos que se utilizarán a lo largo de este trabajo y son indispensables para comprender su contenido.

2.1. Conceptos generales de la robótica

A la definición de robot, término cuya delimitación resulta compleja, debido a que cada arquitectura posee características únicas; se añaden calificativos que permiten acotar la palabra y describir con mayor precisión sus particularidades. Por ejemplo, el adjetivo humanoide indica que se trata de un dispositivo cuya configuración emula a la morfología humana.

Por lo tanto, la definición que en este trabajo se adaptó es la de robot manipulador industrial, ya que se asemeja en cuanto a configuración y arquitectura al robot MENTOR. Según Barrientos et al. (2012), "Un robot industrial es un manipulador multifuncional reprogramable, capaz de mover materias, piezas, herramientas o dispositivos especiales, según trayectorias variables, programadas para realizar tareas diversas".

Otro concepto relevante es la configuración del robot, un parámetro de clasificación que depende de la forma y la estructura de la máquina. Existen diversas configuraciones; en este trabajo se atenderá únicamente la configuración antropomórfica, ya que el robot MENTOR presenta esta estructura. Diversos autores definen al robot antropomórfico como un manipulador cuya disposición de articulaciones rotacionales imita la anatomía del brazo

humano.

2.1.1. Tipos de articulaciones

En la literatura se utilizan indistintamente los términos junta, articulación o enlace; esto depende del autor o el texto que se está revisando. Sin embargo, se puede asumir que son sinónimos. La definición más conocida de este componente dice que "se trata de un elemento que une dos eslabones permitiendo el movimiento relativo entre cada dos eslabones consecutivos" (Barrientos et al., 2012, p. 31). Cada articulación puede provocar un movimiento de desplazamiento o giro, y cada uno de estos movimientos independientes que realiza la articulación con respecto a la anterior se denomina *grado de libertad*.

Existen varios tipos de articulaciones o juntas, aunque la mayor parte de los robots manipuladores cuentan con dos tipos: articulaciones de revolución y prismática. Según Siciliano et al. (2009), "una unión prismática crea un movimiento traslacional relativo entre los dos enlaces, mientras que una unión de revolución crea un movimiento rotacional relativo entre los dos enlaces". En el caso del robot MENTOR, solo se cuenta con articulaciones de revolución.

2.1.2. Análisis cinemático

Un efector final se define formalmente como el dispositivo periférico situado en el extremo distal de la cadena cinemática de un manipulador robótico. Es el componente encargado de materializar la interacción física entre el sistema de control del robot y su entorno de trabajo para ejecutar una tarea específica.

Partiendo de la definición de Siciliano et al. (2009), “el análisis cinemático de la estructura mecánica de un robot se refiere a la descripción del movimiento con respecto a un marco de referencia cartesiano fijo, ignorando las fuerzas y momentos que provocan el movimiento de la estructura” (p. 30). A partir de este concepto se desprende la cinemática, la cual describe la relación analítica entre la posición de las articulaciones y la posición y orientación del efector final con respecto al marco de referencia fijo. Para su estudio se emplean técnicas de álgebra lineal.

De aqui se concluye que: “el espacio de trabajo representa esa porción del entorno a la que el efector final del manipulador puede acceder. Su forma y volumen dependen de la estructura del manipulador, así como de la presencia de límites mecánicos en las articulaciones”. Conocer este espacio de trabajo es fundamental, ya que determina los puntos en el espacio que son alcanzables por el robot y en donde puede realizar tareas. (Siciliano et al. (2009))

2.2. Cinemática

De este concepto se derivan dos problemas: el de cinemática directa y el de cinemática inversa. La cinemática directa consiste en determinar un método sistemático que describe el movimiento del efector final en función del movimiento de las articulaciones. Por el contrario, la cinemática inversa consiste en transformar un movimiento deseado dentro del espacio de trabajo en un correspondiente movimiento de las articulaciones. Para la descripción de ambas es necesario comprender el concepto de rotación elemental.

2.2.1. Rotaciones elementales

Las rotaciones elementales se utilizan en la robótica porque permiten rotar un marco de referencia con respecto a otro un ángulo determinado alrededor de alguno de los ejes x, y y z como se muestra en la Figura 2.1. Para cada eje existe una matriz de rotación elemental, como se expresa en la ecuación (2.1).

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, \tag{2.1}$$

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}.$$

Donde:

- $R_z(\alpha)$ es la matriz de rotación elemental alrededor del eje z por un ángulo α.
- $R_y(\beta)$ es la matriz de rotación elemental alrededor del eje y por un ángulo β.
- $R_x(\gamma)$ es la matriz de rotación elemental alrededor del eje x por un ángulo γ.
- α, β y γ representan los ángulos de rotación alrededor de los ejes z, y y x, respectivamente.

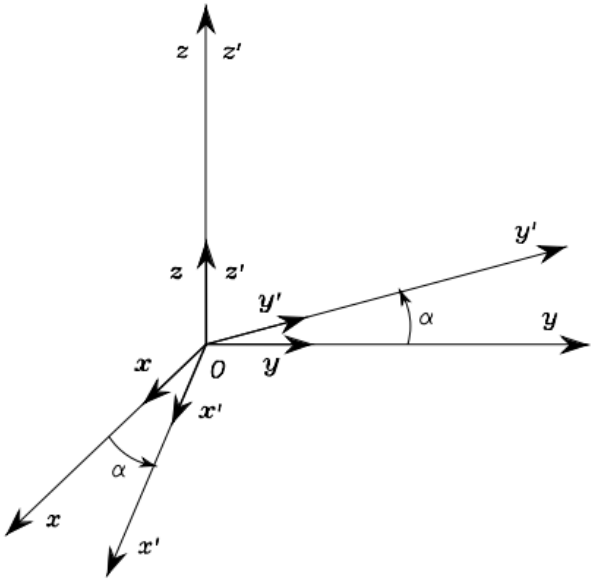


Figura 2.1: Rotación sobre el marco O–xyz un ángulo α alrededor del eje z

Fuente: (Siciliano Bruno et al. (2009)).

2.2.2. Composición de matrices de rotación

Para realizar la composición de matrices rotación del marco de referencia existen dos tipos:

- Rotaciones sobre el marco actual.
- Rotaciones sobre sistema fijo.

En el primero, las rotaciones se realizan sobre el marco de referencia actual; es decir, "cada rotación se realiza con respecto al último sistema coordenado (sistema actual)". La composición de rotaciones sucesivas se obtiene entonces mediante la postmultiplicación de las matrices de rotación siguiendo el orden dado de las rotaciones" (Siciliano et al., 2009). De manera general, para n rotaciones se cumple:

$$^{0}R_{n} = {}^{0}R_{1}\,{}^{1}R_{2}\cdots{}^{n-1}R_{n} \tag{2.2}$$

Donde:

- $^{0}R_{n}$ es la matriz de rotación que describe la orientación del marco n con respecto al marco base 0.
- El superíndice indica el marco de referencia respecto al cual se expresa la rotación, mientras que el subíndice corresponde al marco rotado.
- $^{n-1}R_{n}$ es la matriz de rotación relativa entre el marco n y el marco $(n-1)$.

El segundo enfoque de rotación elemental se denomina rotaciones sobre el sistema fijo; es decir, las rotaciones se realizan con respecto a un marco de referencia fijo. Generalmente, estas rotaciones están definidas con respecto al marco de referencia base. En este caso, la rotación compuesta se obtiene mediante la premultiplicación de las matrices de rotación individuales.

2.2.3. Ángulos de Euler

Las matrices de rotación proporcionan una descripción de la orientación de un marco de referencia. Se ha demostrado que tres parámetros son suficientes para describir la orientación de un cuerpo rígido; este conjunto mínimo pertenece al grupo especial ortogonal de orden tres, cuyos elementos cumplen las siguientes propiedades:

- Son cerrados con respecto a la multiplicación matricial; es decir, al multiplicar dos matrices de rotación siempre se obtiene otra matriz de rotación.

- Se cumple que su determinante es igual a uno, $\det(R) = 1$, para la representación de sistemas coordenados dextrógiros.

- Su matriz inversa es igual a su transpuesta, lo que implica que $R^{-1} = R^T$.

Tomando como base lo anterior, es posible obtener una representación mínima de la orientación utilizando un conjunto de tres ángulos $\Phi = [\phi\ \vartheta\ \psi]^T$. Una matriz de rotación genérica se puede obtener componiendo una secuencia adecuada de tres rotaciones elementales, garantizando que no se realicen dos rotaciones sucesivas sobre ejes paralelos. Existen 12 conjuntos de posibles combinaciones de matrices básicas de rotación que cumplen con lo anterior; de estas, solo interesan dos que son ampliamente usadas en la robótica. La primera combinación se conoce como *Roll–Pitch–Yaw* (ZYX) y la segunda se conoce como ángulos de Euler (ZYZ). Desarrollando la rotacion de Euler (ZYZ) se obtiene la matriz siguiente:

$$R(\Phi) = R_z(\phi)\, R_{y'}(\vartheta)\, R_{z''}(\psi) = \begin{bmatrix} c_\phi c_\vartheta c_\psi - s_\phi s_\psi & -c_\phi c_\vartheta s_\psi - s_\phi c_\psi & c_\phi s_\vartheta \\ s_\phi c_\vartheta c_\psi + c_\phi s_\psi & -s_\phi c_\vartheta s_\psi + c_\phi c_\psi & s_\phi s_\vartheta \\ -s_\vartheta c_\psi & s_\vartheta s_\psi & c_\vartheta \end{bmatrix}, \tag{2.3}$$

Donde:

- $R(\Phi)$ es la matriz de rotación total resultante de la composición de tres giros sucesivos.

- $R_z(\phi)$ representa la primera rotación elemental alrededor del eje z por un ángulo ϕ.

- $R_{y'}(\vartheta)$ representa la segunda rotación elemental alrededor del eje y (ya desplazado) por un ángulo ϑ.

- $R_{z''}(\psi)$ representa la tercera rotación elemental alrededor del eje z final por un ángulo ψ.

- $c_{\text{ángulo}}$ y $s_{\text{ángulo}}$ representan las funciones trigonométricas cos(ángulo) y sin(ángulo) respectivamente.

Para observar de manera gráfica la rotación descrita en la Ecuación 2.3, se representa en la Figura 2.2.

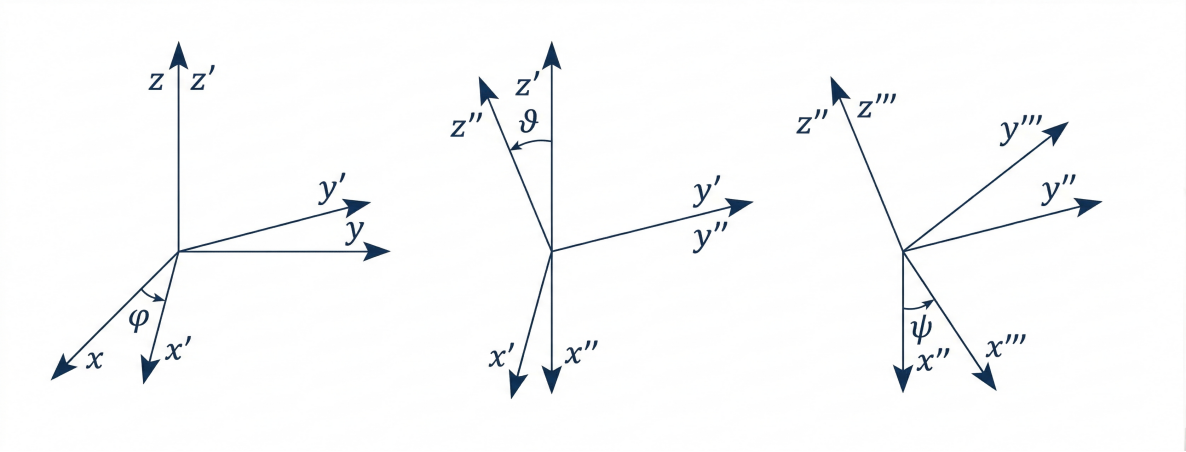


Figura 2.2: Representación de una rotación siguiendo los ángulos de Euler *ZYZ*

Desarrollando la rotacion *Roll–Pitch–Yaw* (ZYX) se obtiene la matriz siguiente:

$$R(\Phi) = R_z(\varphi)\, R_y(\vartheta)\, R_x(\psi) = \begin{bmatrix} c_\varphi c_\vartheta & c_\varphi s_\vartheta s_\psi - s_\varphi c_\psi & c_\varphi s_\vartheta c_\psi + s_\varphi s_\psi \\ s_\varphi c_\vartheta & s_\varphi s_\vartheta s_\psi + c_\varphi c_\psi & s_\varphi s_\vartheta c_\psi - c_\varphi s_\psi \\ -s_\vartheta & c_\vartheta s_\psi & c_\vartheta c_\psi \end{bmatrix}. \tag{2.4}$$

Donde:

- $R(\mathbf{\Phi})$ es la matriz de rotación total.

- $R_z(\phi)$ representa la rotación elemental alrededor del eje z por un ángulo ϕ.
- $R_y(\theta)$ representa la rotación elemental alrededor del eje y por un ángulo θ.
- $R_x(\psi)$ representa la rotación elemental alrededor del eje x por un ángulo ψ.
- $c_{\text{ángulo}}$ y $s_{\text{ángulo}}$ representan las funciones trigonométricas cos(ángulo) y sin(ángulo) respectivamente.

Para ilustrar de manera gráfica la rotación descrita en la Ecuación 2.4, se presenta la Figura 2.3.

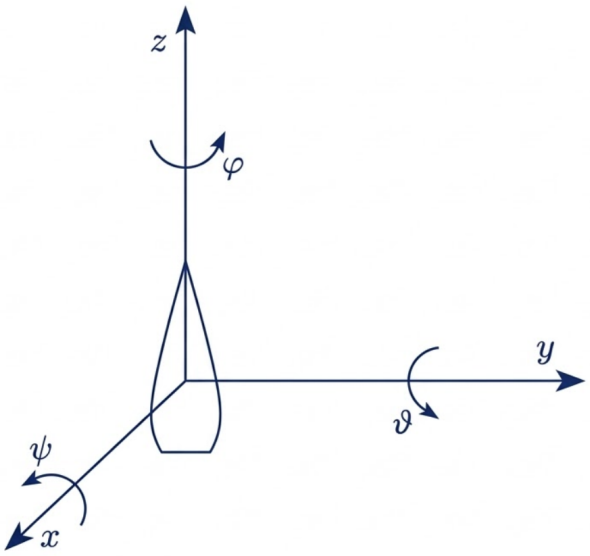


Figura 2.3: Representación de los ángulos de *Roll–Pitch–Yaw* (ZYX)

Las composiciones de rotaciones son útiles en el diseño de la representación virtual, particularmente la configuración *Roll–Pitch–Yaw*. Este enfoque permite describir la orientación y posición del efector final, así como la cinemática en los robots manipuladores.

2.2.4. Transformaciones homogéneas

Como se mencionó, la posición de un cuerpo rígido en el espacio se expresa en términos de la posición de un punto del cuerpo con respecto de un marco de referencia. A esto se le denomina traslación, mientras que la rotación se expresa en términos de las componentes de los vectores unitarios del marco unido al cuerpo con respecto al origen situado en el marco de referencia base. Una transformación homogénea se representa mediante una matriz de dimensiones 4×4.

$$ {}^{a}\mathbf{H}_{b} = \begin{bmatrix} {}^{a}\mathbf{R}_{b} & {}^{a}\mathbf{o}_{b} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{2.5} $$

Donde:

- ${}^{a}\mathbf{R}_{b}$: pertenece a un grupo especial de orden tres y es una matriz de rotación.
- ${}^{a}\mathbf{o}_{b}$: el vector de posición del origen del sistema b con respecto del sistema a.
- $\mathbf{0} = [\,0\ 0\ 0\,]$: una matriz de 1×3 de ceros.

2.3. Control en tiempo discreto

El desarrollo y la amplia disponibilidad de computadoras, microcontroladores y procesadores digitales de señal han impulsado de manera significativa la implementación de sistemas de control en tiempo discreto en diversas áreas de la ingeniería. A diferencia de los sistemas de control clásicos en tiempo continuo, en los cuales las señales se consideran definidas para todo instante de tiempo t, los sistemas de control digitales operan con variables

representadas como secuencias de valores numéricos obtenidos a partir del muestreo de una señal analógica original.

En este contexto, Ogata distingue tres tipos fundamentales de señales: la señal analógica en tiempo continuo, la señal muestreada y la señal digital (Figura 2.4). La señal analógica, representada como $x(t)$, existe para cualquier valor real del tiempo. Cuando esta señal es medida periódicamente mediante un dispositivo de muestreo cada intervalo de tiempo T, se obtiene una secuencia de valores $x(kT)$, con $k = 0, 1, 2, \ldots$, que comúnmente se denota como $x[k]$. Esta secuencia corresponde a una señal en tiempo discreto o señal muestreada.

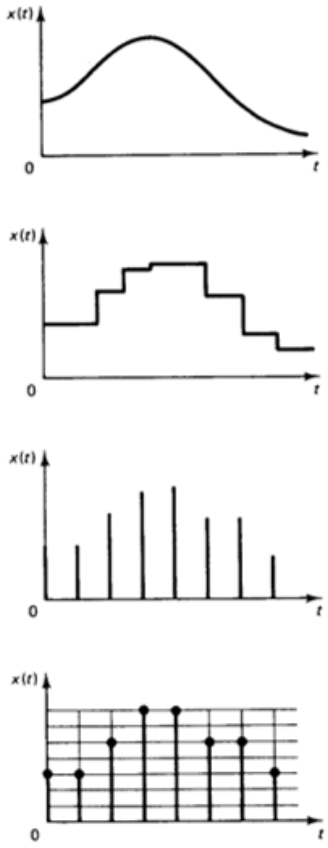


Figura 2.4: Conversión de una señal continua a una señal discreta

En los sistemas digitales reales, además del proceso de muestreo temporal, se lleva a cabo un proceso de cuantificación en amplitud. Durante este proceso, la señal muestreada es procesada por un cuantificador que únicamente puede representar un número finito de niveles de amplitud. En consecuencia, cada muestra es aproximada al nivel discreto más cercano. El resultado final es una señal digital, caracterizada por ser discreta tanto en el dominio del tiempo como en el de la amplitud, lo que permite su procesamiento mediante dispositivos digitales.

2.3.1. Estructura general de un sistema de control digital

Un sistema de control en tiempo discreto está integrado por los siguientes elementos:

1. Planta o proceso físico, de naturaleza continua.

2. Sensor analógico, que mide la variable de salida de la planta.

3. Convertidor analógico–digital (A/D), que transforma la señal analógica del sensor en una secuencia de números mediante muestreo y cuantificación.

4. Controlador digital, implementado en una computadora o microcontrolador, que ejecuta un algoritmo de control utilizando las muestras de entrada y genera valores numéricos de la señal de control.

5. Convertidor digital–analógico (D/A) o una etapa de modulación, que reconstruye una señal analógica a partir de los valores numéricos producidos por el controlador.

6. Retenedor de orden cero, que mantiene constante la señal de salida del D/A durante cada periodo de muestreo.

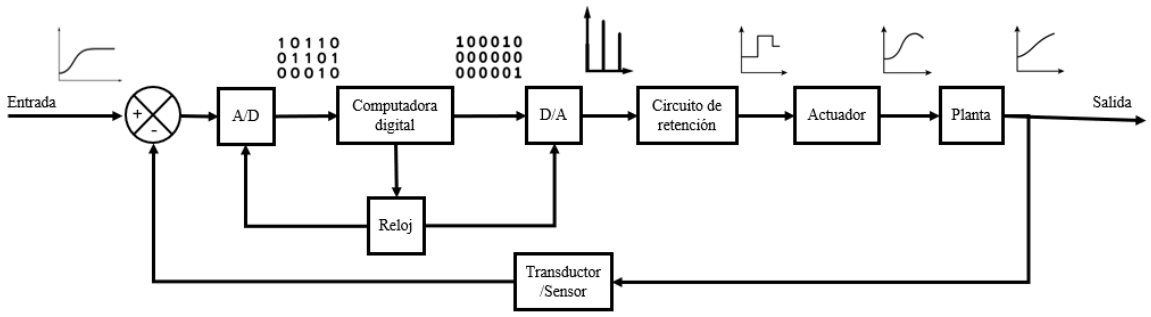


Figura 2.5: Diagrama de bloques de un sistema de control digital, muestra las señales en forma binaria o gráfica

Fuente: Ogata et al. (1996).

En el diagrama de flujo presentado por Ogata en la figura 2.5 esta la estructura general de un sistema de control digital. Una vez digitalizada la información, el microcontrolador procesa estos datos mediante el algoritmo de control implementado. Dicho algoritmo opera exclusivamente con números y, a partir de la señal muestreada, calcula la acción correctiva necesaria para llevar a la planta hacia el comportamiento deseado. El resultado de este procesamiento es una nueva secuencia de números digitales que representan la señal de control en forma discreta. Por último, esta señal digital debe volver a convertirse en una señal física para que pueda actuar sobre la planta. Para ello, la secuencia numérica pasa al convertidor digital analógico, que transforma los valores discretos en niveles de voltaje o corriente. De este modo, se obtiene una señal de tiempo continúo formada por secciones constantes que representan la acción de control aplicada a la planta. Todo este ciclo se sincrónica mediante un reloj interno del sistema digital. Dicho reloj determina el periodo de muestreo y garantiza que las operaciones se ejecuten de manera ordenada. Finalmente, la señal generada puede enviarse directamente a la planta o aplicarse a través de un actuador, según lo requiera el sistema físico. El actuador convierte la señal de control en una acción mecánica, hidráulica, eléctrica o neumática, cerrando así el ciclo del control digital.

Proceso de muestreo

El muestreo es la operación mediante la cual una señal continua se observa en instantes igualmente espaciados. Si T es el periodo de muestreo, las muestras se obtienen en $t = kT$. La elección de T es un aspecto crítico del diseño en tiempo discreto, ya que un valor de T demasiado grande implica una frecuencia de muestreo baja, lo que puede provocar pérdida de información en las dinámicas del sistema e incluso inestabilidad. Por otro lado, un valor de T muy pequeño reduce el error de aproximación al sistema continuo, pero aumenta la carga

computacional y las exigencias del hardware. Ogata et al. discute que el periodo de muestreo debe ser lo suficientemente pequeño para capturar adecuadamente el comportamiento de la planta, pero sin llegar a ser innecesariamente rápido desde un punto de vista práctico Ogata et al. (1996).

Cuantificación

El sistema numérico estándar utilizado para el procesamiento de señales digitales es el sistema binario. En un sistema de codificación numérica, los datos consisten en n pulsos, cada uno de los cuales indica "encendido" o "apagado", como en lógica booleana o en el sistema codificado decimal a binario (BCD). Sin embargo, en el caso de la cuantificación, los n pulsos representan 2^n niveles de amplitud o estados de salida. Al convertir cada muestra analógica en un valor digital, el convertidor redondea la amplitud al nivel más cercano dentro de un conjunto finito de niveles. Por lo tanto, se define el nivel de cuantificación Q como el intervalo entre dos puntos adyacentes de decisión y está dado mediante la Ecuación 2.6:

$$Q = \frac{\text{FSR}}{2^n} \tag{2.6}$$

donde FSR es el intervalo a escala completa y 2^n la cantidad total de niveles disponibles para cuantificar la señal obtenida.

2.4. Control PID de continuo a discreto

El controlador proporcional–integral–derivativo (PID) es, sin duda, el esquema de control realimentado más empleado en sistemas de lazo cerrado, tanto en su versión analógica como en su implementación digital. La literatura de control continuo y discreto coincide

en que su popularidad se debe a la combinación de tres factores: i) estructura sencilla, ii) interpretación física directa y iii) capacidad para lograr un equilibrio adecuado entre exactitud en estado permanente y comportamiento dinámico aceptable Franklin et al. (1997); Massaccesi (2014).

El PID actúa sobre el error de seguimiento $e(t) = r(t) - y(t)$ mediante la suma de tres aportaciones: una acción proporcional al error instantáneo, una acción integral del error acumulado y una acción derivativa asociada a la velocidad de cambio del error. Esta estructura básica se ha presentado con variaciones mínimas a lo largo de los años y puede encontrarse en diversos textos de control discreto Abdul-Hamid (2007); Massaccesi (2014); Franklin et al. (1997); Ogata et al. (1996).

Acciones PID continuo En el dominio continuo, la acción proporcional se escribe como:

$$u(t) = K\, e(t) \tag{2.7}$$

donde $u(t)$ es la señal de control y K la ganancia proporcional. Este término genera una acción correctiva directamente proporcional al error presente, reduciendo su magnitud, pero sin garantizar la eliminación del error en estado estacionario Ogata et al. (1996).

La acción integral introduce una dependencia de $u(t)$ respecto al error acumulado en el tiempo, cuya ecuación es:

$$u(t) = \frac{K}{T_I} \int_0^t e(\tau)\, d\tau \tag{2.8}$$

donde T_I es el tiempo integral o tiempo de reajuste. Este término corrige el error persistente, ya que la salida del controlador crece mientras exista un error distinto de cero, pero también puede hacer más lenta la respuesta o incrementar el sobreimpulso si la sintonía no es adecuada

Abdul-Hamid (2007); Ogata et al. (1996).

La acción derivativa, por su parte, depende de la velocidad de cambio del error y se formula como:

$$u(t) = K\, T_D\, \dot{e}(t) \tag{2.9}$$

donde T_D es el tiempo derivativo. Este término introduce un efecto anticipatorio, ya que reacciona a la tendencia futura del error y, en consecuencia, tiende a mejorar el amortiguamiento del sistema y a reducir el sobreimpulso, aunque también puede amplificar el ruido de medición si no se utiliza el filtro adecuado Massaccesi (2014); Franklin et al. (1997).

Las tres acciones conducen a la ecuación general mostrada a continuación:

$$u(t) = K\left[e(t) + \frac{1}{T_I}\int_0^t e(\tau)\, d\tau + T_D\, \dot{e}(t)\right] \tag{2.10}$$

Mientras que, en el dominio de Laplace, la función de transferencia del controlador queda:

$$H(s) = \frac{u(s)}{e(s)} = K\left(1 + \frac{1}{T_I s} + T_D s\right) \tag{2.11}$$

Acciones PID discreto

Cuando las acciones PID se implementan en una computadora digital o microcontrolador, es necesario trasladar la formulación anterior a ecuaciones en tiempo discreto. Para ello se consideran muestras del error y de la señal de control tomadas cada periodo de muestreo T, de modo que $e(k) = e(kT)$ y $u(k) = u(kT)$. La aproximación numérica de la integral y de la derivada puede realizarse mediante esquemas de diferencias finitas, como los métodos de Euler y la diferencia hacia atrás, los cuales se utilizan comúnmente en desarrollos de control

digital Franklin et al. (1997); Ogata et al. (1996). Bajo estas condiciones, las tres acciones en forma discreta se describen en las Ecuaciones 2.12–2.14. Asimismo, la Ecuación 2.15 presenta la expresión final del PID discreto.

$$u_P(k) = K\, e(k) \tag{2.12}$$

$$u_I(k) = u_I(k-1) + \frac{K\,T}{T_I}\, e(k) \tag{2.13}$$

$$u_D(k) = \frac{K\,T_D}{T}\left[e(k) - e(k-1)\right] \tag{2.14}$$

$$u(k) = u(k-1) + K\left[\left(1 + \frac{T}{T_I} + \frac{T_D}{T}\right)e(k) - \left(1 + 2\frac{T_D}{T}\right)e(k-1) + \frac{T_D}{T}e(k-2)\right] \tag{2.15}$$

Como se observa en las ecuaciones anteriores, la acción proporcional (Ecuación 2.12) mantiene la misma estructura algebraica que en el caso continuo, pero aplicada a la muestra actual del error. La acción integral discreta (Ecuación 2.13) se implementa como una suma acumulativa del error, donde $u_I(k)$ representa la contribución integral en el instante k; esta ecuación corresponde a una integración rectangular del error a lo largo del tiempo Abdul-Hamid (2007); Ogata et al. (1996). Por último, la acción derivativa (Ecuación 2.14) se aproxima mediante diferencias finitas entre muestras consecutivas, donde la diferencia $e(k) - e(k-1)$ proporciona una estimación de la velocidad de cambio del error en el intervalo de muestreo Franklin et al. (1997). Al combinar estas expresiones y reordenar términos, se

obtiene una forma incremental del PID discreto que relaciona la nueva salida del controlador con la salida anterior y con las últimas muestras del error (Ecuación 2.15).

2.5. Sintonización del PID

Además de la estructura matemática del controlador PID, Ogata Katsuhiko describe diversos métodos empíricos para determinar los parámetros del controlador a partir del comportamiento dinámico de la planta Ogata (2010). Entre ellos, las reglas de sintonía propuestas por Ziegler y Nichols constituyen uno de los procedimientos más conocidos para fijar la ganancia proporcional K_P, el tiempo integral T_I y el tiempo derivativo T_D a partir de ensayos experimentales sobre el proceso. Estas reglas se basan en el análisis de la respuesta transitoria de la planta y permiten obtener valores de los parámetros del controlador PID sin requerir un modelo matemático detallado del sistema.

Siguiendo esta metodología, el procedimiento general consiste en excitar la planta, registrar su respuesta y, a partir de ciertas características medibles, calcular K_P, T_I y T_D mediante fórmulas o tablas de sintonización. A continuación, se describen algunos de los métodos de sintonización más utilizados.

Método de sintonización con respuesta al escalón

En este método, la respuesta de la planta a una entrada escalón unitario se obtiene de manera experimental, tal como se muestra en la Figura 2.6. Si la planta no contiene integradores ni polos dominantes complejos conjugados, entonces la curva de respuesta a escalón unitario tendrá una forma de S (Figura 2.7). Esta curva con forma de S se caracteriza por dos parámetros: el tiempo de retardo L y la constante de tiempo T.

El tiempo de retardo y la constante de tiempo se determinan dibujando una recta tangente en el punto de inflexión de la curva con forma de S y determinando las intersecciones de esta tangente con el eje de tiempo y con la línea $c(t) = K$ (Figura 2.7).

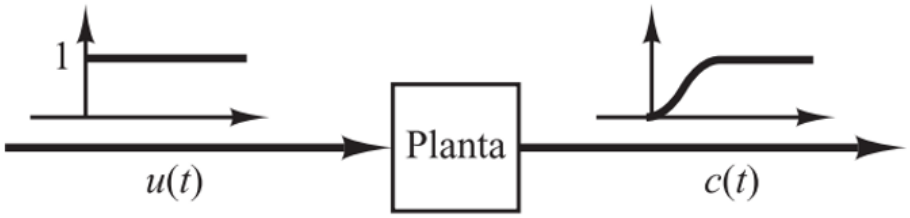


Figura 2.6: Respuesta a un escalón unitario

Fuente: Ogata Katsuhiko (2010).

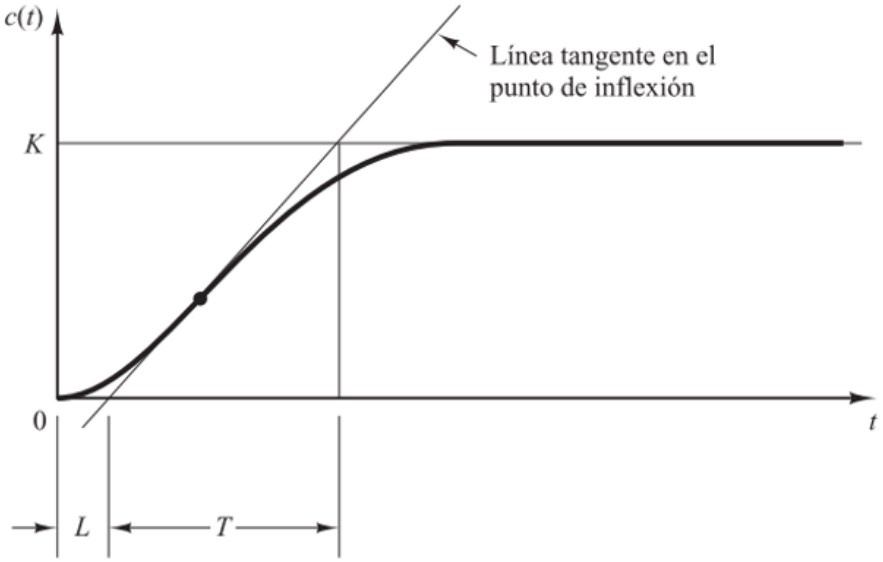


Figura 2.7: Curva de respuesta en forma de S.

Fuente: Ogata Katsuhiko (2010).

Tabla 2.1: Regla de sintonía de Ziegler–Nichols basada en la respuesta a escalón

Tipo de controlador	K_P	T_I	T_D
P	$\frac{T}{L}$	∞	0
PI	$0.9\,\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\,\frac{T}{L}$	$2L$	$0.5L$

Fuente: Ogata Katsuhiko (2010).

Ziegler y Nichols sugirieron establecer los valores de K_P, T_I y T_D de acuerdo con los valores que se muestran en la Tabla 2.1. El controlador PID sintonizado mediante este método puede expresarse como en la Ecuación 2.16, donde se muestra la función de transferencia aproximada del PID que se aplica al sistema.

$$G_C(s) = \frac{0.6\,T\left(s + \frac{1}{L}\right)^2}{s} \tag{2.16}$$

Método de sintonización con acción proporcional

El segundo método se basa en la búsqueda experimental de una ganancia proporcional crítica. Para ello, se configura inicialmente un controlador puramente proporcional, fijando $T_I = \infty$ y $T_D = 0$. A continuación, se incrementa gradualmente la ganancia K_p desde un valor pequeño hasta alcanzar un valor crítico K_{cr}, para el cual la salida del sistema presenta oscilaciones sostenidas de amplitud aproximadamente constante. El periodo de estas oscilaciones se denomina periodo crítico P_{cr}.

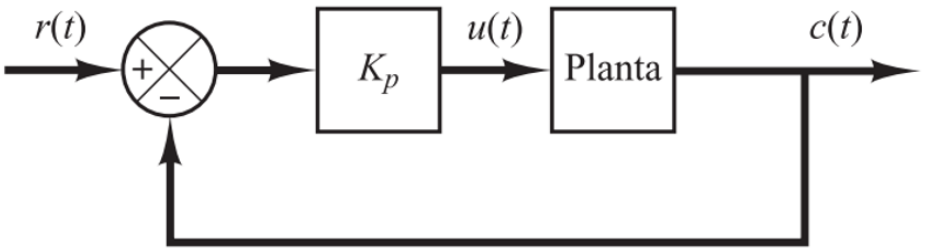


Figura 2.8: Sistema en lazo cerrado con un control proporcional.

Fuente: Ogata Katsuhiko (2010).

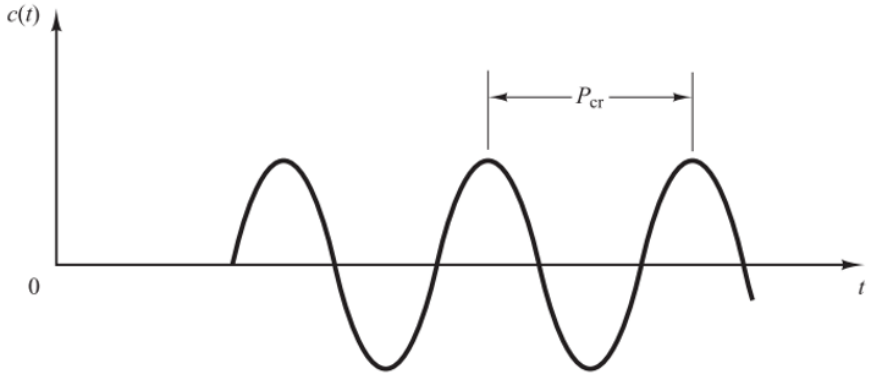


Figura 2.9: Oscilación sostenida con periodo P_{cr}.

Fuente: Ogata Katsuhiko (2010).

Una vez determinados experimentalmente K_{cr} y P_{cr}, las reglas de sintonización propuestas por Ziegler–Nichols proporcionan expresiones empíricas para calcular los parámetros del controlador. En particular, las fórmulas para el caso PID relacionan K_p, T_i y T_d con K_{cr} y P_{cr}, de forma que, por ejemplo, la ganancia proporcional se toma como una fracción de K_{cr} y los tiempos integral y derivativo se fijan como múltiplos del periodo crítico. Estas relaciones se presentan en la Tabla 2.2, diferenciando entre controladores P, PI y PID. Por último, se obtiene una función de transferencia del nuevo controlador PID ya sintonizado

(Ecuación 2.17).

Tabla 2.2: Regla de sintonía de Ziegler–Nichols basada en la ganancia crítica

Tipo de controlador	K_P	T_I	T_D
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Fuente: Ogata Katsuhiko (2010).

$$G_C(s) = \frac{0.075\,K_{cr}P_{cr}\left(s+\frac{4}{P_{cr}}\right)^2}{s} \tag{2.17}$$

Los dos métodos de Ziegler–Nichols constituyen una herramienta práctica para la sintonía inicial de controladores PID cuando se dispone de la planta en operación y es posible realizar ensayos experimentales.

En conclusión, el marco teórico y el desarrollo técnico presentado en el Capitulo 2 establecen las bases conceptuales y experimentales necesarias para la implementación integral del sistema de control del robot MENTOR. La definición de la arquitectura, los criterios de diseño electrónico y la estrategia de control permiten estructurar un sistema coherente y funcional. Con estos fundamentos consolidados, el Capitulo 3 abordara la materialización física del proyecto, detallando la fabricación del hardware, la integración de la tarjeta MINICON_STF4B y la validación experimental del sistema completo.

3. Exploración de la plataforma existente

Previo al diseño e implementación de un nuevo sistema de control y comunicación para el robot MENTOR, resulta indispensable realizar un análisis detallado de la pataforma existente. Esto permite identificar capacidades, limitaciones mecánicas y cinemáticas.

Con el fin de establecer una nomenclatura clara y consistente para el análisis cinemático y de control, se realizó la identificación de las articulaciones que conforman el robot antropomórfico MENTOR. En la Figura 3.1 se muestran las distintas articulaciones del manipulador, las cuales se nombran de acuerdo con su función mecánica y su ubicación en la estructura del robot.

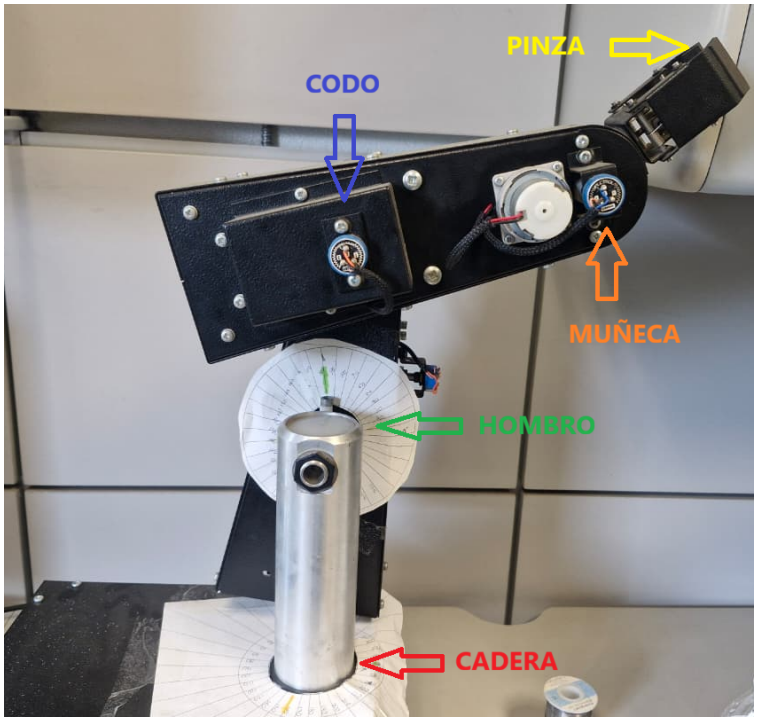


Figura 3.1: Identificación de las articulaciones del robot antropomórfico MENTOR.

