UNIVERSIDAD PENÍNSULA DE SANTA ELENA

FACSISTEL

INGENIERÍA EN ELECTRÓNICA Y AUTOMATIZACIÓN

TRABAJO DE INTEGRACIÓN CURRICULAR

DESARROLLO DE UN SISTEMA DE CONTROL CON EL ALGORITMO LQG APLICADO AL SEGUIMIENTO DE TRAYECTORIA DE UN BRAZO ROBÓTICO

VASCONEZ A. LAURA, TOMALÁ P. LIZANDRO

Dirigido por:
ING. FIGUEROA O. JUNIOR

La Libertad - 2025

Desarrollo de un sistema de control con el algoritmo LQG aplicado al seguimiento de trayectoria de un brazo robótico.

Vasconez A. Laura, Tomalá P. Lizandro

2025-07-29

DEDICATORIA

Dedico este trabajo, en primer lugar, a Dios, por ser mi guía y fortaleza en cada paso de mi vida, por iluminar mi camino con sabiduría y brindarme la perseverancia necesaria para alcanzar mis metas.

A mis padres Hugo Asencio y Lorena Tomalá, por su amor incondicional, su apoyo constante y por creer en mí incluso en los momentos más difíciles. Cada uno de ustedes ha sido una fuente de inspiración y motivación a lo largo de este camino. A mi esposa, Juleidy, por caminar a mi lado con paciencia, amor y comprensión, siendo mi compañera fiel en cada reto. Y a mi hija Raquelita, mi mayor inspiración, cuyo amor y sonrisa me impulsan a seguir adelante y dar siempre lo mejor de mí.

Y también dedico este esfuerzo a todas las personas apasionadas por la tecnología y el desarrollo de la robótica, especialmente en el área de los brazos robóticos. Que este trabajo sirva como un pequeño aporte al conocimiento colectivo y una motivación para quienes desean innovar y construir un futuro impulsado por la ingeniería.

Tomala Panchana Lizandro Vinicio

DEDICATORIA

Con profundo agradecimiento y cariño dedico esta tesis a mi familia, cuyo amor incondicional y apoyo constante constituyeron el refugio y la fortaleza que impulsaron cada paso en este proceso académico.

A mis amigos, por su comprensión, aliento y compañía, que hicieron más llevadero el camino de retos y aprendizajes compartidos.

Finalmente, me dedico a mí misma, reconociendo la perseverancia, la pasión y la disciplina que permitieron transformar cada obstáculo en una oportunidad de crecimiento profesional y personal.

Como se afirma, "los grandes logros nacen de pequeños esfuerzos repetidos día tras día." Que este trabajo refleje la dedicación y constancia que sostienen el avance académico y personal.

Vasconez Alava Laura Lorena

AGRADECIMIENTO

Desde lo más profundo de mi corazón le doy gracias a Dios por estar cada segundo en mi vida y brindarme la oportunidad de ser feliz con mi familia. A mi familia, mi padre Hugo Asencio y su hermano Sgto. Edinson Holguín, a mi mamá Lorena Tomalá y mis hermanas Carla y Andrea que con mucho sacrificio me ofrecieron todo lo necesario para poderme desarrollar como un profesional.

A mi esposa Juleidy y mi pequeña Raquelita quienes se han convertido en un ser muy importante en mi vida estando presente en los mejores y peores momentos de mi vida en los últimos años. Extiendo mi gratitud a toda su familia que me han acogido como un miembro más de su hogar.

A mi tutor Ing. Junior Figueroa por compartir su conocimiento para el buen desarrollo de este trabajo de titulación y a mi compañera de tesis Laura Vasconez, por su compromiso, apoyo y colaboración constante durante todo este proceso.

Tomala Panchana Lizandro Vinicio

AGRADECIMIENTO

Agradezco, ante todo, a Dios, por ser guía constante en este camino. Su presencia me dio serenidad en los momentos de incertidumbre, fortaleza en la dificultad y claridad para seguir adelante con fe y esperanza.

A mis padres, Solanda Alava y Fabián Vasconez, les expreso mi más profundo agradecimiento por su amor incondicional, su apoyo inquebrantable y su confianza en mí. A mi madre, por su ternura, su entrega sin límites y por estar presente en cada paso con paciencia, cuidado y palabras de aliento. A mi padre, por su presencia firme, sus consejos serenos y por brindarme la seguridad de saber que siempre podía contar con él. A mis hermanos, Brandon y Zuleyka, gracias por su compañía cercana, por el aliento en los momentos más desafiantes y por ser una fuente constante de afecto y apoyo.

Mi gratitud también es para Carlos Mora, cuya paciencia, comprensión y apoyo han sido fundamentales tanto en lo personal como en lo académico. Su presencia ha significado un verdadero sostén en los momentos de mayor exigencia, motivándome a seguir con determinación. Del mismo modo, agradezco profundamente a mis amigos Bryan Sánchez y Josué Vera, por su respaldo sincero y su cercanía constante durante esta etapa.

Finalmente, extiendo un reconocimiento especial al Ingeniero Junior Figueroa, tutor de esta tesis, por su guía y dedicación, y a mi compañero Lizandro Tomala, por su compromiso y colaboración, que han sido claves para la culminación exitosa de este proyecto académico.

Vasconez Alava Laura Lorena

APROBACIÓN DEL TUTOR

En mi calidad de tutor del trabajo de titulación denominado: **"Desarrollo de un sistema de control con el algoritmo LQG aplicado al seguimiento de trayectoria de un brazo robótico."**, elaborado por el estudiante **Vasconez Alava Laura Lorena, Tomalá Panchana Lizandro Vinicio**, de la carrera de **Ingeniería en Electrónica y Automatización** de la Universidad Estatal Península de Santa Elena, declaro que luego de haber orientado, estudiado y revisado, lo apruebo en todas sus partes y autorizo al estudiante que inicie los trámites legales correspondientes.

JUNIOR RAFAEL FIGUEROA OLMEDO

Firmado digitalmente por JUNIOR RAFAEL FIGUEROA OLMEDO
Fecha: 2025.06.27 08:31:00 -05'00'

Ing. Junior Figueroa Olmedo, Mgtr.
Tutor

DECLARACIÓN

El contenido del presente trabajo de titulación es de nuestra responsabilidad, el patrimonio intelectual del mismo le pertenece a la Universidad Estatal Península de Santa Elena.

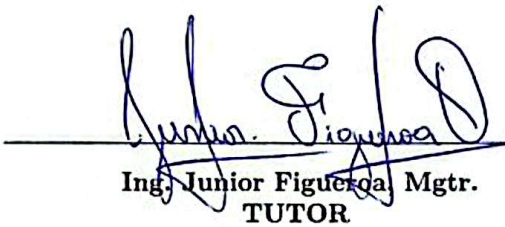
Vasconez Alava Laura Lorena
Autor

Tomalá Panchana Lizandro Vinicio
Autor

TRIBUNAL DE GRADO

Ing. Ronald Ruvira Jurado, Ph.D.
DIRECTOR DE CARRERA

Ing. Edisson Coral, Mgtr.
DOCENTE ESPECIALISTA



Ing. Junior Figueroa, Mgtr.
TUTOR

Ing. Luis Enrique Chuquimarca, Mgtr.
DOCENTE GUÍA UIC

Ing. Corina Gonzabay de la A, Mgtr.
SECRETARIA DEL TRIBUNAL

RESUMEN

El presente trabajo de titulación aborda el diseño e implementación de un sistema de seguimiento de trayectorias aplicado a un brazo robótico de cinco grados de libertad, integrando técnicas de modelado, control y visualización. Para el modelado cinemático y dinámico del sistema se utilizó el enfoque de Euler-Lagrange, implementado en MATLAB, permitiendo obtener las ecuaciones de movimiento que representan el comportamiento del manipulador. A partir de este modelo se diseñó un controlador óptimo LQG (Linear Quadratic Gaussian), el cual fue programado en el entorno de desarrollo Spyder utilizando el lenguaje Python. Este algoritmo combina un regulador LQR y un filtro de Kalman para estimar estados no medibles y minimizar el error cuadrático esperado bajo presencia de ruido y perturbaciones.

Las trayectorias a seguir, fueron generadas mediante instrucciones G-code exportadas desde Inkscape, utilizando comandos de interpolación lineal y circular. Estos datos fueron procesados en MATLAB y convertidos a coordenadas temporizadas compatibles con el sistema de control. Para la validación del controlador, se ejecutaron tres trayectorias distintas con características geométricas.

Además, se desarrolló una interfaz gráfica en MATLAB App Designer que facilita la interacción con el sistema, permitiendo ejecutar movimientos manuales, cargar trayectorias programadas y supervisar variables críticas del sistema. La interfaz está dividida en secciones funcionales que optimizan el control del brazo y mejoran la experiencia del usuario durante las pruebas de simulación.

ABSTRACT

This thesis presents the design and implementation of a trajectory tracking system applied to a five-degree-of-freedom robotic arm, integrating modeling, control, and visualization techniques. The kinematic and dynamic modeling of the system was carried out using the Euler-Lagrange approach, implemented in MATLAB, which allowed the derivation of the motion equations representing the manipulator's behavior. Based on this model, an optimal LQG (Linear Quadratic Gaussian) controller was designed and programmed in the Spyder development environment using the Python language. This algorithm combines an LQR regulator with a Kalman filter to estimate unmeasurable states and minimize the expected quadratic error under the presence of noise and disturbances.

The trajectories to be followed were generated using G-code instructions exported from Inkscape, employing linear and circular interpolation commands. These data were processed in MATLAB and converted into time-coordinated paths compatible with the control system. For controller validation, three distinct trajectories with different geometric characteristics were executed.

Moreover, a graphical user interface (GUI) was developed in MATLAB App Designer to facilitate interaction with the system, enabling manual movement execution, programmed trajectory loading, and real-time monitoring of critical system variables. The interface is divided into functional sections that optimize arm control and enhance the user experience during simulation tests.

Índice general

Índice de figuras

1 Introducción

En este capítulo, se presenta la relevancia y la importancia práctica del proyecto, junto con una evaluación del estado actual del problema abordado. el presente proyecto expone lo que son avances en el campo de la robótica, destacando el desarrollo de sistemas de control cada vez más precisos y capaces de operar en entornos dinámicos. Uno de los principales desafíos ha sido lograr que los brazos robóticos sigan trayectorias de manera precisa, en especial al realizar tareas que requieran mover objetos o realizar movimientos. [1].

Para que un manipulador ejecute estas acciones con precisión, es necesario generar una trayectoria, es decir, una serie de puntos por los cuales el manipulador debe pasar. En este trabajo, se utiliza un brazo robótico comercial de cinco grados de libertad; aunque no se trata de un robot industrial, este permite probar técnicas de control avanzadas en condiciones reales.