De esta manera, el sistema se compone de una articulación de cadera, una de hombro,

una de codo, dos articulación correspondiente a la muñeca y una articulación asociada a la pinza. Esta identificación es importante para el desarrollo del modelo cinemático, la asignación de marcos de referencia y la implementación de los controladores de posición para cada grado de libertad.

La metodología propuesta para la revisión del hardware consiste en dos niveles: análisis mecánico y análisis electrónico, los cuales se detallan a continuación.

3.1. Inspección mecánica

El inspección mecánica está orientado a evaluar la configuración física y las restricciones articulares; es decir, los límites de operación de cada articulación. Para identificar estos parámetros se diseñaron goniómetros, integrándolos en cada articulación del robot. Posteriormente, se asignó una referencia en una parte fija del robot y cada articulación se accionó hasta un punto cercano al tope mecánico, registrando los ángulos mínimo y máximo. Esto permitió integrar una tabla con los límites angulares de cada articulación.

Tabla 3.1: Límites de operación de las articulaciones de robot MENTOR.

Articulación	**Rango (°)**	**Límite inferior (°)**	**Límite superior (°)**
Cadera	220	-110	110
Hombro	180	-90	90
Codo	180	0	180
Muñeca	150	-75	75

La determinación de estos valores permitió establecer los límites de operación aplicados en el diseño de la interfaz de usuario, en el sistema de control y en la representación virtual del robot MENTOR.

3.2. Inspección eléctrica

El análisis eléctrico estuvo orientado principalmente a determinar las características eléctricas del robot, así como el tipo de sensor utilizado para medir las variables articulares y la distribución del cableado en los conectores, de modo que la implantación de la solución implique los menores cambios posibles en el robot.

3.2.1. Sensores

Se identificó que el robot MENTOR está equipado con seis sensores de posición, ubicados en cada eje de rotación asociado a un motor. Estos sensores corresponden a potenciómetros de rotación continua de la marca *Vishay*, con una resistencia nominal de $5\,\mathrm{k}\Omega$, utilizados para medir la posición articular mediante una señal analógica proporcional al ángulo. Cada articulación tiene asociado un sensor; en el caso del conjunto de muñeca, se emplean dos sensores debido a que esta sección integra dos actuadores. Adicionalmente, la pinza cuenta con un sensor dedicado que permite retroalimentar y controlar su apertura y cierre. Para facilitar la identificación física de estos elementos dentro del sistema, en la Figura 3.2 se muestra la ubicación de los sensores en el robot.

Figura 3.2: Ubicación de los sensores de posición en el robot MENTOR.

Mediante mediciones experimentales se identificó que la relación entre el voltaje entregado por cada sensor y la posición angular es lineal, por lo que se puede modelar mediante la ecuación de la recta. Para establecer esta relación se midió el voltaje en tres puntos de interés: límite inferior, posición de origen (*home*) y límite superior. La Tabla 3.2 muestra los resultados obtenidos para cada sensor cuando la polarización es de 3.3 V:

Tabla 3.2: Límite de operación en voltaje de los sensores de posición con polarización de 3.3 V.

Sensor	**Límite superior (V)**	**Límite inferior (V)**	**Posición de home (V)**
Cadera	2.27	0.15	1.14
Hombro	2.38	0.49	1.445
Codo	2.42	0.59	0.59
Muñeca	1.70	0.60	1.15
Pinza	1.21	0.42	0.81

A partir de los valores contenidos en la Tabla 3.2 y de los límites angulares de cada articulación mostrados en la Tabla 3.1, se define un modelo lineal. Donde la pendiente m y la ordenada al origen b se obtienen mediante las siguientes expresiones.

Calculo de la pendiente:

$$m = \frac{\theta_{\text{máx}} - \theta_{\text{mín}}}{V_{\text{máx}} - V_{\text{mín}}}. \tag{3.1}$$

Calculo de la ordenada al origen:

$$b = \theta_{\text{mín}} - m\, V_{\text{mín}}. \tag{3.2}$$

Donde:

- $\theta_{\text{máx}}$ y $\theta_{\text{mín}}$ son los límites superior e inferior del rango angular válido de la articulación (en grados).
- $V_{\text{máx}}$ y $V_{\text{mín}}$ son los límites superior e inferior del rango de voltaje entregado por el sensor dentro del recorrido útil de la articulación (en volt).

Por lo que el modelo general es el siguiente:

$$\theta(V) = m\, V + b. \tag{3.3}$$

Donde:

- $\theta(V)$ es el ángulo estimado (o posición angular) de la articulación, expresado como función del voltaje medido V.
- V es el voltaje de salida del sensor (potenciómetro) asociado a la articulación.
- m es la pendiente del modelo lineal.
- b es la ordenada al origen del modelo lineal.

Estas ecuaciones se utilizaron en los capítulos posteriores para el diseño del controlador y la representación virtual .

3.2.2. Identificación de conexiones

Otra de las actividades importantes fue la identificación de las conexiones eléctricas. Se constató que el robot emplea dos conectores principales para agrupar los cables de polarización de los sensores y los de alimentación de los motores. Ambos conectores son tipo Molex de 12 terminales y paso de 2.54 mm. Con la finalidad de documentar la distribución del cableado se elaboraron la Tabla 3.3 y la Tabla 3.4, donde se indica la posición, el color y el componente asociado. Además, se incluyó la Figura 3.3, que muestra un esquema con la numeración de las terminales propuesto.

Figura 3.3: Dibujo de conector Molex 12 terminales con numeración

Tabla 3.3: Identificación de conexiones en el conector número uno del robot

Conector número 1		
Número de terminal	**Descripción**	**Color de cable**
1	Referencia	Negro
2	Polarización de sensores	Rosa
3	No se usa	—
4	Positivo motor codo	Café
5	Negativo motor codo	Rojo
6	Sensor codo	Naranja
7	Positivo motor hombro	Amarillo
8	Negativo motor hombro	Verde
9	Sensor hombro	Azul
10	Positivo motor cadera	Morado
11	Negativo motor cadera	Gris
12	Sensor cadera	Blanco

Tabla 3.4: Identificación de conexiones en el conector número dos del robot

Conector numero 2		
Numero de terminal	**Descripción**	**Color de cable**
1	Referencia	Negro
2	Polarización de sensores	Rosa
3	No se usa	—
4	Positivo motor pinza	Café
5	Negativo motor pinza	Rojo
6	Sensor pinza	Naranja
7	Positivo motor muñeca I.	Amarillo
8	Negativo motor muñeca I.	Verde
9	Sensor muñeca I.	Azul
10	Positivo motor muñeca D.	Morado
11	Negativo motor muñeca D.	Gris
12	Sensor muñeca D.	Blanco

Para finalizar este apartado de exploración de hardware conviene sintetizar los hallazgos y su relevancia para el desarrollo del sistema de control. El análisis mecánico permitió establecer los límites de operación de cada articulación para acotar el espacio de trabajo; por otra parte, en el análisis eléctrico se identificaron los sensores y se confirmó su comportamiento. Todo lo anterior se utilizó para la implementación del nuevo sistema de control y comunicación para el robot MENTOR como se desarrollo en los próximos capítulos.

4. Implementación del hardware

Como primera parte y ya descritos los elementos que constituyen al robot, se ha decidido diseñar e implementar un nuevo hardware que hace operativo al robot MENTOR, la implementación del hardware es representada en dos partes, considerados como elementales: el microcontrolador y el circuito electrónico del sistema.

4.1. Microcontrolador

Se conoce que un microcontrolador es aquel dispositivo programable con una muy alta escala de integración, encargado, además, de realizar todas las tareas que realiza una maquina a través de una búsqueda cíclica de instrucciones y de datos con el objetivo de realizar secuencias de instrucciones y operaciones. Este dispositivo cuenta con memoria, módulos de entrada y salida, módulos embebidos de comunicación o tiempo, entre otros.

A partir de lo anterior, se observa que, tanto en el ámbito académico como en el comercial, existen tarjetas de desarrollo, las cuales son circuitos electrónicos que integran uno o varios microcontroladores. Estas son ampliamente utilizadas en la enseñanza debido a su facilidad de manejo y programación. En la Facultad de Ingeniería se ha desarrollado una tarjeta de desarrollo propia llamada "MINICON_STF4B", basada en el MCU STM32F446, la cual puede ser utilizada en entornos de desarrollo de licencia libre para ser programada conforme a las necesidades del usuario. Tras la investigación realizada para seleccionar la nueva unidad de procesamiento del robot, a continuación se presenta la comparación mostrada en

la tabla 4.1.

Tabla 4.1: Comparativa general de los MCU integrados en las tarjetas de desarrollo

Tarjeta MINICON_STF4B	**Tarjeta TIVA**
Arquitectura Cortex–M4	Arquitectura Cortex–M4
Reloj interno de 180 MHz	Reloj interno de 120 MHz
Memoria flash de 512 KB, RAM de 128 KB	Memoria flash de 1024 KB, RAM de 256 KB
Módulo embebido ADC de hasta 12 bits, 2 módulos DAC de 12 bits	Módulo embebido ADC de hasta 12 bits
Cuarenta pines GPIO	Noventa pines GPIO
Dos canales PWM	Ocho canales PWM
Interfaces de comunicación: 4 USART, 2 UART, 4 SPI, 3 I^2C	Interfaces de comunicación: 8 UART, 4 SPI, 10 I^2C

Fuente: Datos tomados de las páginas oficiales de STMicroelectronics y Texas Instruments.

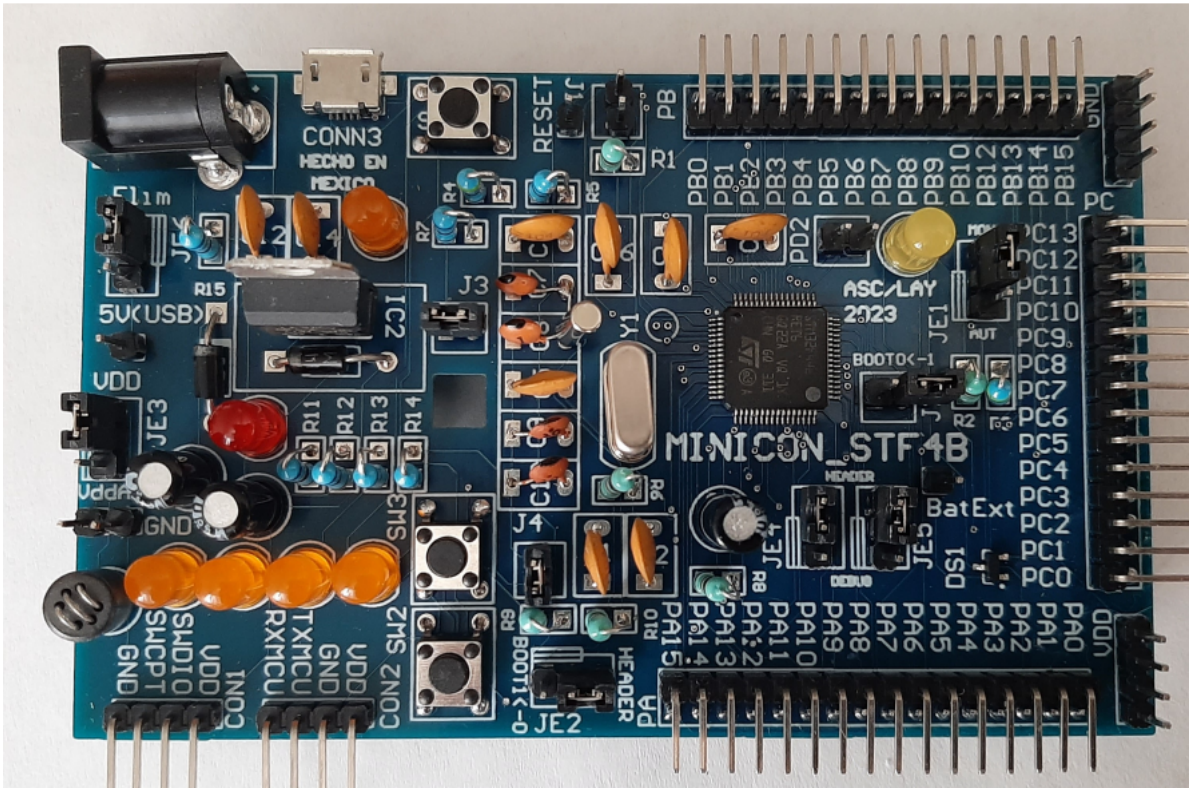


Figura 4.1: Tarjeta de desarrollo MINICON_STF4B con MCU STM32F446

Fuente: Guia de operaciones básicas de la tarjeta MINICON_STF4B | dctrl.fi-b.unam.mx (2024). https://dctrl.fi-b.unam.mx/~salva/Curso_AIDA_CM4_enero2024/Documentos/GB_MINICON_STF4B.pdf

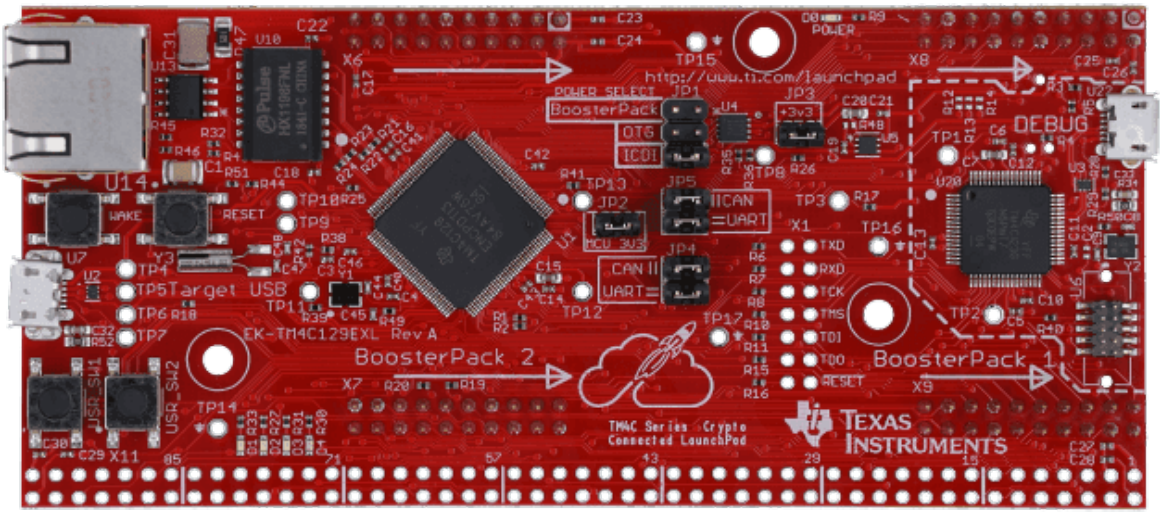


Figura 4.2: Tarjeta de desarrollo TIVA con MCU TM4C129x

Fuente: EK-TM4C129EXL Evaluation board | TI.com. (n.d.). `https://www.ti.com/tool/EK-TM4C129EXL`

A partir del análisis de los antecedentes revisados, se decidió emplear la tarjeta de desarrollo *MINICON_STF4B* como plataforma principal para la implementación del presente trabajo. Esta elección responde al interés de impulsar el uso y desarrollo de soluciones basadas en sistemas embebidos diseñados dentro del propio Departamento de Control y Robótica de la Facultad de Ingeniería.

La tarjeta *MINICON_STF4B* constituye una herramienta académica desarrollada con fines formativos y de investigación, utilizada en asignaturas especializadas del área de Control y Robótica, lo que ha permitido consolidar un entorno de aprendizaje orientado al diseño, programación y experimentación con sistemas embebidos aplicados a la ingeniería.

Asimismo, el empleo de esta plataforma dentro de la presente tesis busca contribuir a la difusión y aprovechamiento del trabajo académico realizado a lo largo de varios años en

el desarrollo de esta tecnología, por parte del M.I. Antonio Salvá Calleja, cuyo esfuerzo y dedicación para la creación, documentación y consolidación de esta herramienta es extraordinario.

4.2. Controlador electrónico

Para el diseño del controlador electronico se toman en cuenta los siguientes requerimientos:

1. Alimentación: Voltaje de entrada (127 Vac/3 A), Voltaje de salida (12 Vdc / 3 A y 3.3 Vdc / 500 mA).

2. Controladores de los motores (Señales de control de 3.3 V y Voltaje de alimentación del motor de 12 V)

3. E/S del microcontrolador: 6 entradas analogicas, 6 salidas PWM, 12 salidas digitales y comunicación UART.

La construcción del circuito se realizó procurando no afectar el área de trabajo del robot, sin modificar el diseño original del MENTOR y, además, permitiendo que, en caso de ser necesario, pudiera reconectarse al circuito electrónico original sin requerir ajustes mayores. El robot MENTOR cuenta con dos arneses de cables que se conectan a los motores y sensores; estos mismos se reutilizaron en el nuevo circuito electrónico. En la Figura 4.3 se presenta, de manera general, la estructura del nuevo circuito electrónico.

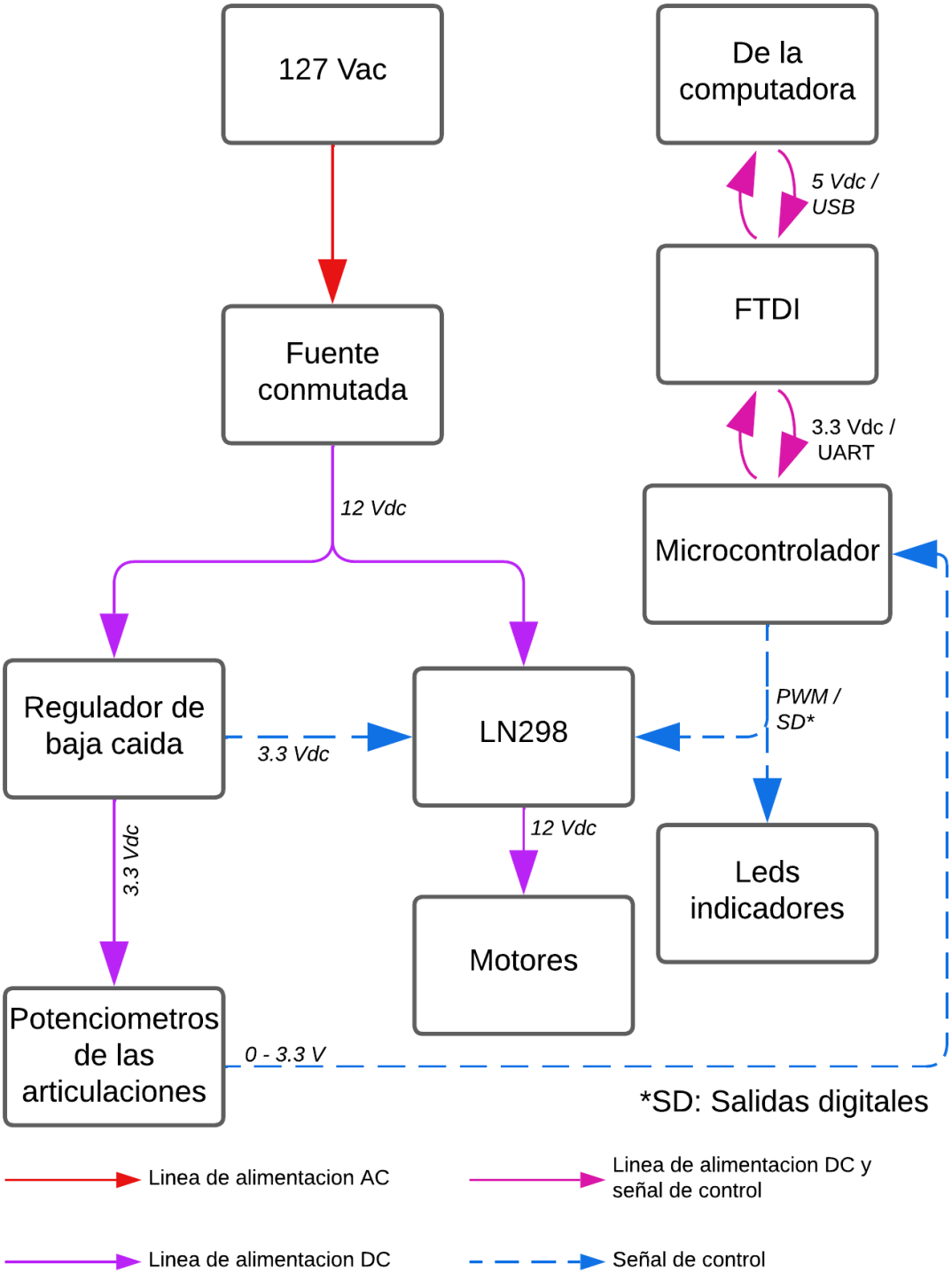


Figura 4.3: Diagrama de flujo de la tarjeta del circuito electrónico

4.2.1. Alimentación del circuito

Para la alimentación del circuito se adoptó una arquitectura híbrida que combina fuentes conmutadas para la etapa principal de potencia y reguladores lineales de baja caída (LDO) para las etapas de referencia, instrumentación y medición.

De acuerdo con ON Semiconductor (2014), los convertidores conmutados presentan ventajas relevantes en aplicaciones de potencia, entre las que destacan su alta eficiencia, menor volumen físico respecto a fuentes lineales y mejor desempeño cuando se requieren corrientes medias o elevadas. Estas características los hacen adecuados para la alimentación de sistemas donde la disipación térmica y la densidad de potencia constituyen factores de diseño importantes.

En el sistema desarrollado, dichas propiedades resultan particularmente convenientes debido a que los módulos controladores de los motores demandan corrientes relativamente altas. Por esta razón, se empleó una fuente conmutada para suministrar energía a los motores, a los módulos de control y a los elementos asociados al sistema de accionamiento.

No obstante, los convertidores conmutados introducen rizado de alta frecuencia y componentes de ruido asociados al proceso de conmutación ON Semiconductor (2014). Este comportamiento puede afectar circuitos sensibles de medición analógica. En el presente trabajo, los potenciómetros se emplean como sensores de posición y se conectan directamente a las entradas analógicas del convertidor analógico–digital del microcontrolador, por lo que requieren una alimentación estable y con bajo nivel de ruido.

Con el objetivo de garantizar condiciones adecuadas para la adquisición de estas señales, se incorporó un regulador lineal de baja caída con capacitores que actúan como

filtros de baja y alta frecuencia para alimentar la etapa de medición. Los reguladores LDO proporcionan una salida con bajo rizado y elevada estabilidad, lo que favorece la precisión en la conversión analógica–digital.

De esta manera, la combinación de una fuente conmutada para la alimentación de potencia y reguladores lineales de baja caída para las etapas de medición permite aprovechar las ventajas de ambas tecnologías, manteniendo eficiencia energética en el sistema y estabilidad en las señales analógicas utilizadas para el control del robot MENTOR.

4.2.2. Módulos controladores de motor

Para el control de los motores del robot se seleccionó el módulo L298N puente H dual, un controlador de potencia ampliamente utilizado en aplicaciones de robótica, automatización y sistemas embebidos. La elección de este dispositivo se fundamenta tanto en sus características eléctricas como en criterios de practicidad, robustez, disponibilidad comercial y compatibilidad con el microcontrolador STM32F446RE empleado en esta tesis.

De acuerdo con Handson Technology (2018), este módulo está basado en el circuito integrado L298, el cual incorpora dos puentes H completos que permiten controlar de manera independiente dos motores de corriente directa en ambas direcciones, lo que lo convierte en un controlador versátil capaz de adaptarse a diversas fuentes de alimentación, incluidas las fuentes conmutadas. El L298N es especialmente adecuado para esta tesis porque los motores utilizados no requieren corrientes muy elevadas, por lo que la capacidad del módulo resulta suficiente para lograr un accionamiento confiable. Asimismo, la capacidad de ser controlado con niveles lógicos de 3.3 V permite integrarlo con el microcontrolador utilizado, simplificando

el diseño final del circuito electrónico.

Otra ventaja del módulo es la incorporación de elementos protectores y auxiliares, entre los cuales se encuentran diodos de protección tipo *flyback* para cargas inductivas, un regulador de 5 V integrado que puede alimentar circuitos lógicos externos si fuera necesario y un LED indicador de estado que facilita la verificación visual del funcionamiento del sistema. Retomando todo lo anterior, se determinó que el módulo L298N fue elegido por cuestiones prácticas: es un módulo económico, de fácil acceso y cuenta con abundante documentación, esquemas, guías de usuario y ejemplos de aplicación en línea.

4.2.3. Otros componentes

El circuito electrónico cuenta con leds indicadores que dan un entendimiento visual del accionamiento individual de los PWM y la dirección de giro del motor correspondiente a cada una de las seis articulaciones que conforman el sistema del robot MENTOR.

El circuito cuenta con dos interruptores manuales que detienen la corriente de los motores o eliminan la alimentación del voltaje que alimenta al microcontrolador. Ayudando como un interruptor de paro si es que no se calculó el espacio de trabajo necesario que ocupa el robot MENTOR o si el usuario necesita resetear el microcontrolador respectivamente.

Por último, la comunicación entre el microcontrolador y cualquier unidad central de procesamiento se realiza a través de la comunicación serial UART (Universal Asyncronus Reception Transmissor), para eso se implementa un módulo conversor de USB a UART, permitiendo que se habilite un puerto serial y se establezca la comunicación con MATLAB, del cual se hablara a detalle en las consecuentes páginas de la tesis. También se le da energía

al microcontrolador a través de este módulo de comunicación, porque puede entregar voltajes de alimentación de 3.3 V y 5 V respectivamente, que, en nuestro caso, la tarjeta de desarrollo necesita un voltaje de alimentación de 3.3 V.

Esquema del hardware

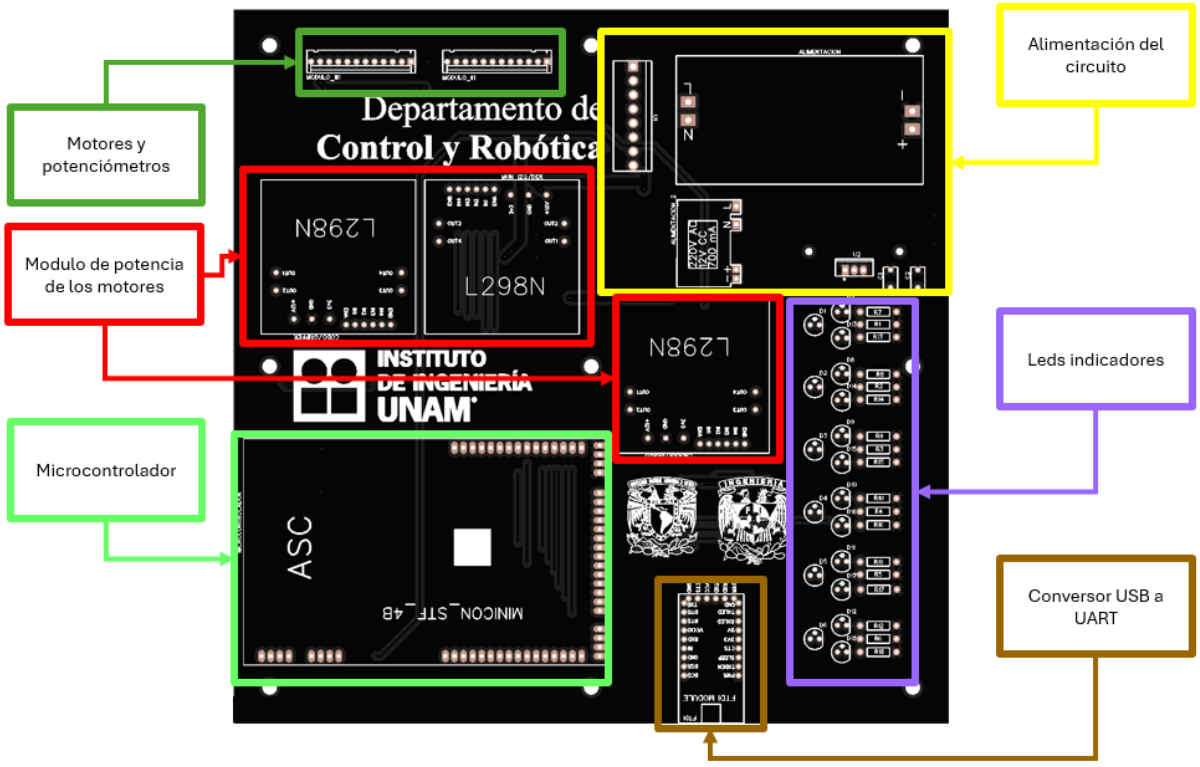


Figura 4.4: Identificación en la placa final del esquema general

5. Firmware

Después de seleccionar el microcontrolador, se procedió a definir la asignación de recursos internos del mismo, estableciendo las entradas analógicas, las salidas digitales y las señales de modulación PWM, así como los pines de comunicación que se emplearon. Esta distribución se resume en la Tabla 5.1.

La programación del firmware del sistema se realizó en lenguaje C, utilizando el entorno de desarrollo STM32CubeIDE, el lenguaje C permite el control preciso sobre el hardware, una gestión eficiente de memoria y tiempos de ejecución, convirtiéndose en una excelente elección para el diseño de bajo nivel requerido por el microcontrolador, durante el desarrollo de esta tesis se utilizó el lenguaje C para la configuración del sistema, inicialización de relojes, temporizadores, ADC, PWM, USART y GPIO's, que se explicaran en los siguientes subtemas con mayor detalle.

Tabla 5.1: Selección de las entradas y salidas de los pines de la tarjeta de desarrollo MINICON_STF4B

ADC	**PWM**	**IN1/IN4**	**IN2/IN3**
PA0	PC6	PB3	PB4
PA1	PC7	PB5	PB6
PA2	PC8	PB7	PB9
PA6	PC9	PB10	PC10
PA7	PA11	PB12	PB13
PB0	PB8	PB14	PB15
PB1	PC6	PB3	PB4

5.1. Bibliotecas

La decisión de separar el código de bajo nivel responde a principios de ingeniería de software. Dividir las funciones en unidades independientes permite que cada módulo se diseñe, pruebe y depure en forma aislada, evitando crear dependencias innecesarias y asegurando la robustez del sistema. Durante el desarrollo del *firmware* del sistema, se implementó una arquitectura similar, permitiendo que el código se mantuviera organizado, escalable y fácilmente corregible, sin afectar las otras bibliotecas del *firmware*. Las bibliotecas creadas para la tesis se dividen en: ADC, CONTROL, MOTOR, PWM, TIMER y USART1.
Como se menciona, la organización del código de bajo nivel de esta manera posibilita que cualquier modificación o mejora a futuro no afecte las otras partes del *firmware*, manteniendo la estabilidad del código.

ADC (Convertidor digital-analógico)

La biblioteca llamada ADC tiene como propósito administrar la adquisición de señales analógicas provenientes de los potenciómetros del robot. Esta define los canales analógicos utilizados (ver Tabla 5.2), inicializa el ADC y provee funciones de lectura filtrada en milivoltios. También incorpora un filtro pasa bajas, fundamental para garantizar mediciones estables y precisas.

Control

El propósito que tiene esta biblioteca es implementar controladores PID independientes utilizados para cada articulación. Esta contiene la estructura interna del PID, sus parámetros,

memorias de estado, saturación, filtros derivativos y zonas de banda muerta, también se definen los independientes por articulación.

Motor

El propósito es ejecutar funciones de bajo nivel, para accionar los motores mediante los módulos puente H, definimos la asignación de pines que giran en sentido de las manecillas del reloj o en contrasentido de estas y aplica un PWM calculado para cada motor, también tiene funciones que inicializan o apagan el motor. En esta biblioteca se encapsulan todo lo relacionado con la electrónica de potencia, permitiendo que el controlador (PID) solo produzca una señal de "potencia deseada", sin preocuparse por cómo se realiza el accionamiento físico.

PWM (Modulación por ancho de pulso)

Esta biblioteca está diseñada solo para generar señales PWM de alta frecuencia (20 KHz) que necesitan los seis motores individualmente, primero se define la configuración de los temporizadores TIM1, TIM3 y TIM4, así como la funciones para ajustar el ciclo de trabajo por canal, permitiendo que tanto el firmware de control o la trama en la que se ajuste el PWM, solo se necesite indicar el porcentaje requerido y el canal para configurar la señal de salida.

TIMER2

En este apartado se configura el temporizador 2 de la tarjeta MINICON_STF4B a 100 Hz para ejecutar el lado de control periódico. El lazo de control (lectura ADC, cálculo PID y envío del PWM) se ejecuta dentro de la interrupción del temporizador, garantizando un comportamiento determinístico y en tiempo real.

USART1 (Receptor/Transmisor Universal Síncrono/Asíncrono)

Esta última biblioteca maneja la comunicación serie entre el MCU y MATLAB mediante un

protocolo asíncrono de comunicación de recepción y transmisión, se incluye la inicialización, él envió, la recepción la clasificación de tramas, así como la velocidad y el parseo de la comunicación. La comunicación y las tramas permitidas se hablan a detalla en la sección Comunicación y tramas

Interrupciones

El código de bajo nivel desarrollado para el sistema de control de robot incluye en su arquitectura algunas interrupciones, que permiten garantizar un funcionamiento determinístico, en tiempo real y con baja latencia. En particular, se emplean dos interrupciones principales: la interrupción periódica del temporizador TIM2 y la interrupción de recepción de la interfaz USART1, El uso de ambas está fundamentado tanto en los requerimientos funcionales del sistema como en el libro TM32 ARM Programming for Embedded Systems de Mazidi et al. (2018)

Interrupción del temporizador TIM2

El temporizador TIM2 se configuró para generar interrupciones a una frecuencia de 100 Hz, definiendo así el tiempo de muestreo del lazo de control. Dentro del manejador de interrupción (`TIM2_IRQHandler`) se ejecutan las tareas críticas del sistema, tales como: adquisición de señales analógicas, cálculo de las acciones de control PID y actualización del PWM aplicado a cada motor. Todo lo anterior se sustenta en Mazidi et al. (2018), donde se explica que los temporizadores de la serie STM32F4 están diseñados para producir interrupciones periódicas que permiten ejecutar tareas repetitivas con precisión temporal. Asimismo, en el mismo capítulo se resalta que los temporizadores son apropiados para implementar actividades periódicas del sistema y para controlar procesos que dependen de una temporización estable. La ejecución del control dentro de la interrupción asegura que la temporización no se vea

afectada por la carga del procesador ni por la comunicación, preservando la estabilidad del sistema.

Interrupción de recepción UART

El *firmware* utiliza también la interrupción RXNE de la interfaz USART1 para gestionar la llegada de datos desde MATLAB. Cada vez que el módulo UART recibe un byte, la interrupción se activa y el *firmware* ensambla el mensaje en un *buffer* hasta detectar el carácter de fin de línea. De este modo, la comunicación serie se maneja sin bloquear el flujo del programa y sin interferir con el lazo de control. Las interrupciones de USART permiten capturar datos byte por byte de manera inmediata y sin pérdidas de información. En la referencia consultada se menciona que el uso de interrupciones en USART permite recibir datos sin bloquear la CPU, reaccionando inmediatamente cuando llega un nuevo byte (Mazidi et al., 2018, p. 201). Además, el uso de interrupciones es el método más recomendado para comunicaciones asíncronas en sistemas embebidos.

5.2. Comunicación y tramas

En la comunicación entre el microcontrolador y la interfaz de MATLAB se implementó el protocolo UART (*Universal Asynchronous Receiver–Transmitter*). Esta decisión se basa en el análisis presentado en Gupta and Charan (2024), donde se demuestra que UART es uno de los métodos de comunicación más utilizados en sistemas embebidos debido a su simplicidad, estabilidad, bajo costo y mínima carga de hardware. El protocolo funciona de manera asíncrona, lo que elimina la necesidad de un reloj compartido entre los dispositivos y permite que el transmisor y el receptor mantengan sincronía mediante el uso de bits de inicio,

datos y parada para estructurar cada paquete transmitido. Esto coincide con lo expuesto en dicho artículo, donde se afirma que la comunicación asíncrona se caracteriza por un diseño estable y sencillo. Asimismo, se describe que UART es un protocolo de comunicación utilizado para enviar información hacia dispositivos periféricos de computadora.

En esta tesis, MATLAB se comunica por medio de un adaptador USB–Serial, lo cual permite que las tramas enviadas y recibidas sean interpretadas sin necesidad de hardware adicional, controladores especializados o protocolos más complejos. A diferencia de interfaces síncronas como SPI, I^2C o CAN, UART no requiere líneas adicionales ni temporización estricta. El receptor genera su propio reloj interno y tolera variaciones temporales significativas, lo cual también es señalado en el artículo. En conjunto, al tratarse de una comunicación con carga de software mínima, se permite dedicar mayor capacidad de procesamiento al lazo de control del robot.

A continuación, se describen todas las tramas en la Tabla 9 y la Tabla 10, donde se listan las funciones contenidas en la biblioteca `USART1.h/.c`.

Tabla 5.2: Tramas aceptadas desde MATLAB al MCU del STM32

Trama	Función
VE: PRUEBA	Verifica el enlace de comunicación.
PARO: ALTO	Activa una función parecida a un paro de emergencia; detiene los motores y apaga los pines digitales que indican el giro del motor.
PARO: CONTINUAR	Libera el paro y permite la continuación de instrucciones que van de MATLAB al MINICON_STF4B o la continuación de instrucciones internas del MCU desde el último movimiento del robot.
PARO: REINICIAR	Libera el paro y regresa al robot a la posición definida como "Home".
MINICON_STF4B: INFO	Solicita toda la información de tramas que puede aceptar y enviar el MCU.
PWM: <id>,<ciclo>,<CW/CCW>	Para el control manual de un motor, indicando potencia (PWM) y giro.
DATOS: <frecuencia><ON/OFF>	Activa o desactiva la transmisión periódica de datos de todos los ADC vinculados a las articulaciones del robot.
SETI: <id>, <SPmV>, <Kp>, <Ki>, <Kd>	Actualiza un canal PID correspondiente a una articulación del robot; los parámetros esperados son: setpoint y ganancias proporcional, integral y derivativa.
SETD: <error_mV>	Ajusta la banda muerta del setpoint.
SETSP: <sp0>, <sp1>, <sp2>, <sp3>, <sp4>, <sp5>	Envía los datos para la actualización de los seis setpoints correspondientes a los seis canales de las articulaciones del robot.

Tabla 5.3: Tramas transmitidas desde el STM32 hacia MATLAB

Trama	**Función**
VE: Exitosa	Respuesta a la trama recibida "VE:PRUEBA", confirmando que el enlace es correcto.
PARO: ACTIVO	Indica que la función de paro de emergencia se encuentra activada.
PARO: LIBERADO	Indica que la función de paro de emergencia se encuentra desactivada.
EJPROG:FINALIZADO	Indica que un paso de trayectoria enviada desde MATLAB ha sido completado exitosamente.
MINICON_STF4B LISTO	Mensaje inicial del MCU enviado una única vez cuando la comunicación ha sido enlazada exitosamente.
V: <m0>, <m1>, <m2>, <m3>, <m4>, <m5>	Envía las mediciones filtradas en milivolt correspondientes a cada potenciómetro lineal de las articulaciones del robot.

En este capítulo se presentó el diseño e implementación del código embebido desarrollado para la actualización del robot antropomórfico. Asimismo, se estableció un protocolo de comunicación serial robusto que permite la interacción bidireccional entre la plataforma de supervisión en MATLAB y el microcontrolador, posibilitando la actualización en línea de parámetros de control, consignas articulares y criterios de terminación de movimiento.

Con la culminación del desarrollo del firmware, el sistema de control digital queda completamente integrado y funcional, sentando las bases para su evaluación experimental y validación sobre el robot real.

En el siguiente capítulo se abordará el diseño y desarrollo de la aplicación de supervisión y control, donde se describe la interfaz gráfica, consolidando así la interacción entre hardware y software dentro del sistema propuesto.

6. Diseño del software

6.1. Discusión de elección de software

En este apartado se explica la razón por la cual se seleccionó *MATLAB App Designer* para el desarrollo de la interfaz de usuario destinada al control del sistema implementado para el robot MENTOR. De acuerdo con *MathWorks* (s/f), "App Designer es un entorno de desarrollo interactivo para diseñar una aplicación y programar su comportamiento. Proporciona una versión totalmente integrada del editor de *MATLAB* y un gran conjunto de componentes interactivos de la IU" (MathWorks, sf).

Entre las principales ventajas de esta herramienta destaca su compatibilidad nativa con *Simulink* y con numerosas *Toolboxes* de *MATLAB*, en particular con la de *Robotics System Toolbox*, utilizada de manera continua en este trabajo para la simulación del robot, la planificación de trayectorias y la validación del espacio de trabajo. Asimismo, *MATLAB* ofrece soporte integrado para diversos protocolos de comunicación —como UART— que funcionan sin problemas dentro de *App Designer*, lo cual facilita la interacción con el robot.

MATLAB App Designer también permite empaquetar y distribuir aplicaciones mediante un generador integrado, lo que posibilita su instalación tanto en equipos con licencia completa como en usuarios sin acceso directo a *MATLAB*, requiriendo únicamente los componentes mínimos para la ejecución de la aplicación. Esto reduce el espacio de almacenamiento necesario y favorece la portabilidad del sistema.

Finalmente, el licenciamiento institucional por parte de la UNAM garantiza que el

trabajo desarrollado sea escalable y sostenible, permitiendo que otros estudiantes puedan dar continuidad a las aplicaciones, algoritmos y estrategias de control descritas en este trabajo. Esto facilita la evolución del proyecto y fortalece la infraestructura académica en torno al robot MENTOR.

6.2. Esquema general de la interfaz de control

En esta sección se presenta una visión general de la arquitectura de la interfaz de control desarrollado para el robot MENTOR. Dicho software fue implementado en *MATLAB App Designer* y se diseñó bajo un enfoque modular, lo que permite integrar nuevos programas de manera más sencilla y eficiente. Esta modularidad se logra mediante un programa principal, encargado de invocar y gestionar diversos subprogramas almacenados en la carpeta de recursos del proyecto.

El flujo de operación del sistema consta, básicamente, de un programa principal que ofrece acceso a cinco subprogramas diferentes. Cada subprograma posee funciones específicas, por lo que únicamente se puede ejecutar uno a la vez. Para ilustrar esta estructura propuesta, en la Figura 6.1 se muestra un esquema de flujo del programa desarrollado.

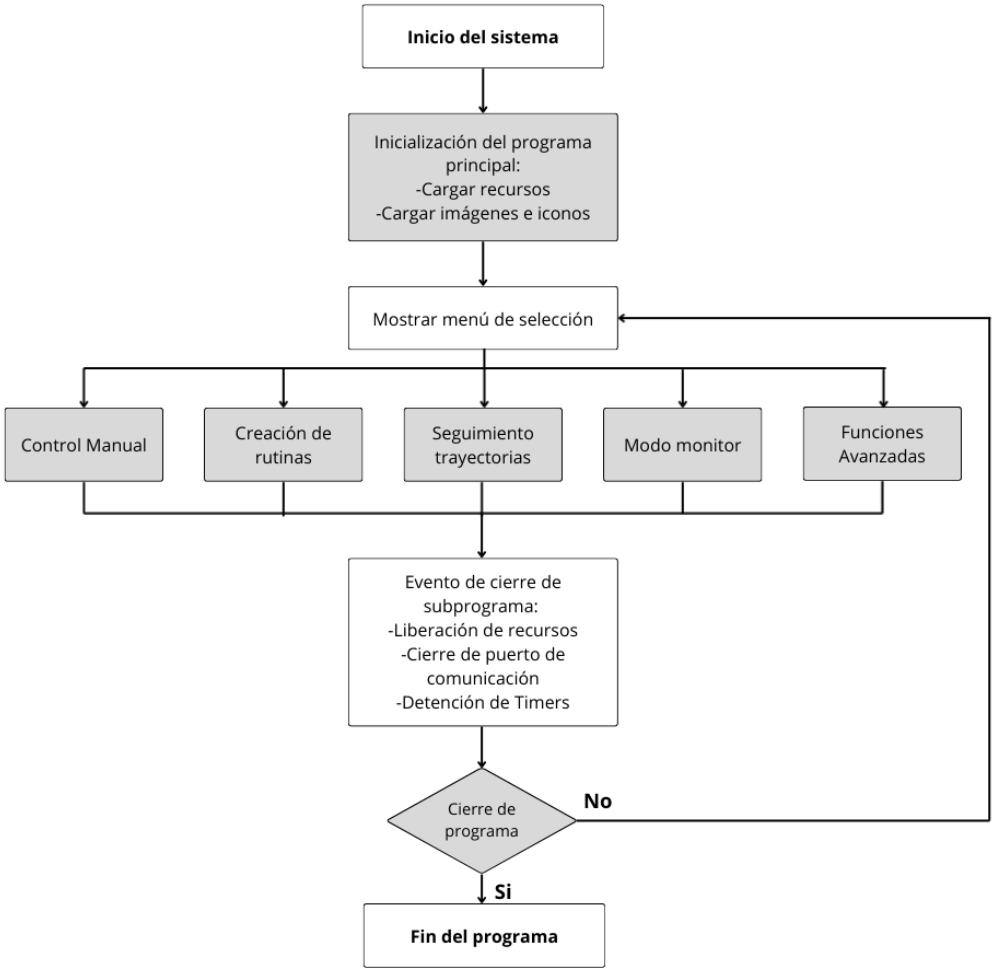


Figura 6.1: Esquema de flujo de ejecución de programa de control para robot MENTOR

A partir del esquema de flujo se identifican tres bloques principales. El primero corresponde a la inicialización del sistema, donde se habilitan los recursos necesarios. El segundo despliega el menú de selección, el cual permanece activo hasta que el usuario elige alguno de los subprogramas. El tercer bloque se ejecuta hasta que se detecta un evento de cierre, realizando la liberación de recursos, el cierre de los puertos de comunicación y la detención de todas las tareas periódicas asociadas a los temporizadores necesarios para la ejecución del programa. Una vez finalizado este proceso, el sistema regresa al menú de selección, formando

así un bucle continuo que solo termina cuando se cierra la ventana principal, concluyendo la ejecución del programa.

En los subcapítulos siguientes se explicará con mayor detalle el funcionamiento de cada uno de los subprogramas desarrollados como parte de esta tesis.

6.3. Control manual

La App Control Manual es la interfaz para la operación directa e inmediata de cada una de las articulaciones del robot MENTOR. Este módulo fue diseñado con el propósito de permitir al usuario manipular el robot en tiempo real, facilitando tanto la evaluación del sistema de control como la ejecución de movimientos básicos sin necesidad de cargar trayectorias preprogramadas.

La interfaz presenta un conjunto de cinco controles deslizantes, cada uno asociado a una articulación del robot: cadera, hombro, codo, elevación y mecanismo de pinza. Estos controles permiten modificar de manera continua el valor de referencia enviado a cada controlador PID. Conforme el usuario desplaza un control, la aplicación genera de forma inmediata la trama correspondiente y la envía al microcontrolador, lo que produce el movimiento directo de la articulación seleccionada. Además de los

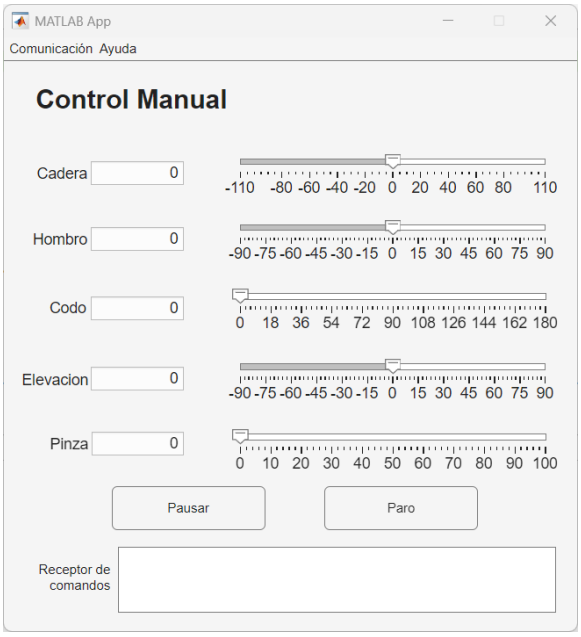


Figura 6.2: App Control Manual

controles deslizantes, la aplicación incorpora una consola de mensajes que informan en tiempo real sobre él envió de la trama actual de cada sensor de posición.

La app también incluye un panel de configuración básica que permite seleccionar el puerto serial, establecer la velocidad de comunicación, activar o desactivar la transmisión continua de datos y ejecutar funciones esenciales como el paro de emergencia. El paro puede activarse desde la interfaz y provoca que todas las señales de control se anulen inmediatamente, garantizando la seguridad del operador y del equipo en caso de fallos.

6.4. Creación y ejecución de programas

La App Creación y Ejecución de Programas constituye el módulo encargado de permitir al usuario generar, editar, almacenar y ejecutar secuencias de movimiento para el robot MENTOR. Mientras que la App Control Manual está orientada al control manual y la evaluación inmediata de cada articulación, esta app introduce un nivel superior de automatización, posibilitando la programación estructurada de trayectorias compuestas por múltiples pasos y configuraciones articulares. Su diseño funcional permite que estudiantes y operadores elaboren rutinas completas sin necesidad de conocimientos avanzados de programación embebida o comunicación serial.

La interfaz integra una tabla de programación, donde cada fila representa un paso del programa que el robot debe ejecutar. En cada paso pueden definirse los valores de referencia de cada articulación, el tipo de movimiento (por ejemplo, desplazamiento directo o interpolado), el tiempo asignado para completar la transición, y la duración de retardo o espera al finalizar el movimiento. Esta estructura permite crear secuencias complejas mediante la

combinación de movimientos elementales, generando trayectorias reproducibles y adecuadas para prácticas de laboratorio o tareas específicas de demostración.

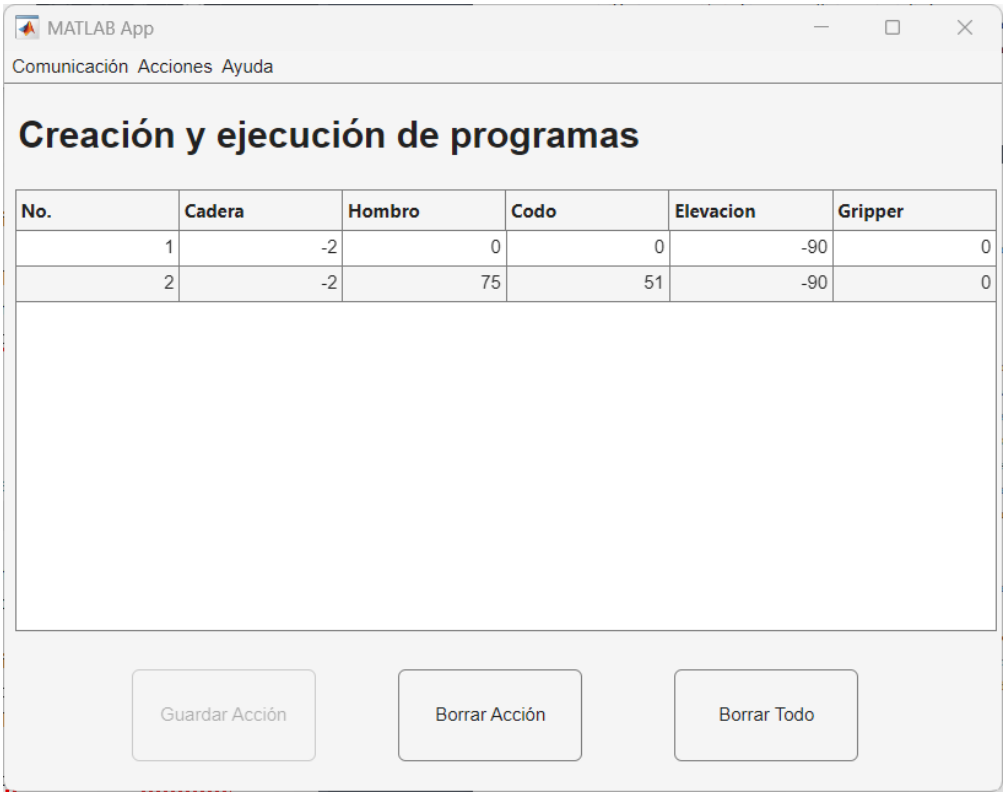


Figura 6.3: App Creación y ejecución de programas

La aplicación incorpora también herramientas para agregar, eliminar, reordenar y modificar pasos dentro del programa. Estas funciones permiten construir rutinas de forma intuitiva, similar a un editor de instrucciones industriales. Cada cambio realizado en la tabla se refleja inmediatamente en la estructura interna de datos, lo que garantiza que el usuario tenga control completo sobre el flujo de ejecución. Asimismo, la App permite guardar y cargar programas desde archivos externos, lo que facilita la reutilización de rutinas y la creación de una biblioteca de ejercicios para las prácticas de la materia de Robótica Industrial.

Para la ejecución del programa, la App utiliza la comunicación anterior ya estableci-

da en la app de Control Manual con el microcontrolador del robot, enviando los comandos correspondientes paso por paso y esperando las confirmaciones necesarias. Además, la aplicación dispone de funciones de inicio, pausa, reanudación y cancelación, proporcionando control total sobre la ejecución del programa y garantizando seguridad en caso de que ocurra algún evento inesperado.

6.5. Programa de monitoreo

En esta sección se presenta una descripción general de la aplicación de monitoreo desarrollada como parte del sistema de control del robot MENTOR. Esta aplicación tiene como objetivo ilustrar de manera gráfica la posición del robot en tiempo real, usando como base la representación virtual.

Descripción general

La aplicación desarrollada permite al usuario seleccionar una configuración cinemática definiendo los valores de las variables articulares mediante una interfaz gráfica, para posteriormente enviar dichos valores al MCU. Por otra parte, el MCU envía continuamente los valores de cada variable articular; con estos datos, *MATLAB* actualiza la representación virtual del robot. En la Figura 6.4 se presenta un esquema de comunicación.

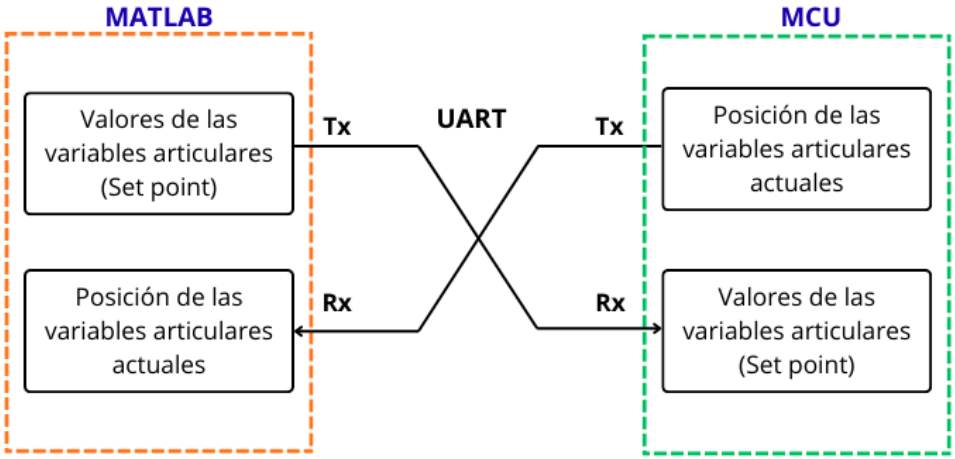


Figura 6.4: Esquema de flujo de comunicación del programa de control para robot MENTOR

Interfaz gráfica y funcionalidades

La aplicación desarrollada cuenta con tres bloques principales. El primero corresponde a la barra de menú superior, donde se pueden realizar ajustes relacionados con la comunicación con el dispositivo y cargar el archivo URDF en caso de que el usuario cierre la ventana donde se muestra la representación virtual del robot. En el segundo bloque se encuentran cinco campos editables y cinco deslizadores, con los cuales se pueden seleccionar los valores de las articulaciones en grados; el usuario únicamente puede seleccionar ángulos dentro del rango de operación. El tercer bloque consta de tres botones con los que se pueden enviar comandos al robot: llevar al robot a la posición *Home*, enviar la configuración cinemática y por último, un botón de paro de emergencia. Este último envía una trama específica al MCU para activar una interrupción de alta prioridad que detiene inmediatamente el movimiento del robot. Esta función resulta crucial para evitar colisiones y salvaguardar tanto el entorno como la integridad mecánica del robot.

Por último, en esta sección se adjunta una captura de pantalla de la interfaz desarro-

llada.

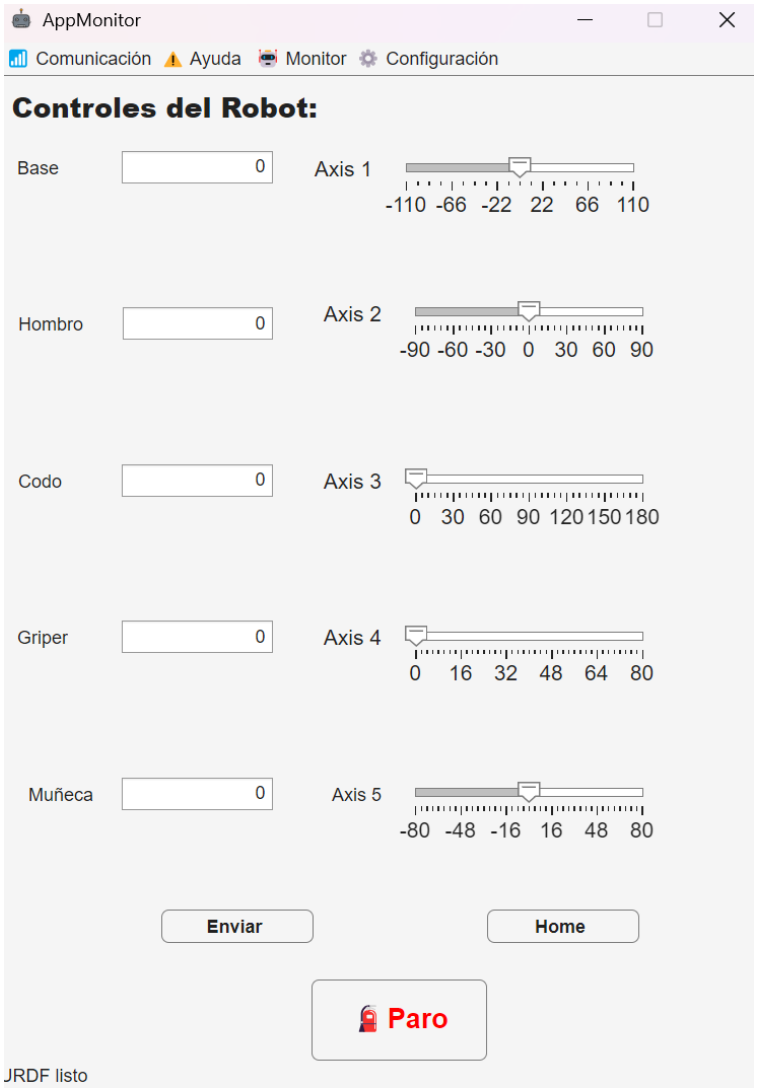


Figura 6.5: Captura de pantalla de la interfaz para aplicación de modo monitor

En el siguiente apartado se explica el funcionamiento del programa de seguimiento de trayectorias desarrollado como parte de este trabajo.

6.6. Programa de seguimiento de trayectorias

En esta sección se presenta una descripción general de la aplicación de seguimiento de trayectorias desarrollada como parte del sistema de control del robot MENTOR. Su diseño se fundamenta en lo expuesto en el Capítulo referente al seguimiento y planeación de trayectorias con *MATLAB* y *Toolbox*. Debido a la importancia del seguimiento de trayectorias en robótica y control, se decidió desarrollar una aplicación que ilustre de manera cualitativa este tópico.

Descripción general

La aplicación desarrollada permite al usuario realizar una simulación de la trayectoria entre tres puntos; este esquema se muestra en la Figura 8.5, "Flujo de trayectoria planeada". En la aplicación se permite al usuario enviar la trayectoria simulada al MCU mediante UART para que el robot la ejecute y, con ello, evaluar de manera cualitativa el desempeño de la ejecución de la trayectoria con respecto a la simulación. El esquema de flujo del programa se presenta a continuación.

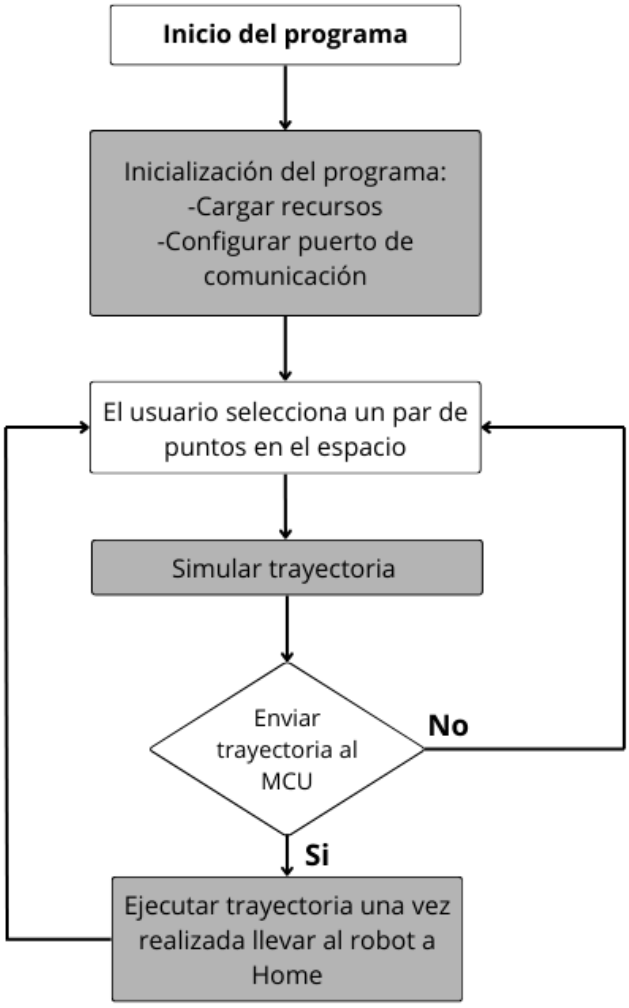


Figura 6.6: Esquema de flujo de ejecución de programa "Seguimiento de trayectorias".

Interfaz gráfica y funcionalidades

El desarrollo de la interfaz gráfica se realizó completamente con *MATLAB App Designer*. La aplicación cuenta con una barra de menú superior donde se puede acceder a algunos ajustes, un par de menús desplegables para la selección de puntos que están dentro del espacio de trabajo y, por último, un panel con los controles del robot y la simulación. En la Figura 6.7 se muestra una captura de pantalla de la interfaz desarrollada.

Figura 6.7: Captura de pantalla de la interfaz para aplicación de seguimiento de trayectorias

En la barra de menú superior se puede acceder a configuraciones relacionadas con la comunicación con el robot, tales como la configuración del puerto de comunicación, la verificación de la conexión con el dispositivo y el cierre de la comunicación. En lo correspondiente a controles del robot se cuenta con cinco botones: dos corresponden a la simulación y los tres restantes al control del robot.

Durante la simulación, la representación virtual se mueve en tiempo real, lo cual proporciona una representación intuitiva del movimiento planeado. Una vez finalizada la simulación, la aplicación habilita el botón *Enviar*, que inicia la transmisión de la trayectoria planeada hacia el MCU. El microcontrolador recibe la secuencia punto a punto y ejecuta el movimiento, lo que permite comparar la trayectoria simulada frente a la trayectoria realizada. En la Figura 6.8 se observa una captura del programa en ejecución.

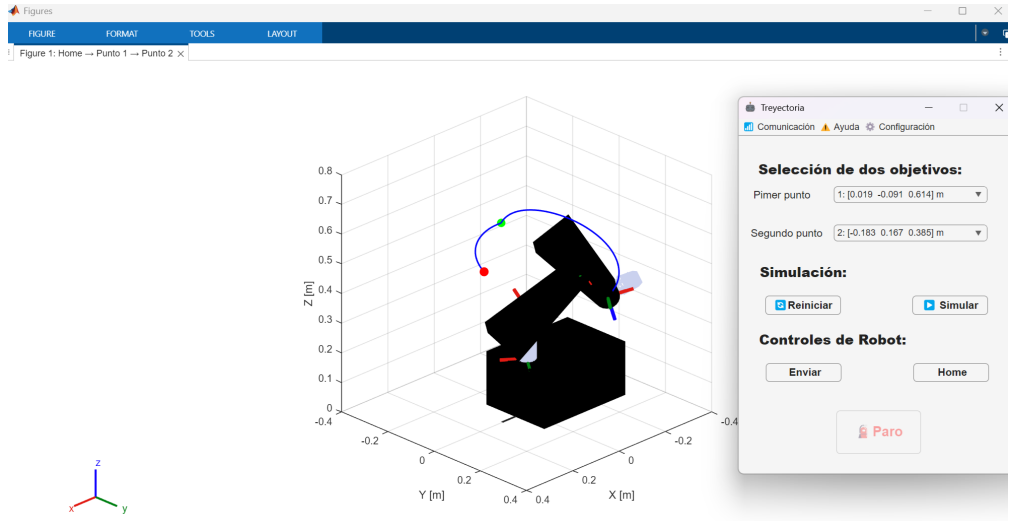


Figura 6.8: Captura de pantalla de la interfaz para aplicación de seguimiento de trayectorias

La aplicación también incorpora un botón de paro de emergencia, el cual envía una trama específica al MCU, de manera idéntica al funcionamiento del botón de paro de emergencia utilizado en el programa de monitoreo.

6.7. App funciones avanzadas

La *App Funciones Avanzadas* fue desarrollada como un entorno complementario destinado a proporcionar funciones avanzadas de diagnóstico, análisis y soporte para la operación del robot MENTOR. El propósito central de esta aplicación es ofrecer una plataforma donde puedan explorarse alternativas de control tales como control por modos deslizantes, control PD con compensación de fricción, control basado en modelos simplificados, control adaptativo, linealización por realimentación o estrategias híbridas entre control clásico y heurístico. La aplicación permite configurar los parámetros del algoritmo elegido, ejecutar las pruebas directamente sobre el robot en tiempo real y observar el efecto inmediato de la estrategia

implementada. De esta manera, el usuario puede comparar rendimientos, analizar estabilidad y determinar la viabilidad de esquemas que eventualmente sustituyan o complementen al PID principal.

Dentro de las Funciones Avanzadas se incluyen funciones para la visualización en tiempo real de datos, particularmente útiles durante el ajuste de parámetros, la validación de sensores y la resolución de problemas de comunicación. La aplicación recibe tramas desde el microcontrolador y las presenta de manera estructurada, permitiendo al usuario identificar valores anómalos, saltos en las mediciones o inconsistencias en los datos transmitidos. Esto facilita la detección temprana de fallas relacionadas con errores en la configuración del *firmware*.

Otra característica importante de esta aplicación es su capacidad para manejar y gestionar *scripts* de MATLAB (Figura 6.9) asociados al software interno del robot MENTOR. El módulo ofrece un mecanismo para crear, abrir, editar y guardar *scripts* dentro del directorio de recursos del proyecto, permitiendo organizar herramientas adicionales que complementan las funciones del sistema. De esta manera, el usuario puede generar utilidades personalizadas, rutinas de análisis, guiones de simulación o procedimientos de depuración sin abandonar la interfaz principal. Esta integración favorece un flujo de trabajo continuo y eficiente, especialmente en actividades de investigación y documentación.

El Módulo de Herramientas también incorpora opciones para monitorear la comunicación serial utilizada por las demás aplicaciones. A través de indicadores visuales y cuadros de registro, es posible observar el estado del puerto, la disponibilidad de datos y la tasa de transmisión. Esta funcionalidad es esencial para garantizar que el enlace entre MATLAB y el microcontrolador se mantenga estable, especialmente en situaciones donde se requie-

ren transmisiones continuas de datos o cuando se ejecutan tareas complejas que implican múltiples aplicaciones simultáneas.

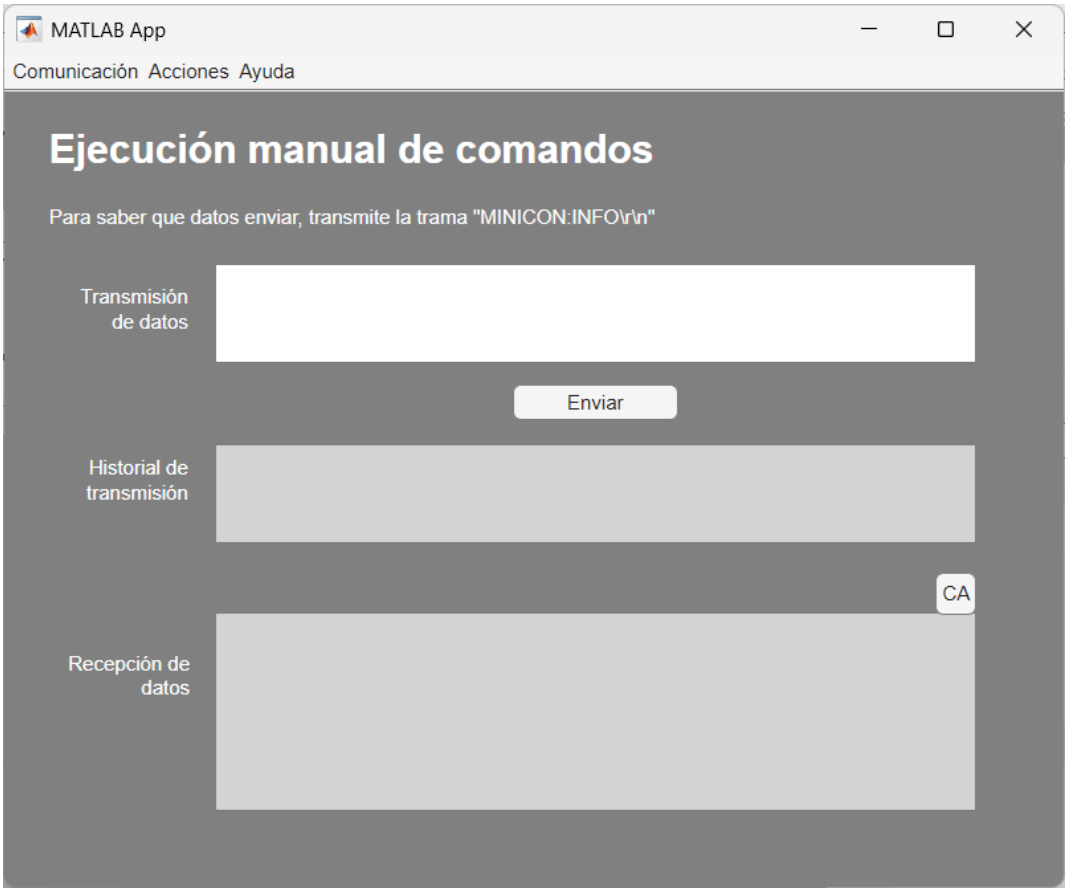


Figura 6.9: App Funciones Avanzadas

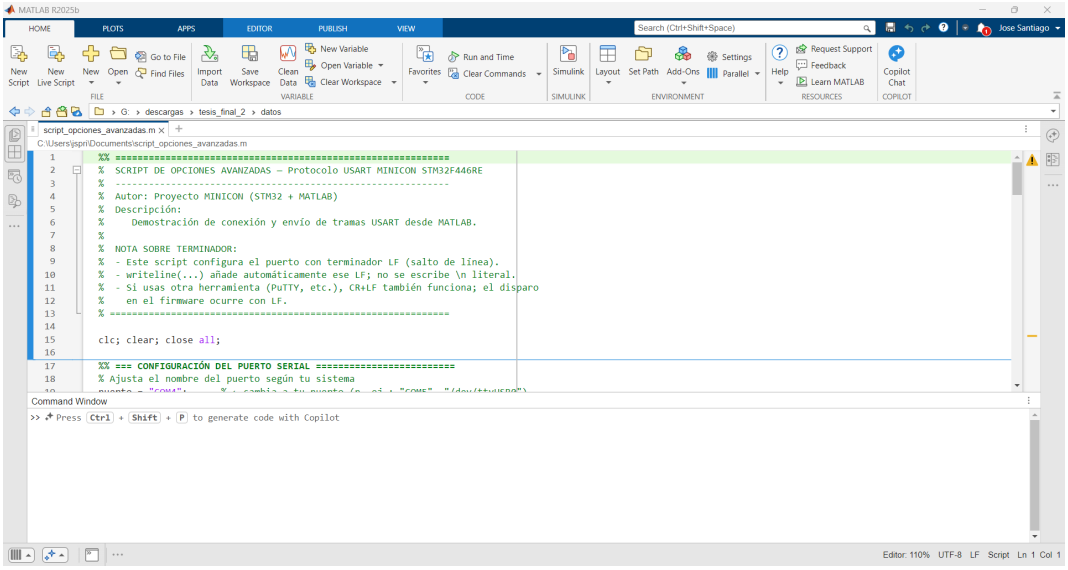


Figura 6.10: Script base creado desde la app Funciones Avanzadas

En conclusión, en este capítulo se documenta el trabajo realizado para el diseño e implementación del software de supervisión, integrando en una misma plataforma todo el software manejador del robot MENTOR. Este desarrollo permite contar con una herramienta funcional para visualizar variables en tiempo real, enviar consignas y ejecutar acciones de operación de manera estructurada, contribuyendo a la consolidación del sistema propuesto. Con lo anterior completado, el siguiente capítulo se enfocará en el control del robot, presentando el procedimiento seguido para su evaluación, así como el análisis del desempeño obtenido a partir de la implementación y ajuste de los controladores en las articulaciones.

7. Control del robot

El control del robot se diseñó sobre un esquema de lazo cerrado con retroalimentación de posición basado en la medición proporcionada por lo potenciómetros de la configuración inicial del hardware del robot MENTOR, en donde cada potenciómetro se encuentra instalado en cada articulación del robot, teniendo como objetivo que el sistema de control garantice el alcance y la posición deseada aun en presencia de perturbaciones externas, variaciones en la carga mecánica y no linealidades inherentes al sistema, para lograrlo se evaluaron métodos de control clásicos, cuyo análisis se presenta a continuación.

7.1. Elección de control

El diseño del sistema de control para el robot se hizo con una selección de un método robusto, eficiente y compatible con las limitaciones inherentes tanto al sistema, como al microcontrolador. En sistemas mecatrónicos, en especial aquellos accionados con motores de corriente directa y con sensores analógicos como potenciómetros, la calidad del control depende de la estabilidad del algoritmo, de la regularidad de tiempo de muestreo y de la capacidad para compensar variaciones en la dinámica producto de fricción, carga y no linealidades mecánicas, en este contexto, la estrategia más viable para garantizar un comportamiento estable y predecible son los controladores tipo Proporcional-Integral-Derivativo (PID).

La relevancia del PID en sistemas embebidos actuales, está documentado, por ejemplo, Zhao (2025) describe que, a pesar de técnicas avanzadas basadas en modelos matemáticos y

optimización numérica, el PID continua siendo la opción más empleada debido a sus estructura versátil, bajo costo computacional e independencia parcial del modelo, la flexibilidad lo convierte en un controlador capaz de interactuar con sistemas cuya dinámica no esté completamente caracterizada o presente variaciones en el tiempo. Condiciones que se encuentran en el robot de nuestra tesis.

Uno de los factores más determinantes en la elección del método de control fue la ausencia de un modelo matemático del robot. La creación de una función de transferencia completa para el robot es la complicación que tenemos en cada articulación del mismo, aunque tenemos otros factores como la interacción mecánica entre estabones, la presencia de fricción, el acoplamiento dinámico y la variación de carga dependiendo de la postura, mismos argumentos que encontramos en artículos como el de Xiang et al. (2025), donde comunican que la mayoría de los sistemas no lineales presentan dificultades para ser completamente modelados, lo que hace que el PID con o sin modificaciones adaptativas continue siendo una alternativa válida cuando la retroalimentación es confiable y se busca un desempeño estable sin recurrir a un modelado intensivo. De forma paralela, Song et al. (2017) señala que incluso en sistemas multivariables con incertidumbre severa en el comportamiento, el PID puede servir como base para esquemas robustos o adaptativos, manteniendo el nivel de adaptabilidad aceptable, por lo tanto, su uso se justifica plenamente en el robot MENTOR.

Otro aspecto fundamental es la frecuencia de actualización de control. El sistema fue diseñado para operar a una frecuencia fija de 100 Hz, ejecutando el controlador en la interrupción periódica del temporizador, esta tasa se encuentra dentro del rango de frecuencias usuales para actuadores electromecánicos de baja inercia, de acuerdo con Laskawski and Wcislik (2016), quienes mencionan que un tiempo de muestreo adecuado es crucial para

preservar la estabilidad del control digital y evitar errores derivados de aproximaciones discretas. La elección de un periodo de 10 ms permite alcanzar una respuesta apropiada entre la capacidad del sistema y las restricciones del microcontrolador, evitando cargas computacionales que pueda comprometer las ejecuciones de otras tareas que se están realizando simultáneamente en la tarjeta de desarrollo.

La implementación de controladores PID en microcontroladores ARM ha sido ampliamente documentada en la literatura, como lo demuestran Nithyasree and Kandasamy (2012). Estos autores demuestran que el PID es especialmente adecuado para procesadores de propósito general que deben cumplir con tareas de control con tiempos de ejecución deterministas. Sus hallazgos confirman que la estructura compacta del PID, basada en cálculos directos mediante diferencias sucesivas, permite su ejecución en intervalos muy breves sin comprometer el rendimiento del microcontrolador ni saturar sus recursos de procesamiento.

Finalmente, un controlador PID discreto en el robot asegura un comportamiento estable durante el seguimiento de trayectorias, proporciona transiciones suaves entre posiciones y ofrece robustez frente a perturbaciones.

7.2. Controlador PID

El controlador proporcional ,integral, derivativo (PID) constituye el núcleo del sistema de control implementado en cada articulación del sistema MENTOR. La estructura funcional adoptada en este proyecto se basa en la arquitectura clásica de lazo cerrado, donde el error de posición, obtenido a partir de los potenciómetros, se procesa para calcular la acción de control y generar la señal que acciona el actuador, tal como se ilustra en la Figura 18.