La relevancia de este trabajo radica en que, si bien los controladores tradicionales son efectivos en muchas situaciones, no siempre son suficientes para garantizar un rendimiento óptimo cuando se requiere precisión en entornos dinámicos. En estos casos, los factores como el ruido, los errores de medición o los cambios en el entorno pueden dificultar que los controladores convencionales mantengan una trayectoria precisa. Por esta razón, se hace necesario recurrir a enfoques más avanzados, como el controlador LQG, que integra un controlador LQR con un estimador de estados, como el filtro de Kalman. Esta combinación permite obtener una estimación más precisa del estado del sistema, lo que mejora la exactitud y la confiabilidad en las tareas.[1].

Este trabajo tiene como objetivo desarrollar un sistema de control de trayectoria y posición de un brazo robótico de 5 grados de libertad mediante el uso de un controlador LQG, se modela matemáticamente el robot, se generan las trayectorias de referencia, se diseña e implementa el controlador y, finalmente, se valida su desempeño en un ambiente controlado. Con este enfoque, se pretende demostrar que técnicas avanzadas de control pueden superar a los métodos tradicionales, incluso en manipuladores de uso no industrial.

1.1 Justificación

Hoy en día, la robótica tiene un papel fundamental en áreas como la industria manufacturera, la medicina avanzada y los servicios automatizados. En particular, dentro del desarrollo de sistemas robóticos manipuladores, uno de los principales desafíos es lograr que los brazos robóticos puedan seguir trayectorias con alta precisión. Este aspecto es especialmente complicado cuando los brazos robóticos operan en entornos variables, donde pueden presentarse interferencias, ruidos o cambios inesperados. Estas condiciones pueden afectar directamente a la ejecución de tareas que requieren movimientos precisos, como el ensamblaje de componentes pequeños o procedimientos médicos delicados.[2], [3].

El objeto de estudio en este trabajo se enfoca en el desarrollo de un sistema de control con el algorimo LQG aplicado al seguimiento de trayectorias en un brazo robótico de cinco grados de libertad. Aunque este tipo de robot no pertenece a la categoría industrial, representa una plataforma adecuada para validar estrategias de control modernas en condiciones reales [4]. La elección de este objeto responde a la necesidad de superar las limitaciones de controladores clásicos, los cuales si bien son ampliamente utilizados por su simplicidad, no ofrecen un desempeño óptimo cuando el sistema se ve afectado por ruidos, incertidumbres o variaciones en su entorno operativo [5]. El control óptimo es una estrategia de control que ha sido estudiada durante siglos, y se le atribuye su nacimiento al problema sobre el braquistócrono, propuesto por el matemático Johann Bernoulli en 1696 [6]. Desde entonces, muchos investigadores han diseñado distintas técnicas de control moderno basadas en el control óptimo, tales como: Control MPC (Model Predictive Control) y Control H-infinity.

A partir de esta problemática, surge la necesidad de explorar métodos más robustos que permitan una mejor estimación del estado del sistema y una regulación más eficiente de sus movimientos. En este sentido, el empleo del controlador LQG, ofrece una alternativa prometedora. [2]. Esto permite resaltar la necesidad de investigar y abordar este problema mediante el desarrollo e implementación de técnicas de control más avanzadas, contribuyendo así al avance tecnológico y a la formación de soluciones más robustas y eficientes.

1.2 Panorama actual

En los últimos años, el desarrollo de sistemas de control aplicados a la robótica ha sido objeto de múltiples investigaciones, debido a la creciente demanda de precisión y fiabilidad en tareas automatizadas. Particularmente, el control de trayectoria en manipuladores robóticos representa un área de estudio fundamental, ya que influye directamente en el rendimiento y la eficiencia de procesos industriales, médicos y de servicio.

En esta sección se presenta un análisis detallado de las investigaciones previas que sirven como referencia para el desarrollo del presente trabajo de titulación. También se analiza la problemática y se identifican los hallazgos más importantes que sirven de base para esta propuesta. La información se toma de fuentes confiables, como libros y artículos científicos, con el fin tener un respaldo teórico del proyecto.

Según el artículo "*Controlador LQG experimental de propósito didáctico*" el cual fue expuesto por el Congreso anual de la AMCA, cuyo propósito de esta labor es exponer la creación y uso de un controlador digital con un enfoque didáctico, programado en C++. Este controlador tiene la finalidad de facilitar la experimentación donde se relaciona con el esquema de control óptimo gaussiano, un tema estudiado en los cursos de control lineal. El sistema se ha diseñado para respaldar las actividades prácticas de los estudiantes, permitiéndoles aplicar el control LQG en procesos lineales de hasta quinto orden. El sistema ha sido probado en diversos procesos físicos y se representan los resultados obtenidos al controlar una planta lineal de tercer orden como parte de la aplicación práctica de este enfoque didáctico [7]

La tesis titulada "*Integración de sensores de proximidad en brazo robot comercial*" el proyecto se centra en la integración de seis sensores de proximidad en un brazo robótico comercial, específicamente en el modelo MyCobot 320 PI, que posee seis grados de libertad, el objetivo principal es potencial la capacidad del brazo robótico para interactuar de manera efectiva con su entorno. Se inicia con la comprensión del robot y con pruebas electrónica para usar los sensores fijados con piezas diseñadas. Se optimiza la electrónica para reducir el cableado y se desarrolla una programación que controla el robot y gestiona los datos de los sensores, estableciendo un sistema bidireccional [8].

En el trabajo de titulación, modalidad de proyectos de investigación y desarrollo de la ESPOCH el proyecto titulado "Modelamiento y simulación de

un algoritmo para el control del brazo robótico" se ha desarrollado el modelo cinemático para el control y simulación del brazo robótico KAWASAKI RS 003N, ubicado en la facultad de la ESPOCH. El diseño y modelado del brazo robótico se llevaron a cabo mediante SolidWorks y CAD. Posteriormente, se importó el modelo CAD a Simulink, donde se definió la estructura mecánica del brazo utilizando la herramienta SimMechanics. Se implementó un algoritmo de control para generar trayectorias en manipulador, seguido de pruebas de simulación y control de movimientos del brazo robótico. Los resultados que se obtuvieron revelan que el algoritmo Denavit-Hartenberg posibilita un control preciso de los movimientos del robot [9].

El análisis del panorama actual permitió identificar una falta de aplicación de técnicas avanzadas de control en brazos robóticos no industriales. Esto muestra la necesidad de investigaciones que validen estas técnicas en plataformas comerciales y realicen comparaciones con métodos tradicionales, para mejorar la precisión y eficiencia en tareas que requieren movimientos exactos.

1.3 Objetivos

1.3.1 Objetivos generales

Desarrollar un sistema de control de trayectoria y posición de un brazo robótico de 5 grados de libertad mediante el ajuste computacional de parámetros de un controlador LQG.

1.3.2 Objetivos específicos

- Examinar y analizar información relacionada con el control de brazos robóticos y el algoritmo LQG para comprender el estado actual de la investigación en este campo.

- Identificar el sistema del brazo robótico, mediante su descripción cinemática y dinámica, con la finalidad de obtener el modelo matemático expresado en matrices de estado.

- Evaluar el comportamiento óptimo del sistema modelado mediante la simulación en MATLAB/SIMULINK, con el objetivo de identificar las condiciones más contraproducentes y aplicar los respectivos estimadores o filtros de estado.

- Ajustar las matrices de ponderación de estado y control del regulador LQR (Q, R) y de filtro de Kalman (Qe, Re) mediante métodos de optimización de trayectoria usando MATLAB, de forma que se cumplan los requisitos de rendimiento.

- Desarrollar un diagrama de bloques basado en el modelo en Simulink, de modo que se pueda visualizar la simulación avanzada del comportamiento del sistema.

- Implementar el sistema diseñado basado en el controlador LQG en un microcontrolador para la manipulación del brazo robótico en el prototipo escogido.

- Establecer una comunicación entre dos dispositivos empleados para transmitir los datos.

- Realizar experimentos y pruebas en el brazo robótico controlado por el algoritmo LQG para evaluar su desempeño en términos de precisión, velocidad y estabilidad.

1.4 Fundamentos teóricos

En esta sección, se describe una visión general de lo que es un brazo robótico, abordando su clasificación y los aspectos fundamentales que definen su funcionamiento.

Se describen los conceptos clave como los grados de libertad, que sirven para entender la versatilidad de un brazo robótico según sus movimientos posibles. Se detalla la importancia de la generación de trayectorias para el control de estos sistemas, así como la planificación y ejecución de trayectorias precisas que permitan al brazo robótico llevar a cabo tareas complejas. Además, se presenta el modelo en el espacio de estados, que permite describir y analizar el comportamiento dinámico de los robots, que ofrece una presentación matemática eficiente de sus movimientos y dinámicas.

Además, se expone uno de los controladores más utilizados en robótica; se menciona cómo es su funcionamiento, así como su aplicación tradicional en la robótica. También se abordarán los desafíos que enfrentan estos controladores en términos de precisión y estabilidad frente a perturbaciones y ruidos. Se

evalúa la información sobre el algoritmo de control LQG en el seguimiento de trayectorias de un brazo robótico y cómo puede mejorar la precisión, la estabilidad y el rendimiento.

1.4.1 Conceptos generales de robótica

La robótica se define como un campo de estudio y desarrollo de robots, que son mecanismos automáticos programables capaces de ejecutar tareas de manera autónoma o semiautónoma. Según [10] un robot es un mecanismo actuado y reprogramable que interactúa con su entorno para realizar una variedad de tareas, mediante acciones autónomas (inteligentes) generadas a partir de la información captada por sus sensores.

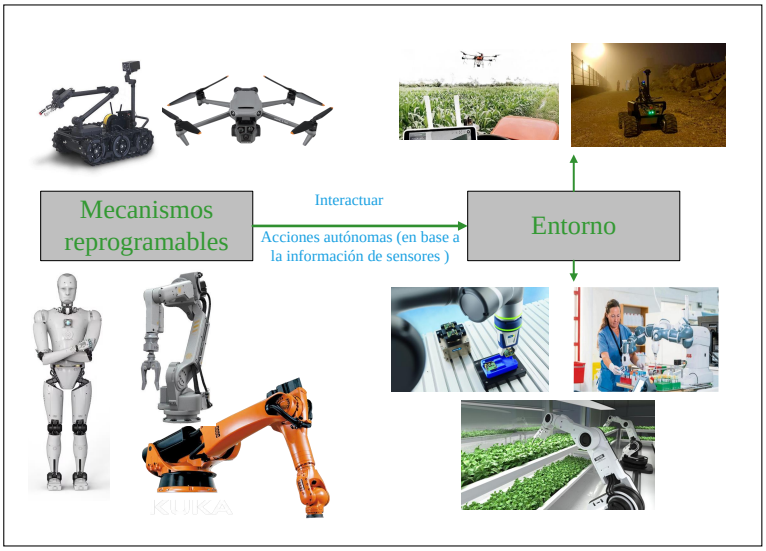


Figura 1.1: Interacción entre mecanismos reprogramables. Esquema adaptado de [10]

En la Figura. 1.1 se aborda la definición de robot al representar distintos tipos de mecanismos reprogramables que interactúan con su entorno a través de acciones autónomas basadas en la información de sensores; la imagen muestra la relación de los robots y su entorno, resaltando su autonomía y capacidad de toma de decisiones [10].