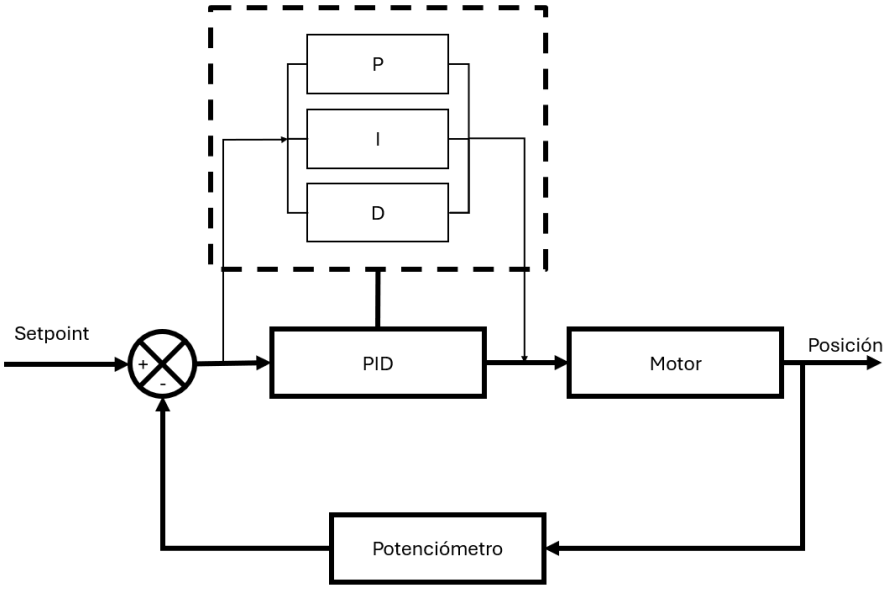


Figura 7.1: Control de Lazo Cerrado

7.2.1. Estructura del algoritmo

El controlador PID utilizado fue desarrollado como una librería modular dentro del *firmware* de bajo nivel, estructurada para administrar seis controladores independientes: uno por cada articulación del robot. Cada controlador está definido como una estructura llamada `PID_t` y se inicializa con parámetros específicos de ganancia, saturación, tiempo de muestreo y constantes de filtrado. Esta arquitectura modular permite que cada articulación cuente con su propio conjunto de parámetros sin interferir con las demás.

La implementación base del algoritmo corresponde a una función llamada `PID_Step_Base` incluida en el archivo de código `control.c`. Dentro de esta función se integran tres elementos esenciales del controlador:

- **Acción proporcional:** La acción proporcional evalúa el error actual y lo multiplica por la ganancia correspondiente:

```
float P = p->Kp * e_eff;
```

Este término corrige la desviación inmediata entre la posición deseada y la real, generando una respuesta proporcional en magnitud a la discrepancia presente. Es el componente que más contribuye a la rapidez de reacción del sistema.

- **Acción integral:** La integral se implementa mediante una acumulación discreta:

```
float I_pre = p->i_acc + p->Ki * p->Ts * e_eff;
```

Esta ecuación constituye la versión digital del integrador continuo, donde el término $K_i T_s e[k]$ se va sumando gradualmente para corregir errores persistentes en el tiempo. El integrador se actualiza únicamente cuando la salida no está saturada, el control generado supera la banda muerta de salida y el error supera la banda muerta de entrada. Estas condiciones permiten evitar el fenómeno conocido como *windup*, que ocurriría si la integral continuara creciendo incluso cuando el actuador ya no puede generar mayor esfuerzo.

- **Acción derivativa filtrada:** Sirve para estimar la tendencia futura del error; se emplea un cálculo derivativo suavizado mediante un filtro de primer orden:

```
float v = (e_eff - p->e_prev) / p->Ts;
p->d_filt += beta * (v - p->d_filt);
float D = p->Kd * p->d_filt;
```

El filtro derivativo depende de la frecuencia de muestreo porque la derivada digital se basa en diferencias finitas entre muestras sucesivas, y su estabilidad numérica, sensibilidad al ruido y rapidez de respuesta están determinadas directamente por el valor de T_s. Al incluir T_s en el cálculo del derivado filtrado, el controlador asegura una representación coherente de la dinámica del error, evitando la amplificación de ruido y logrando un comportamiento amortiguado y estable en cada articulación del robot.

A nivel general, el control calculado es la suma de las tres acciones, quedando descritas en la Ecuación 7.1.

$$u[k] = K_p\, e[k] + I[k-1] + K_i\, T_s\, e[k] + K_d\, D_f[k] \tag{7.1}$$

Cada una de las acciones del controlador puede habilitarse o deshabilitarse según se requiera. Aunque el PID cuenta con ganancias previamente ajustadas conforme a su sintonización individual, estas pueden modificarse para implementar otros esquemas de control, como P, PI, PD o PID. La configuración del tipo de control deseado se logra desactivando las acciones no utilizadas, asignando valores de ganancia igual a cero a las correspondientes funciones.

7.2.2. Otras funcionalidades

Aunque el controlador utiliza la estructura clásica del PID, en el código se incorpora una serie de mecanismos adicionales que permiten mejorar la estabilidad, reducir el ruido y prevenir comportamientos indeseados típicos de sistemas reales. Estas herramientas provienen tanto de la teoría del control digital como de la experiencia práctica con actuadores y

sensores.

1. **Banda muerta del error:** En la banda muerta del error se define un umbral mínimo por debajo del cual el sistema considera que el error es equivalente a cero. Esto elimina microoscilaciones producidas por ruido en los potenciómetros, evita movimientos innecesarios del motor y mejora la estabilidad cerca del *setpoint*.

2. **Antiwindup:** Con esto se evita que la acción integral continúe acumulándose cuando el actuador alcanza saturación. Este mecanismo es esencial para prevenir sobreimpulsos cuando el sistema vuelve a una región de operación normal. La implementación considera condiciones estrictas para mantener la coherencia entre la acción integral y el efecto real del actuador.

7.3. Selección de ganancias y resultados del control

La sintonía de los seis controladores PID se realizó de manera experimental debido a la naturaleza no lineal del robot MENTOR, las fricciones internas, la heterogeneidad mecánica entre articulaciones y la sensibilidad de los potenciómetros de realimentación. Para cada articulación se determinó un conjunto de ganancias proporcional (K_p), integral (K_i) y derivativa (K_d) capaz de garantizar seguimiento preciso de referencia, mínima sobreoscilación y estabilidad frente a perturbaciones originadas por el movimiento de otros grados de libertad. Estas ganancias no fueron arbitrarias: cada articulación presenta relaciones distintas entre par, inercia y rango útil del sensor, por lo que ganancias elevadas en K_p (entre 1500 y 3000) fueron necesarias para compensar la fricción estática y garantizar una acción proporcional suficiente para vencer el régimen de arranque. Por otra parte, las ganancias integrales se

mantuvieron moderadas con el fin de evitar acumulación excesiva del error, mientras que K_d se empleó para reducir los picos de velocidad y suavizar la transición entre niveles de referencia.

Con estas ganancias, el desempeño general del sistema de control demuestra una respuesta estable y un seguimiento adecuado del *setpoint* en todas las articulaciones. En las gráficas correspondientes se observa que la señal medida se ajusta al valor de referencia con un error transitorio reducido y sin presentar sobreoscilaciones significativas. Las pendientes pronunciadas del *setpoint* son seguidas de manera eficiente gracias al equilibrio entre la acción proporcional y la derivativa, evitando variaciones abruptas o comportamientos impulsivos. Además, una vez alcanzado el nivel deseado, el error se mantiene cercano a cero, lo que indica un buen efecto de la acción integral sin generar acumulación excesiva o fenómenos de *windup*.

En articulaciones como codo, muñecas y pinza, donde la fricción interna y la relación de engranes producen mayor resistencia inicial al movimiento, se observan picos de error ligeramente superiores durante los cambios bruscos de referencia. Este comportamiento es consistente con la mecánica del robot y no compromete la estabilidad, ya que los controladores logran amortiguar rápidamente estos picos y llevar la señal medida al valor deseado en tiempos cortos. Las ganancias más altas de K_p y K_d utilizadas en estas articulaciones permiten contrarrestar las cargas dinámicas y suavizar los transitorios.

Por otro lado, las articulaciones de cadera y hombro muestran un comportamiento aún más estable debido a que presentan menor fricción relativa y una estructura mecánica más robusta. Sus errores transitorios son pequeños, los tiempos de establecimiento son breves y prácticamente no se aprecia sobrepaso. Esto confirma que los valores moderados de K_i y K_d

seleccionados son suficientes para garantizar una respuesta estable, evitando amplificación del ruido o exceso de esfuerzo de control.

Tabla 7.1: Ganancias finales de los controladores PID

Articulación	K_p	K_i	K_d
Cadera	1600	50	30
Hombro	1675	80	25
Codo	2000	80	25
Muñeca Izquierda	3000	150	60
Muñeca derecha	3000	150	60
Pinza	1500	50	20

A lo largo del presente capítulo se justificó la elección del controlador PID como estrategia principal para el control articular del robot, considerando su simplicidad de implementación, su robustez y su adecuación a las condiciones reales de operación del sistema. Posteriormente, se describió el proceso seguido para su implementación en el microcontrolador, así como las funcionalidades complementarias incorporadas para mejorar el desempeño y la seguridad del control, tales como criterios de saturación, manejo de banda muerta, mecanismos de mitigación de acumulación integral y validación de estabilidad en la ejecución de movimientos. Finalmente, se presentaron las ganancias obtenidas para cada articulación y los criterios empleados para su ajuste, dejando establecido el conjunto de parámetros con los que opera el sistema. Con este marco de control definido e integrado, el Capítulo 8 introduce el desarrollo de la representación virtual del robot, abordando el modelado e integración en el entorno de simulación y su uso como herramienta de visualización, verificación y apoyo a la supervisión del sistema.

8. Representación virtual del Robot MENTOR

Un gemelo digital se define, según Chinnasamy et al. (2023), como una "representación de un objeto físico, proceso, estructura o equipo en un espacio virtual". Este tipo de modelos resultan de utilidad para monitorizar, analizar y optimizar procesos, así como para predecir el rendimiento a lo largo de su ciclo de vida, entre muchas otras aplicaciones. En el desarrollo de este trabajo se implementó una representación virtual considerando aspectos cinemáticos, con el objeto de determinar el espacio de trabajo del robot manipulador, así como la planeación de trayectorias con el uso de *MATLAB* y la *Robotics System Toolbox*, con la intención de facilitar la integración futura de otras estrategias de control.

Para la construcción de la representación virtual se elaboró primero un modelo CAD del robot manipulador en un programa de diseño asistido por computadora. Posteriormente, el modelo CAD se exportó a un archivo URDF (*Universal Robot Description Format*). En el apartado siguiente se presentan con mayor detalle los conceptos relacionados con este tipo de archivos.

8.1. Descripción de URDF

Un archivo URDF puede generarse a partir de un modelo CAD mediante herramientas de exportación o bien escribirse y editarse en un entorno de desarrollo integrado (IDE). Un archivo de este tipo se define como: "El URDF (*Universal Robotic Description Format*) es un formato basado en XML diseñado para describir la estructura de un robot: sus partes rígidas

(*links*), sus uniones móviles (*joints*) y, además, propiedades físicas, visuales y de colisión" (ROS.org, 2023).

El XML de un URDF contiene las siguientes especificaciones; es importante mencionar que no todas son estrictamente necesarias:

1. **Robot:** describe las propiedades generales del robot.
2. **Sensor:** especifica dispositivos tales como sensores, una cámara o sensor de distancia.
3. **Link:** detalla las propiedades cinemáticas y dinámicas de cada eslabón.
4. **Transmission:** conecta los actuadores a las juntas y representa su acoplamiento mecánico.
5. **Joint:** describe las propiedades cinemáticas y dinámicas de las articulaciones.
6. **Gazebo:** contiene parámetros específicos para la simulación en Gazebo.
7. **Model State:** indica el estado de un modelo en un momento determinado.
8. **Model:** define las propiedades cinemáticas y dinámicas de la estructura de un robot.

A continuación, se describen las especificaciones empleadas en el desarrollo de la representación virtual. Se comenzará por describir la articulación y sus atributos y elementos.

Joint (Articulación)

Una *joint* (articulación) es un elemento de unión que describe la cinemática y dinámica entre dos *links* (eslabones). En este elemento se pueden definir rangos de operación en radianes, así como la relación jerárquica entre la junta padre y la junta hijo. La Figura 8.1 muestra la relación y los marcos de referencia entre padre e hijo.

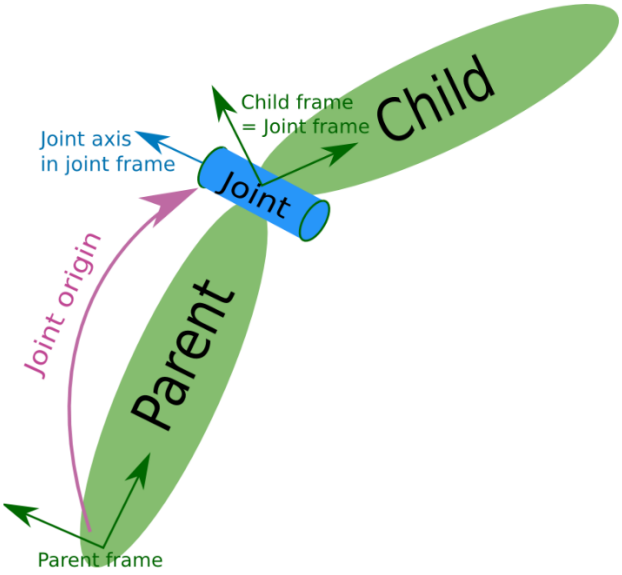


Figura 8.1: Relación padre e hijo y sus marcos de referencia

Fuente: ROS.org (2023).

La Figura 8.1 muestra la relación entre el padre y el hijo, así como la relación entre los marcos de referencia a partir de los cuales se pueden calcular las transformaciones homogéneas entre dichos marcos. Los atributos obligatorios de la junta son: `name`, que debe ser único por articulación, y `type`, que especifica el tipo de junta. Esta puede ser de revolución, prismática, fija, flotante, plana y continua. En el caso del modelo desarrollado, todas las juntas son de revolución.

En los elementos se pueden tener múltiples opciones dependiendo de lo que se desee modelar. En este trabajo es de interés analizar propiedades cinemáticas, por lo que se abordaron únicamente los elementos empleados en el modelo:

1. **Axis (eje):** es el eje de la articulación y determina el eje de rotación.

2. **Limit (límite):** existen dos, el inferior y el superior; estos atributos establecen el rango de movimiento de cada articulación.

Los atributos mencionados anteriormente son fundamentales para garantizar que las configuraciones cinemáticas sean consistentes con el robot físico. Existen además parámetros de carácter dinámico, por ejemplo, velocidad máxima, fricción y amortiguación, pero su modelado excede el alcance de este trabajo y por lo tanto no fueron incluidos en el desarrollo de este modelo.

Link

Un *link* o cuerpo rígido en robótica se entiende como un elemento de enlace que conecta articulaciones para formar una cadena cinemática. En el archivo URDF se pueden definir ciertas propiedades de cada *link*, entre las que destacan: aspectos dinámicos y de inercia, características visuales y propiedades de colisión. Estos atributos permiten representar tanto la geometría como el comportamiento físico del eslabón en simulación y análisis.

En el modelo desarrollado para este trabajo se consideraron únicamente los aspectos visuales de los *links*; la inclusión de parámetros dinámicos y de inercia se deja como mejora para trabajos futuros, con el fin de obtener una representación virtual más robusta y completa.

8.1.1. Diagrama de bloques de URDF del modelo

Se elaboró un diagrama de bloques para el modelo, basado en la estructura del URDF, con el fin de ejemplificar de manera clara la construcción de la representación virtual (Figura 8.2).

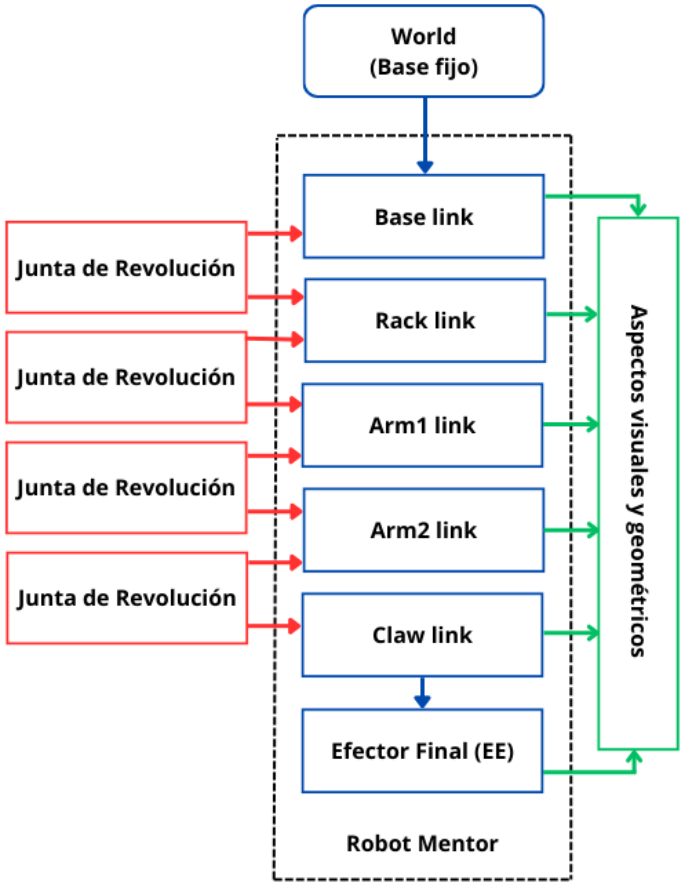


Figura 8.2: Diagrama de bloques del robot basado en la estructura del URDF

A partir de la Figura 8.2 se puede describir la estructura general de la representación virtual: primero se define un sistema de referencia fijo en la base y, a partir de este sistema, se construyen los demás marcos de referencia. El bloque denominado *Base link* es un cuerpo rígido que representa, en el modelo, la cadera; este tiene asignado un sistema de referencia que corresponde al marco del padre. Posteriormente, se une a través de una junta de revolución a su hijo, que en este caso se denomina *Rack link*. Este último ahora es padre del siguiente cuerpo rígido, de nombre *Arm1 link*, y así sucesivamente hasta llegar al efector final (*EE*). Cada cuerpo rígido tiene propiedades visuales y geométricas y, en algunos modelos, puede

incluir propiedades dinámicas; en este trabajo no se consideraron estas últimas.

8.2. Obtención de modelo cinemático a partir del archivo URDF

Es importante para el desarrollo de este apartado comprender algunos conceptos básicos para desarrollar el modelo cinemático del robot a partir del archivo URDF. Para ello, conviene recordar que una matriz de transformación homogénea "representa de manera conjunta la posición y orientación de un sólido en el espacio", de acuerdo con Barrientos et al. (2012). Esta se define mediante una matriz de dimensión 4×4 y representa la transformación de un vector de coordenadas homogéneas de un sistema a otro.

Dado que en el URDF se tiene una jerarquía de padre a hijo, existe una ecuación que determina la transformación homogénea entre el marco de referencia del padre y el de su hijo, la cual se muestra en la Ecuación 8.1.

$$^{parent}T_{child} = T_{origin}\, R_{axis}(q). \tag{8.1}$$

Donde:

- $^{\text{parent}}T_{\text{child}}$ es la matriz de transformación homogénea que describe la orientación del marco *child* con respecto al marco *parent*.
- T_{origin}: es la rotación fija y se determina mediante los ángulos de Euler (XYZ).
- $R_{axis}(q)$: es la rotación alrededor del eje de giro del marco de la junta.

- q: es la variable articular asociada a la junta.

En la Ecuación 8.1 se muestra el caso particular para obtener la transformación de una junta entre un par de cuerpos. Sin embargo, existe un modelo general para calcular la transformación homogénea a través de varias juntas; por ejemplo, la transformación entre el marco de referencia base y el efector final se muestra en la Ecuación 8.2.

$$ {}^{0}T_{n} = \prod_{i=1}^{n} \left({}^{i-1}T_{i}^{orig} \, Q_i(q_i) \right). \tag{8.2} $$

Donde:

- ${}^{0}T_{n}$ es la matriz de transformación homogénea total que describe la posición y orientación del marco n (efector final) con respecto al marco base 0.
- $\prod_{i=1}^{n}(\cdot)$ denota el producto ordenado de matrices de transformación homogénea a lo largo de la cadena cinemática, desde la primera hasta la n-ésima articulación.
- ${}^{i-1}T_{i}^{\text{orig}}$ es la transformación homogénea fija entre los marcos consecutivos $(i-1)$ e i, la cual define la posición y orientación inicial del marco de la junta i con respecto al marco anterior.
- $Q_i(q_i)$ es la transformación homogénea asociada al movimiento de la i-ésima articulación, definida como función de la variable articular q_i. Esta matriz representa la rotación o traslación producida por la junta.
- q_i es la variable articular correspondiente a la junta i.
- El subíndice n indica el número total de articulaciones que conforman el manipulador.

Es importante recalcar que la Ecuación 8.2 es la base con la que *MATLAB* realiza las transformaciones homogéneas para determinar la posición y orientación del efector final en función de las variables articulares. Haciendo uso de las ecuaciones antes mencionadas se obtiene el modelo cinemático del robot, el cual se muestra más adelante. Para ello, se comienza por calcular las transformaciones homogéneas para cada una de las articulaciones.

$$
\begin{aligned}
{}^{0}T_1(q_1) &= \begin{bmatrix} C_1 & -S_1 & 0 & 0 \\ S_1 & C_1 & 0 & 0 \\ 0 & 0 & 1 & 1.18 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad {}^{1}T_2(q_2) = \begin{bmatrix} C_2 & -S_2 & 0 & P_{x12} \\ 0 & 0 & -1 & P_{y12} \\ S_2 & C_2 & 0 & P_{z12} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\
{}^{2}T_3(q_3) &= \begin{bmatrix} C_3 & -S_3 & 0 & P_{x23} \\ S_3 & C_3 & 0 & P_{y23} \\ 0 & 0 & 1 & P_{z23} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad {}^{3}T_4(q_4) = \begin{bmatrix} C_4 & -S_4 & 0 & P_{x34} \\ S_4 & C_4 & 0 & P_{y34} \\ 0 & 0 & 1 & P_{z34} \\ 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{8.3}
$$

Donde:

- q_n: variable articular asociada a la junta.
- $C_n = \cos(q_n)$: coseno del ángulo de la variable articular.
- $S_n = \sin(q_n)$: seno del ángulo de la variable articular.
- P_{xnm}, P_{ynm}, P_{znm}: posición del sistema de referencia de la articulación con respecto del marco base, en metros.

Partiendo de la Ecuación 8.3 y la Ecuación 8.2 se puede deducir que la ecuación que modela el comportamiento cinemático de la representación virtual es el producto de las transformaciones homogéneas, quedando de manera general de la siguiente forma.

$$ {}^{0}T_{4}(q) = {}^{0}T_{1}(q_1)\,{}^{1}T_{2}(q_2)\,{}^{2}T_{3}(q_3)\,{}^{3}T_{4}(q_4). \tag{8.4} $$

Para validar el modelo antes propuesto se realiza la importación del modelo al espacio de trabajo de *MATLAB* y, mediante el comando `show`, se obtuvo la visualización general de la representación virtual que se muestra en la Figura 8.3.

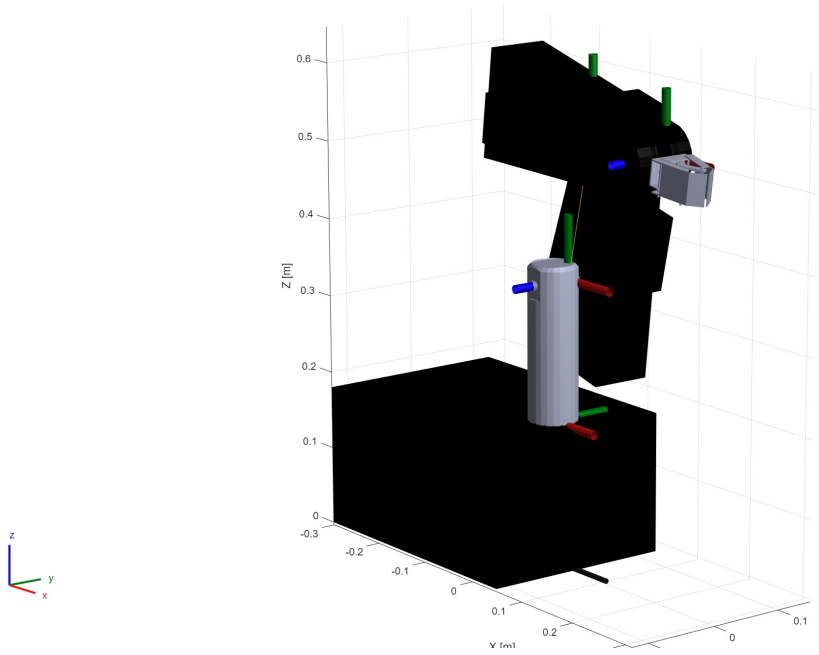


Figura 8.3: Importación de representación virtual al espacio de trabajo de MATLAB

Con esto se valida que no existe ningún error al asignar la jerarquía en el exportador de URDF. Posteriormente, se realiza un *script* en el que se programa el modelo cinemático propuesto y se asignan valores a las variables articulares para validarlo mediante la función `getTransform`, con la cual se puede obtener la transformación homogénea entre los marcos

de referencia de dos cuerpos a partir de una configuración dada. Se obtienen los siguientes resultados, donde se compara la posición del efector final con respecto al marco base.

Tabla 8.1: Validación del modelo cinemático vs `getTransform` de *MATLAB*

Valor de las variables articulares [rad].	**Posición del EE obtenidos con `getTransform` [m].**	**Posición del EE obtenidos con modelo cinemático [m].**
$q_1 = \frac{\pi}{2};\ q_2 = \frac{\pi}{16}$ $q_3 = \frac{\pi}{20};\ q_4 = \frac{\pi}{8}$	$x = -0.0148$ $y = 0.1548$ $z = 0.6017$	$x = -0.014$ $y = 0.154$ $z = 0.6017$
$q_1 = 0;\ q_2 = 0$ $q_3 = 0;\ q_4 = 0$	$x = 0.1988$ $y = 0.0148$ $z = 0.5433$	$x = 0.198$ $y = 0.014$ $z = 0.543$

Los resultados muestran que el modelo que describe la cinemática de la representación virtual, obtenido de manera analítica mediante transformaciones homogéneas, arroja resultados de posición y rotación consistentes con los obtenidos mediante funciones contenidas en la *Robotics System Toolbox* de *MATLAB*. Con lo anterior validado, es posible proseguir con el análisis y la planificación de trayectorias, así como con el análisis del espacio de trabajo, sabiendo que las transformaciones homogéneas están correctamente definidas. Con esta correspondencia establecida, se procede a usar alguna de las propiedades que tienen los modelos URDF para la planeación y seguimiento de trayectorias tema que se desarrolla en el siguiente capitulo.

8.3. Planeación del seguimiento de trayectorias con MATLAB y Toolbox

El seguimiento de trayectorias es uno de los elementos centrales del control en robótica, ya que es de suma importancia cuando se trata de realizar una aplicación en un entorno real donde existen otros cuerpos que limitan el movimiento del robot. Algunos autores, como Barrientos et al. (2012), definen que "el control cinemático establece cuáles son las trayectorias que debe seguir cada articulación del robot a lo largo del tiempo para lograr los objetivos fijados por el usuario". Esto se realiza desde un punto de origen hasta un punto destino con especificaciones tales como calidad, velocidad y aceleración, las cuales deben atender restricciones físicas propias de los actuadores.

El esquema que se propone para realizar el seguimiento de trayectorias se describe de manera general en la siguiente ilustración.

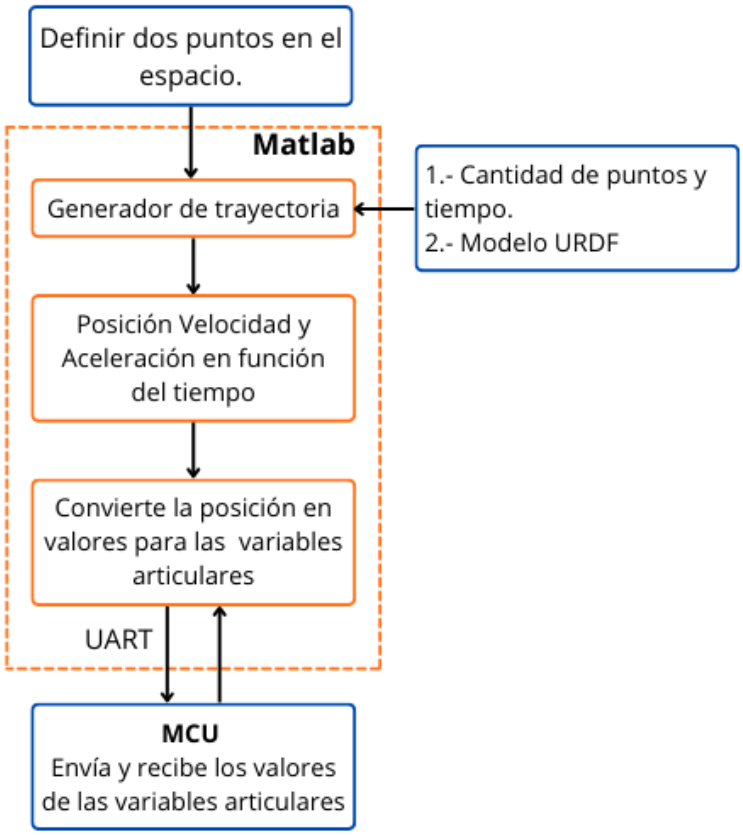


Figura 8.4: Esquema propuesto para analizar el seguimiento de trayectorias

Para explicar de manera más detallada el esquema propuesto para validar el seguimiento de trayectorias mostrado en la Figura 8.4, es necesario mencionar que se hace uso de algunas funciones contenidas en *Robotics System Toolbox*. Una de ellas es el generador de trayectorias `trapveltraj`, el cual genera una trayectoria con perfiles de aceleración trapezoidal.

La función recibe como parámetros el punto inicial y final para el cual se realizará la trayectoria, además del número de puntos entre el punto inicial y final. Por último, se especifica el tiempo total del movimiento. La función devuelve tres matrices: la primera con los ángulos de las variables articulares, la segunda con las velocidades articulares y la tercera

con la aceleración en cada instante de la trayectoria. En el desarrollo de este trabajo solo se utilizan las variables articulares, puesto que el esquema de control implementado no tiene retroalimentación de velocidad o aceleración y únicamente cuenta con retroalimentación de posición.

El siguiente paso consiste en enviar los valores de las variables articulares al MCU mediante comunicación UART, equiespaciados a lo largo del tiempo indicado como parámetro en la función `trapveltraj`. El MCU responde con el valor de las posiciones de las variables articulares mediante una interrupción periódica con una frecuencia de 33 Hz. MATLAB guarda los datos en una matriz para su posterior análisis.

El diseño del experimento consta de dos trayectorias: una va desde la posición *Home* a un punto en el espacio denominado *P1* y otra desde *P1* hacia un segundo punto denominado *P2*, como se ilustra en la siguiente figura.

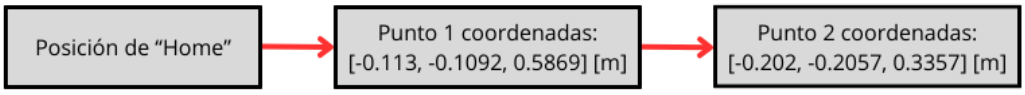


Figura 8.5: Flujo de la trayectoria planeada

Nota: Las coordenadas se expresan en el siguiente orden: $[x, y, z]$ [m] con respecto al marco de referencia situado en la cadera del robot.

Los resultados derivados de la implementación de este esquema se presentan y analizan en el Capítulo 9 correspondiente, donde se muestran de manera detallada los comportamientos obtenidos para cada una de las articulaciones del robot.

9. Resultados

En este capítulo se presentan los resultados obtenidos durante la implementación y validación del sistema de control del robot **MENTOR**, empleando la tarjeta de desarrollo **MINICON_STF4B**.

Los resultados expuestos a continuación demuestran el desempeño del sistema en términos de respuesta dinámica, estabilidad y correcta comunicación entre el entorno de supervisión y la plataforma embebida.

9.1. Análisis del desempeño del controlador PID en cada articulación

En esta sección se presenta la respuesta experimental del controlador PID implementado en la tarjeta MINICON_STF4B para cada una de las articulaciones del robot MENTOR.

En las Figuras 9.1–9.6 se muestran tres señales principales: el *setpoint* (línea roja discontinua), la señal de medición proveniente del sensor de posición (línea azul) y el error de control (línea magenta). Los cambios escalonados en la referencia permiten evaluar el comportamiento dinámico del sistema ante variaciones abruptas de posición.

Se evaluaron parámetros clásicos de desempeño en sistemas de control, tales como el tiempo de establecimiento, el sobreimpulso, el error en estado estacionario y la estabilidad ante cambios escalonados de referencia.

El tiempo de establecimiento observado en las diferentes articulaciones es reducido, ya que la señal de medición converge rápidamente al valor de referencia después de cada transición. El sobreimpulso es mínimo o prácticamente inexistente en la mayoría de los casos, lo que indica una adecuada relación entre las acciones proporcional y derivativa del controlador.

El error en estado estacionario tiende a cero una vez alcanzada la referencia, evidenciando la correcta acción integral del PID. Asimismo, no se observan oscilaciones sostenidas ni comportamiento inestable, lo que confirma que el sistema opera dentro de una región estable para las ganancias seleccionadas.

En conjunto, los resultados demuestran que la sintonización implementada permite un seguimiento preciso de la referencia, con transitorios controlados y comportamiento dinámico consistente en todas las articulaciones del robot.

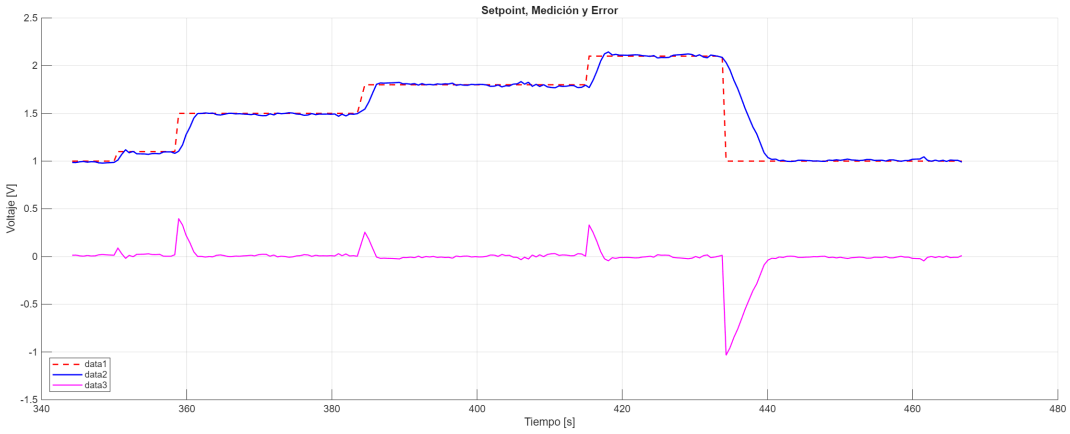


Figura 9.1: Análisis del comportamiento del controlador PID: Cadera

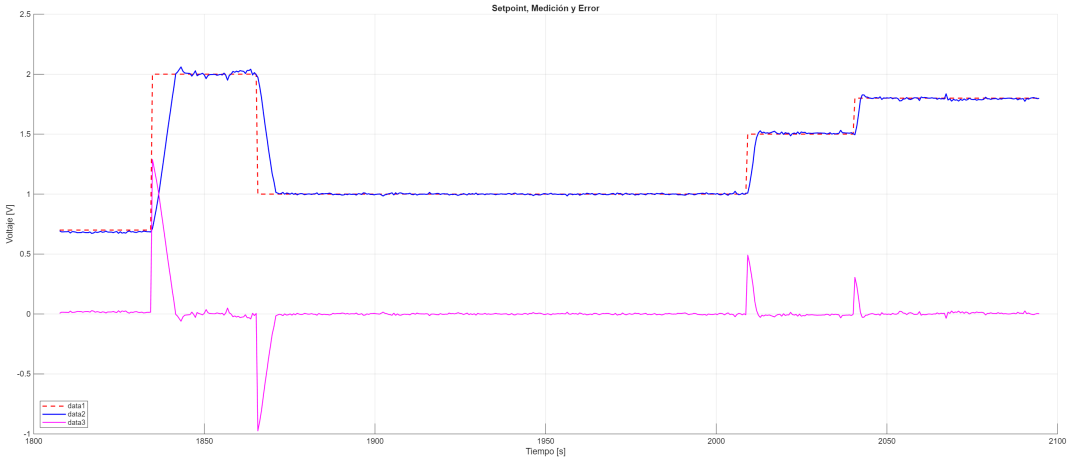


Figura 9.2: Análisis del comportamiento del controlador PID: Hombro

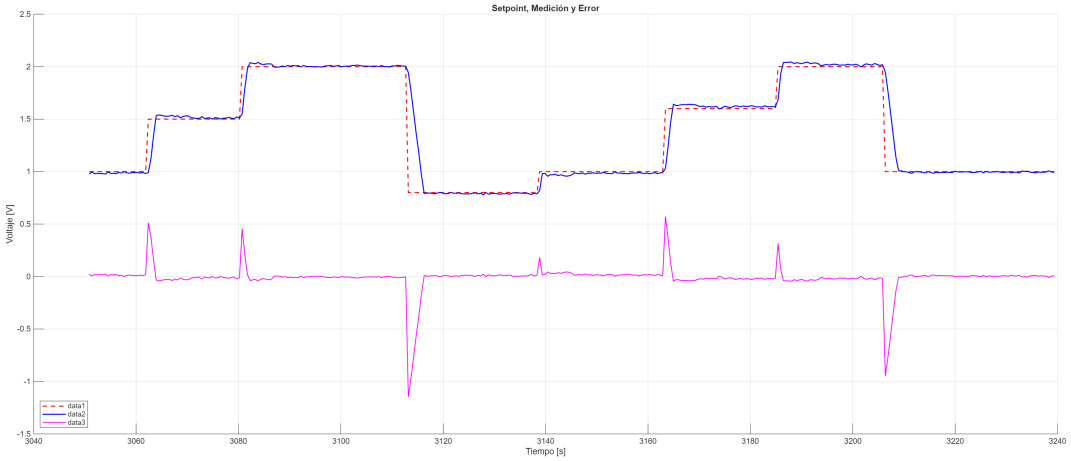


Figura 9.3: Análisis del comportamiento del controlador PID: Codo

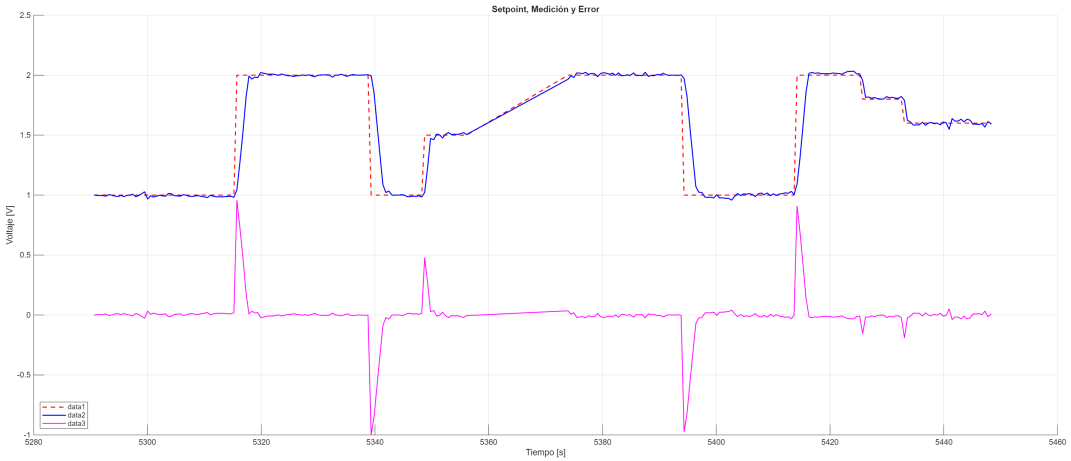


Figura 9.4: Análisis del comportamiento del controlador PID: Muñeca izquierda

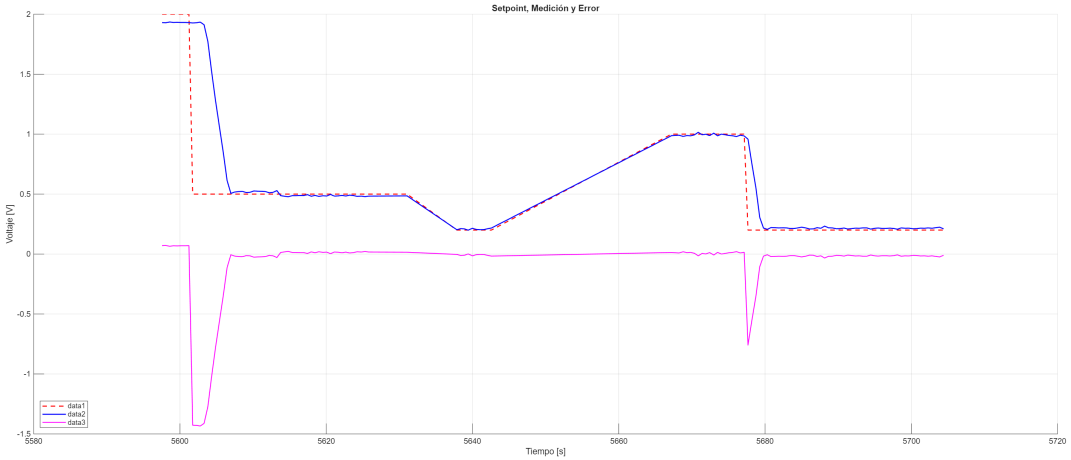


Figura 9.5: Análisis del comportamiento del controlador PID: Muñeca derecha

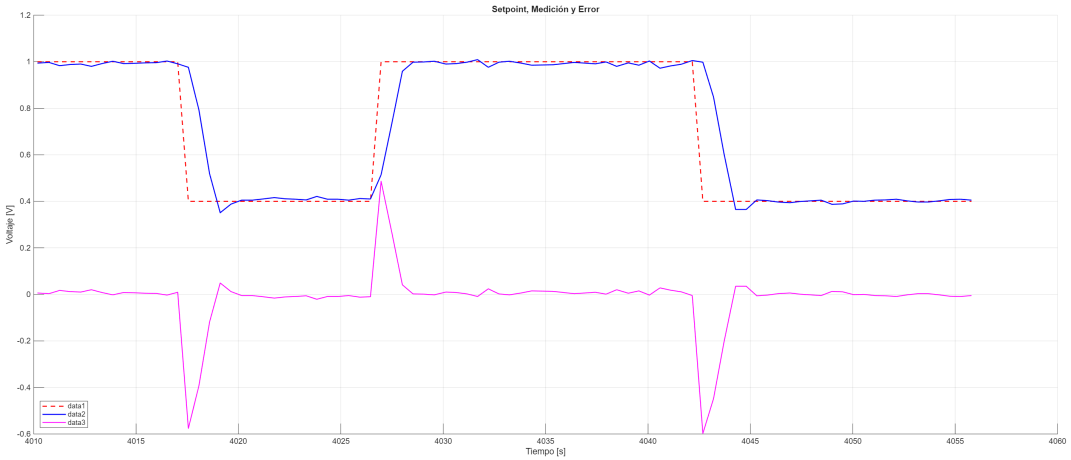


Figura 9.6: Análisis del comportamiento del controlador PID: Pinza

9.2. Espacio de trabajo obtenido mediante la representación virtual

En los apartados 2.1 y 2.2 de este trabajo se abordaron los conceptos básicos para el desarrollo de la representación virtual del robot antropomórfico MENTOR. De igual manera, se obtuvo un modelo analítico que describe la cinemática a partir del archivo URDF, y se verificó que fuera consistente y correcto mediante el uso de herramientas de cómputo. El

paso siguiente fue determinar el espacio de trabajo del robot, información clave para la implementación del programa de control y la HMI.

El cálculo del espacio de trabajo se realizó mediante el modelo URDF utilizando funciones contenidas en la *Robotics System Toolbox* de *MATLAB*. Para ello, fue importante definir en el archivo URDF los límites de las articulaciones derivados del análisis de hardware existente (Tabla 3.1). A continuación, se importó el modelo URDF al espacio de trabajo de *MATLAB* para ejecutar el análisis del espacio de trabajo con una cantidad considerable de puntos, con el fin de obtener un volumen que modele el espacio en el que el robot tiene alta maniobrabilidad. El resultado del análisis se muestra en la siguiente serie de ilustraciones.

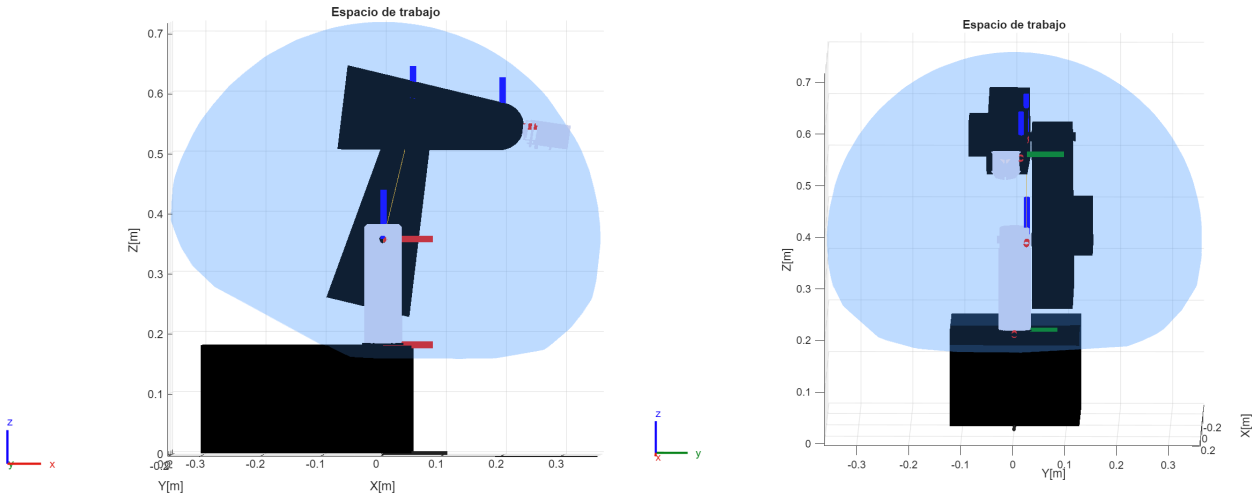


Figura 9.7: Espacio de trabajo modelado en *MATLAB* vista lateral y frontal

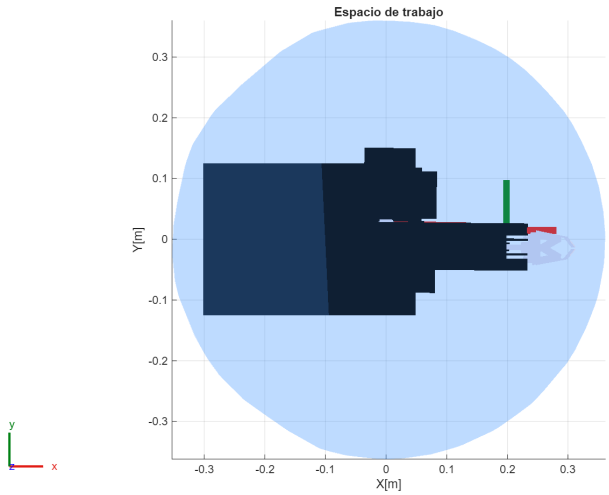


Figura 9.8: Espacio de trabajo modelado en *MATLAB* vista superior

Con base en los resultados obtenidos en las ilustraciones anteriores, se concluye que el espacio de trabajo calculado coincide con lo reportado para robots con configuración antropomórfica, como es el caso del robot MENTOR. Además, es muy similar al reportado por otros investigadores, alumnos de otras instituciones y trabajos de investigación publicados.

9.3. Validación experimental del seguimiento de trayectorias

Los resultados obtenidos de implementar el esquema descrito en el capitulo 8.3 consistieron en una simulación en MATLAB de la trayectoria propuesta y su posterior validación mediante la ejecución en el robot real.

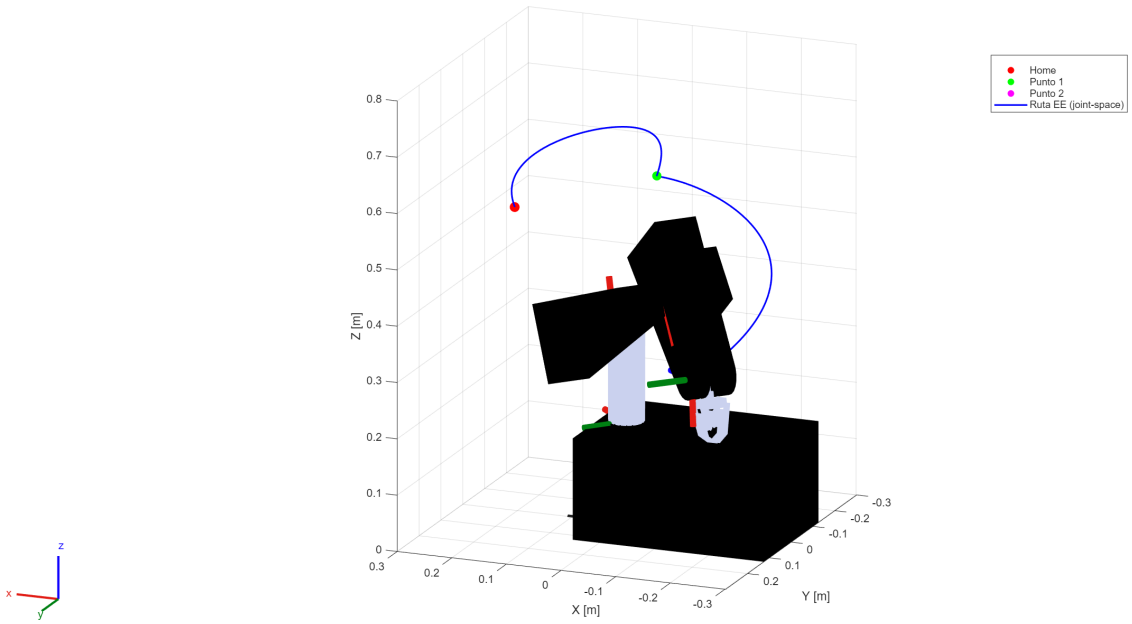


Figura 9.9: Simulación de la trayectoria propuesta

La figura anterior muestra la ruta esperada del efector final. Una vez terminada la simulación, se enviaron los datos al MCU y se graficó la trayectoria propuesta contra la trayectoria realizada para cada una de las articulaciones.

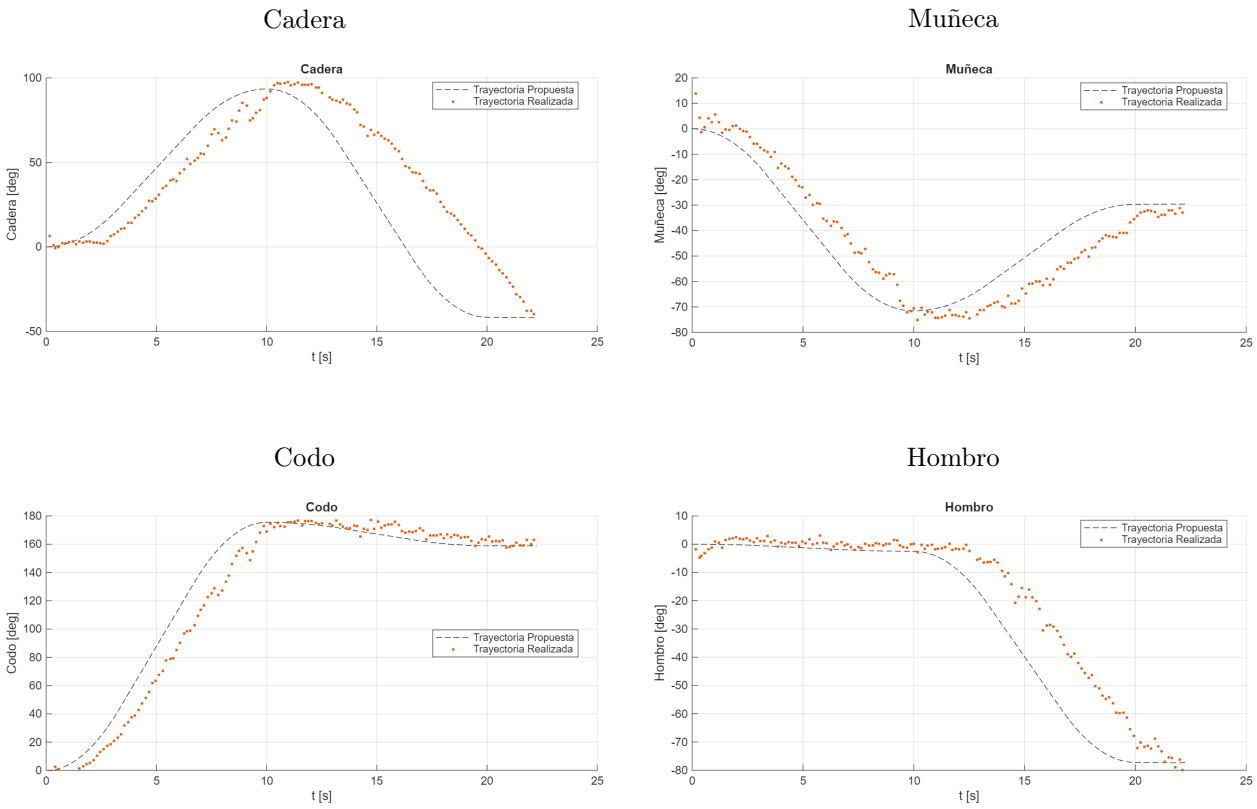


Figura 9.10: Gráficas de seguimiento de trayectorias para las articulaciones del robot

Al comparar los resultados obtenidos se observa que el robot sigue, de manera general, la forma del movimiento propuesto. No obstante, presenta ciertos retrasos y desviaciones respecto a la referencia, los cuales pueden atribuirse a limitaciones del controlador implementado.

Como se mencionó previamente, el controlador únicamente cuenta con retroalimentación de posición; el seguimiento podría mejorarse si además se incorpora retroalimentación de velocidad. Otra mejora posible consiste en considerar los datos de aceleración generados por el planificador de trayectorias, los cuales fueron omitidos en esta implementación.

Existen además factores dinámicos que afectan el seguimiento, tales como fricción, inercia y centros de masa. En conjunto, estas consideraciones representan oportunidades

de mejora para incrementar el rendimiento del controlador y obtener un seguimiento de trayectoria más preciso.

9.4. Fabricación y validación experimental del hardware

Las Figuras 9.11–9.13 presentan el desarrollo físico de la tarjeta MINICON_STF4B, la cual fue diseñada, fabricada mediante proceso industrial de PCB y posteriormente ensamblada para su validación experimental.

Posteriormente, se llevó a cabo el montaje de los componentes electrónicos, incluyendo el microcontrolador STM32F446RE, reguladores de tensión, etapas de acondicionamiento analógico y módulos de potencia tipo puente H.

Durante la fase de pruebas se evaluó el comportamiento eléctrico bajo operación continua, verificando estabilidad en las tensiones de alimentación, integridad de señales PWM, ausencia de ruido significativo en las entradas analógicas y correcto funcionamiento del sistema de adquisición. Asimismo, se realizaron pruebas de carga prolongadas en los actuadores, monitoreando temperatura en los controladores de potencia y en las pistas de alimentación.

Después de múltiples ciclos de operación y ensayos dinámicos con el robot en movimiento, no se detectaron fallos funcionales, reinicios inesperados ni sobrecalentamientos en los componentes críticos. La disipación térmica se mantuvo dentro de rangos seguros y no se observaron degradaciones en el desempeño eléctrico.

Estos resultados validan la robustez del diseño del PCB, la adecuada selección de

componentes y la correcta implementación del sistema de potencia y control, confirmando la confiabilidad del hardware desarrollado para el robot MENTOR.

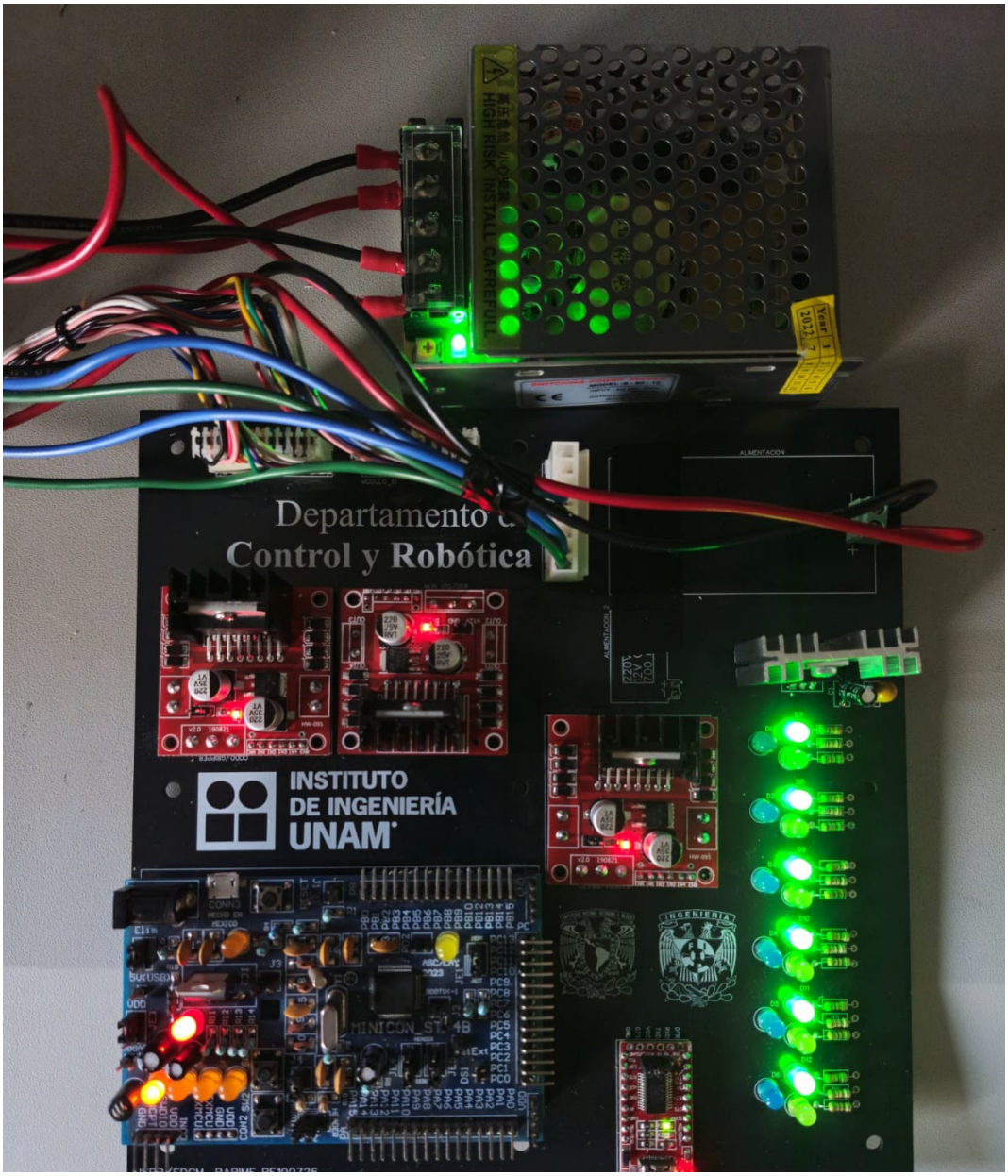


Figura 9.11: Montaje físico del sistema MINICON_STF4B durante pruebas experimentales.

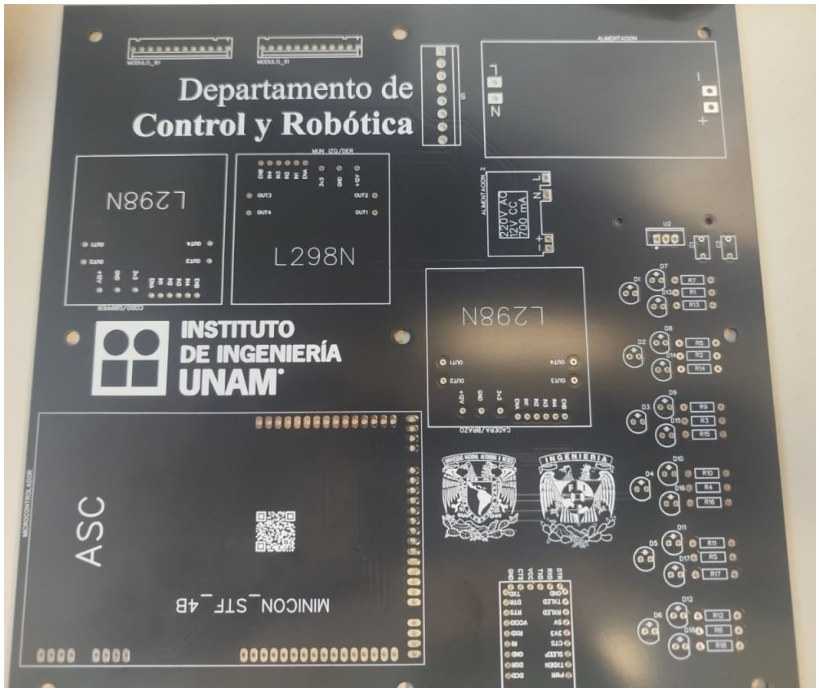


Figura 9.12: Vista frontal del PCB MINICON_STF4B fabricado y serigrafiado.

Figura 9.13: Vista posterior del PCB mostrando el ruteo de pistas y distribución eléctrica.

9.5. Interfaces de la aplicación de supervisión y control

La Figura 9.14 muestra la pantalla principal de la aplicación desarrollada en *MATLAB App Designer* para el control del robot MENTOR. La interfaz se organiza en cinco módulos funcionales, los cuales estructuran el flujo de operación del sistema y facilitan su identificación durante la presentación de resultados.

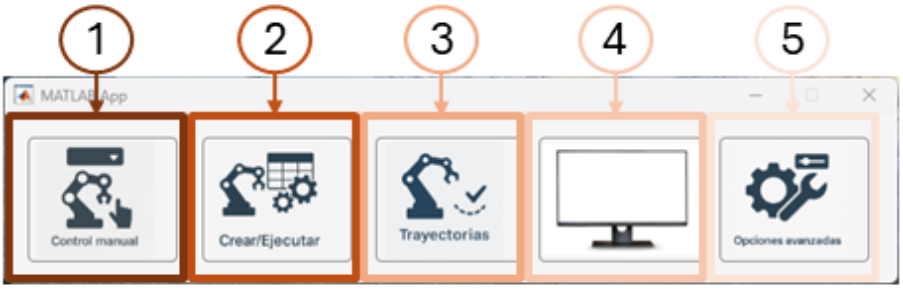


Figura 9.14: Interfaz principal de la aplicación MATLAB para control, programación y supervisión del robot.

1. **Control manual:** permite el accionamiento directo del robot mediante deslizadores o controles individuales por articulación. Se utiliza para pruebas básicas, posicionamiento manual y verificación del funcionamiento de cada eje.

2. **Crear/Ejecutar:** módulo destinado a la programación por puntos (*point-to-point*). Permite registrar posiciones articulares, almacenarlas en una secuencia y ejecutarlas de forma automática.

3. **Trayectorias:** sección orientada a la generación y validación de trayectorias planificadas. Permite definir movimientos continuos y evaluar su comportamiento antes de la ejecución física.

4. **Simulación / Monitorización:** corresponde al entorno de visualización del modelo virtual (URDF), donde se observa en tiempo real el movimiento del robot y la respuesta del sistema de control.

5. **Opciones avanzadas:** área destinada a configuraciones técnicas adicionales, como ajuste de parámetros de control, calibración o configuraciones del sistema.

Posteriormente, se presentan de manera individual las interfaces de cada módulo (Figuras 9.15 a 9.20), con el propósito de detallar su estructura, funcionalidad y los elementos de control implementados en el software final del robot MENTOR.

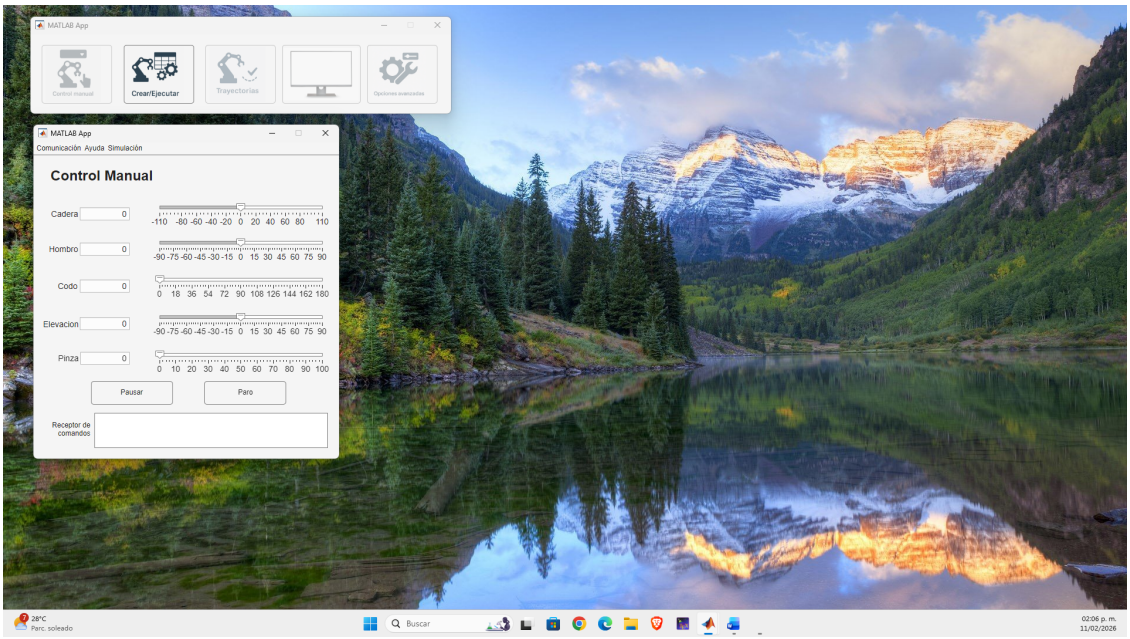


Figura 9.15: Interfaz del módulo Control Manual.

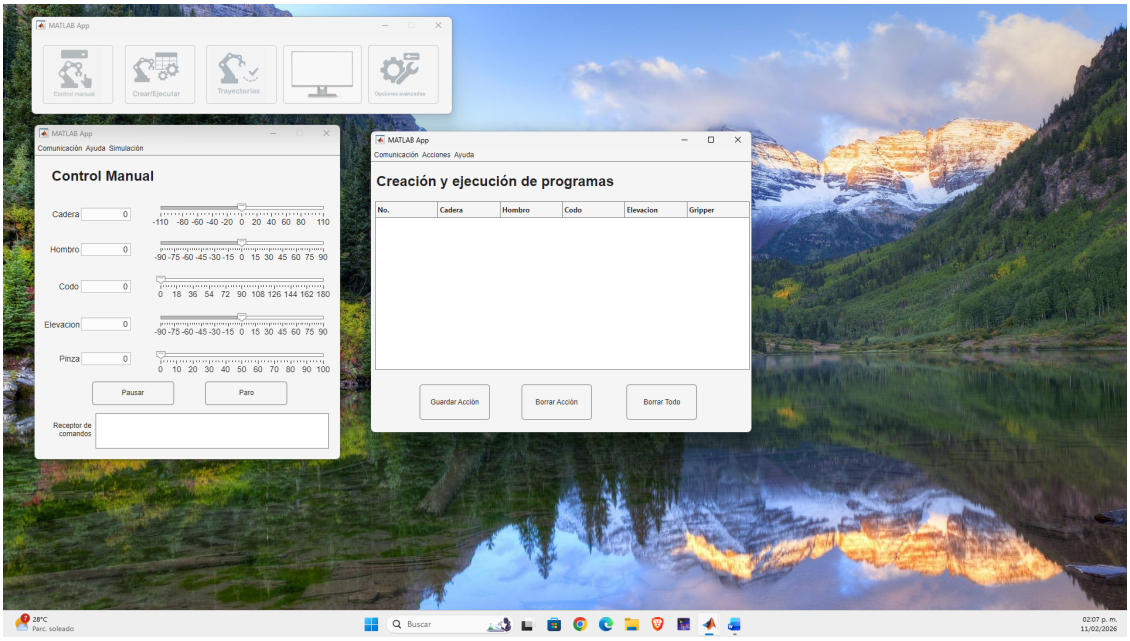


Figura 9.16: Interfaz del módulo Crear/Ejecutar.

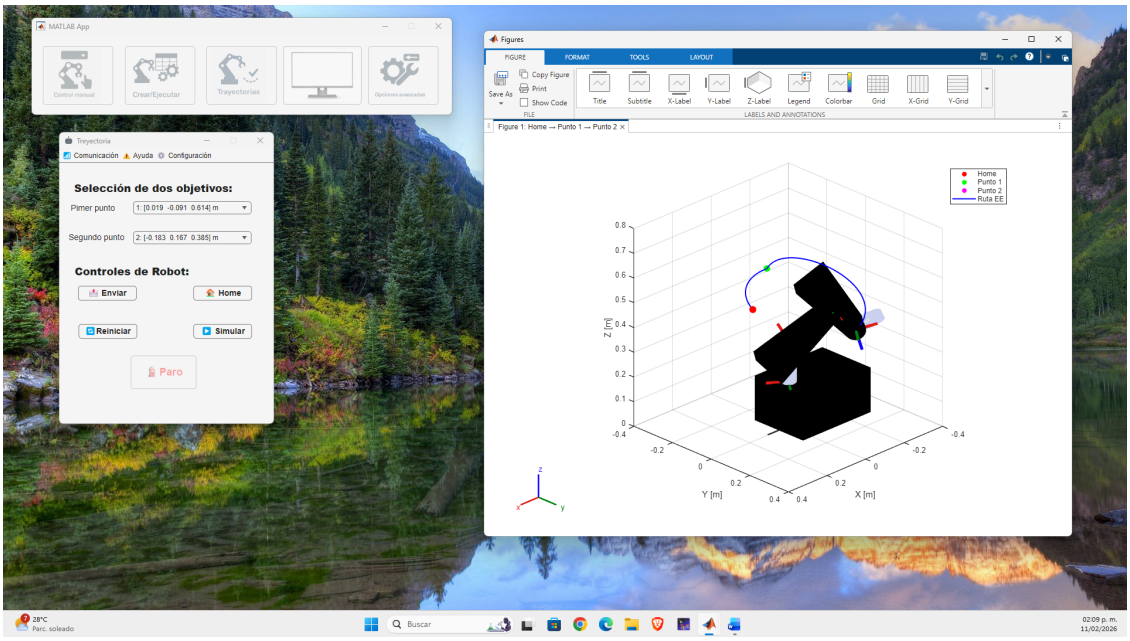


Figura 9.17: Interfaz del módulo Trayectorias.

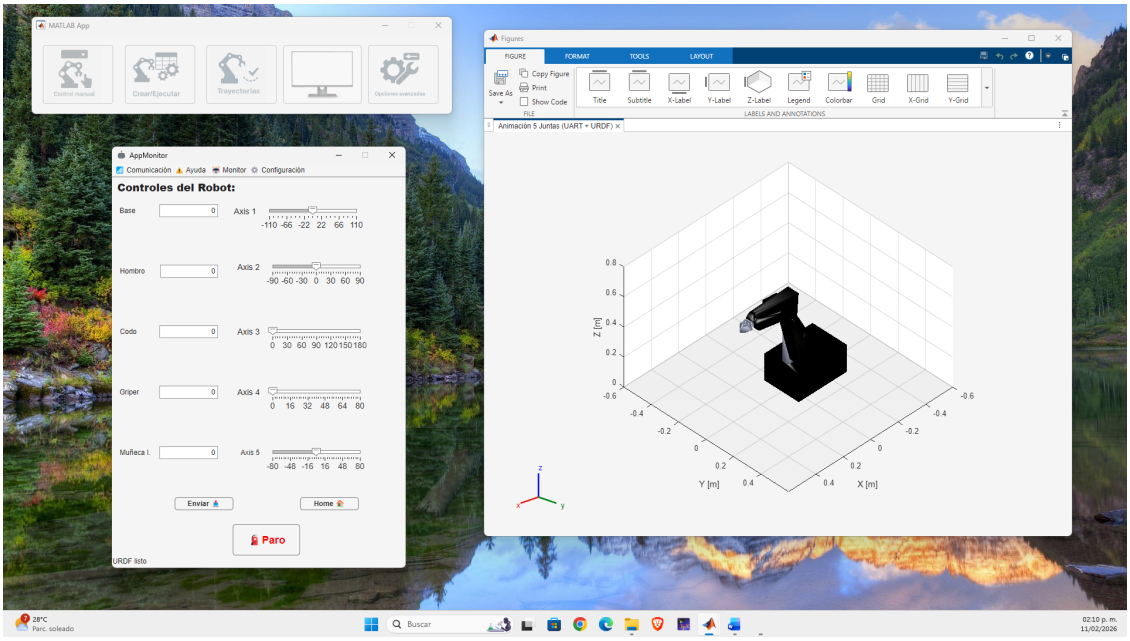


Figura 9.18: Interfaz del módulo de Monitorización.

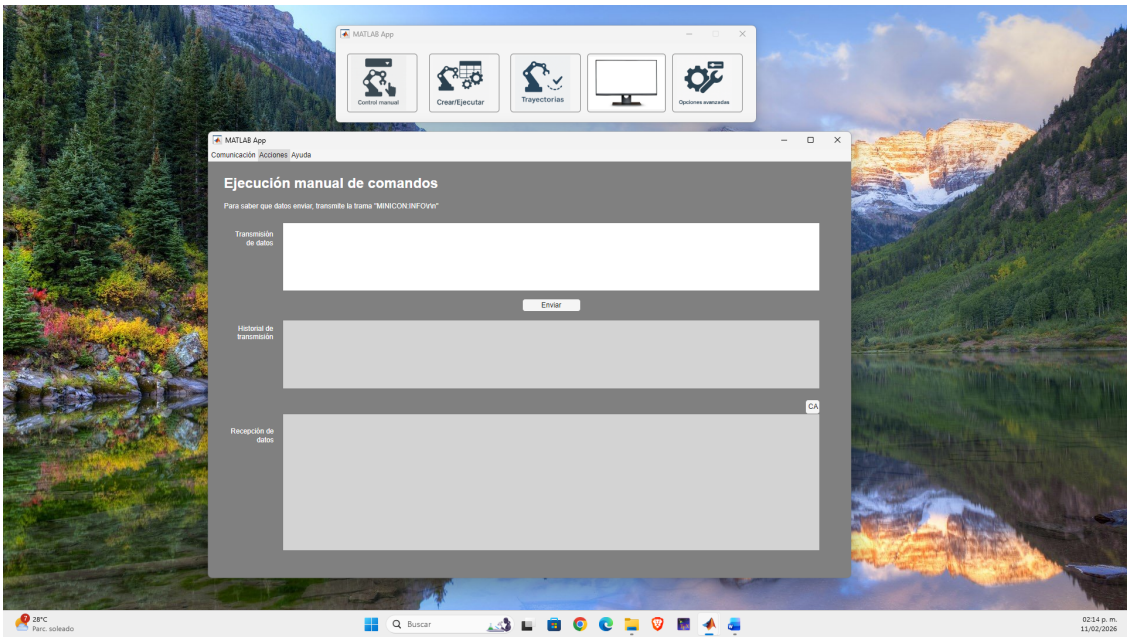


Figura 9.19: Interfaz del módulo Opciones Avanzadas.

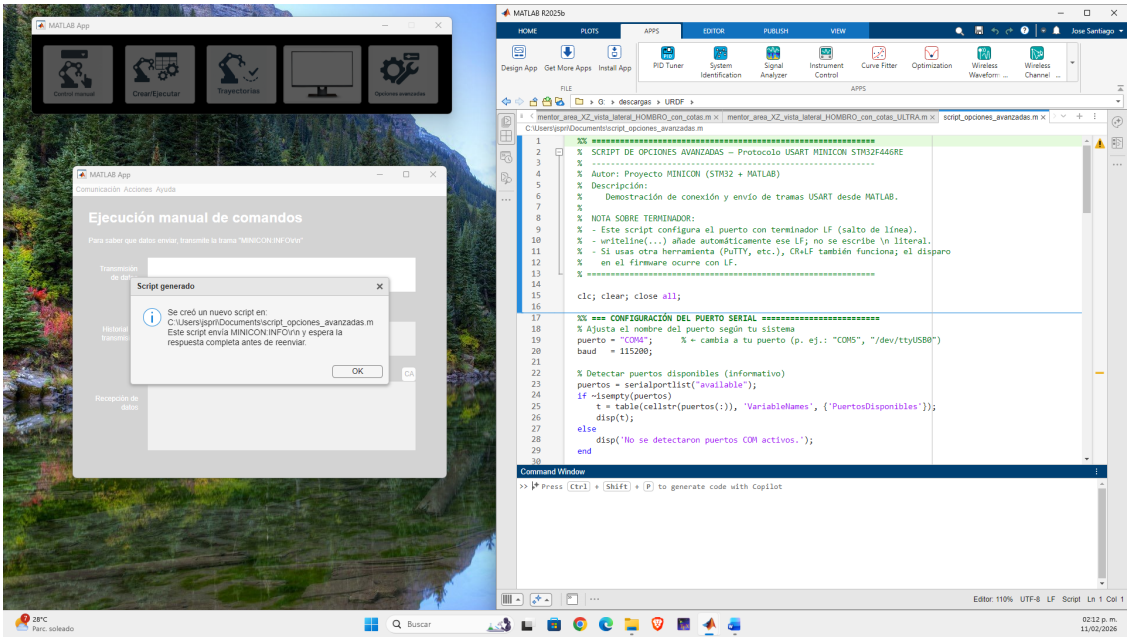


Figura 9.20: Interfaz complementaria del sistema de supervisión (vista adicional).

En el presente capítulo se expusieron los resultados obtenidos a lo largo del desarrollo de esta tesis. Se presentaron las curvas de comportamiento, las señales de control generadas y las ganancias finales empleadas en cada eje, estableciendo así la evidencia cuantitativa del desempeño alcanzado por el sistema. Con la información experimental ya documentada, el capítulo siguiente desarrolla el análisis de resultados, donde se interpretan los datos obtenidos, se evalúa el comportamiento dinámico del robot y se contrastan los resultados con los objetivos planteados al inicio de este documento.

10. Análisis de resultados

Para concluir este trabajo se presenta un resumen de los resultados más relevantes obtenidos a lo largo de los capítulos anteriores. Estos hallazgos se agrupan en seis grandes apartados que reflejan las principales aportaciones técnicas y metodológicas del presente trabajo.