1.4.2 Clasificación general de los robots

Hoy, existe una diversidad significativa de robots, respaldada por muchos fabricantes dedicados a su diseño, producción y comercialización. La clasificación de los robots puede realizarse utilizando distintos criterios, dependiendo del enfoque y el propósito del análisis; en este caso, la categorización se basa en la movilidad del robot dentro del entorno, dividiéndose en dos grandes grupos [10]:

- **Robots fijos o estacionarios**

  Se caracterizan por un desplazamiento restringido y/o limitado, permaneciendo en un área fija durante su operación.

- **Robots móviles**

  Formados por los robots que se desplazan en su entorno de trabajo y, por tanto, cambian la posición y orientación de su chasis (estructura) en el mismo.

En la categoría de robos estacionarios, su desplazamiento es limitado, por lo que es importante mencionar que la subclasificación del robot industrial manipulador Figura.1.1 es usada en procesos de manufactura, producciones industriales, entre otros procesos [10]. Las tareas que este tipo de robot ejecuta son básicamente las de manipular objetos o materiales en un entorno industrial.

Cada articulación está representada con flechas curvas, que muestran los grados de libertad de movimiento que tienes. Gracias a esto, el robot puede realizar movimientos, emulando la flexibilidad y precisión de un brazo humano. El efector final, está ubicado en el xtremo del manipulador; es el que interactúa directamente con el entorno, realizando tareas como agarre, mover objetos, entre otros [10].

Este tipo de robots, con su estructura y capacidad de movimiento, resulta esencial para realizar trabajos que requieren gran precisión y versatilidad. Además de esta clasificación, existen otros tipos de robots según la aplicación para la cual fueron diseñados, destacando principalmente dos categorías:

- **Robot industrial**

  Estos fueron diseñados para realizar tareas en entornos de manufactura, ensamblaje o procesos industriales. Están orientados a mejorar la eficiencia, precisión y seguridad en la producción y se caracterizan por su capacidad para trabajar en condiciones de alta demanda.

- **Robot de servicio**

  Fueron diseñados para asistir a las personas en diversas actividades fuera del ámbito industrial. Pueden ser utilizados en aplicaciones como investigación, educación, y asistencia en taras específicas.

Existen varias clasificaciones de robots que son mucho más específicas y que abordan mucha más información referente a la robótica. En este proyecto, el resto del capítulo se enfocará en los robots manipuladores.

1.4.3 Robots manipuladores

Existen varias definiciones para describir lo que es un robot manipulador; sin embargo, la que proporciona una mejor descripción lo define como un manipulador multifuncional reprogramable diseñado para mover materiales, partes, herramientas o dispositivos especializados mediante movimientos programados; esto le permite la ejecución de una variedad de tareas [10].

La Figura. 1.2 muestra una representación esquemática de un robot manipulador antropomórfico. Se puede observar claramente cómo su estructura imita la anatomía humana, con partes como el hombro, el brazo, el codo, el antebrazo, la muñeca y el efector final que asemeja a la mano [10].

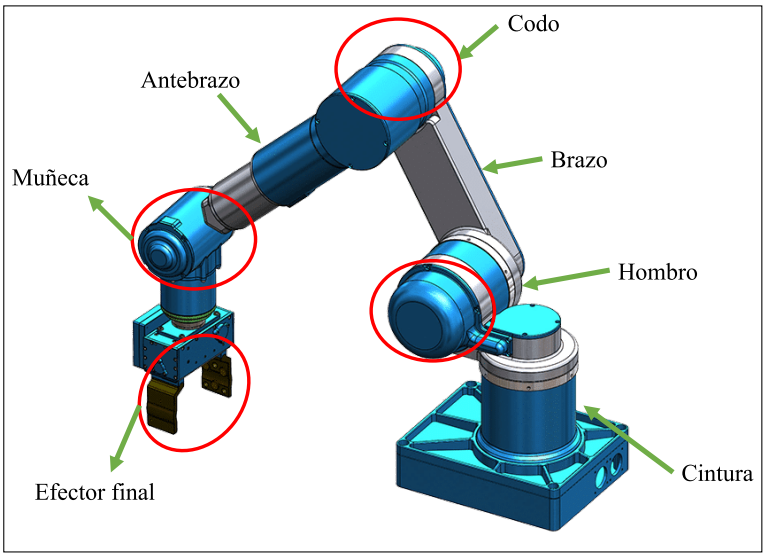


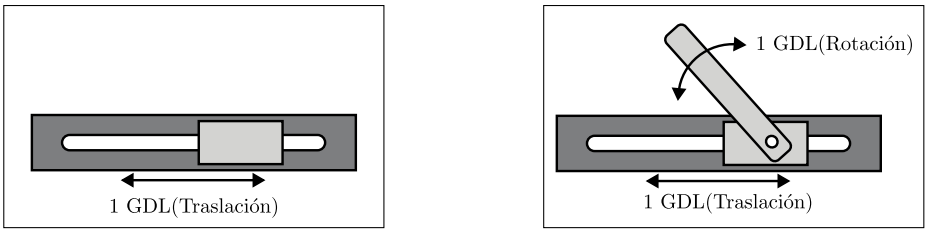
Figura 1.2: Partes principales de un brazo robótico antropomórfico. Fuente: Elaboración propia

1.4.4 Grados de Libertad

Los grados de libertad (GDL) de un robot son una medida fundamental que indica el número de movimientos independientes que se pueden realizar. Esta característica es crucial para determinar la versatilidad y la capacidad de un robot para interactuar con su entorno [11].

La cantidad de grados de libertad en un robot no solo afecta la capacidad de movimiento, sino también su aplicación en diferentes campos. Por ejemplo, los robots con más GDL son preferidos en aplicaciones industriales donde se requiere precisión y flexibilidad [12].

En la Figura.1.3 muestra dos ejemplos sencillos para entender qué son los grados de libertad en una estructura mecánica. En el primer caso (a), la estructura puede moverse solo hacia adelante y hacia atrás, lo que se llama un grado de libertad por traslación. En el segundo caso (b), además de moverse hacia adelante y hacia atrás, una parte de la estructura también puede girar, por lo que tiene dos grados de libertad: uno por el movimiento lineal y otro por el movimiento de rotación. Esto explica que, mientras más grados de libertad tenga una estructura, más formas tiene de moverse.



(a) Estructura de 1GDL. (b) Estructura de 2GDL.

Figura 1.3: Grados de libertad GDL. Fuente: Elaboración propia

El número de grados de libertad de una cadena cinemática puede ser obtenido mediante la fórmula de Grübler, según la cual:

$$NGDL = \lambda \cdot (n - j - 1) + \sum_{i=1}^{j} f_i \tag{1.1}$$

Donde:

- λ: GDL del espacio de trabajo (Típicamente tres en el plano, seis en el espacio).
- n: Número de eslabones (debe incluirse el eslabón fijo o base).
- j: Número de articulaciones.
- f_i: Grados de libertad permitidos a la articulación i.

El conjunto de eslabones y articulaciones se denomina cadena cinemática, se dice que cada cadena cinemática es abierta si cada eslabón se conecta mediante articulaciones exclusivamente al anterior y al siguiente, exceptuado el primero, que suele fijar a un soporte, y al último, cuyo extremo final queda libre.

1.4.5 Articulaciones de un brazo robótico

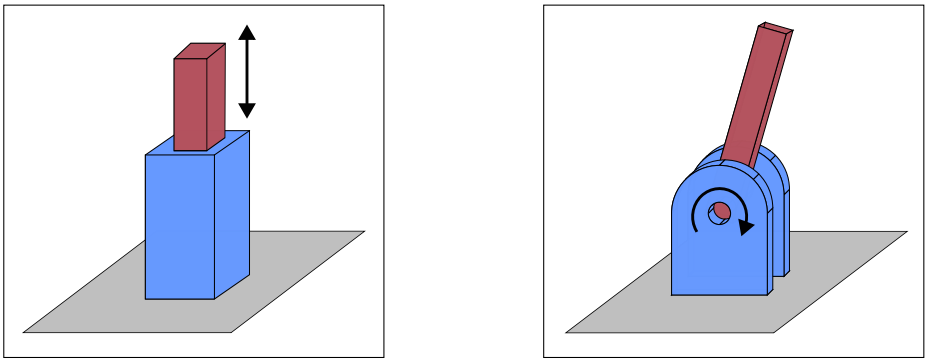
Las articulaciones son conexiones móviles entre dos eslabones sucesivos que otorgan al robot manipulador restricciones cinemáticas que determinan cómo pueden moverse en su espacio de trabajo [13] . Estas articulaciones se presentan en muchas formas, en la Tab.1.1 se muestran a varias con el respectivo GDL.

Articulación	GDL translacionales	GDL rotacionales	Total GDL
de 6-GDL	3	3	6
con Cojinetes	1	3	4
de Buje	3	3	6
Cartesiana	3	0	3
Homocinética	0	2	2
Cilíndrica	1	1	2
Gimbal	0	3	3
Tornillo de avance	1(traslacional-rotacional acoplada)		1
Pasador y ranura	1	1	2
Planar	2	1	3
Prismática	1	0	1
Rectangular	2	0	2
Rotacional	0	1	1
Esférica	0	3	3
Telescópica	1	3	4
Universal	0	2	2
Soldada	0	0	0

Tabla 1.1: Resumen de los GDL que ofrecen los distintos tipos de articulación

En esta investigación se consideran únicamente dos tipos de articulaciones básicas: la articulación rotacional, que imita el movimiento de las extremidades humanas, como brazos y piernas; y la articulación prismática, que suele encontrarse en mecanismos como pinzas, garras u otras herramientas similares.

La Figura. 1.4 muestra dos tipos de articulaciones comunes que se utilizan en los robots manipuladores. La figura (a) representa una articulación prismática, la cual permite que una parte del robot se desplace hacia arriba o hacia abajo en línea recta, como si se deslizara. La figura (b) muestra una articulación rotacional, que permite que una parte del robot gire alrededor de un punto fijo, similar al movimiento de una bisagra. Estas articulaciones son fundamentales para que los robots puedan moverse y realizar tareas con precisión.



(a) Articulación prismática.

(b) Articulación rotacional.

Figura 1.4: Articulación más comunes en robots manipuladores. Fuente: Elaboración propia

Las articulaciones en robótica suelen estar equipadas con actuadores y sensores para controlar el movimiento con precisión y mejorar su posición u orientación. [14] .

1.4.6 Configuraciones típicas de brazos robóticos

La estructura típica de un manipulador consiste en un brazo compuesto por elementos con articulaciones entre ellos. En el último enlace se coloca un efector final; en el siguiente apartado se considera la clasificación con base en las estructuras más utilizadas en brazos robóticos.