El primer apartado corresponde al desarrollo del sistema de control. Se implementó un sistema de control embebido con retroalimentación de posición para cada una de las articulaciones del robot; es decir, para cada eje acoplado a su respectivo motor. Se determinaron las ganancias adecuadas para garantizar un comportamiento estable y eficiente del controlador, optimizando el desempeño de cada articulación bajo las condiciones de operación establecidas. Esta etapa permitió validar la funcionalidad del control independiente por articulación y sentó las bases para el desarrollo de las interfaces de operación.

El segundo apartado se refiere al diseño, desarrollo e integración del hardware necesario para la implementación del sistema de control. Para ello, se seleccionaron y ensamblaron diversos módulos: el convertidor USB–serial, los controladores para los motores, indicadores luminosos, fuente de poder y reguladores de voltaje. Todos estos componentes se integraron en una sola tarjeta mediante el diseño de un circuito impreso, realizado con software especializado. Tanto el esquemático como los archivos de fabricación se incluyen en los anexos de este documento. La Figura 10.1 muestra el resultado final del PCB ya ensamblado y en operación.

Este diseño representa una mejora en organización, mantenimiento y fiabilidad del

sistema, y permite la integración o desarrollo de nuevas estrategias de control.

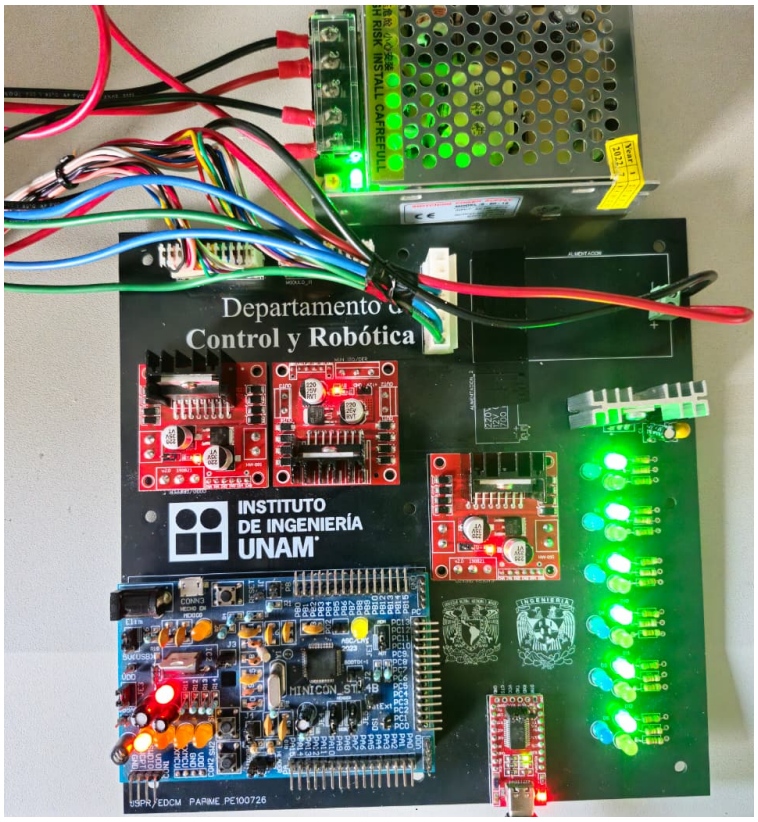


Figura 10.1: Circuito impreso desarrollado para control de robot MENTOR

El tercer resultado corresponde al desarrollo de la representación virtual del robot MENTOR. Esta herramienta puede ser utilizada dentro del entorno de *MATLAB* y permite simular diferentes estrategias de control, analizar el espacio de trabajo, planear trayectorias con obstáculos y restricciones, evaluar distintos solucionadores cinemáticos y validar configuraciones antes de ejecutarlas en el robot físico. Este modelo no solo acelera el proceso de diseño y prueba, sino que también reduce riesgos operativos y facilita la enseñanza de conceptos básicos de robótica en el laboratorio.

El cuarto resultado consiste en la documentación generada, en la cual se describió con detalle la estructura mecánica, las conexiones eléctricas, la distribución de pines, los sensores

utilizados y los rangos de operación de cada articulación. Esta documentación ofrece una referencia sólida para futuras mejoras, mantenimiento, ampliaciones del sistema o incluso el desarrollo de nuevas soluciones basadas en la plataforma desarrollada, tanto a nivel de software como de hardware.

El quinto resultado, uno de los más relevantes para los objetivos planteados en este trabajo, es el desarrollo de un entorno de software para la operación integral del robot. Este incluye cinco aplicaciones principales, enumeradas a continuación:

1. Control manual.
2. Creación y ejecución de programas.
3. Modo monitor.
4. Seguimiento de trayectorias.
5. Funciones avanzadas.

Cada una de estas aplicaciones fue analizada en capítulos anteriores y constituye una herramienta de gran utilidad tanto para la operación directa del robot MENTOR como para la creación de nuevo material didáctico, así como para la ejecución y desarrollo de prácticas de laboratorio. Con este entorno se habilita una interacción más intuitiva con el robot, se favorece el aprendizaje, la experimentación y el desarrollo de futuros proyectos académicos y de investigación.

Por último, se propone un manual de prácticas que abarca temas fundamentales para la enseñanza de la robótica y que están alineados con los contenidos del plan de estudios vigente. Se espera que este material sea integrado al desarrollo de las clases en el Laboratorio

de Control Robótica, con el propósito de ilustrar tanto conceptos básicos como otros de mayor profundidad, fortaleciendo así el proceso de aprendizaje y la aplicación práctica de los conocimientos impartidos.

11. Conclusiones y trabajo a futuro

11.1. Conclusiones

El desarrollo presentado en este trabajo demuestra que la modernización integral de un robot antropomórfico obsoleto puede lograrse mediante la integración de técnicas contemporáneas de diseño de hardware y software embebido. El robot MENTOR, originalmente concebido para un entorno computacional analógico–digital de los años ochenta, fue completamente rehabilitado mediante la sustitución total de su arquitectura electrónica, la implementación de un sistema de control de posición basado en PID y la creación de una plataforma de software moderna compatible con *MATLAB*.

Una aportación importante del proyecto fue la creación de una representación digital del robot en formato URDF. Si bien este modelo no contiene parámetros dinámicos ni una descripción cinemática completa, sí implementa los elementos estrictamente necesarios para visualizar el movimiento en tiempo real: los orígenes geométricos de cada eje de giro, los límites mecánicos correspondientes a cada articulación y el mapeo directo entre el voltaje medido por el microcontrolador y la posición angular mostrada en la interfaz gráfica.

El controlador PID implementado en cada articulación fue optimizado incorporando funciones como *anti-windup* y banda muerta tanto en el error como en la salida. Esto permitió corregir problemas característicos del sistema original: oscilaciones, ruido en la realimentación, inestabilidad y errores residuales de posicionamiento. La validación experimental demostró que las ganancias seleccionadas permiten un comportamiento estable y con tiempos

de asentamiento adecuados para aplicaciones académicas.

El software desarrollado en *MATLAB* complementa la solución proporcionando interfaces intuitivas para control manual, monitoreo cinemático en tiempo real, ejecución estructurada de programas y seguimiento de trayectorias. Estas herramientas transforman al robot MENTOR en una plataforma alineada con la enseñanza universitaria actual, facilitando su incorporación a prácticas de laboratorio del Departamento de Control y Robótica.

11.2. Trabajo a futuro

El sistema desarrollado durante este proyecto permitió rehabilitar, digitalizar y modernizar el robot antropomórfico MENTOR mediante la integración de electrónica embebida, algoritmos y herramientas de software contemporáneas. No obstante, el potencial de expansión tecnológica del robot es amplio y puede fortalecerse mediante la incorporación de técnicas avanzadas de control, simulación, modelado y supervisión. El presente capítulo propone un conjunto de líneas de mejora cuyo propósito es ampliar la funcionalidad, precisión, seguridad y capacidad didáctica del sistema, tanto en el plano físico como virtual.

- **Modelado cinemático y dinámico completo en URDF o SDF:** el URDF actual describe únicamente orígenes, ejes y límites. Incluir masas, inercias, longitudes de eslabones y actuadores permitiría simulaciones físicamente realistas en *MATLAB*, habilitando análisis de cargas y trayectorias avanzadas.

- **Sistema de calibración automática de articulaciones:** podría incluir rutinas de búsqueda de topes mecánicos y estimación de no linealidades de los potenciómetros. Esto reduce errores acumulados y mejora la precisión del modelo virtual.

- **Adaptación de nuevos efectores finales y sensores ópticos para entrenamiento de lógica avanzada:** se propone una modificación en la estructura mecánica del robot MENTOR que permita la integración de distintos efectores finales y, con ello, implementar un sistema modular de efectores que amplíe significativamente las aplicaciones académicas del robot. Esto posibilitaría tareas complejas como ensamble, manipulación precisa de objetos, clasificación, reconocimiento de patrones o manipulación asistida por visión. En el mismo rubro, la incorporación de sensores ópticos habilitaría la creación de rutinas de lógica avanzada, seguimiento de objetos y retroalimentación sensorial adicional para algoritmos de control. Esta combinación convertiría al robot en una plataforma completa para enseñar conceptos de percepción, decisión y reacción automatizada.

Referencias

Abdul-Hamid, B. M. (2007). *Sistemas discretos de control*. Visión Net.

Barrientos, A., Peñín, L., Balaguer, C., and Aracil, R. (2012). *Fundamentos de robótica*. McGraw-Hill, Madrid, 2 edition.

Chinnasamy, S. K., Sura, H. P., Saleem, A., Kathirvel, A., and Rangan, P. (2023). Digital twin of robot manipulator using ros. *AIP Conference Proceedings*, 2946(1).

Franklin, G. F., Powell, J. D., and Workman, M. (1997). *Digital control of dynamic systems*. Addison-Wesley, 3 edition.

Gupta, A. and Charan, C. (2024). Analysis of universal asynchronous receiver-transmitter (uart). In *Proceedings of the 2nd IEEE International Conference on Device Intelligence, Computing and Communication Technologies (DICCT)*, pages 194–198.

Handson Technology (2018). L298n dual h-bridge motor driver user guide.

Laskawski, M. and Wcislik, M. (2016). Sampling rate impact on the tuning of pid controller parameters. *International Journal of Electronics and Telecommunications*, 62(1):43–48.

Massaccesi, D. (2014). *Técnicas de control automático: Un estudio comparativo*. Editorial Académica Española.

MathWorks (s/f). App designer. `https://www.mathworks.com/products/matlab/app-designer.html`. Consultado el 29 de enero de 2026.

Mazidi, M. A., Chen, S., and Ghaemi, E. (2018). *STM32 ARM programming for embedded systems: Using C language with STM32 Nucleo*. Pearson.

Nithyasree, M. and Kandasamy, K. V. (2012). A generic pid controller based on arm processor. *Procedia Engineering*, 38:1044–1049.

Ogata, K. (2010). *Ingeniería de control moderna*. Pearson, 5 edition.

Ogata, K., Aranda Pérez, G., Rodríguez Ramírez, F., and Sánchez García, G. (1996). *Sistemas de control en tiempo discreto*. Prentice-Hall.

ON Semiconductor (2014). Switch-mode power supply reference manual.

ROS.org (2023). Urdf (universal robot description format). `https://wiki.ros.org/urdf`. Consultado el 29 de enero de 2026.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Robotics: Modelling, planning and control*. Springer.

Song, Y., Huang, X., and Wen, C. (2017). Robust adaptive fault-tolerant pid control of mimo nonlinear systems with unknown control direction. *IEEE Transactions on Industrial Electronics*, 64(6):4876–4884.

Xiang, K., Song, Y., and Ioannou, P. (2025). Nonlinear adaptive pid control for nonlinear systems. *IEEE Transactions on Automatic Control*, 70(10):7000–7007.

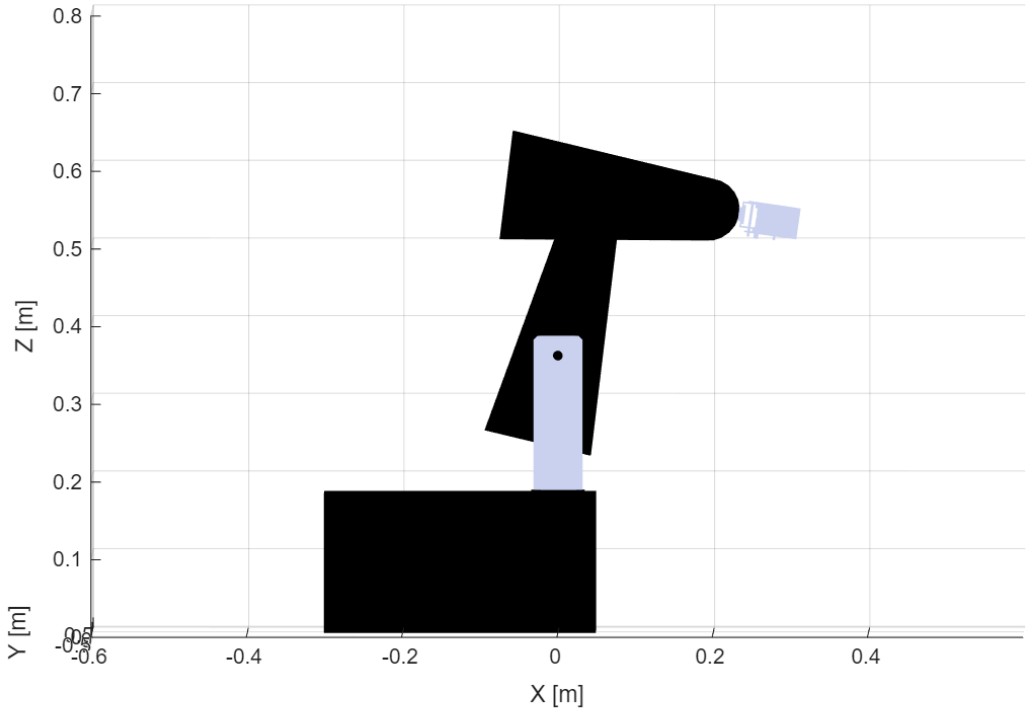
Zhao, M. (2025). A review of pid controller implementation methods: From analog computers to digital systems. *Applied and Computational Engineering*, 169(1):60–67.

ANEXOS

MANUAL DE PRÁCTICAS DEL LABORATORIO DE ROBÓTICA INDUSTRIAL

Práctica Nº 1

Robots industriales: estructura, componentes y entorno de operación



Nombre completo del alumo		Firma	Calificación
Nº de brigada:_____	Fecha de elaboración:_____	Grupo:	

1 Práctica Nº 1: Robots industriales: estructura, componentes y entorno de operación

1.1 Objetivos

El alumno describirá los fundamentos de la robótica industrial, identificando sus antecedentes históricos, los componentes y la estructura de los robots industriales, así como los conceptos básicos de las celdas de trabajo robóticas empleadas en la industria.

1.2 Recursos

Software

- Entorno de desarrollo y ejecución MATLAB R2025a o superior.
- Sistema operativo Windows 10 o superior.

Recursos de equipo

- Computadora de escritorio.
- Robot antropomórfico de cinco grados de libertad (5 GDL), modelo MENTOR.

1.3 Seguridad en la ejecución de la actividad

Peligro o fuente de energía	Riesgo asociado	Medidas de control
Energía eléctrica (alimentación del sistema)	Electrocución	Verificar las conexiones y los niveles de voltaje antes de energizar el sistema y evitar el contacto con partes energizadas.
Energía eléctrica en corriente continua	Daño al equipo	Verificar la polaridad y el nivel de tensión antes de realizar la conexión de dispositivos o componentes.
Movimiento mecánico del robot (articulaciones y efector final)	Golpes o atrapamiento	Mantener manos y objetos fuera del área de trabajo del robot y no intervenir durante la ejecución del movimiento.

1.4 Introducción

La manufactura industrial moderna se encuentra en un proceso de transformación acelerada impulsado por la demanda de productos altamente personalizados, ciclos de vida cada vez más cortos y una presión constante

por reducir costos sin sacrificar calidad. En este contexto, los procesos de manufactura tradicionales, basados en máquinas herramienta dedicadas y altamente especializadas, comienzan a mostrar limitaciones frente a escenarios productivos caracterizados por la flexibilidad y la reconfiguración continua. Entre dichos procesos, el maquinado ocupa un lugar central, ya que constituye una de las etapas más críticas en la transformación de materias primas en productos finales con tolerancias estrictas y acabados superficiales controlados.

Históricamente, las máquinas de control numérico computarizado (CNC) han sido la solución predominante para operaciones de maquinado debido a su elevada rigidez estructural, precisión geométrica y estabilidad dinámica. No obstante, estas ventajas vienen acompañadas de costos elevados de adquisición, instalación y mantenimiento, así como de una limitada versatilidad frente a cambios frecuentes en el diseño del producto o en el proceso. Esta situación ha motivado la exploración de alternativas tecnológicas capaces de ofrecer un equilibrio entre precisión, flexibilidad y rentabilidad, dando lugar al creciente interés en el uso de robots industriales como plataformas de maquinado.

Los robots industriales han sido ampliamente utilizados durante décadas en tareas como manipulación de materiales, soldadura, pintura, ensamblaje y paletizado. Su popularidad se debe, en gran medida, a su alta flexibilidad, amplio espacio de trabajo, facilidad de reprogramación y capacidad para integrarse en sistemas automatizados complejos. Sin embargo, el uso de robots industriales en aplicaciones de maquinado representa un desafío significativo, ya que estas tareas imponen exigencias mecánicas y dinámicas considerablemente mayores que las aplicaciones tradicionales.

El concepto de maquinado robótico surge formalmente a finales de la década de 1980, cuando se propuso utilizar robots industriales para sustituir a operadores humanos en tareas de desbaste, perforado, rectificado y rebabado. Desde entonces, la investigación en este campo ha evolucionado de manera notable, impulsada por los avances en sensores, sistemas de control, modelado dinámico y planificación de trayectorias. A diferencia de las máquinas CNC, los robots industriales presentan una estructura serial con múltiples grados de libertad, lo que les confiere una gran capacidad de movimiento, pero al mismo tiempo introduce problemas asociados con baja rigidez, errores de posicionamiento y vibraciones inducidas durante el proceso de corte.

Uno de los principales factores que limita la adopción industrial del maquinado robótico es la relación entre la rigidez estructural del robot y la tasa de remoción de material. En operaciones de baja tasa de remoción, como pulido, desbarbado y rectificado fino, los robots industriales han demostrado un desempeño competitivo e incluso superior al trabajo manual, especialmente cuando se integran sistemas de control basados en fuerza y retroalimentación sensorial. En contraste, las operaciones de alta tasa de remoción, como fresado y taladrado, exigen niveles de rigidez y estabilidad dinámica que superan las capacidades inherentes de muchos robots industriales convencionales.

El desempeño de un sistema de maquinado robótico no depende únicamente del manipulador, sino del conjunto de elementos que conforman la celda de trabajo. Una celda de maquinado robótico integra el manipulador industrial, el husillo o herramienta de corte, el sistema de sujeción de la pieza, los sensores, el sistema de control y el entorno físico de operación. La interacción entre estos elementos determina factores clave como la precisión dimensional, la calidad superficial y la estabilidad del proceso.

En años recientes, el avance de tecnologías como el aprendizaje automático y el análisis de datos ha abierto nuevas oportunidades para el maquinado robótico, permitiendo estrategias de control adaptativo

y optimización global del sistema. Estas tendencias apuntan hacia sistemas más inteligentes, capaces de adaptarse de manera autónoma a condiciones cambiantes del entorno y del proceso.

1.4.1 Robot MENTOR

El robot MENTOR es un manipulador antropomórfico de seis ejes diseñado originalmente como una plataforma didáctica para la enseñanza de conceptos fundamentales de control, electrónica y robótica. Su arquitectura combina un sistema mecánico ligero con un conjunto de servomotores de corriente directa, potenciómetros de retroalimentación y un esquema de control analógico–digital.

Cada una de las articulaciones del MENTOR está accionada mediante motores de corriente directa acoplados a cajas reductoras de alta relación. La posición angular de cada eje se mide a través de potenciómetros montados directamente en los ejes correspondientes, lo que permite obtener una señal analógica proporcional al ángulo de rotación.

Los movimientos de la muñeca —izquierda y derecha— se obtienen mediante la combinación de dos motores que operan de manera conjunta o diferencial. Por su parte, la pinza utiliza un mecanismo de cable de nylon con resortes de torsión, permitiendo distintos grados de cierre programable.

El robot MENTOR está equipado con seis sensores de posición ubicados en cada eje de rotación asociado a un motor. Estos sensores son potenciómetros lineales de rotación continua con una resistencia nominal de $5\,\mathrm{k}\Omega$. Cada articulación cuenta con un sensor, salvo la muñeca, que integra dos sensores debido a su configuración de doble motor. Adicionalmente, la pinza dispone de un sensor que permite controlar la apertura y el cierre del efector final.

1.5 Trabajo Previo

1.5.1 Preguntas de trabajo previo

1. ¿Cuáles fueron las principales necesidades industriales que dieron origen a la robótica industrial y en qué tipo de procesos se implementaron los primeros robots?

2. ¿Qué se entiende por robot industrial y cuáles son las características que lo diferencian de otros sistemas automatizados?

3. Describe los componentes principales de un robot industrial e indica la función que desempeña cada uno dentro del sistema.

4. Menciona al menos tres ventajas del uso de robots industriales en comparación con procesos manuales tradicionales y explica brevemente cada una.

1.5.2 Investigación: Cinemáticas de robots más comunes

El alumno deberá investigar y describir las características principales, la estructura cinemática y las aplicaciones industriales de los siguientes tipos de robots:

- Robot cartesiano
- Robot cilíndrico
- Robot esférico (o polar)
- Robot SCARA
- Robot articulado

La investigación deberá enfocarse en identificar:

- Tipo de movimiento (prismático o rotacional).
- Número de grados de libertad.
- Geometría del espacio de trabajo.
- Ejemplos de aplicaciones industriales típicas para cada configuración.

1.6 Desarrollo

1.6.1 Parte 1. Identificación de los elementos del robot industrial

En esta parte, el alumno identificará visualmente los principales elementos que conforman un robot industrial, a partir de la imagen proporcionada por el docente.

Instrucción A partir de la figura mostrada, el alumno deberá:

1. Identificar los componentes estructurales del robot industrial señalados con flechas.
2. Nombrar correctamente cada elemento identificado.

1.6.2 Evidencias a entregar (Parte 1)

- Imagen del robot con los componentes correctamente identificados. 1
- Describir brevemente la función general que desempeña cada componente dentro del sistema robótico.

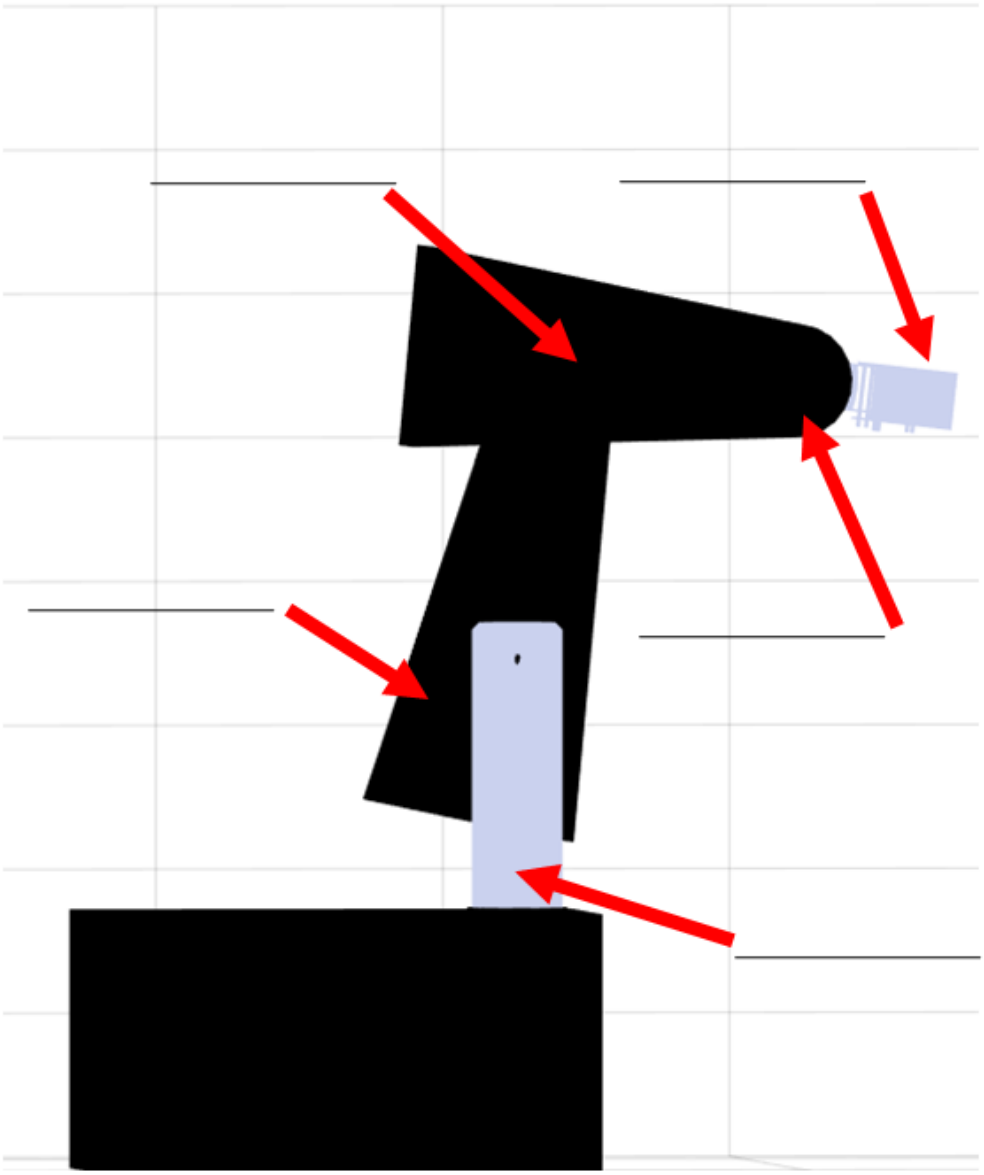


Figure 1: Elementos estructurales de un robot industrial

1.6.3 Actividad 2. Identificación de componentes en el robot real

El alumno deberá tomar fotografías del robot disponible en el laboratorio y, a partir de dichas imágenes, identificar los principales componentes mecánicos, eléctricos y de sensado que conforman el sistema.

En las fotografías deberán señalarse e identificarse, como mínimo, los siguientes elementos:

- Base del robot.
- Eslabones.
- Articulaciones mecánicas.
- Potenciómetros.
- Efector final (pinza, herramienta o dispositivo instalado).
- Otros componentes relevantes observados en el robot.

Para cada elemento identificado, el alumno deberá indicar brevemente su función general dentro del sistema robótico.

1.6.4 Evidencias a entregar (Actividad 2)

El alumno deberá integrar en el reporte las siguientes evidencias correspondientes a la Actividad 2:

1. Conjunto de fotografías del robot real tomadas en el laboratorio (mínimo 3, desde diferentes ángulos) donde se observe claramente la estructura general del robot.
2. Fotografías anotadas o marcadas (con flechas, recuadros o etiquetas) en las que se identifiquen, como mínimo:
   - Base del robot.
   - Eslabones.
   - Articulaciones mecánicas.
   - Potenciómetros.
   - Efector final (pinza o herramienta instalada).
   - Otros componentes relevantes observados en el robot.
3. Tabla o listado descriptivo donde se indique para cada componente identificado:
   - Nombre del componente.
   - Ubicación aproximada (eje/articulación o zona del robot).
   - Función general dentro del sistema.

1.6.5 Actividad 3. Identificación de componentes internos del robot (opcional)

Instrucción En caso de contar con la autorización expresa del personal docente o responsable del laboratorio, el alumno podrá retirar parcial o totalmente la tapa de alguna articulación del robot (por ejemplo, hombro o codo) con el fin de observar su estructura interna.

El alumno deberá tomar fotografías del interior de la articulación y, a partir de ellas, identificar los componentes internos visibles, tales como:

- Motor eléctrico o actuador.
- Sistemas de transmisión (engranes, reductores o acoplamientos).
- Potenciómetros u otros sensores de retroalimentación.
- Elementos de soporte mecánico.
- Cableado interno y conexiones.

Para cada componente identificado, el alumno deberá describir su función general dentro del movimiento y control del robot.

1.6.6 Parte 2. Control manual del robot industrial

El alumno establecerá comunicación entre la aplicación y el robot MENTOR, utilizando la interfaz de **CONTROL MANUAL**.

Instrucciones

1. Encender el robot industrial y verificar que el sistema de alimentación esté funcionando correctamente.
2. Abrir la aplicación principal del robot en el equipo de cómputo asignado.

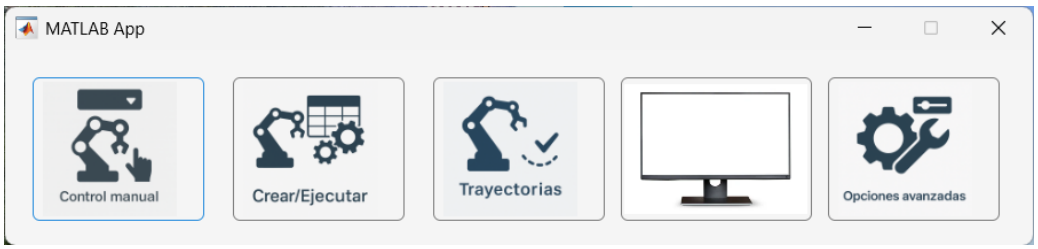


Figure 2: Aplicación principal del robot.

3. Abrir el módulo **Control Manual** desde la aplicación principal.

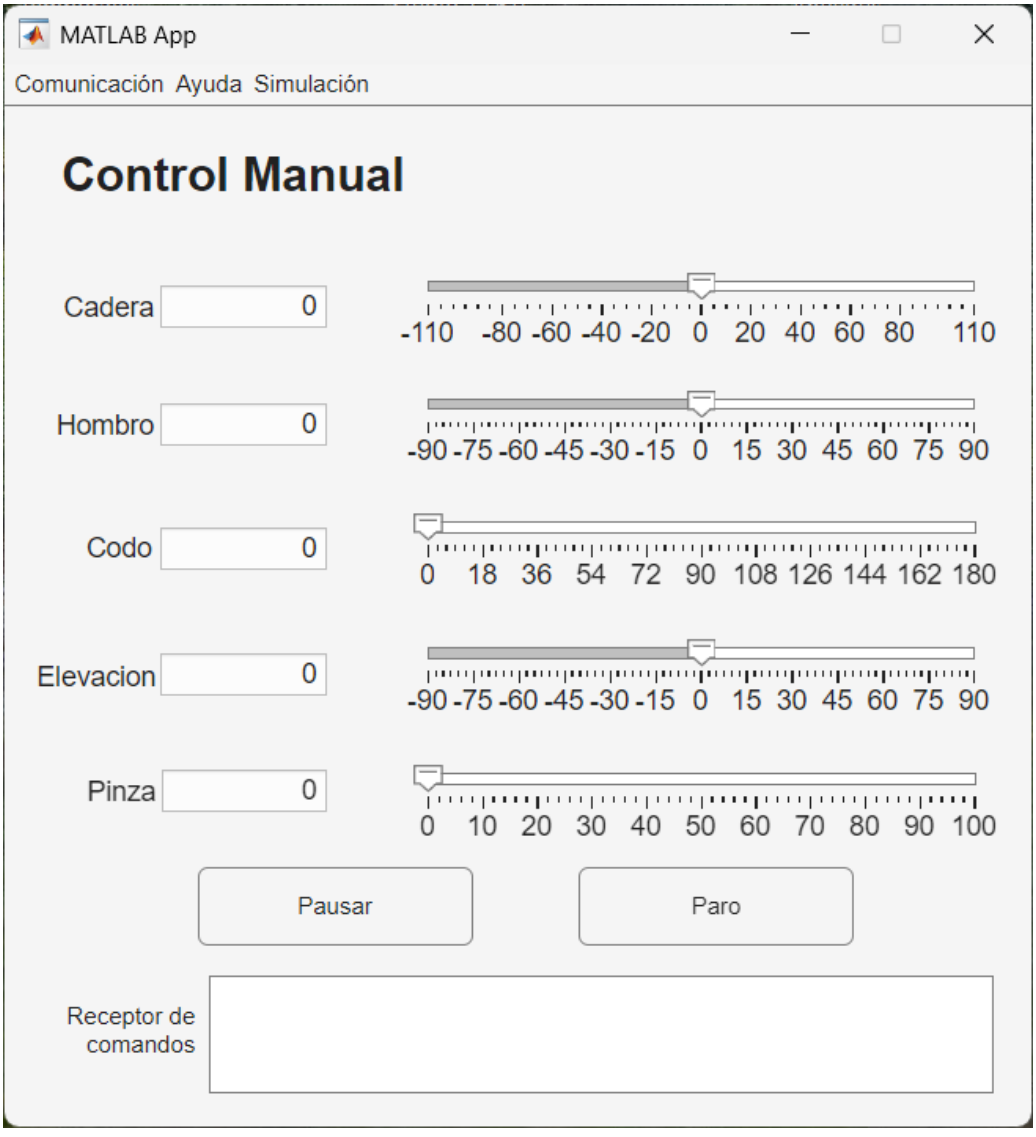


Figure 3: Módulo CONTROL MANUAL.

4. En el menú principal del módulo, seleccionar la opción **Comunicaciones** → **Verificar enlace**.

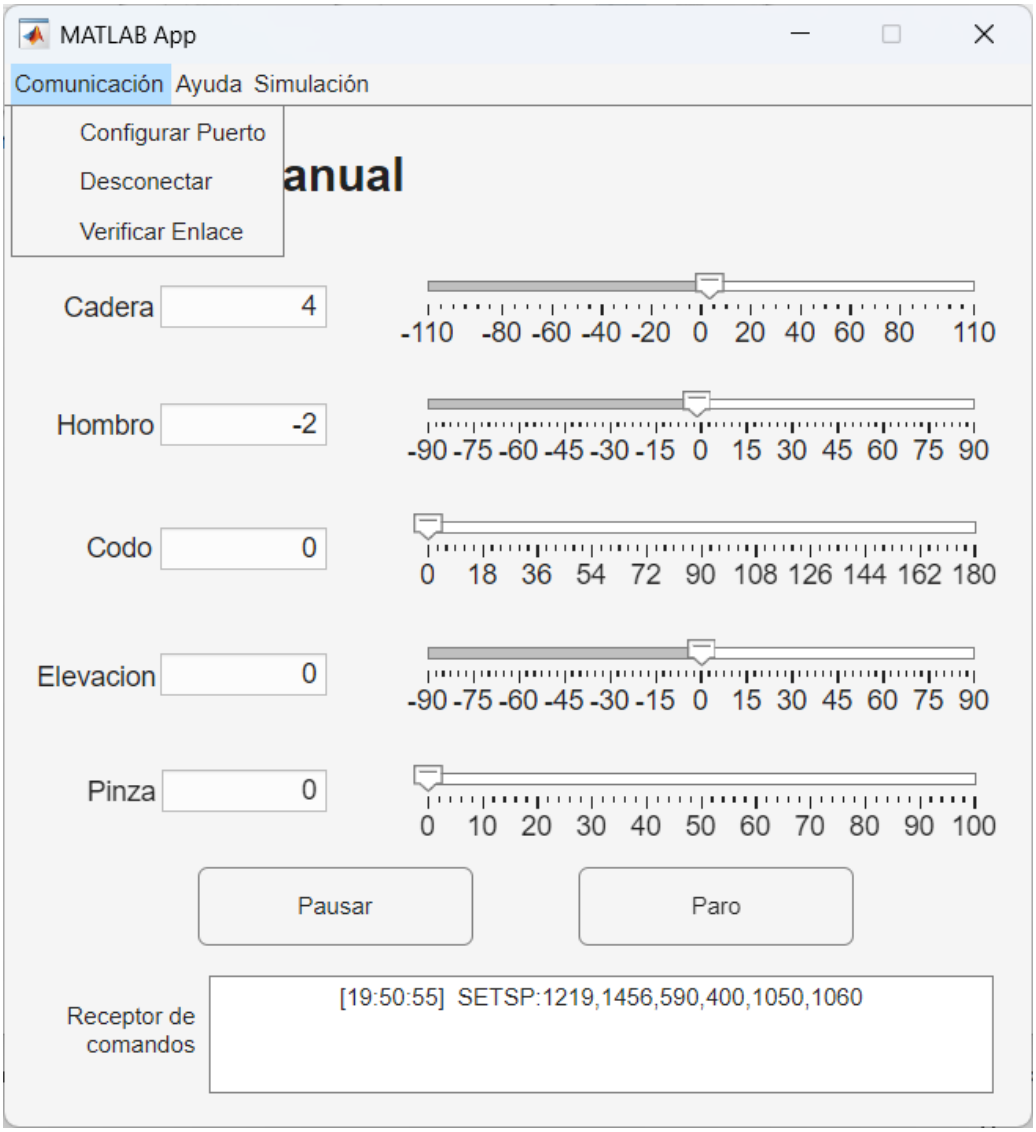


Figure 4: Sección COMUNICACIÓN del módulo CONTROL MANUAL.

5. Ejecutar la verificación de enlace y confirmar que la comunicación con el robot sea exitosa.
6. Utilizando los deslizadores o controles disponibles en la interfaz:
   - Mover cada articulación del robot de forma individual.
   - Observar la respuesta física del robot.

- Verificar que el movimiento corresponda con el control seleccionado.

7. Realizar movimientos suaves y dentro de los límites permitidos por el sistema.

Consideraciones de seguridad

- No forzar manualmente ninguna articulación mientras el sistema esté energizado.
- No exceder los límites mecánicos del robot.
- Detener inmediatamente la operación si se detecta un comportamiento anormal.

Evidencias a entregar (Parte 2) El alumno deberá registrar y reportar observaciones relacionadas con:

- Precisión del movimiento.
- Respuesta del sistema ante comandos individuales.
- Comportamiento de cada articulación durante el control manual.

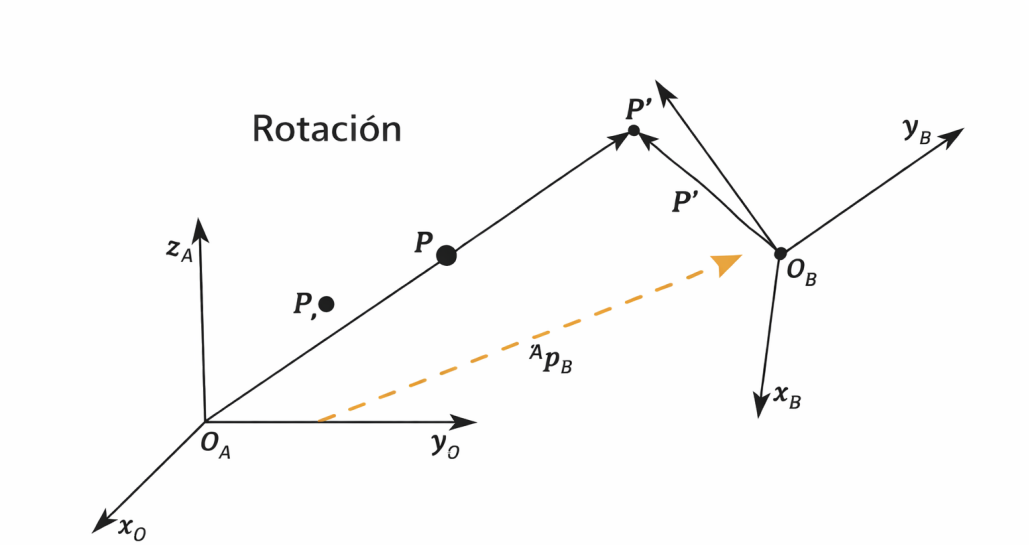
1.7 Conclusiones

1.8 Referencias

1. Ji, W., & Wang, L. (2019). Industrial robotic machining: a review. *The International Journal of Advanced Manufacturing Technology, 103*(1), 1239–1255.