El espacio de trabajo es el conjunto de puntos en los que se puede situar el efector final del manipulador. Corresponde al volumen encerrado por superficies que determinan estos puntos a los que puede acceder el manipulador.

Configuración cartesiana

Esta tiene tres articulaciones prismáticas (3GDL estructura PPP), lo que permite que el efector final se desplace a lo largo de los ejes (X, Y, Z) esta disposición es particularmente útil en aplicaciones donde se requiere alta precisión en la ubicación de un punto en específico [15].

Sin embargo, a pesar de su facilidad, dicha configuración presenta limitaciones significativas; no es adecuada para acceder a puntos situados en espacios cerrados, lo que restringe su volumen de trabajo en comparación con otras configuraciones más flexibles. Esta configuración cartesiana es utilizada en entornos industriales para tareas de ensamblaje, soldadura y manipulación de materiales donde la precisión es necesaria [15].

La Figura. 1.5 muestra un robot cartesiano que se mueve en línea recta sobre tres ejes (X, Y y Z) para manipular objetos con una pinza. Su área de trabajo está representada por un recuadro transparente, e incluye un objeto cercano con el que puede interactuar.

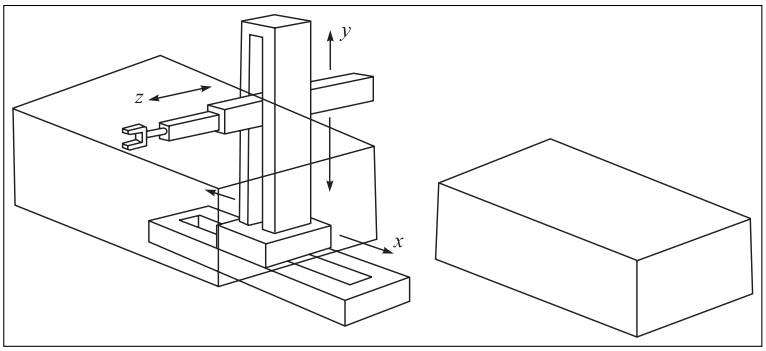


Figura 1.5: Robot manipulador cartesiano y su espacio de trabajo. Fuente: [16]

Configuración cilíndrica

Esta incluye dos articulaciones prismáticas y una rotativa (2 GDL, estructura RPP). Esta disposición permite que el manipulador pueda moverse dentro de un espacio cilíndrico, facilitando operaciones que se encuentren alrededor de un eje central. En este caso, la posición se especifica en términos de radio y altura, lo que resulta natural para muchas aplicaciones industriales.

El volumen de trabajo en esta estructura RPP (o PRP) se asemeja a un toro con sección cuadrada, proporcionando un área considerable para operar en comparación con la configuración cartesiana. Esta configuración es ideal para celdas de fabricación flexibles donde el brazo robótico esté situado en el centro y pueda servir a diversas máquinas que se encuentren a su alrededor.

La Figura.1.6 se muestra un robot manipulador cilíndrico que puede girar en su base, extender su brazo hacia adelante y moverlo hacia arriba o abajo. Al final del brazo, tiene una pinza para agarrar objetos. El área donde el robot puede trabajar está en forma de cilindro, y se indica con un espacio transparente alrededor. A un lado, hay un objeto cilíndrico con una ranura, que parece ser una pieza con la que el robot interactúa.

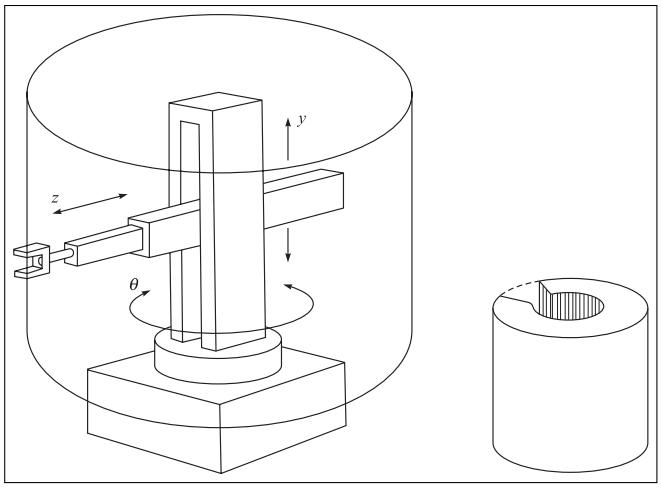


Figura 1.6: Robot manipulador cilíndrico y su espacio de trabajo. Fuente: [16]

Configuración polar o esférica

Esta configuración, de tipo polar o esférica, se compone de dos articulaciones rotacionales y una prismática (3 GDL, configuración RRP); esta disposición permite que el efecto final se mueva en un espacio tridimensional definido por coordenadas polares. En este caso, las variables articulares expresan la posición del extremo del tercer enlace, haciendo referencia a los ángulos y la distancia radial.

El volumen de trabajo abarca el espacio entre dos esferas concéntricas, lo que proporciona una gran flexibilidad para alcanzar diferentes posiciones. Esta configuración es utilizada en aplicaciones donde se requiere un amplio rango de movimiento, como brazos robóticos configurados para tareas como pintura o soldadura, así como en manipuladores diseñados para operaciones complejas en entornos tridimensionales [15].

En la Figura.1.7 se puede observar que el manipulador puede girar, levantar/bajar y estirarse, lo que le permite moverse dentro de un área esférica. Esto permite que los movimientos, puedan alcanzar diferentes posiciones y realizar tareas como agarrar objetos o manipular piezas dentro de ese espacio. Su diseño le da flexibilidad para trabajar en áreas amplias y realizar tareas precisas en 3D.

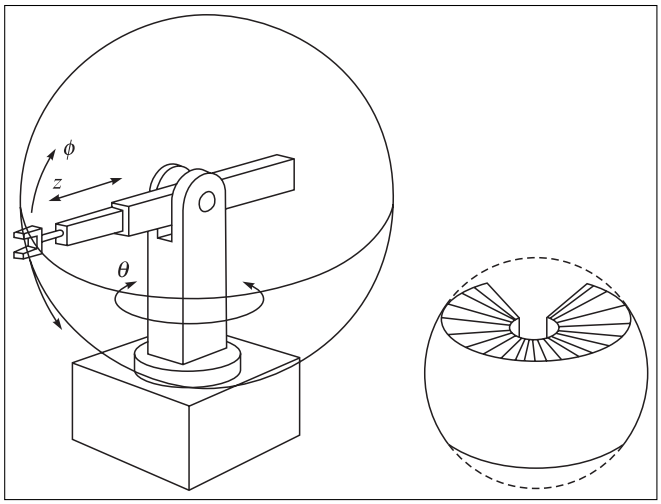


Figura 1.7: Robot manipulador polar y su espacio de trabajo Fuente: [16].

Configuración antropomórfica

En la Figura. 1.8 se muestra la configuración antropomórfica, también conocida como manipulador de codo o configuración angular vertical, se caracteriza por la similitud en los movimientos de un brazo humano. Esta estructura ofrece una gran accesibilidad y maniobrabilidad, lo que lo convierte en uno de los modelos más versátiles en el ámbito de la robótica. Sin embargo, esta configuración presenta ciertos inconvenientes: cuando opera con cargas pesadas y altas velocidades, puede generar inercias de giro que afectan la precisión del posicionamiento [15].

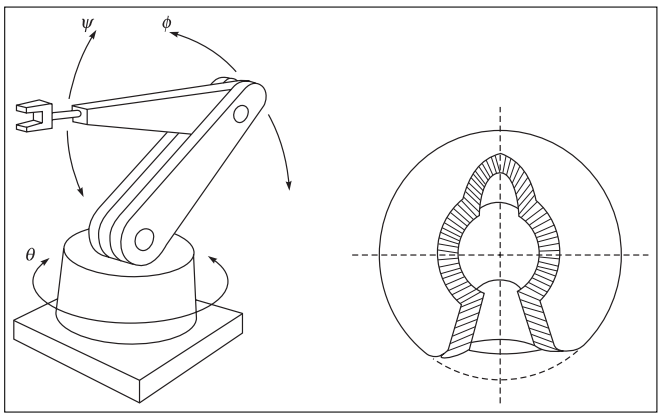


Figura 1.8: Robot manipulador antropomórfico y su espacio de trabajo. Fuente: [16]

1.4.7 Componentes de un manipulador

La Figura 1.9 muestra que todos los sistemas robóticos, sin importar su tipo o aplicación, están formados por varios componentes organizados en diferentes subsistemas. Estos subsistemas trabajan de forma conjunta para que el robot pueda funcionar correctamente. Los más importantes son: el sistema mecánico, que incluye la estructura física y los enlaces del robot; el sistema de actuación

o accionamiento, encargado de generar el movimiento mediante motores o actuadores; el sistema de sensado o sensorial, que proporciona información del entorno o del propio robot mediante sensores; y el sistema de control, que procesa esa información y envía las órdenes necesarias a los actuadores para ejecutar las tareas deseadas.

Además, en algunos casos, el robot puede incorporar elementos terminales, como pinzas, herramientas de soldadura o dispositivos de corte, que le permiten realizar funciones específicas dependiendo de la aplicación para la que ha sido diseñado [10].

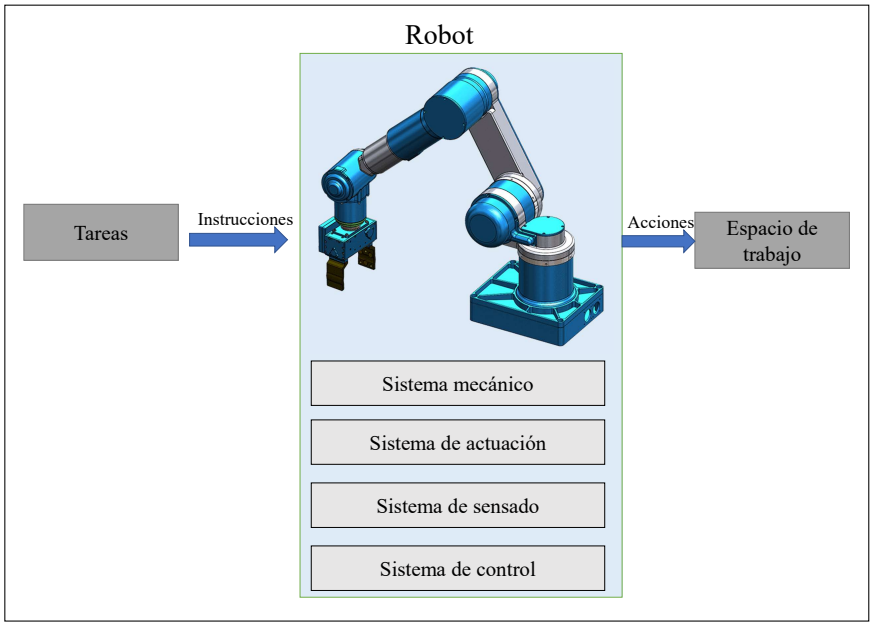


Figura 1.9: Componentes de un manipulador. Esquema adaptado de [10]

Sistema mecánico

En la Figura.1.10 se puede observar el conjunto de componentes físicos que facilitan el movimiento y la interacción del robot con su entorno, este sistema constituye elementos o eslabones unidos mediante articulaciones que forman la estructura principal del robot, permitiendo realizar muchos movimientos. El movimiento de las articulaciones puede ser lineal, de giro, o una mezcla de ambos [10].

Los elementos finales o herramientas, que son efectores finales, forman parte del sistema mecánico debido a que son un elemento que se coloca al extremo del último enlace del manipulador y suman la capacidad de agarre.

Según [17] , la precisión en el diseño de estos componentes es fundamental para que el robot ejecute movimientos controlados con alta fiabilidad; la resis-

tencia de la estructura es especialmente importante en diversas aplicaciones, donde se requieren robots que operen durante largos periodos sin fallo. Las integraciones de estos componentes deben ser cuidadosamente planificadas, ya que cualquier fallo en el sistema mecánico puede comprometer la efectividad del robot.

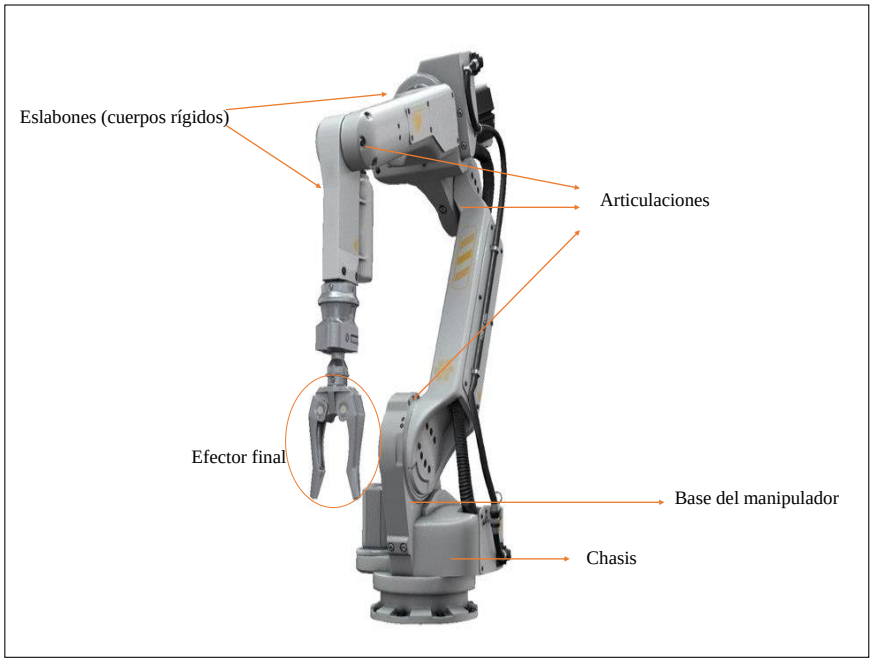


Figura 1.10: Sistema mecánico de un brazo robótico. Fuente: Elaboración propia

Sistema de actuación

Los actuadores son los que generan la fuerza para animar la estructura mecánica acorde a los mandos dispuestos por el sistema de control. Son los responsables de la calidad de movimientos del robot, ya sean suaves o bruscos, dependen directamente de la naturaleza del tipo de actuadores utilizados. [10].

En la Figura.1.11 se puede observar cómo funciona un sistema de actuación en robótica. Para este estudio, que se enfoca en robots manipuladores, se encuentran compuestos por motores eléctricos y se utilizan fuentes de alimentación AC o DC. El proceso comienza con la fuente de poder, que suministra la energía eléctrica necesaria. Dicha energía pasa a ser ajustada a un amplificador de potencia, que envía la señal al actuador, donde este se encarga de convertir el movimiento [10].

Cuando el movimiento pasa a través de la transmisión, que utiliza componentes mecánicos como engranajes y correas, el movimiento llega a la articulación donde realiza la acción deseada [10].

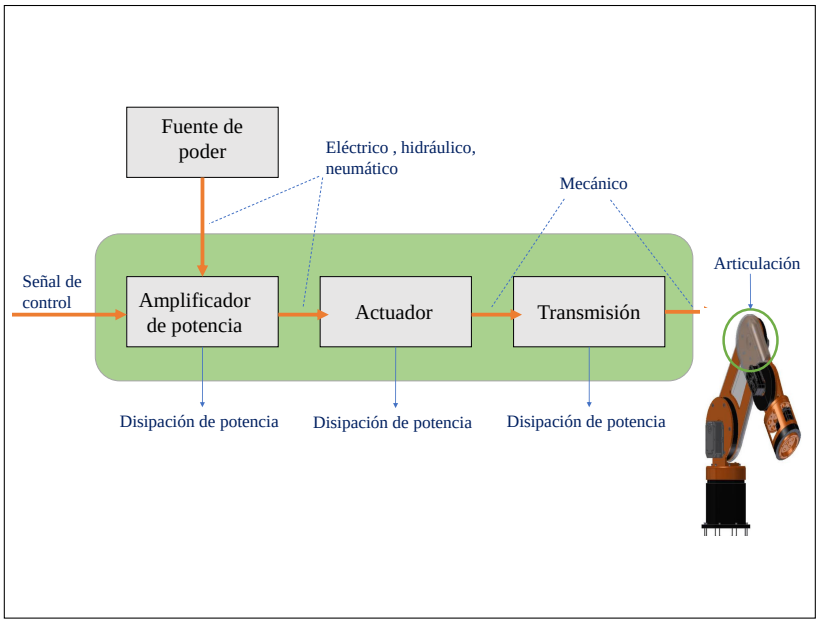


Figura 1.11: Sistema de actuación. Esquema adaptado de [10]

Sistema de sensado

En la Figura.1.12 se muestra cómo el sistema permite que el robot obtenga la información de su entorno, como de su propio estado; esta información llega a partir de distintas señales obtenidas por los sensores. El objetivo de este sistema es que el robot pueda determinar las diferentes acciones a ejecutar sobre los actuadores con una alta precisión y exactitud. El sensor es aquel dispositivo eléctrico, mecánico y/o químico que convierte un fenómeno físico en una medida cuantitativa [10].

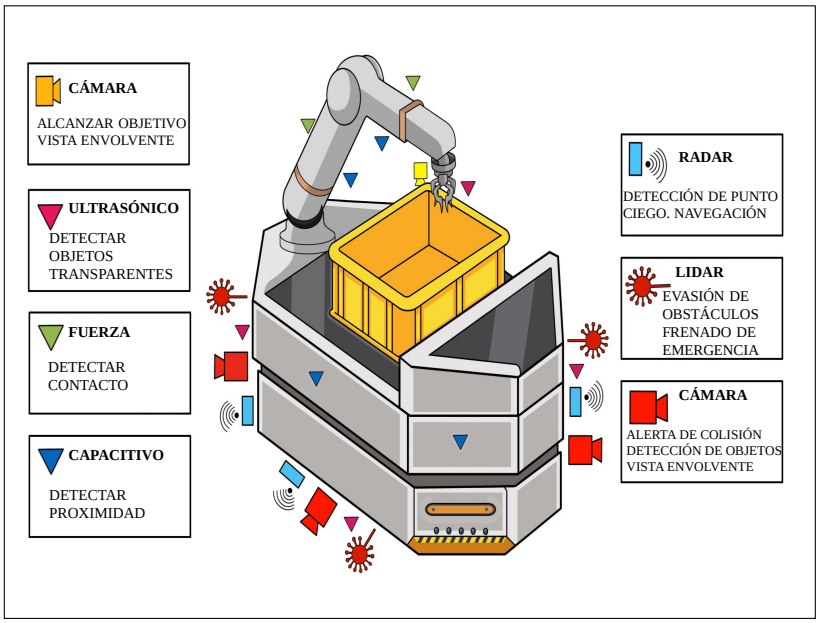


Figura 1.12: Localización de sensores. Esquema adaptado de [10]

Sistema de control

Los dispositivos electrónicos computacionales son los encargados de realizar los cálculos para que el robot desarrolle tareas específicas para las que se diseñó. El controlador suele recibir la información de los programas base que permiten mejorar el desempeño, recibe señales del sistema sensorial, las interpreta y luego decide qué acciones se realizarán [10].

1.4.8 Cinemática del brazo robótico

La Figura. 1.13 representa el estudio de la cinemática del brazo robótico, la cual analiza su movimiento en relación con un sistema de referencia sin considerar las fuerzas involucradas. La cinemática describe de manera analítica el desplazamiento espacial del brazo robótico como una función del tiempo, estableciendo la relación entre la posición y la orientación del efector final [18]. En esta sección se analizarán los conceptos fundamentales de la cinemática directa e inversa.

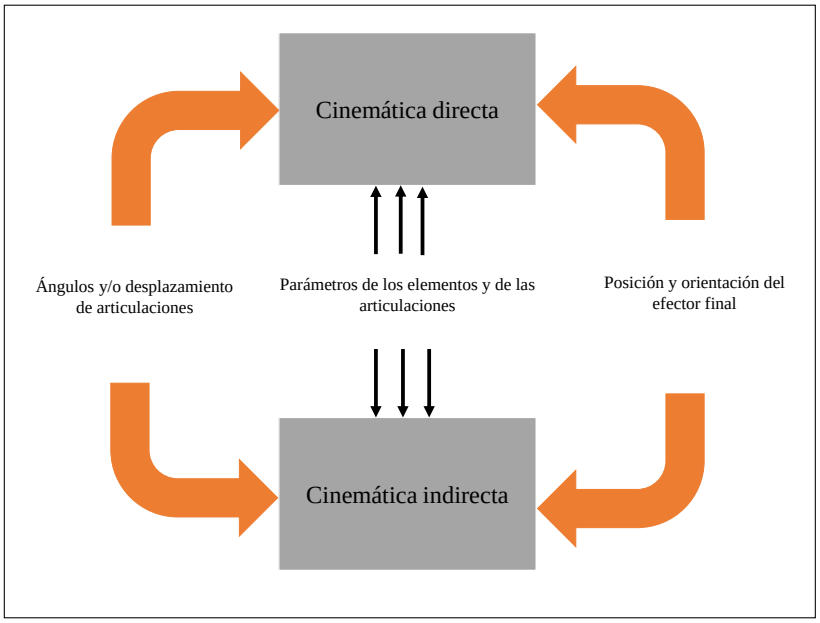


Figura 1.13: Relación de cinemática directa e inversa en robots manipuladores. Fuente: Elaboración propia.