Práctica Nº 2

Rotaciones, composición de rotaciones y transformaciones homogéneas



Rotaciones y Transformaciones Homogéneas

Nombre completo del alumo		Firma	Calificación
Nº de brigada:_____	Fecha de elaboración:_____	Grupo:	

2 Práctica 2: Rotaciones, composición de rotaciones y transformaciones homogéneas

2.1 Objetivo

Comprender y aplicar los conceptos fundamentales de rotación, composición de rotaciones mediante desarrollo analítico y transformaciones homogéneas, empleándolos en la descripción de la cinemática de un robot y validando los resultados con el apoyo de herramientas computacionales.

2.2 Recursos

Software

- Entorno de desarrollo: MATLAB R2025a o superior.
- Sistema operativo: Windows 10 o superior.

Recursos de equipo

- Computadora de escritorio.
- Robot antropomórfico de cinco grados de libertad (5 GDL), modelo MENTOR (o modelo de simulación equivalente).

2.3 Seguridad

Peligro o fuente de energía	**Riesgo asociado**	**Medidas de control**
Energía eléctrica (alimentación del sistema)	Electrocución	Verificar conexiones y niveles de voltaje antes de energizar el sistema. Evitar el contacto con partes energizadas.
Movimiento mecánico del robot (articulaciones y efector final)	Golpes o atrapamiento	Mantener manos y objetos fuera del área de trabajo. No intervenir durante la ejecución. Activar paro de emergencia si es necesario.
Movimiento en simulación y comandos erróneos	Daño al equipo / ejecución no deseada	Validar límites y referencias antes de ejecutar en el robot real. Probar primero en simulación y ejecutar a baja velocidad.

2.4 Fundamento teórico

La descripción del movimiento de un robot manipulador requiere representar de manera consistente la orientación y la posición de sus eslabones respecto a marcos de referencia. Para ello, se utilizan matrices de rotación y transformaciones homogéneas, herramientas matemáticas que permiten modelar la cinemática de un sistema rígido en tres dimensiones. Una matriz de rotación en $\mathbb{R}^3$ pertenece al grupo especial ortogonal $SO(3)$ y codifica un cambio de orientación preservando distancias y ángulos. La composición de rotaciones se realiza mediante multiplicación matricial, lo que permite encadenar orientaciones relativas entre marcos consecutivos [2].

Cuando se requiere describir simultáneamente orientación y traslación, se emplean transformaciones homogéneas en $SE(3)$, representadas por matrices 4×4 que integran una matriz de rotación R y un vector de traslación p. Estas transformaciones se utilizan para definir la relación entre marcos coordenados a lo largo de la cadena cinemática de un robot, permitiendo obtener la pose del efector final con respecto a la base a partir de la composición de transformaciones entre eslabones [1].

En aplicaciones prácticas, la representación de orientación puede expresarse mediante distintos parámetros (ángulos de Euler, rotaciones elementales alrededor de ejes, o ángulo-eje). En esta práctica se desarrollarán rotaciones elementales, su composición analítica, y la construcción y validación de transformaciones homogéneas mediante herramientas computacionales, como apoyo para la interpretación cinemática de un robot manipulador.

2.4.1 Rotaciones elementales y matrices de rotación

Las rotaciones elementales alrededor de los ejes x, y y z se expresan mediante matrices 3×3, típicamente denotadas como $R_x(\alpha)$, $R_y(\beta)$ y $R_z(\gamma)$. Estas matrices describen el cambio de orientación de un marco al rotarlo un ángulo alrededor de un eje fijo. La composición de rotaciones se realiza multiplicando matrices en el orden correspondiente, lo que refleja que las rotaciones en 3D no conmutan en general, es decir, $R_a R_b \neq R_b R_a$ [2].

2.4.2 Transformaciones homogéneas

Las transformaciones homogéneas nos modelan la posición de un cuerpo rígido en el espacio puede describirse en términos de la **posición** de un punto del cuerpo con respecto a un marco de referencia. A este desplazamiento se le denomina *traslación*. Por otro lado, la **orientación** del cuerpo se describe mediante una *rotación*, la cual se expresa en términos de las componentes de los vectores unitarios del marco unido al cuerpo con respecto al marco de referencia base. En robótica, la combinación de traslación y rotación se representa de manera compacta mediante una **transformación homogénea**, escrita como una matriz de dimensiones 4×4:

$$ {}^{a}H_b = \begin{bmatrix} {}^{a}R_b & {}^{a}o_b \\ 0\ 0\ 0 & 1 \end{bmatrix} \tag{1} $$

Donde:

- ${}^{a}R_b$: pertenece al grupo especial ortogonal de orden tres ($SO(3)$) y corresponde a la **matriz de rotación** que expresa la orientación del sistema b respecto al sistema a.
- ${}^{a}o_b$: **vector de posición** del origen del sistema b con respecto al sistema a (traslación).
- $[0\ \ 0\ \ 0]$: matriz fila 1×3 de ceros.
- 1: elemento escalar que permite trabajar con coordenadas homogéneas.

La utilidad principal de ${}^{a}H_b$ es que permite transformar puntos (o vectores de posición) expresados en el sistema b al sistema a, además de facilitar la **composición** de movimientos mediante el producto de transformaciones homogéneas en cadenas cinemáticas.

2.5 Trabajo previo

1. Defina qué es una matriz de rotación en $\mathbb{R}^3$ e indique dos propiedades que debe cumplir para pertenecer a $SO(3)$.
2. Escriba las matrices de rotación elementales $R_x(\alpha)$, $R_y(\beta)$ y $R_z(\gamma)$.
3. Explique por qué la composición de rotaciones en 3D no es conmutativa y proporcione un ejemplo conceptual.
4. Describa la estructura de una transformación homogénea ${}^{A}T_B$ e indique qué representa cada submatriz.
5. Investigue dos aplicaciones en robótica donde sea indispensable el uso de transformaciones homogéneas (por ejemplo: cinemática directa, calibración, control en espacio cartesiano).

2.6 Desarrollo de la actividad

2.6.1 Parte 1. Rotaciones elementales y validación computacional

En esta parte se construirán rotaciones elementales alrededor de ejes coordenados y se verificará su comportamiento mediante simulación computacional. El objetivo es relacionar el significado geométrico de la rotación con su representación matricial y confirmar propiedades básicas como ortonormalidad y determinante unitario.

1. Abra MATLAB y cree un nuevo script.
2. Defina un vector unitario de prueba (por ejemplo, $v = [1 \;\; 0 \;\; 0]^T$) y un conjunto de ángulos (por ejemplo 0°, 30°, 60°, 90°).
3. Implemente $R_x(\alpha)$, $R_y(\beta)$ y $R_z(\gamma)$ y aplíquelas al vector v.
4. Verifique para al menos tres ángulos que:
   - $R^T R = I$ (ortonormalidad),
   - $\det(R) = 1$.
5. Represente en 3D el vector original y el vector rotado para visualizar el efecto de cada rotación.

2.6.2 Evidencia a entregar (Parte 1)

- Tabla con los ángulos evaluados y el vector resultante para R_x, R_y y R_z.
- Evidencia de verificación de propiedades ($R^T R = I$ y $\det(R) = 1$) para al menos una rotación y tres ángulos.
- Observaciones breves sobre el significado geométrico de cada rotación.

2.6.3 Parte 2. Composición de rotaciones: desarrollo analítico y comparación

En esta parte se realizará la composición de dos y tres rotaciones, comparando el resultado analítico con el obtenido mediante cálculo computacional. Se busca evidenciar el efecto del orden de aplicación y su impacto en la orientación final.

1. Seleccione dos rotaciones elementales distintas (por ejemplo $R_z(\gamma)$ y $R_y(\beta)$) con ángulos definidos por el profesor.
2. Calcule analíticamente la matriz compuesta $R = R_z(\gamma)R_y(\beta)$.
3. Calcule también la composición en orden inverso $R' = R_y(\beta)R_z(\gamma)$.
4. Compare R y R' e indique si son iguales o diferentes.
5. Valide los resultados en MATLAB y evalúe el efecto sobre un vector de prueba v.

2.6.4 Evidencia a entregar (Parte 2)

- Desarrollo analítico de la composición seleccionada (matriz resultante).
- Comparación entre R y R' (indicando conclusiones sobre el orden).
- Captura o tabla de MATLAB mostrando la diferencia numérica entre ambas composiciones.

2.6.5 Parte 3. Construcción de transformaciones homogéneas y encadenamiento

En esta parte se construirán transformaciones homogéneas a partir de una rotación y una traslación, y se evaluará su composición. El objetivo es comprender cómo se obtiene la pose resultante al encadenar marcos de referencia.

1. Defina una rotación ${}^{a}R_{b}$ (por ejemplo, una composición de Parte 2) y una traslación ${}^{a}p_{b}$.
2. Construya la transformación homogénea ${}^{a}H_{b}$.
3. Defina una segunda transformación ${}^{b}H_{c}$ con rotación y traslación.
4. Calcule la transformación compuesta ${}^{a}H_{c} = {}^{a}H_{b}\,{}^{b}H_{c}$.
5. Verifique el resultado aplicando ${}^{a}H_{c}$ a un punto homogéneo $q = [x\ y\ z\ 1]^{T}$.

2.6.6 Evidencia a entregar (Parte 3)

- Matrices ${}^{a}H_{b}$, ${}^{b}H_{c}$ y ${}^{a}H_{c}$.
- Un ejemplo numérico de aplicación a un punto homogéneo antes y después de la transformación.
- Observaciones sobre la interpretación física de la composición (marcos y pose resultante).

2.7 Conclusiones

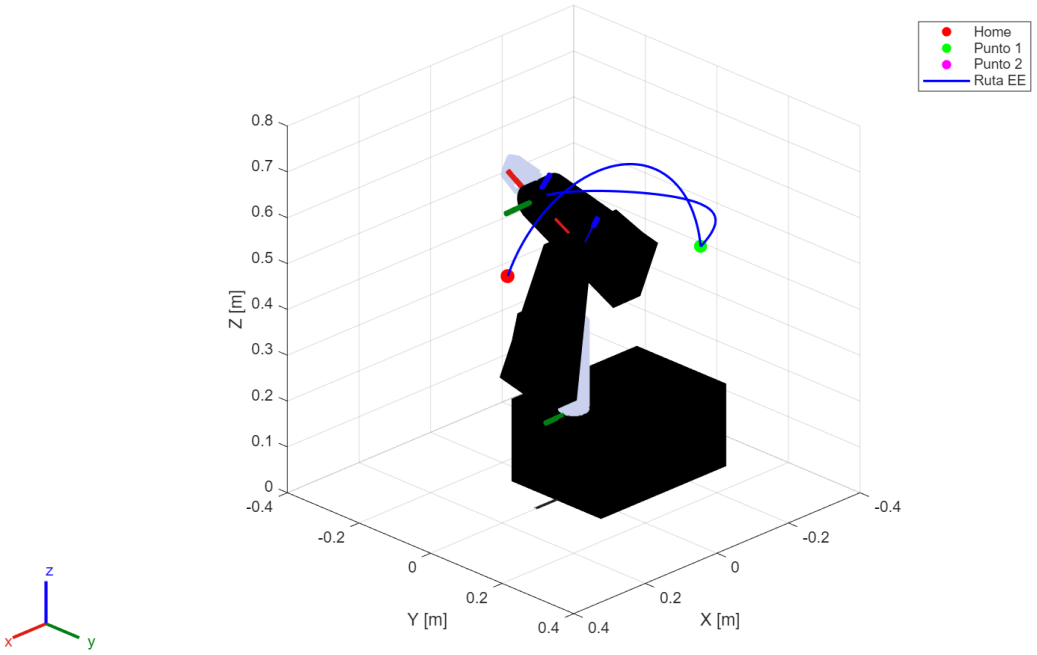
References

[1] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Pearson, 2005.

[2] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2010, doi:10.1007/978-1-84628-642-1.

[3] A. Barrientos, L. Peñín, C. Balaguer y R. Aracil, *Fundamentos de Robótica*, 2a ed. McGraw-Hill España, 2012.

Práctica Nº 3
Seguimiento y simulación de trayectorias



Nombre completo del alumo		Firma	Calificación
Nº de brigada:_____	Fecha de elaboración:_____	Grupo:	

3 Práctica 3: Seguimiento y simulación de trayectorias

3.1 Objetivo

Realizar el seguimiento y la simulación de trayectorias entre puntos del espacio de trabajo mediante una aplicación desarrollada en MATLAB App Designer, comparando de manera **cualitativa** el movimiento planeado (gemelo digital) frente a la ejecución del robot real, e identificando criterios de seguridad y validación durante la transmisión y ejecución de la trayectoria.

3.2 Recursos

Software

- Entorno de desarrollo y ejecución: MATLAB R2025a o superior.
- Sistema operativo: Windows 10 o superior.

Recursos de equipo

- Computadora de escritorio.
- Robot antropomórfico de cinco grados de libertad (5 GDL), modelo MENTOR.
- Cable de conexión de comunicación USB “tipo c”

3.3 Seguridad

Peligro o fuente de energía	**Riesgo asociado**	**Medidas de control**
Energía eléctrica (alimentación del sistema)	Electrocución / daño al equipo	Verificar conexiones, niveles de voltaje y puesta a tierra antes de energizar. Evitar contacto con partes energizadas.
Movimiento mecánico del robot	Golpes, atrapamiento, colisiones	Mantener manos y objetos fuera del área de trabajo. Mantener postura de observación segura. Ejecutar a baja velocidad cuando aplique.
Ejecución de trayectoria no validada	Movimiento inesperado / colisión	Simular primero la trayectoria. Confirmar que los puntos seleccionados están dentro del espacio de trabajo. Validar postura inicial (Home) antes de enviar.
Comunicación UART/serie activa	Comandos no deseados / pérdida de control	Verificar puerto y conexión antes de habilitar envío. Tener disponible el paro de emergencia. Cerrar comunicación al finalizar.

3.4 Fundamento teórico

El **seguimiento de trayectorias** consiste en planear y ejecutar un movimiento que conecta puntos del espacio de trabajo del robot, garantizando una evolución continua (o suficientemente suave) de la postura entre un punto inicial y uno final. En un entorno didáctico, una forma efectiva de introducir este tópico es mediante un enfoque **cualitativo**: observar cómo cambia el movimiento al modificar los puntos, identificar comportamientos deseables (suavidad, continuidad, ausencia de saturaciones) y reconocer discrepancias entre la simulación y la ejecución real.

En esta práctica se utiliza una aplicación desarrollada en **MATLAB App Designer** que permite: (i) seleccionar puntos dentro del espacio de trabajo, (ii) simular la trayectoria con un **gemelo digital** que se mueve en tiempo real, (iii) enviar la trayectoria al microcontrolador mediante **UART** para que el robot real ejecute la secuencia punto a punto y (iv) comparar cualitativamente el resultado real respecto a lo simulado. Además, se dispone de un **paro de emergencia** que envía una trama específica al MCU para detener la ejecución.

3.4.1 Flujo general del programa de seguimiento

El flujo típico de operación se resume en:

- Inicio e inicialización (carga de recursos y configuración de comunicación).

- Selección de puntos dentro del espacio de trabajo.
- Simulación de trayectoria (visualización en tiempo real).
- Envío de trayectoria al MCU (cuando aplique).
- Ejecución en robot real y retorno a Home al finalizar.

3.4.2 Criterios cualitativos de evaluación

Para esta práctica, el desempeño se juzga mediante observación y criterios descriptivos:

- **Suavidad del movimiento**: ausencia de cambios bruscos.
- **Continuidad**: transición estable entre segmentos.
- **Coherencia simulación–real**: similitud general del movimiento observado.
- **Seguridad**: uso correcto de Home, control del área, y paro de emergencia.

3.5 Trabajo previo

1. Defina qué se entiende por trayectoria en el espacio de trabajo de un robot manipulador.
2. Mencione dos diferencias típicas entre simulación (gemelo digital) y ejecución real (por ejemplo: fricción, holguras, límites físicos, discretización por puntos).
3. Describa tres criterios cualitativos para evaluar la ejecución de una trayectoria (por ejemplo: suavidad, continuidad, repetibilidad).

3.6 Desarrollo de la actividad

3.6.1 Parte 1. Preparación del entorno y verificación de comunicación

En esta parte se prepara el entorno de ejecución y se verifica que la aplicación se encuentre lista para simular y, si aplica, comunicarse con el robot.

1. Encender el equipo de cómputo y ejecutar el programa principal.
2. Abrir la aplicación **Seguimiento de trayectorias**.

3. En la barra de menú de la aplicación, acceder a la sección de configuración de comunicación (puerto, verificación y cierre).

4. Verificar la conexión con el dispositivo (según la opción disponible en la aplicación). La Figura 5 muestra la aplicación abierta y lista para operar.

Figure 5: Aplicación de seguimiento de trayectorias abierta.

3.6.2 Evidencia a entregar (Parte 1)

- Captura de pantalla de la aplicación abierta y lista para operar.
- Nota breve indicando el estado de la comunicación (verificada/no verificada) y acciones realizadas.

3.6.3 Parte 2. Selección de puntos y simulación de trayectoria (gemelo digital)

En esta parte se ejecuta la simulación cualitativa del movimiento planeado. El objetivo es observar el comportamiento del gemelo digital durante el seguimiento entre puntos.

1. Seleccionar dos puntos dentro del espacio de trabajo mediante los menús desplegables de la aplicación.
2. Ejecutar la opción **Simular trayectoria**.
3. Observar el movimiento del gemelo digital en tiempo real. La Figura 6 muestra un ejemplo del resultado esperado.
4. Repetir el procedimiento con al menos dos conjuntos distintos de puntos:
   - Conjunto A: puntos cercanos entre sí (trayectoria corta).
   - Conjunto B: puntos más separados (trayectoria más amplia).

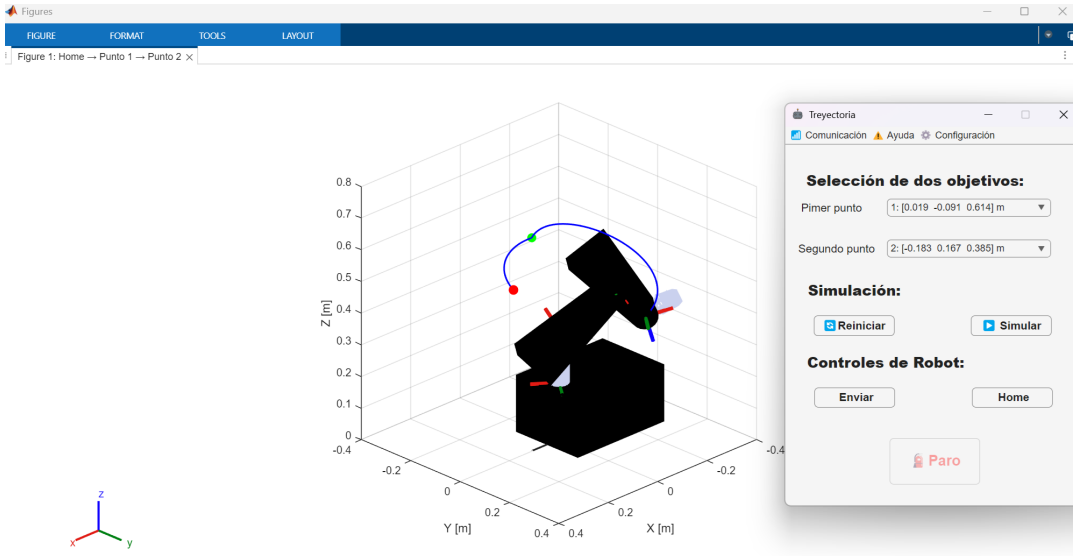


Figure 6: Resultado esperado de la simulación de trayectoria en la aplicación.

Registro cualitativo sugerido: describir con palabras el comportamiento (suave/brusco, continuo/discontinuo, lento/rápido, estable/inestable) y anotar cualquier observación relevante.

3.6.4 Evidencia a entregar (Parte 2)

- Capturas de pantalla (o breve descripción) de al menos dos simulaciones.
- Tabla descriptiva breve con criterios cualitativos (por ejemplo: “Suavidad: alta/media/baja”, “Continuidad: alta/media/baja”).

3.6.5 Parte 3. Envío de trayectoria al MCU y ejecución en robot real

En esta parte se envía una trayectoria simulada al MCU para que el robot ejecute el movimiento, con el fin de comparar cualitativamente la ejecución real contra la simulación.

Condición: esta parte se realiza únicamente si el profesor autoriza la ejecución en robot real y el sistema está en condiciones seguras y en la posición de "home".

1. Confirmar que la trayectoria fue simulada correctamente (Parte 2) y que el robot está en posición segura.
2. Habilitar la opción **Enviar** (cuando el sistema la habilite al finalizar la simulación).
3. Enviar la trayectoria al MCU y observar la ejecución del robot.
4. Comparar cualitativamente la ejecución real contra la simulación:
   - similitud general del recorrido,
   - presencia de retrasos, vibraciones o cambios bruscos,
   - desviaciones visibles en puntos de llegada (si se aprecian).
5. Confirmar el retorno a Home al finalizar la trayectoria.

3.6.6 Evidencia a entregar (Parte 3)

- Descripción cualitativa comparativa simulación vs. ejecución real (mínimo un conjunto de puntos).
- Observaciones de seguridad aplicadas durante la ejecución.

3.6.7 Parte 4. Paro de emergencia y validación de seguridad

En esta parte se verifica el comportamiento del paro de emergencia, enfatizando el criterio de seguridad sobre el resultado del movimiento.

Nota: El procedimiento exacto será indicado por el profesor. Realizar esta parte bajo supervisión.

1. Iniciar la ejecución de una trayectoria autorizada (simulada o real, según indicación).
2. Activar el **paro de emergencia** desde la aplicación cuando el profesor lo indique.
3. Verificar que el sistema detiene la ejecución conforme al comportamiento esperado.
4. Restablecer el sistema siguiendo el procedimiento del laboratorio (por ejemplo: detener, regresar a estado seguro, y reintentar solo si se autoriza).

3.6.8 Evidencia a entregar (Parte 4)

- Registro breve del comportamiento observado al activar paro de emergencia.
- Descripción de acciones de recuperación realizadas.

3.7 Conclusiones

References

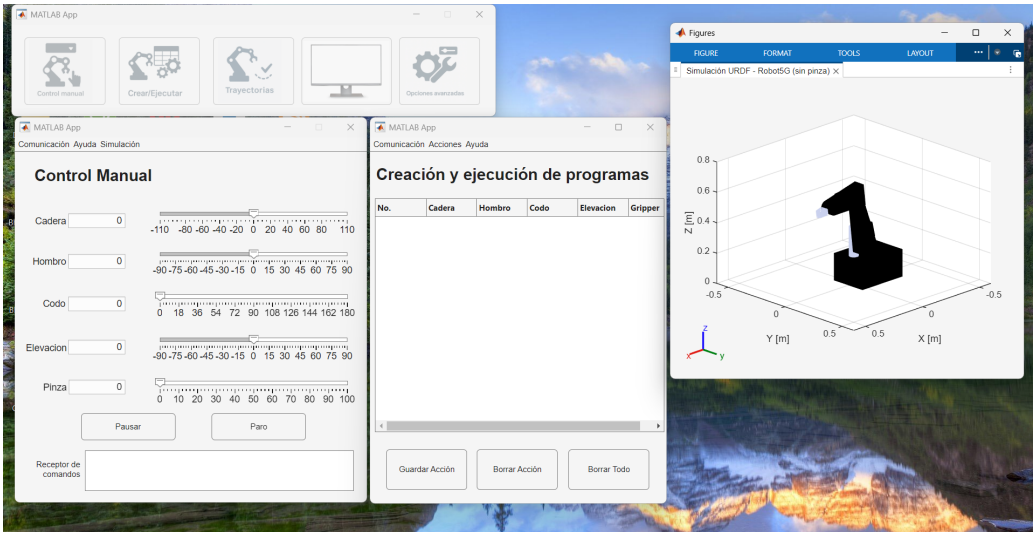
[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer, 2010.

[2] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Pearson, 2005.

[3] A. Barrientos, L. Peñín, C. Balaguer y R. Aracil, *Fundamentos de Robótica*, 2a ed. McGraw-Hill España, 2012.

Práctica Nº 4

Programación por puntos y ejecución secuencial de un robot industrial



Nombre completo del alumo		Firma	Calificación
Nº de brigada:_____	Fecha de elaboración:_____	Grupo:	

4 Práctica 4: Programación por puntos y ejecución secuencial de un robot industrial

4.1 Objetivo

Implementar, administrar y ejecutar secuencias de movimiento de un robot industrial mediante una interfaz de supervisión, verificando la comunicación con el sistema de control, el registro y almacenamiento de acciones, y la validación de la ejecución por pasos y en modo continuo con confirmación de finalización.

4.2 Recursos

Software

- Entorno de desarrollo y ejecución: MATLAB R2025a o superior.
- Sistema operativo: Windows 10 o superior.

Recursos de equipo

- Computadora de escritorio.
- Robot antropomórfico de cinco grados de libertad (5 GDL), modelo MENTOR.

4.3 Seguridad

Peligro o fuente de energía	**Riesgo asociado**	**Medidas de control**
Energía eléctrica (alimentación del sistema)	Electrocución	Verificar las conexiones y los niveles de voltaje antes de energizar el sistema y evitar el contacto con partes energizadas.
Energía eléctrica en corriente continua	Daño al equipo	Verificar la polaridad y el nivel de tensión antes de realizar la conexión de dispositivos o componentes.
Movimiento mecánico del robot (articulaciones y efector final)	Golpes o atrapamiento	Mantener manos y objetos fuera del área de trabajo del robot y no intervenir durante la ejecución del movimiento.

4.4 Fundamento teórico

Los robots industriales se han consolidado como un componente esencial de los sistemas de automatización modernos, debido a su capacidad para ejecutar tareas con precisión y repetibilidad dentro de celdas de trabajo. En las últimas décadas, la evolución de estos sistemas ha impulsado el desarrollo de esquemas de operación en los que el robot deja de ser únicamente una herramienta automatizada y se integra como un colaborador dentro del proceso productivo, lo cual incrementa la importancia de la supervisión, la programación estructurada y la validación segura de su ejecución [1]. Bajo este enfoque, la programación secuencial de movimientos mediante acciones discretas, así como el almacenamiento y recuperación de programas, permiten estandarizar rutinas, mejorar la trazabilidad de la operación y facilitar la repetición controlada de tareas. Por ello, la presente práctica se orienta a la implementación y ejecución de secuencias de movimiento a través de una interfaz de supervisión, verificando la comunicación con el sistema de control, la ejecución por pasos y en modo continuo, y la confirmación de finalización como criterio de validación del comportamiento del robot en un entorno de laboratorio.

4.4.1 Programación de robots por puntos (Point-to-Point)

La programación de robots industriales por puntos, conocida como point-to-point (PTP), es una de las estrategias más empleadas en entornos industriales debido a su simplicidad, robustez y facilidad de implementación. En este tipo de programación, el robot es instruido para desplazarse entre un conjunto finito de posiciones predefinidas, denominadas puntos de enseñanza, sin imponer restricciones explícitas sobre la trayectoria seguida entre dichos puntos. El objetivo principal del control PTP es garantizar que el robot alcance con precisión cada posición final, respetando los límites articulares, de velocidad y de aceleración del sistema [2].

Desde el punto de vista del control, la programación por puntos se basa en la definición de consignas articulares que son ejecutadas por el sistema de control del robot, el cual utiliza la retroalimentación de posición para verificar el cumplimiento de cada movimiento. Este enfoque resulta especialmente adecuado para tareas de manipulación, ensamblaje, carga y descarga, donde la trayectoria intermedia no afecta el resultado del proceso. De acuerdo con Barrientos et al., la programación PTP permite una rápida implementación de secuencias de movimiento y facilita la repetibilidad de las operaciones, características esenciales en sistemas de automatización industrial [2].

En aplicaciones modernas, la programación por puntos suele integrarse con interfaces de supervisión que permiten almacenar, editar y ejecutar secuencias de movimientos de forma estructurada. Estas interfaces proporcionan al operador herramientas para capturar posiciones articulares, organizar acciones en programas y validar su ejecución antes de operar el robot en modo continuo. Este esquema incrementa la seguridad y confiabilidad del sistema, especialmente en contextos de enseñanza y laboratorio.

4.4.2 Ejecución secuencial y control de eventos

La ejecución secuencial de movimientos en robots industriales consiste en la activación ordenada de acciones previamente programadas, donde cada acción se ejecuta únicamente cuando se cumplen ciertas condiciones definidas por el sistema de control. Este tipo de ejecución se apoya en el concepto de control de eventos, en el cual el avance del programa depende de la detección de sucesos específicos, tales como la finalización de un movimiento, la estabilización del sistema o la confirmación de una señal de retroalimentación [2].

El control de eventos permite estructurar el comportamiento del robot en estados bien definidos, como espera, ejecución, verificación y finalización. Esta organización facilita la supervisión del sistema y reduce la probabilidad de errores durante la operación, ya que impide que el robot avance al siguiente paso sin haber completado correctamente la acción anterior. En celdas de trabajo industriales, este enfoque es fundamental para coordinar el robot con otros dispositivos, como sensores, actuadores periféricos o sistemas de seguridad.

De acuerdo con Ferreira y Fletcher, los sistemas robóticos contemporáneos han evolucionado hacia esquemas de operación más colaborativos y supervisados, donde la interacción con el entorno y con el operador exige mecanismos confiables de validación del estado del robot antes de continuar con la ejecución del programa [1]. En este contexto, la ejecución secuencial con confirmación de eventos se convierte en un elemento clave para garantizar una operación segura, predecible y compatible con entornos de trabajo compartidos.

4.5 Trabajo previo

1. ¿Qué se entiende por programación de robots por puntos (point-to-point) y cuáles son sus principales características en aplicaciones industriales?

2. ¿Cuál es la diferencia entre la programación por puntos y la programación por trayectorias continuas, y en qué tipo de tareas resulta más conveniente cada una?

3. ¿Qué condiciones o criterios pueden emplearse para confirmar que un movimiento ha finalizado correctamente antes de avanzar al siguiente paso en una secuencia programada?

4. Investigue cuáles son las aplicaciones industriales más comunes de la programación de robots por puntos (point-to-point) y qué características del proceso hacen adecuado este tipo de programación.

4.6 Desarrollo de la actividad

4.6.1 Parte 1. Validación del modelo URDF y reconocimiento del comportamiento articular en simulación

En esta parte se verificará que el modelo virtual del robot (URDF) represente de forma consistente la estructura cinemática del manipulador antropomórfico y que responda correctamente a los valores articulares

definidos desde la interfaz. La finalidad es establecer una referencia confiable para la programación por puntos, evitando definir acciones fuera de rango o configuraciones que puedan inducir colisiones o saturaciones durante la ejecución.

1. Como primer paso, abra la aplicación.
2. En la pantalla principal seleccione la opción **Control manual** para acceder al control articular directo (ver Figura 7).

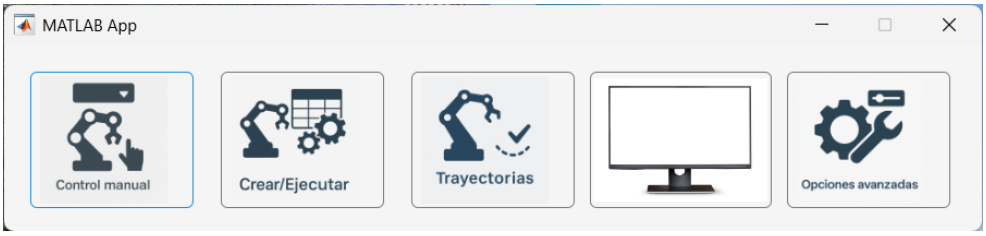


Figure 7: Pantalla inicial de la aplicación (módulos).

La ventana de **Control Manual** muestra los controles de **Cadera, Hombro, Codo, Elevación** y **Pinza**, cada uno con un campo numérico y un deslizador asociado (ver Figura 8).

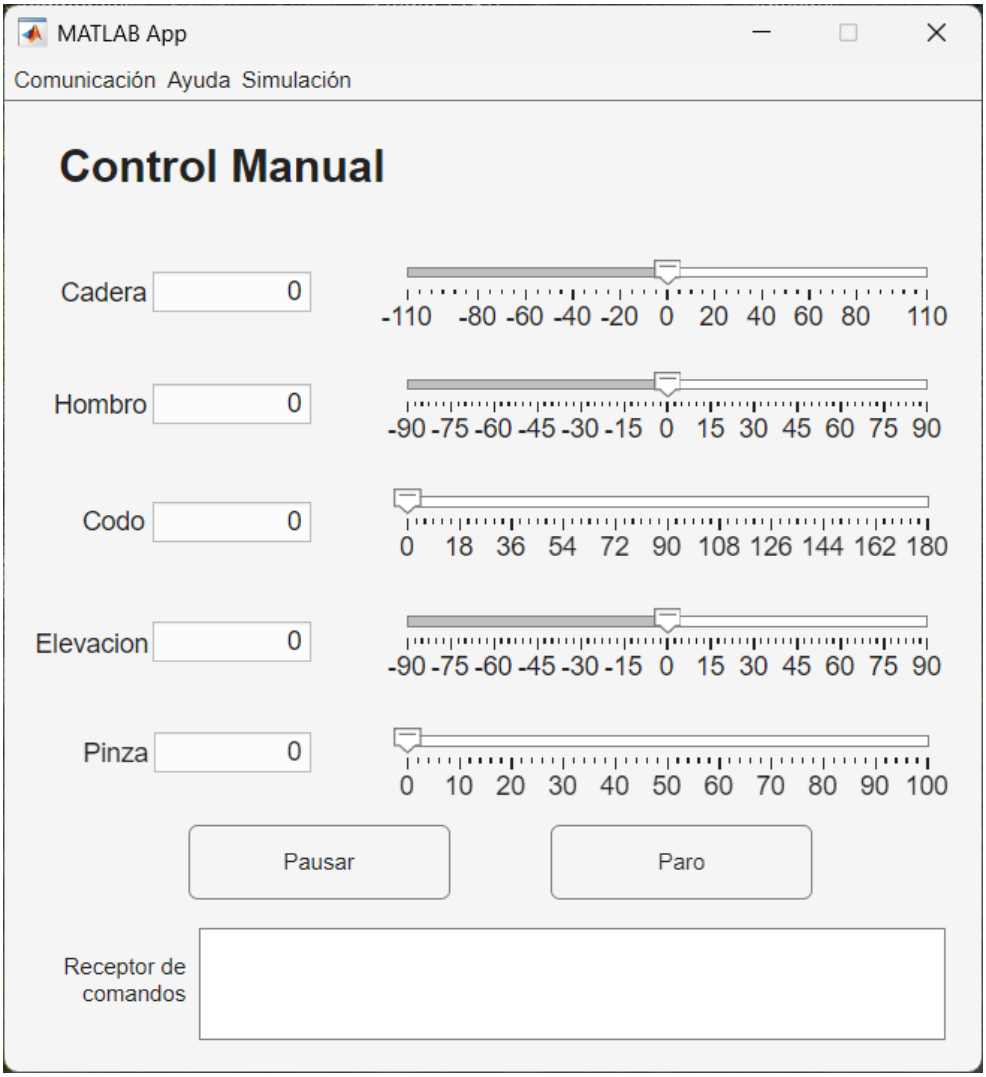


Figure 8: Ventana Control Manual con deslizadores articulares.

3. En la ventana **Control Manual**, abrir el menú **Simulación**.
4. Seleccionar la opción **Iniciar simulación** (ver Figura 9).
5. Confirmar que se abre una ventana de figura con el modelo del robot en 3D (ver Figura 10).

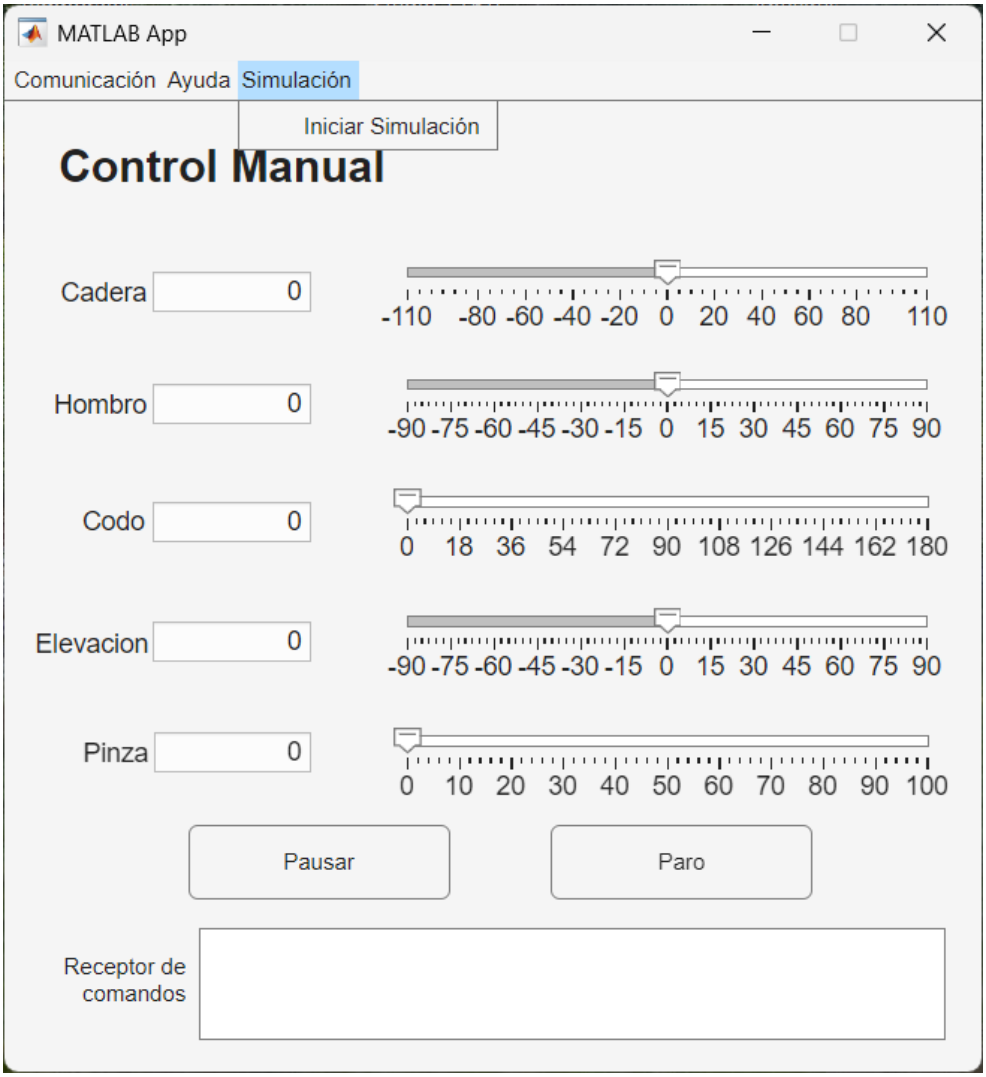


Figure 9: Menú Simulación → Iniciar simulación dentro de Control Manual.