1.4.8.1 Cinemática directa

La resolución de la cinemática directa es un proceso fundamental que permite determinar con precisión la posición y orientación del efector final. Este cálculo se basa en las variables articulares, que son los ángulos de las articulaciones del robot. Los sensores que se encuentran en cada articulación miden los ángulos y envían los datos a la unidad de control del sistema, que los procesa para calcular la ubicación exacta del efector final en un espacio tridimensional.

El cálculo de la cinemática directa se realiza mediante la aplicación de transformaciones matemáticas, como las matrices de Denavit- Hartenberg, que permiten relacionar cada eslabón del robot con el sistema de coordenadas global. Cuando se lo resuelve, se obtiene una descripción precisa de cómo los movimientos de las articulaciones afectan la posición del efector final [19].

La importancia de la cinemática directa es conocer la relación entre las variables articuladas y la posición del efector, es posible de planificar trayectorias y optimizar los movimientos del robot.

1.4.8.2 Cinemática inversa

La cinemática inversa tiene como objetivo determinar las coordenadas de las articulaciones de un robot, para que su efector final alcance una posición específica en el espacio, definida previamente por el usuario. Este proceso permite determinar valores precisos en las articulaciones [20].

Para resolver la cinemática inversa, existen diversos métodos que pueden implementarse utilizando métodos de programación. Sin embargo, muchos de estos procedimientos son iterativos y tienden a ser bastante complejos, esto puede generar retrasos en el procesamiento de datos, especialmente en sistemas con múltiples grados de libertad, donde la cantidad de cálculos aumenta de manera considerable [20] .

A diferencia de la cinemática directa, donde se obtiene una sola solución para el efector final dada una configuración específica de las articulaciones, para el problema de la cinemática inversa, puede haber múltiples conjuntos de coordenadas articulares que logran posicionar y orientar el efector final del robot, la variedad de soluciones puede ser ventajosa, ya que permite flexibilidad en el movimiento [20].

1.4.9 Método de Denavit Hartenberg

Es un método matricial que permite determinar con precisión la ubicación y orientación de los sistemas de coordenadas relacionados con cada eslabón de una cadena articulada, este método ofrece un modelado que permite analizar los movimientos autónomos de manera eficiente. Al establecer relaciones geométricas definidas entre los eslabones, esto nos permite tener una representación de las transformaciones necesarias para poder comprender y controlar el comportamiento del sistema [21].

La Figura.1.14 muestra la representación gráfica de la cinemática de un

manipulador robótico utilizando la convención de Denavit-Hartenberg, es un modelo estándar en el modelo de sistema de cadena cinemática abierta, en la imagen también se pueden identificar los eslabones y articulaciones del sistema, junto con los sistemas de referencia asignados a cada eslabón, los parámetros geométricos como los desplazamientos y ángulos de las articulaciones, permiten que, al definir dichas transformaciones homogéneas necesarias para describir con precisión la posición y la orientación del manipulador en el espacio [21].

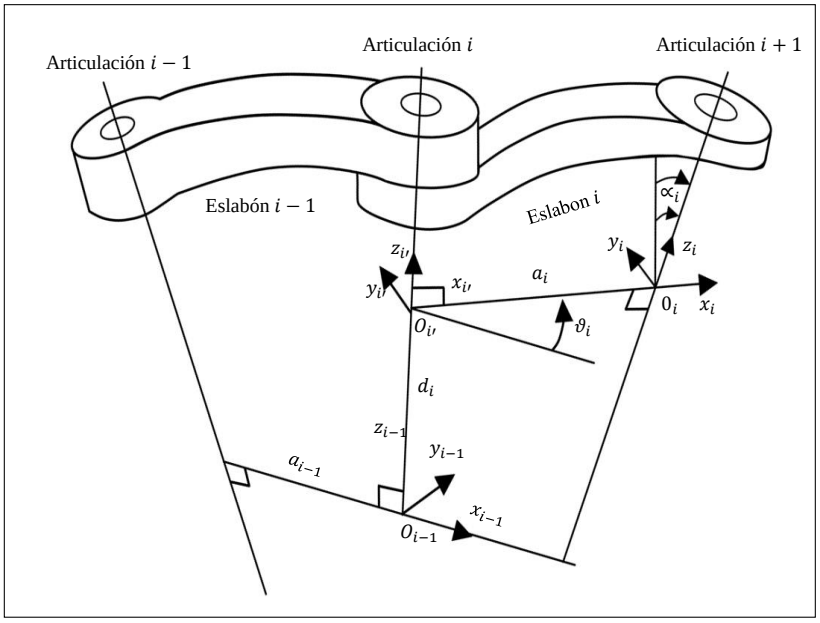


Figura 1.14: Representación cinemática de un manipulador robótico, mediante el modelo de Denavit Hartenberg Fuente: [22]

En el análisis de robots, es importante definir un sistema de referencia para cada cuerpo que forma la cadena cinemática. Un sistema de referencia es un conjunto de ejes coordenados que se utilizan para describir la posición y orientación de un cuerpo en el espacio [21].

El convenio de Denavit-Hartenberg es una metodología clásica para definir sistemas de referencia para cuerpos robóticos. Esta metodología utiliza cuatro parámetros para definir la posición y orientación de un sistema de referencia respecto del anterior[21], [23]:

- θ: el ángulo entre los ejes z de los dos sistemas de referencia.
- d: la distancia entre los ejes z de los dos sistemas de referencia.
- α: el ángulo entre los ejes x de los dos sistemas de referencia.
- a: la distancia entre los ejes x de los dos sistemas de referencia.

La utilización de cuatro parámetros en lugar de seis permite simplificar los cálculos necesarios para analizar la cinemática de un robot [21]. La secuencia de estas transformaciones es la siguiente Figura.1.15.

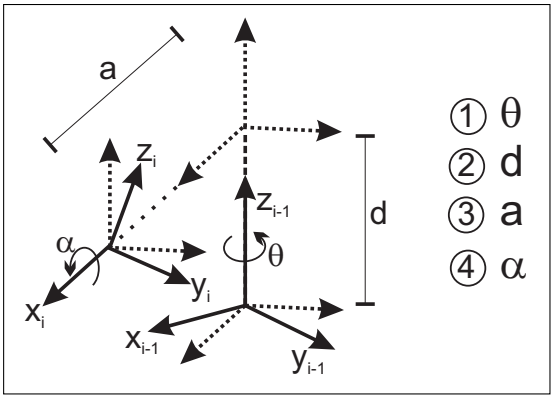


Figura 1.15: Transformaciones básicas de Denavit-Hartenberg Fuente: [21]

1. Rotación respecto a z_{i-1} de magnitud θ_i.
2. Traslación respecto a z_{i-1} de magnitud d_i.
3. Rotación respecto a x_{i-1} de magnitud α_i.
4. Traslación respecto a x_{i-1} de magnitud a_i.

$$ {}^{i-1}_{i}T(\theta, d, \alpha, a) = Rot\,(\theta_i, z_{i-1}) \cdot Trans\,(d_i, z_{i-1}) \cdot Rot\,(\alpha_i, x_{i-1}) \cdot Trans\,(a_i, x_{i-1}) \tag{1.2} $$

$$ {}^{i-1}_{i}T(\theta, d, \alpha, a) = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.3} $$

$$ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.4} $$

$$ {}^{i-1}_{i}T(\theta, d, \alpha, a) = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & 0 \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \sin\theta_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

A continuación, se detallarán los pasos para resolver el método de Denavit-Hartenberg.

1. Enumerar los eslabones en orden creciente, siendo 0 la base fija del robot, y n el último eslabón.
2. Enumerar las articulaciones de forma creciente, siendo 1 el primer GDL y n el último.
3. Localizar el eje de cada articulación. Si la articulación es rotativa, el eje será el eje de giro; si es prismática, será aquel en el que se produce el desplazamiento.
4. Para $i = 0$ a $n - 1$, situar el eje z_i sobre el eje de la articulación $i + 1$.
5. Situar el origen del sistema de la base $\{S_0\}$ en "cualquier punto" del eje z_0, mientras que los ejes x_0 e y_0 se ubicarán creando una terna ortogonal directa.
6. Para $i = 1$ a $n - 1$, situar el origen de la base $\{S_i\}$ en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i. En particular:
   - Si los ejes se cortan, el origen se ubica en el punto de corte.
   - Si los ejes son paralelos, el origen se sitúa en la articulación $i + 1$.
7. Situar x_i en la línea normal común a z_{i-1} y z_i.
8. Situar y_i de modo que forme un sistema ortogonal directo con x_i y z_i.
9. Situar el sistema $\{S_n\}$ en el extremo del robot, de modo que z_n coincida con la dirección de z_{n-1} y x_n sea normal a z_{n-1} y z_n.
10. Determinar los ángulos θ_i. Se define como el ángulo entre el eje x_{i-1} y x_i, girando alrededor del eje z_i. Este ángulo describe la rotación entre los sistemas de coordenadas de dos eslabones consecutivos.
11. Determinar las distancias d_i, que es la medida a lo largo de z_i desde el sistema de coordenadas S_{i-1} hasta la intersección de las normales comunes entre los ejes.

1.4.10 Tipos de trayectorias

La planificación de trayectorias es un aspecto crucial en el control de robots, ya que determina como se moverán sus articulaciones para alcanzar posiciones específicas, existiendo varios tipos de trayectoria que se pueden clasificar según diferentes criterios [24].

1.4.10.1 Trayectorias punto a punto

Las trayectorias punto a punto se caracterizan por el movimiento de cada articulación desde la posición inicial hasta una posición final, sin considerar el estado o la evolución de las demás articulaciones del sistema. En este enfoque, cada actuador se esfuerza por llevar su respectiva articulación al punto de destino en el menor tiempo posible. Se puede distinguir dos modalidades dentro de este tipo de trayectorias: el movimiento eje a eje y el movimiento simultáneo de ejes. En el caso del movimiento simultáneo de ejes, los ejes pueden operar de manera independiente o coordinada, lo que permite una mayor flexibilidad en las operaciones [24].

Movimiento de eje a eje

En el movimiento de eje a eje, solo un eje se desplaza a la vez. Este proceso comienza con la primera articulación que inicia su movimiento. Presenta la ventaja de un menor consumo de potencia instantánea por parte de los actuadores y la posibilidad de multiplexar una única etapa de potencia. Sin embargo, su aplicación en la práctica actual es bastante limitada debido a estas desventajas [24]. En la Figura. 1.16 se puede observar cómo es el desplazamiento de un eje a otro durante el movimiento.

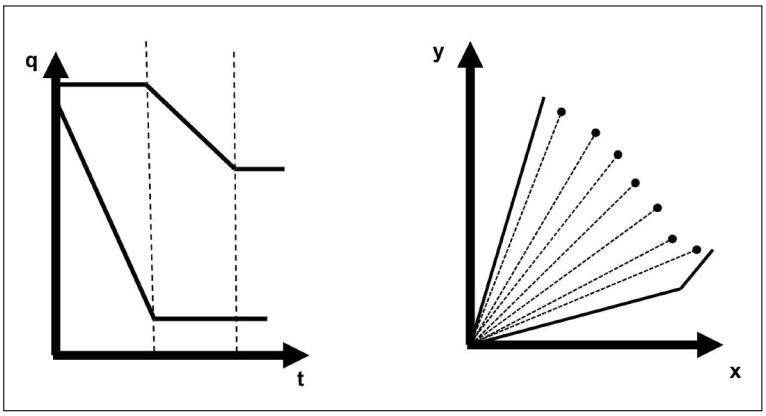


Figura 1.16: Representación de movimiento de eje a eje. Fuente:[22]

Movimiento simultáneo de ejes

Los ejes mueven las articulaciones del robot a la vez, todos los actuadores a una velocidad adaptada a sus características. Dado que las distancias que deben recorrer y las velocidades pueden variar significativamente entre los ejes, cada articulación finalizará su movimiento en momentos diferentes. Este enfoque permite una mayor eficiencia y rapidez en la ejecución de tareas complejas, optimizando así el rendimiento general del sistema robótico [24]. En la Figura. 1.17 se observa la sincronización del movimiento entre los ejes durante

la trayectoria.

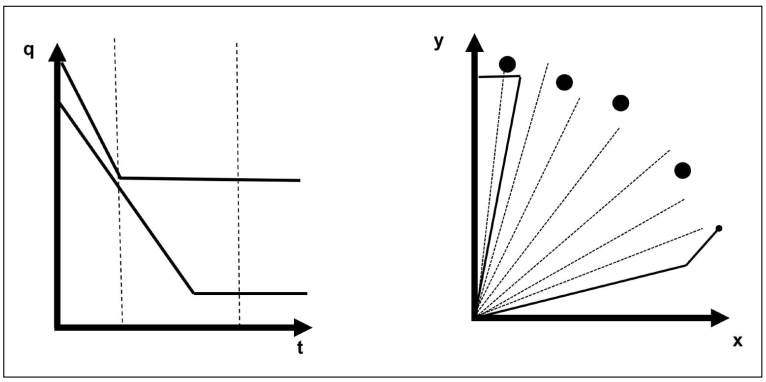


Figura 1.17: Movimiento simultáneo de ejes. Fuente:[22]

1.4.10.2 Trayectorias coordinadas o isócronas

Para evitar que ciertos actuadores operen bajo condiciones de esfuerzo debido a altas velocidades y aceleraciones, lo que podría generar tiempos de espera hasta que la articulación más lenta complete su movimiento, se puede realizar un cálculo previo para identificar dicha articulación y determinar el tiempo que requiere. A partir de esta información, se ajusta la velocidad del resto de los ejes de modo que todos finalicen su desplazamiento de manera sincronizada. Esto garantiza una óptima coordinación en que cada articulación inicia y concluye su movimiento al mismo tiempo, adaptándose a la más lenta [22].

De este modo, el tiempo total empleado en el desplazamiento es el mínimo posible, evitando exigir a los actuadores velocidades y aceleraciones innecesarias desde la perspectiva del usuario, la trayectoria seguida por el extremo del robot no es fácilmente predecible ni relevante, aunque, con un conocimiento adecuado del modelo y del control cinemático, es posible determinarla [27]. En la Fig. 1.18 se observa la ejecución de trayectorias coordinadas e isócronas, donde los ejes se desplazan de manera sincrónica para mantener un movimiento uniforme en la trayectoria establecida.

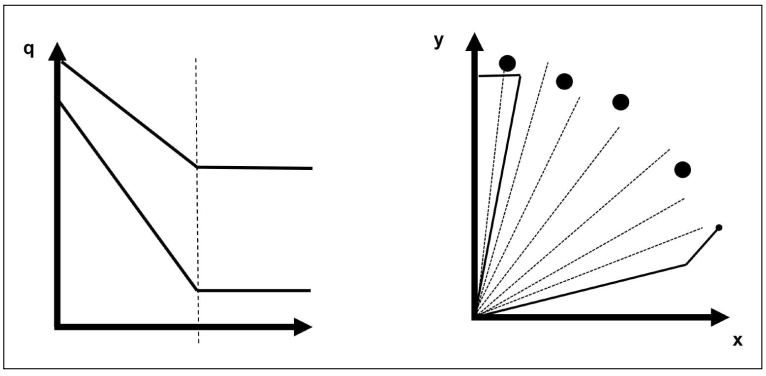


Figura 1.18: Trayectorias coordinadas o isócronas. Fuente:[22]

1.4.10.3 Trayectorias continuas

Cuando se pretende que la trayectoria que sigue el extremo del robot sea conocida por el usuario (trayectoria en el espacio cartesiano o de la tarea), es preciso calcular y controlar de manera continua las trayectorias articulares, que, por lo general, presentarán una compleja evolución [22]. La Figura. 1.19 nos muestra la ejecución de trayectorias continuas, donde los ejes se desplazan de manera fluida.

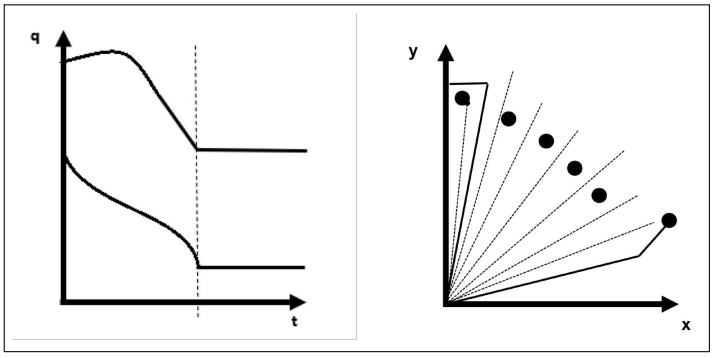


Figura 1.19: Trayectorias coordinadas o isócronas. Fuente:[22]

1.4.11 Generación de trayectoria

En la generación de trayectorias para un manipulador robótico de cinco grados de libertad, es fundamental definir los puntos clave que el efector final debe alcanzar. Estos puntos incluyen no solo la posición deseada en el espacio, sino también la velocidad con la que debe moverse, el tiempo en que debe pasar por cada punto y las restricciones físicas del sistema. El objetivo principal es lograr un desplazamiento suave y preciso entre dos posiciones. La trayectoria más comúnmente utilizada es la línea recta en el espacio cartesiano, ya que su cálculo es sencillo y permite una planificación clara y directa del movimiento. Sin embargo, en ciertas aplicaciones específicas, como el ensamblaje de componentes o tareas de manipulación que requieren una orientación constante, también se emplean trayectorias circulares o curvas más complejas.

Para mejorar el movimiento del brazo robótico, se aplican algoritmos que se ajusten de manera dinámica a la distribución de los puntos que forman la trayectoria. Por ejemplo, se pueden concentrar más puntos en las zonas donde se necesita mayor precisión, como al acercarse a un objetivo o al realizar una operación delicada, y usar menos puntos en tramos donde el movimiento puede ser más rápido y el error aceptable. Este enfoque permite una planificación eficiente del movimiento, manteniendo la precisión en las zonas críticas sin sobrecargar el sistema de control con cálculos innecesarios.

Se opta por una distribución uniforme de puntos a lo largo de la trayectoria, siempre que estén lo suficientemente cercanos entre sí para tener una interpolación suave. Esta solución es fácil de implementar y asegura que las trayectorias en el espacio articular puedan ser generadas y ejecutadas en tiempo real. De esta forma, se mantiene un buen equilibrio entre precisión, velocidad de cálculo y estabilidad del sistema de control. Una planificación adecuada de la trayectoria es clave para asegurar el desempeño eficiente y seguro del brazo robótico durante su operación.

Una vez definida la trayectoria en el espacio cartesiano, se convierte al espacio articular para ser ejecutada por los motores del manipulador. Esta conversión se realiza mediante el proceso de cinemática inversa, el cual permite determinar los ángulos articulares necesarios para que el efector final alcance las posiciones deseadas en cada instante de tiempo. Se obtienen trayectorias para cada articulación, que deben ser suaves, continuas y compatibles con las limitaciones físicas del sistema, como el rango de movimiento o la velocidad máxima de los actuadores.

En este contexto, es importante considerar tanto la precisión de la trayectoria como la viabilidad práctica de su ejecución. Algunas soluciones de cinemática inversa pueden ser matemáticamente válidas, pero no factibles desde el punto de vista mecánico o energético. Por ello, se suelen aplicar filtros o restricciones, la continuidad en el espacio articular sirve para evitar movimientos bruscos que puedan dañar el sistema.

Las trayectorias articulares obtenidas se envían al sistema de control, el cual tiene la función de seguirlas en tiempo real mediante un controlador LQG. Este controlador corrige posibles desviaciones en el movimiento causadas por perturbaciones externas o incertidumbres en el modelo dinámico. De esta forma, la planificación de trayectorias y el control se integran para lograr un funcionamiento coordinado y eficiente del brazo robótico.

1.4.12 Dinámica del brazo robótico

Cuando se busca controlar la precisión de un brazo robótico, es necesario comprender cómo se relacionan los movimientos de sus articulaciones con los torques que se aplican en cada una de ellas. En el caso de un brazo robótico de 5GDL, se requiere un modelo donde se describa cómo este responde ante diferentes condiciones. Ese modelo permite predecir el comportamiento del robot y diseñar un control más eficiente [22].

Una manera efectiva para obtener el modelo dinámico es el uso del método de Euler-Lagrange, el cual se basa en los principios de la energía y permite describir el movimiento de sistemas mecánicos articulados, de una forma más precisa y ordenada.

Este método se basa en calcular dos energías del sistema: la cinemática (relacionada con el movimiento) y la potencial (asociada con la posición y la gravedad). Con ellas se construye la función lagrangiana que se define como la diferencia entre ambas:

$$L(q, \dot{q}) = T(q, \dot{q}) - V(q) \tag{1.5}$$

Donde:

- $T(q, \dot{q})$: Energía cinética total del sistema.
- $V(q)$: Energía potencial total debida a la gravedad.

El vector de posiciones articulares del manipulador se define como:

$$q = [q_1, q_2, q_3, q_4, q_5]^T \tag{1.6}$$

Donde:

- q_i: Representa la posición angular de la articulación i.
- $\dot{q}$: Vector de velocidades articulares.
- $\ddot{q}$: Vector de aceleraciones articulares.

A partir de la función lagrangiana, se obtiene una ecuación diferencial por cada articulación. Esta expresión describe cómo evoluciona el sistema en el tiempo y se expresa como:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_i}\right) - \frac{\partial L}{\partial q_i} = \tau_i \tag{1.7}$$

Donde τ_i es el torque aplicado en la articulación i.

Al aplicar la Ecuación.1.7 a cada una de las cinco articulaciones del brazo robótico, se obtiene un conjunto de cinco ecuaciones que permiten conocer como reacciona el robot ante diferentes entradas.