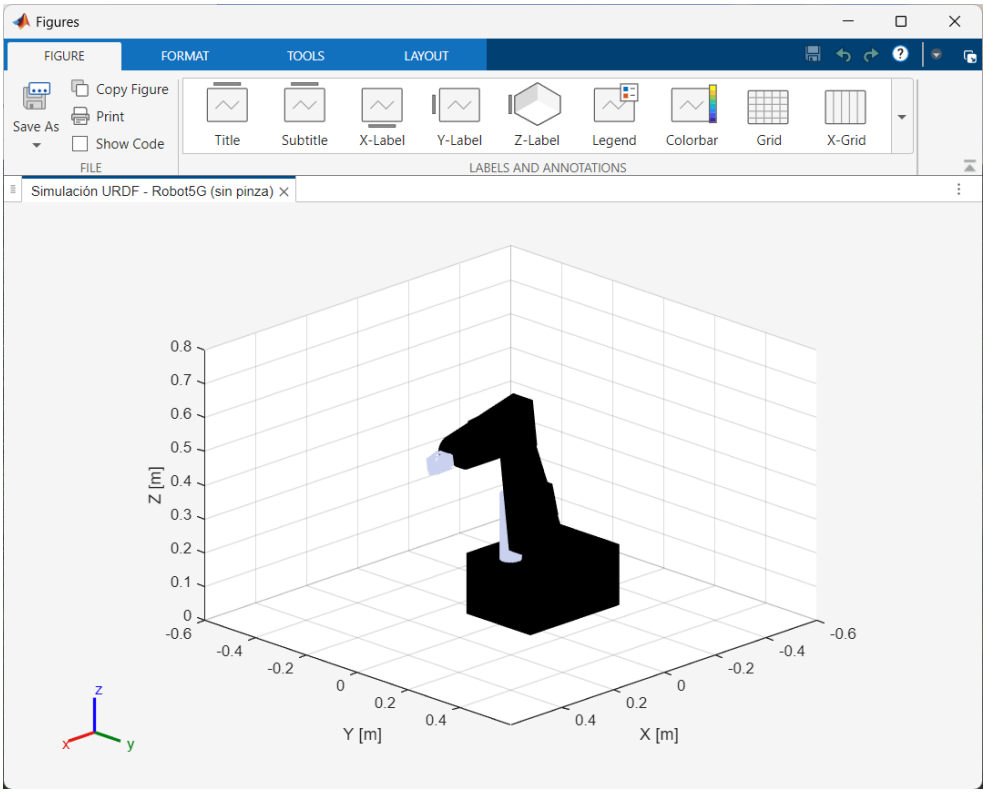


Figure 10: Ventana de figura con visualización del robot en simulación URDF (ejes y unidades).

6. Con la ventana de simulación URDF visible (Figura 10) y la ventana de **Control Manual** abierta (Figura 8):

7. Mover la articulación de **Cadera** lentamente desde el valor cero hacia un valor positivo moderado y posteriormente hacia un valor negativo moderado, observando el movimiento correspondiente en la simulación.

8. Repetir el procedimiento para las articulaciones de **Hombro**, **Codo** y **Elevación**, realizando el ajuste de una articulación a la vez.

Exploración rápida del espacio de trabajo

Con el modelo URDF activo, generar posturas de referencia que serán utilizadas posteriormente en la programación por puntos:

- **Postura A (retraída):** valores articulares moderados que mantengan al robot en una configuración compacta.

- **Postura B (extendida):** valores articulares que lleven el brazo hacia su alcance máximo sin forzar los límites mecánicos.
- **Postura C (lateral):** una postura orientada hacia un costado, variando principalmente la articulación de cadera y una combinación simple de hombro y codo.

4.6.2 Evidencia a entregar (Parte 1)

Se deberá incluir en el reporte:

1. Captura de pantalla o descripción que demuestre que el modelo URDF se visualiza correctamente en la simulación (Figura 10).
2. Nueve posturas de referencia correspondientes a las configuraciones A, B y C, indicando los valores articulares utilizados.
3. Observaciones breves relacionadas con:
   - Límites detectados (identificando qué articulación alcanza primero su límite).
   - Comportamiento general del modelo durante la simulación.

4.6.3 Parte 2. Programación del robot por puntos en entorno de simulación

En esta parte se realizará la programación del robot por puntos (*point-to-point*) utilizando la interfaz de **Creación y ejecución de programas**, registrando posiciones articulares como acciones discretas. Estas acciones conformarán una secuencia de movimiento que posteriormente será validada en simulación.

Todas las acciones definidas en esta parte se realizan exclusivamente en simulación, tomando como base el comportamiento del modelo URDF verificado en la Parte 1.

1. Abra nuevamente la aplicación principal del robot. Posteriormente, abra la aplicación **Control Manual** y active el simulador del robot, siguiendo el mismo procedimiento descrito en la Parte 1.
2. Desde la aplicación principal, seleccione el módulo **Crear/Ejecutar**.
3. Verifique que se despliega la ventana **Creación y ejecución de programas** (ver Figura 11).

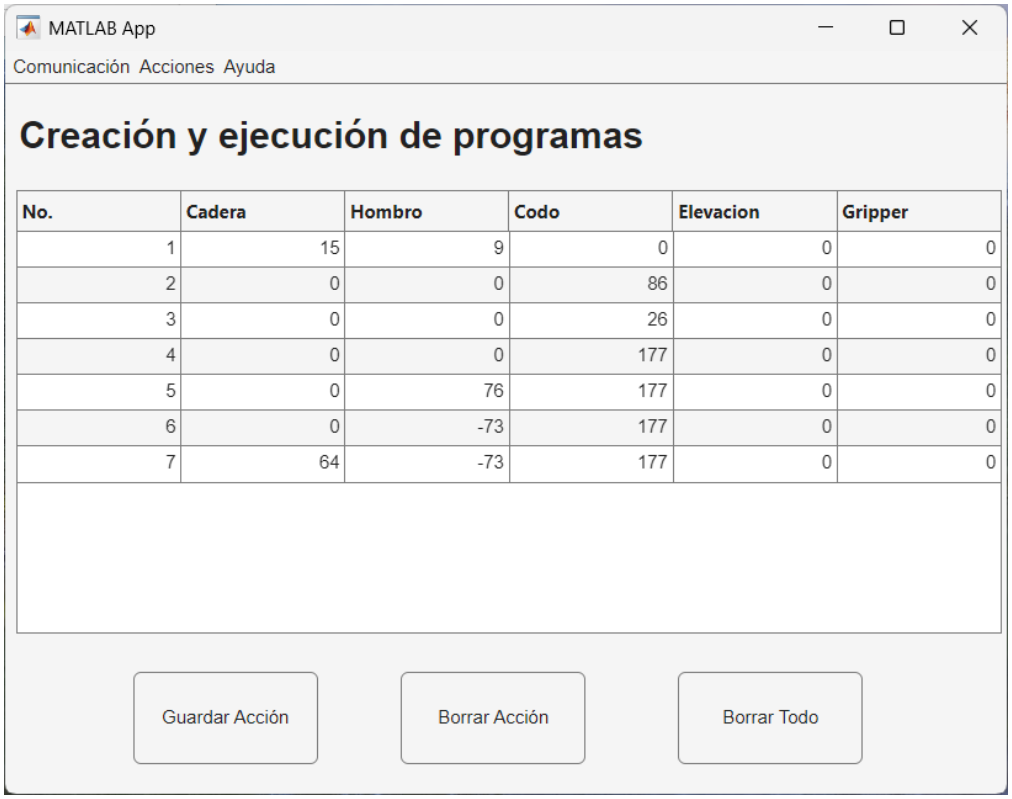


Figure 11: Ventana Creación y ejecución de programas (tabla vacía).

4. Regrese al módulo **Control Manual**.
5. Confirme visualmente en la simulación URDF que la postura es correcta y estable.
6. Regrese a la ventana **Creación y ejecución de programas**.
7. Presione el botón **Guardar Acción**.

Resultado esperado:

- Se agrega la **Posición inicial del robot** a la tabla.
- Los valores articulares registrados corresponden a la postura actual del robot (ver Figura 12).

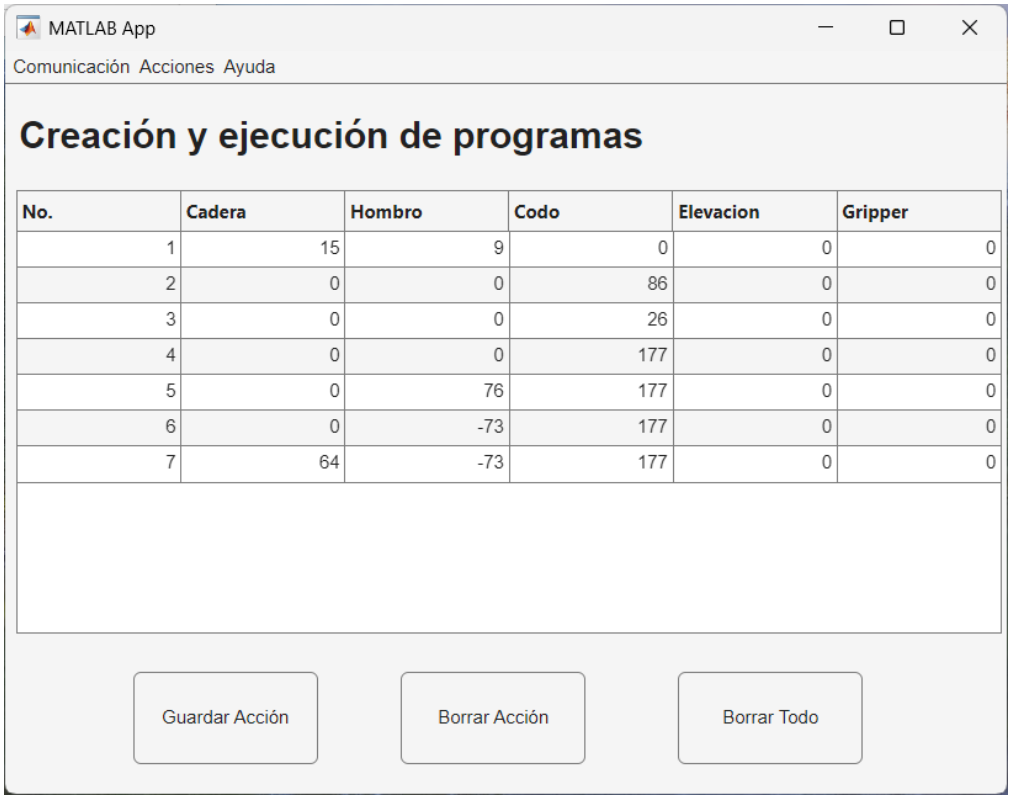


Figure 12: Tabla con acciones capturadas (ejemplo de programa).

9. Regrese nuevamente al módulo **Control Manual**.
10. Modifique los valores articulares para generar una nueva postura distinta a la anterior.
11. Verifique visualmente el movimiento en el modelo URDF.
12. Regrese a la ventana **Creación y ejecución de programas**.
13. Presione **Guardar Acción** para registrar la nueva postura.
14. Repita el procedimiento hasta registrar el programa necesario que indique el profesor.
15. En caso de detectar una acción incorrecta:
    - Seleccione la acción correspondiente.
    - Presione el botón **Borrar Acción** para eliminarla.
16. Si es necesario reiniciar completamente el programa:

- Presione **Borrar Todo**.
- Vuelva a capturar las acciones desde el inicio.

17. Finalmente, en la ventana **Creación y ejecución de programas**, abra el menú **Acciones** y seleccione la opción para guardar el programa (ver Figura 13).

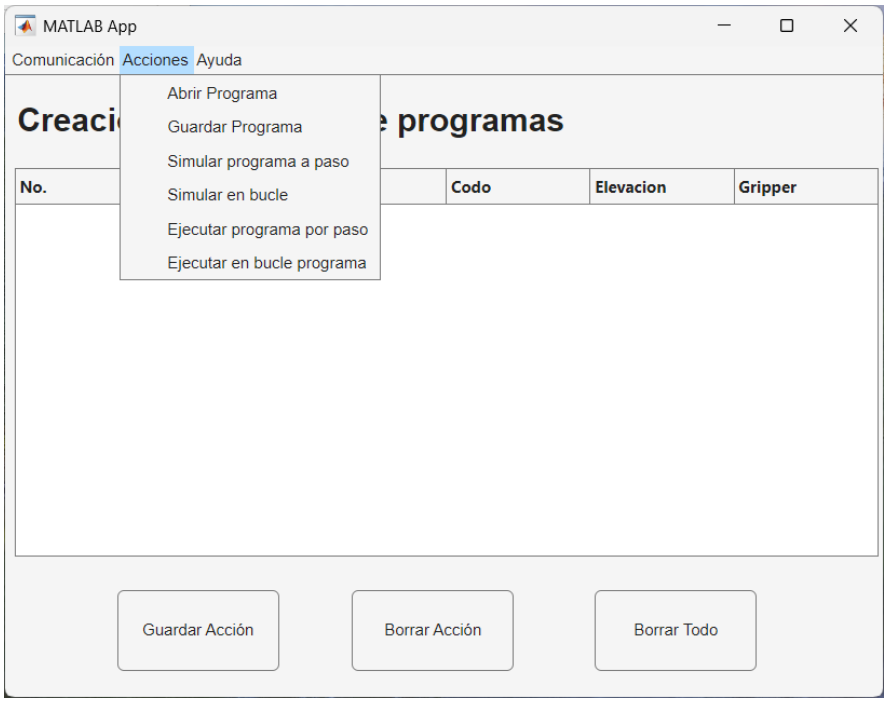


Figure 13: Menú Acciones en la ventana Creación y ejecución de programas.

4.6.4 Evidencia a entregar (Parte 2)

El reporte deberá incluir:

- Tabla final de acciones, indicando los valores articulares correspondientes.
- Descripción breve del criterio utilizado para definir cada punto (postura inicial, intermedia y final).
- Observaciones relacionadas con:
  - Facilidad o dificultad para definir puntos.
  - Articulaciones más sensibles a cambios.
  - Importancia de evitar la definición de puntos cercanos a los límites articulares.

4.6.5 Parte 3. Ejecución y validación del programa en simulación

En esta parte se validará el programa por puntos definido previamente mediante su ejecución en simulación, utilizando los modos de ejecución **por pasos** y **continua (bucle)**. El objetivo es verificar que la secuencia de acciones se ejecute de forma ordenada, sin movimientos inesperados y respetando la lógica de control secuencial antes de transferir el programa al robot real. **Preparación y carga de programa de puntos**

1. Abrir el módulo **Control Manual** (correspondiente a la Parte 1).
2. Activar la simulación del robot desde el menú **Simulación**.
3. Seleccionar **Simulación** → **Iniciar simulación** (ver Figura 13).
4. Verificar que el robot se visualiza en el entorno (URDF) (ver Figura 10).
5. Abrir el módulo **Crear/Ejecutar** (**Creación y ejecución de programas**).
6. En el menú **Acciones**, seleccionar **Abrir Programa** (ver Figura 13).
7. Seleccionar el archivo correspondiente al programa guardado en la Parte 2.
8. Verificar que la tabla se llena con las acciones y que la secuencia está completa y ordenada.

Ejecución en modo por pasos (Simular programa a paso)

Cada acción se ejecutará individualmente, verificando que el robot alcance cada punto antes de avanzar al siguiente.

1. En el menú **Acciones**, seleccionar **Simular programa a paso** (ver Figura 13).
2. Ejecutar la primera acción y observar el movimiento del robot en la ventana URDF.
3. Confirmar visualmente que el robot alcanza la postura del punto actual antes de avanzar.
4. Continuar paso a paso hasta completar la secuencia.

Durante la ejecución por pasos, registrar si ocurre alguno de los siguientes casos:

- Transiciones demasiado bruscas entre puntos.
- Posturas cercanas a límites articulares.
- Configuraciones no recomendables (por ejemplo, cercanía entre eslabones).

Si se detecta algún problema, realizar lo siguiente:

- Detener la simulación.

- Regresar al módulo **Crear/Ejecutar**.
- Eliminar o corregir la acción correspondiente seleccionando la celda y corrigiendo el valor manualmente.
- Repetir la simulación por pasos hasta validar la secuencia.

Ejecución en modo continuo

1. En el menú **Acciones**, seleccionar **Simular en bucle** (ver Figura 13).
2. Observar la ejecución continua del programa completo.
3. Verificar:
   - continuidad del movimiento,
   - repetibilidad,
   - ausencia de configuraciones forzadas.

4.6.6 Evidencia a entregar (Parte 3)

- Captura o registro de que la simulación URDF fue iniciada correctamente.
- Evidencia del programa cargado (tabla con acciones).
- Observaciones de ejecución por pasos y en bucle.
- Ajustes realizados (si existieron) y versión final del programa.

4.6.7 Parte 4. Transferencia del programa validado y preparación para la ejecución en el robot real

Preparación inicial

- Verificar que el robot real se encuentre apagado o desenergizado.
- Verificar que el robot real se encuentre mecánicamente libre de obstáculos y que el área de trabajo esté despejada.
- Encender la fuente de alimentación del sistema robótico.
- Energizar el sistema de control del robot.
- Ejecutar la aplicación.
- Abrir el módulo **Control Manual**.

- Verificar que el enlace de comunicación con el robot se encuentre activo.
- Confirmar que la interfaz no muestre mensajes de error o pérdida de conexión.

Carga del programa en el sistema de ejecución real

- Abrir el módulo **Crear/Ejecutar**.
- En el menú **Acciones**, seleccionar **Abrir Programa**.
- Cargar el programa previamente validado en simulación (Parte 3).
- Verificar que la tabla muestre todas las acciones correctamente.

Ejecución del programa en el robot real (según indicación del profesor)

Nota: El modo de ejecución será indicado por el profesor responsable del laboratorio.

a) Ejecución manual (por pasos)

1. En el módulo **Crear/Ejecutar**, abrir el menú **Acciones**.
2. Seleccionar la opción **Ejecutar programa por paso**.
3. Ejecutar la primera acción del programa.
4. Observar el movimiento del robot hasta que alcance la postura correspondiente.
5. Confirmar la finalización del movimiento antes de avanzar al siguiente paso.
6. Repetir el procedimiento hasta completar la secuencia o hasta que el profesor lo indique.

b) Ejecución continua (en bucle)

1. En el módulo **Crear/Ejecutar**, abrir el menú **Acciones**.
2. Seleccionar la opción **Ejecutar en bucle programa**.
3. Observar la ejecución continua del programa completo.
4. Mantener supervisión constante del comportamiento del robot.
5. Detener la ejecución cuando el profesor lo indique o si se detecta un comportamiento anómalo.

Detención del programa y cierre de la ejecución

1. Detener la ejecución del programa desde la interfaz.
2. Regresar el robot a una posición segura si es necesario.
3. Desenergizar el sistema robótico conforme al procedimiento del laboratorio.

4.6.8 Evidencia a entregar (Parte 4)

El reporte deberá incluir:

1. Confirmación de energización del robot.
2. Evidencia de comunicación correcta con la interfaz.
3. Modo de ejecución utilizado (por pasos o en bucle).
4. Observaciones del comportamiento del robot durante la ejecución real.

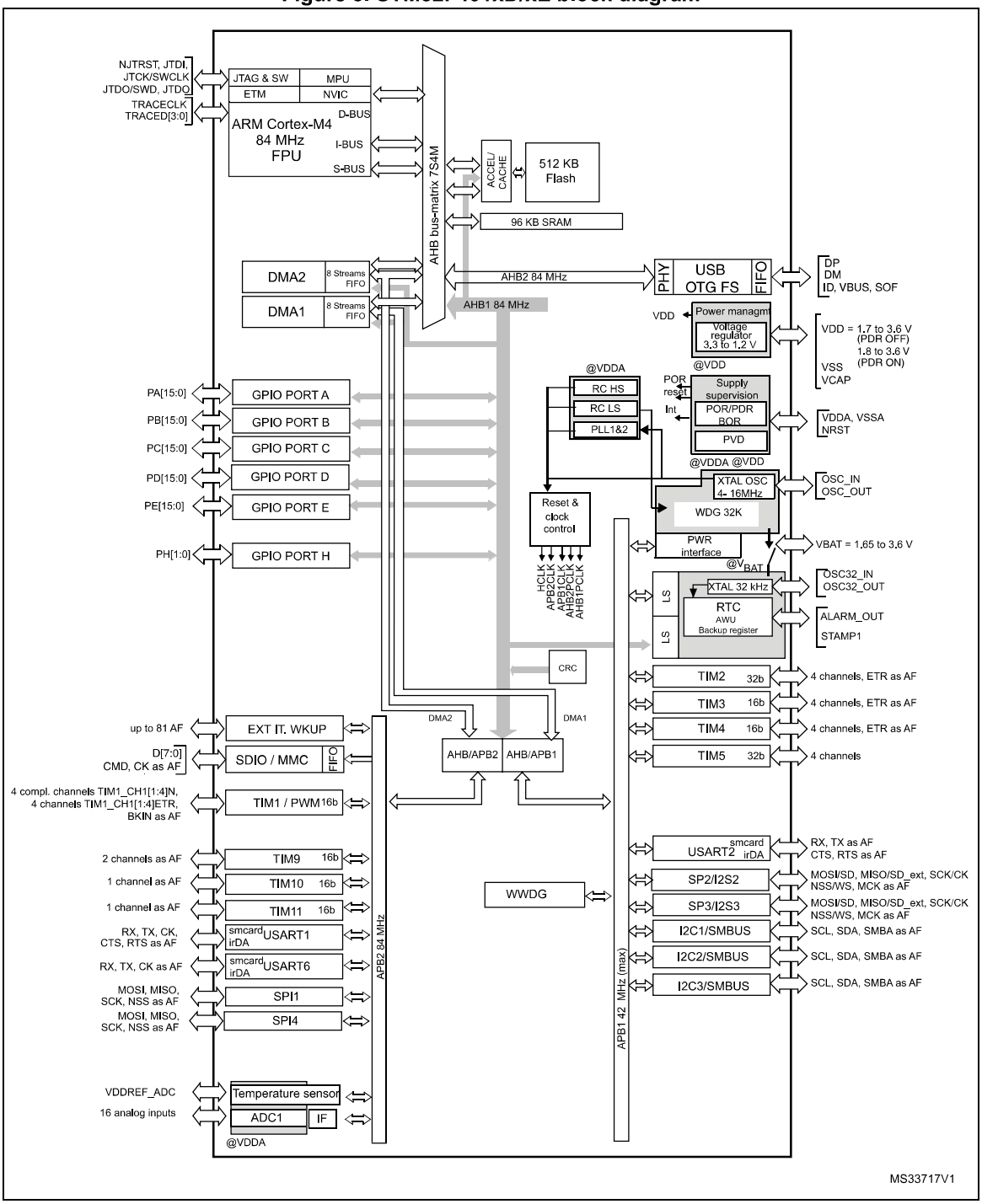
4.7 Conclusiones

References

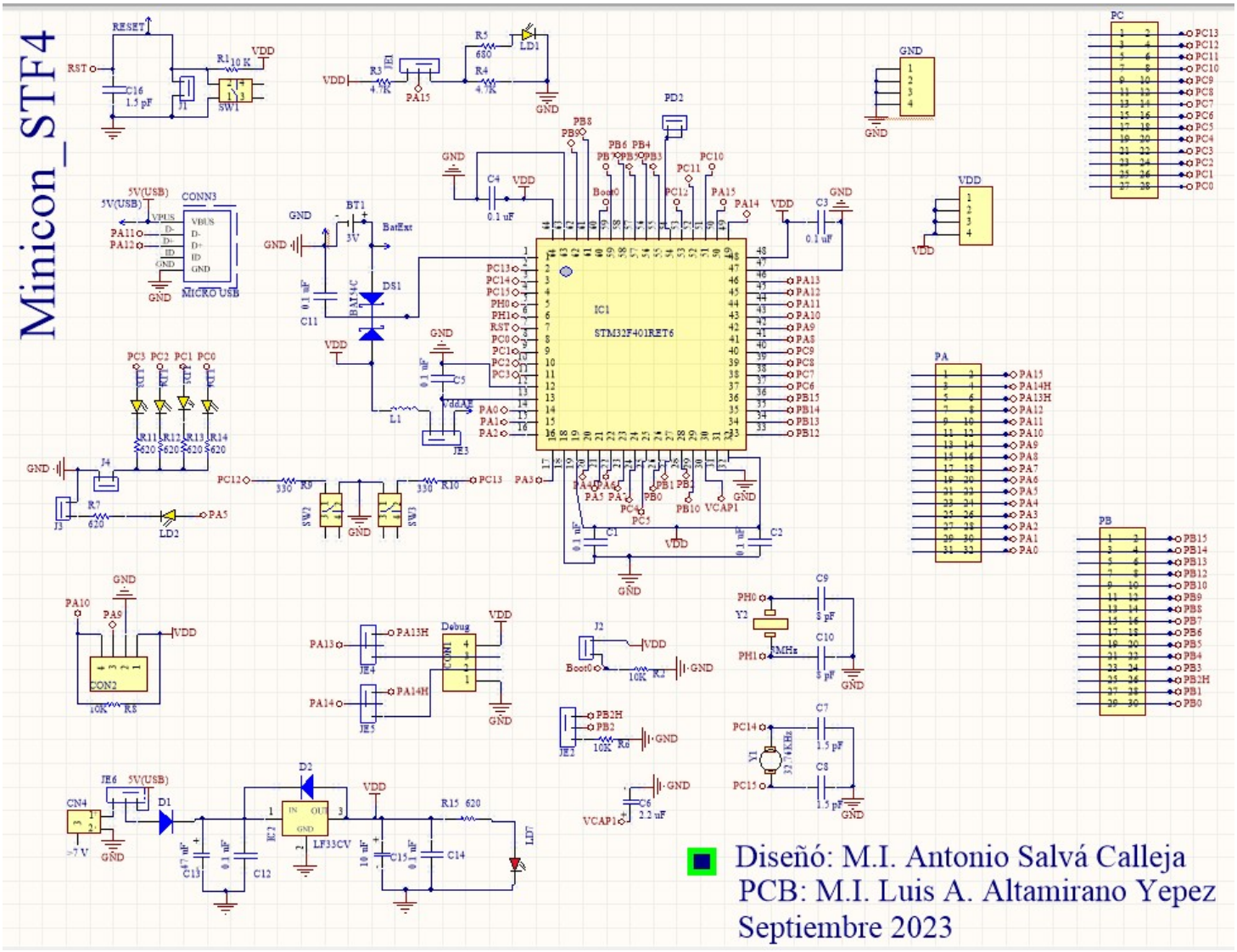
[1] M. I. A. Ferreira and S. R. Fletcher, Eds., *The 21st Century Industrial Robot: When Tools Become Collaborators*. Springer, 2021, doi:10.1007/978-3-030-78513-0.

[2] A. Barrientos, L. Peñín, C. Balaguer y R. Aracil, *Fundamentos de Robótica*. McGraw-Hill, 2007.

Figure 3. STM32F401xD/xE block diagram



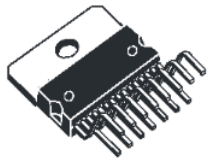
1. The timers connected to APB2 are clocked from TIMxCLK up to 84 MHz, while the timers connected to APB1 are clocked from TIMxCLK up to 42 MHz.


Minicon_STF4
IC1
STM32F401RET6
Diseñó: M.I. Antonio Salvá Calleja
PCB: M.I. Luis A. Altamirano Yepez
Septiembre 2023

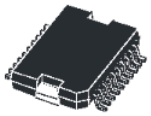
L298

DUAL FULL-BRIDGE DRIVER

- OPERATING SUPPLY VOLTAGE UP TO 46 V
- TOTAL DC CURRENT UP TO 4 A
- LOW SATURATION VOLTAGE
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5 V (HIGH NOISE IMMUNITY)



Multiwatt15



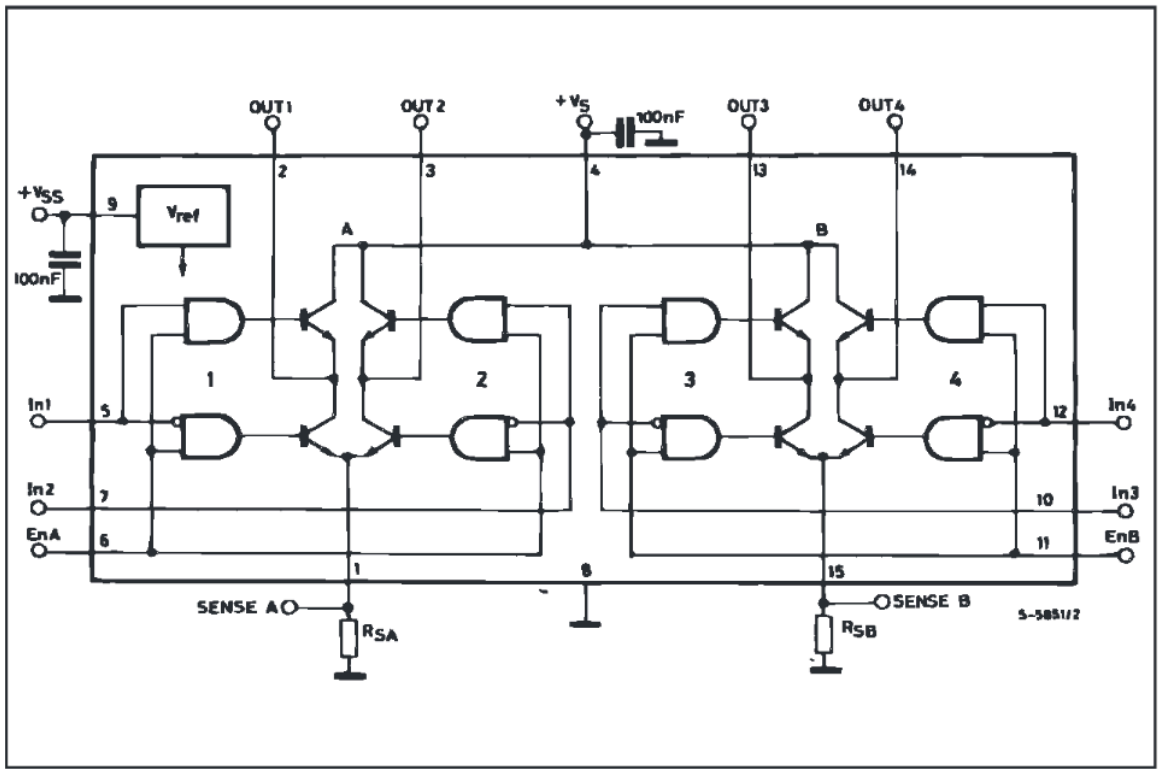
PowerSO20

ORDERING NUMBERS : L298N (Multiwatt Vert.)
L298HN (Multiwatt Horiz.)
L298P (PowerSO20)

DESCRIPTION

The L298 is an integrated monolithic circuit in a 15-lead Multiwatt and PowerSO20 packages. It is a high voltage, high current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors. Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

BLOCK DIAGRAM



MODULO_III
MODULO_II
Departamento de
Control y Robótica
J1
ALIMENTACION
L
N
I
+
MUN_IZQ/DER
ENB IN4 IN3 IN2 IN1 ENA 3v3 GND +12V
OUT3 OUT2
OUT4 OUT1
L298N
ALIMENTACION_2
220V AC
12V CC
700 mA
L
N
+
-
U2
C1
C2
L298N
OUT1 OUT4
OUT2 OUT3
+12V GND 3v3 ENA IN1 IN2 IN3 IN4 ENB
CODO/GRIPPER
L298N
OUT1 OUT4
OUT2 OUT3
+12V GND 3v3 ENA IN1 IN2 IN3 IN4 ENB
CADERA/BRAZO
INSTITUTO
DE INGENIERÍA
UNAM
D1 D7 R7
D13 R1
R13
D2 D8 R8
D14 R2
R14
D3 D9 R9
D15 R3
R15
D4 D10 R10
D16 R4
R16
D5 D11 R11
D17 R5
R17
D6 D12 R12
D18 R6
R18
MICROCONTROLADOR
ASC
MINICON_STF_4B
DTR RXD TXD VCC CTS GND
FTDI MODULE
PWR DCD
TXDEN DSR
SLEEP GND
CTS RI
3V3 RXD
5V VCCIO
RXLED RTS
TXLED DTR
GND TXD
FTDI
JSPR/EDCM PAPIME PE100725

```c
/* ============================================================
 * MINICON STM32F446RE — Firmware principal (Protocolo MATLAB)
 * ------------------------------------------------------------
 * UART : USART1 (PA9 TX, PA10 RX)
 * ADC  : PA0,PA1,PA6,PA7,PB0,PB1
 * PWM  : PC6,PC7,PC8,PC9,PA11,PB8
 * MOT  : PB3–PB15 (IN1/IN2)
 * TIM2 : 100 Hz
 * ============================================================ */

#include "stm32f4xx.h"
#include "usart1.h"
#include "adc1.h"
#include "pwm.h"
#include "motor.h"
#include "control.h"
#include "timer2.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <math.h>

#define NUM_CH 6
/* ============================================================
 * Variables globales
 * ============================================================ */
volatile uint16_t sp_mV[NUM_CH];
volatile uint16_t meas_mV[NUM_CH];
volatile uint8_t  active_id = 0;
static char txbuf[128];

/* === Variables de estado del sistema === */
volatile uint8_t  g_paro_active = 0;
volatile uint8_t  g_tx_enabled  = 0;
volatile float    g_tx_hz       = 33.0f;   // Hz de transmisión
volatile uint16_t g_tx_counter  = 0;

volatile uint8_t  g_step_active = 0;
volatile uint8_t  g_step_done   = 0;
volatile uint8_t  g_step_sent   = 0;
volatile uint16_t g_stable_cnt  = 0;
volatile uint16_t g_dwell_cnt   = 0;



/* ============================================================
 * Prototipos locales
 * ============================================================ */
static void GPIO_Init_PC0_PC3(void);
static void LED_ClearAll(void);

/* ============================================================
 * MAIN
 * ============================================================ */
int main(void)
{
    __disable_irq();

    SystemCoreClockUpdate();

    GPIO_Init_PC0_PC3();
    USART1_Init(115200);
    ADC1_Init_MINICON();
    ADC_LPF_Init_All(150);
    PWM_Init_MINICON(20000.0f);  // 20 kHz
```

```
    Motor_Init_All();
    Motor_Stop_All();
    TIM2_Init_100Hz();

    /* === Posición inicial (HOME) === */
    sp_mV[0] = 1184;
    sp_mV[1] = 1436;
    sp_mV[2] = 590;
    sp_mV[3] = 800;
    sp_mV[4] = 1050;
    sp_mV[5] = 1060;

    /* === Inicialización PID === */
    PID0_Init(1600.0f, 50.0f, 30.0f, 0.01f, 0.02f, 100.0f, g_deadband); // CADERA
    PID1_Init(1675.0f, 80.0f, 25.0f, 0.01f, 0.02f, 100.0f, g_deadband); // HOMBRO
    PID2_Init(2000.0f,100.0f, 50.0f, 0.01f, 0.02f, 100.0f, g_deadband); // CODO
    PID3_Init(1500.0f, 50.0f, 20.0f, 0.01f, 0.02f, 100.0f, g_deadband); // GRIPPER
    PID4_Init(3000.0f,150.0f, 60.0f, 0.01f, 0.02f, 100.0f, g_deadband); // MUN IZQ
    PID5_Init(3000.0f,150.0f, 60.0f, 0.01f, 0.02f, 100.0f, g_deadband); // MUN DER

    PID0_SetDeadband(g_error_db);
    PID1_SetDeadband(g_error_db);
    PID2_SetDeadband(g_error_db);
    PID3_SetDeadband(0.070f);
    PID4_SetDeadband(0.070f);
    PID5_SetDeadband(g_error_db);

    __enable_irq();

    USART1_WriteStr("MINICON listo\r\n");

    /* ============================================================
     * Bucle principal
     * ============================================================ */
    char line[128];

    while (1)
    {
        /* === Procesar comandos recibidos por UART === */
        if (USART1_GetLine(line, sizeof(line)))
            USART1_ProcessLine(line);

        /* === Confirmación de fin de paso === */
        if (g_step_done && !g_step_sent) {
            USART1_SendFinalizado();
            g_step_sent = 1;
        }
    }
}

/* ============================================================
 * TIM2_IRQHandler — lazo de control 100 Hz
 * ============================================================ */
void TIM2_IRQHandler(void)
{
    if (TIM2->SR & TIM_SR_UIF)
    {
        TIM2->SR &= ~TIM_SR_UIF;

        /* --- Si hay PARO, detener motores --- */
        if (g_paro_active) {
            Motor_Stop_All();
            return;
        }

```

```
131         /* --- Lectura ADC y control PID --- */
132         ADC1_ReadAll_mV(meas_mV);
133 
134         int e_tol_mV = (int)(g_error_db * 1000.0f + 0.5f);
135         if (e_tol_mV < 25) e_tol_mV = 25;
136 
137         uint8_t all_ok_err = 1;
138         uint8_t all_ok_u   = 1;
139 
140         for (int i=0; i<NUM_CH; i++)
141         {
142             float e = ((float)sp_mV[i] - (float)meas_mV[i]) / 1000.0f;
143             float u_i = 0.0f;
144 
145             switch(i){
146                 case 0: u_i = PID0_Step(e); break;
147                 case 1: u_i = PID1_Step(e); break;
148                 case 2: u_i = PID2_Step(e); break;
149                 case 3: u_i = PID3_Step(e); break;
150                 case 4: u_i = PID4_Step(e); break;
151                 case 5: u_i = PID4_Step(e); break;
152 
153 
154             }
155 
156             if (u_i > 0.0f)      Motor_Command(i,  u_i, DIR_CW);
157             else if (u_i < 0.0f) Motor_Command(i, -u_i, DIR_CCW);
158             else                 Motor_Command(i, 0.0f, DIR_CW);
159 
160             int abs_err_mV = abs((int)sp_mV[i] - (int)meas_mV[i]);
161             if (abs_err_mV > e_tol_mV)        all_ok_err = 0;
162             if (fabsf(u_i) > 3.0f)            all_ok_u   = 0;
163         }
164 
165         /* --- Transmisión periódica de datos --- */
166         if (g_tx_enabled)
167         {
168             static float tick_accum = 0.0f;
169             float tick_target = 100.0f / g_tx_hz;
170 
171             if (++tick_accum >= tick_target)
172             {
173                 tick_accum = 0.0f;
174                 sprintf(txbuf,"V:%u,%u,%u,%u,%u,%u\r\n",
175                         meas_mV[0],meas_mV[1],meas_mV[2],
176                         meas_mV[4],meas_mV[3],meas_mV[5]);
177                 USART1_WriteStr(txbuf);
178             }
179         }
180     }
181 }
182 
183 /* ============================================================
184  * Utilidades
185  * ============================================================ */
186 static void LED_ClearAll(void){
187     GPIOC->BSRR=(GPIO_BSRR_BR0|GPIO_BSRR_BR1|GPIO_BSRR_BR2|GPIO_BSRR_BR3);
188 }
189 
190 static void GPIO_Init_PC0_PC3(void){
191     RCC->AHB1ENR |= RCC_AHB1ENR_GPIOCEN;
192     GPIOC->MODER &= ~((3u<<(0*2))|(3u<<(1*2))|(3u<<(2*2))|(3u<<(3*2)));
193     GPIOC->MODER |=  ((1u<<(0*2))|(1u<<(1*2))|(1u<<(2*2))|(1u<<(3*2)));
194     LED_ClearAll();
195 }
```