Las ecuaciones que se obtienen con el método anterior se pueden agrupar en una sola expresión en forma matricial, más fácil de usar:

$$\tau = \mathbb{D}(q)\ddot{q} + \mathbb{H}(q, \dot{q}) + \mathbb{C}(q) \tag{1.8}$$

Donde:

- τ: Vector de fuerzas generalizadas (fuerzas lineales y torques) aplicadas en las articulaciones.
- $\mathbb{D}(q)$: Matriz simétrica de dimensión $n \times n$, dependiente únicamente de la posición q. Se conoce como matriz inercial, ya que representa el efecto de las masas en movimiento.
- $\mathbb{H}(q, \dot{q})$: Vector de dimensión $n \times 1$ que representa las fuerzas de Coriolis, centrífugas y centrípetas.
- $\mathbb{C}(q)$: Vector de fuerzas gravitatorias de dimensión $n \times 1$, dependiente únicamente de la configuración q.

1.4.12.1 Dinámica directa

La dinámica directa se refiere al caso en el que se conocen las fuerzas o torques que se aplican en cada articulación y se desea conocer cómo se moverá el brazo robótico. Es decir, a partir de los torques τ, y del estado actual del sistema, se calculan las aceleraciones articulares $\ddot{q}$ [22].

La ecuación se expresa de la siguiente manera:

$$\ddot{q} = \mathbb{D}^{-1}(q)\left[\tau - \mathbb{H}(q, \dot{q}) - \mathbb{C}(q)\right] \tag{1.9}$$

Este modelo es útil en simulaciones dinámicas donde se desea conocer la evolución del sistema en función de las entradas de torque.

1.4.12.2 Dinámica inversa

En cambio, la dinámica inversa se emplea cuando ya se conoce la trayectoria que debe seguir el manipulador, es decir, se conocen las posiciones q, velocidades $\dot{q}$ y aceleraciones $\ddot{q}$, y se busca calcular los torques necesarios en las articulaciones para lograr ese movimiento. Esta formulación se utiliza con frecuencia en el diseño de controladores y planificación de trayectorias [22].

La ecuación correspondiente se expresa así:

$$\tau = \mathbb{D}(q)\ddot{q} + \mathbb{H}(q, \dot{q}) + \mathbb{C}(q) \tag{1.10}$$

Esta expresión permite calcular las fuerzas requeridas para que el robot ejecute un movimiento específico de manera precisa y controlada [22].

1.4.13 Fundamentos del modelado de espacio de estados

Para el desarrollo del sistema de control basado en el algoritmo LQG (Linear Quadratic Gaussian), enfocado al seguimiento de trayectoria de un brazo robótico, es esencial comprender algunos conceptos clave del modelado en espacio de estados: estado, variables de estado, vector de estado y espacio de estados [25].

Estado del sistema

El *estado* de un sistema dinámico que se define como el conjunto mínimo de variables que contienen todo lo que es la información necesaria para comprender y a su vez describir completamente el comportamiento del sistema en un instante determinado y a predecir la evolución futura, con el conocimiento de las entradas externas [25]. En sistemas mecánicos como lo es el brazo robótico, el estado suele estar compuesto por las posiciones y velocidades angulares de sus articulaciones.

Variables de estado

Las *variables de estado* son los componentes individuales del estado del sistema. Dichas variables representan las magnitudes físicas importantes que describen la dinámica interna. En el brazo robótico de cinco grados de libertad, estas variables corresponden típicamente a los ángulos articulares $\theta_1, \theta_2, \ldots, \theta_5$ y sus derivadas temporales $\dot{\theta}_1, \dot{\theta}_2, \ldots, \dot{\theta}_5$ [26].

Vector de estado

El *vector de estado* es la forma compacta de agrupar todas las variables de estado en una representación matricial. Este se expresa como:

$$\mathbf{x}(t) = \begin{bmatrix} \theta_1(t) \\ \dot{\theta}_1(t) \\ \theta_2(t) \\ \dot{\theta}_2(t) \end{bmatrix} \tag{1.11}$$

El uso del vector de estado permite representar el sistema dinámico mediante un modelo matemático estructurado, adecuado para el diseño de con-

troladores modernos [27].

Espacio de estados

El *espacio de estados* es el conjunto de todos los posibles valores que puede tomar el vector de estado. Cada punto en este espacio representa una configuración única del sistema en un momento dado. El modelo en espacio de estados permite expresar la dinámica del sistema mediante un conjunto de ecuaciones diferenciales lineales (o no lineales), base fundamental para la implementación del controlador LQG, el cual combina control óptimo (LQR) y estimación de estados mediante el filtro de Kalman [28, 29].

1.4.14 Generación del modelado en espacio de estado

Uno de los pasos para el desarrollo de los sistemas de control para brazos robóticos es la representación matemática de su dinámica mediante el enfoque de espacios de estados, esto permite describir de forma estructurada cómo van evolucionando las variables internas del sistema, como las posiciones y las velocidades de las articulaciones, a partir de las entradas, como los torques generados por los motores [25]. Este método es adecuado para sistemas multivariables, con multivariables, con múltiples grados de libertad, como es el caso de los robots manipuladores.

Para construir un modelo en espacio de estados, se parte generalmente de las leyes físicas que describen lo que es el sistema. En el caso de un brazo robótico, esto implica identificar las ecuaciones que van a relacionar la entrada del sistema (voltaje del servomotor) con el movimiento que se produce (posición angular). Estas relaciones se obtienen utilizando principios como la segunda ley de Newton para la parte mecánica y la ley de voltaje de Kirchhoff para la parte eléctrica. Una vez planteadas las ecuaciones, se reorganizan para que sean expresadas como un sistema de ecuaciones de primer orden, lo cual es una de las condiciones para que el modelo pueda escribirse en forma de espacio de estado.

Según la teoría presentada por autores [25] y [30] el modelo básico en espacio de estados se componen de dos ecuaciones: la ecuación de estado y la ecuación de estado de salida. La ecuación de estado describe el cómo van a cambiar las variables internas del sistema, mientras que la ecuación de salida describe como esas variables internas se relacionan con las mediciones que pueden obtenerse del sistema. Las ec. se expresan:

Consideremos un sistema de múltiples entradas y múltiples salidas con n integradores, r entradas $u_1(t), u_2(t), ..., u_r(t)$, y m salidas $y_1(t), y_2(t), ..., y_m(t)$. Se define a las n salidas de los integradores como las variables de estado $x_1(t), x_2(t), ..., x_n(t)$. De este modo el sistema puede describirse mediante:

$$\begin{aligned} \dot{x}_1(t) &= f_1(x_1, x_2, ..., x_n; u_1, u_2, ..., u_r; t) \\ \dot{x}_2(t) &= f_2(x_1, x_2, ..., x_n; u_1, u_2, ..., u_r; t) \\ &\vdots \\ \dot{x}_n(t) &= f_n(x_1, x_2, ..., x_n; u_1, u_2, ..., u_r; t) \end{aligned} \tag{1.12}$$

Las salidas $y_1(t), y_2(t), ..., y_m(t)$ del sistema se pueden obtener mediante:

$$\begin{aligned} y_1(t) &= g_1(x_1, x_2, ..., x_n; u_1, u_2, ..., u_r; t) \\ y_2(t) &= g_2(x_1, x_2, ..., x_n; u_1, u_2, ..., u_r; t) \\ &\vdots \\ y_m(t) &= g_m(x_1, x_2, ..., x_n; u_1, u_2, ..., u_r; t) \end{aligned} \tag{1.13}$$

Si se define:

$$x(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}, \quad f(x,u,t) = \begin{bmatrix} f_1(x_1, ..., x_n; u_1, ..., u_r; t) \\ f_2(x_1, ..., x_n; u_1, ..., u_r; t) \\ \vdots \\ f_n(x_1, ..., x_n; u_1, ..., u_r; t) \end{bmatrix},$$

$$y(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_m(t) \end{bmatrix}, \quad g(x,u,t) = \begin{bmatrix} g_1(x_1, ..., x_n; u_1, ..., u_r; t) \\ g_2(x_1, ..., x_n; u_1, ..., u_r; t) \\ \vdots \\ g_m(x_1, ..., x_n; u_1, ..., u_r; t) \end{bmatrix}, \quad u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_r(t) \end{bmatrix} \tag{1.14}$$

Las ecuaciones Ec. (1.13), Ec. (1.14) se convierten en:

$$\dot{x}(t) = f(x,u,t) \tag{1.15}$$

$$y(t) = g(x,u,t) \tag{1.16}$$

En el caso del brazo robótico de 5 grados de libertad, la Ec.(1.15) representa el modelo de estado del sistema, mientras que la Ec.(1.16) corresponde a la ecuación de salida. Dado que las funciones vectoriales **f** y **g** pueden depender

explícitamente del tiempo t, el sistema puede clasificarse como variante en el tiempo.

Al linearizar el modelo dinámico no lineal del brazo robótico de 5 GDL alrededor de un punto de operación, se obtiene un modelo lineal que describe la evolución del sistema en forma matricial, como se muestra a continuación:

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \tag{1.17a}$$

$$y(t) = C(t)x(t) + D(t)u(t) \tag{1.17b}$$

Este modelo es fundamental para el diseño de controladores como el LQG, que requieren una descripción del sistema en el espacio de estados para poder estimar y controlar el movimiento del brazo robótico de manera eficiente.

Las ecuaciones en espacio de estados permiten describir de manera precisa el comportamiento dinámico de un sistema, como es el caso de un brazo robótico de 5 grados de libertad. A continuación, se explican los componentes de estas ecuaciones:

- $\dot{x}(t)$: Es la derivada del vector de estado respecto al tiempo. Describe cómo cambian las posiciones y velocidades del brazo con el tiempo, es decir, su evolución dinámica.
- $A(t)$: Es la matriz de dinámica del sistema. Describe cómo el estado actual del sistema influye en su propia evolución, es decir, cómo las variables del estado se afectan entre sí a lo largo del tiempo.
- $x(t)$: Representa el vector de estado del sistema. En el caso del brazo robótico de 5 GDL, este vector incluye las posiciones angulares y velocidades angulares de cada articulación, por lo que tiene una dimensión de 10 elementos.
- $B(t)$: Representa la matriz de entrada. Indica cómo las entradas del sistema (los torques) afectan el cambio de estado del sistema.
- $u(t)$: Corresponde al vector de entrada del sistema. En un brazo robótico, está compuesto por los torques que se aplican en cada una de las 5 articulaciones para generar movimiento.
- $y(t)$: Es el vector de salida del sistema. Puede representar directamente el vector de estado o alguna transformación del mismo, como por ejemplo la posición del efector final del brazo robótico.

- $C(t)$: Es la matriz de salida. Relaciona los estados internos del sistema con las variables de salida que pueden ser medidas u observadas desde el exterior.
- $D(t)$: Es la matriz de transmisión directa. Determina si existe una relación directa entre las entradas y las salidas del sistema, sin pasar por el estado. En muchos sistemas físicos, esta matriz suele ser nula.

En la Figura. 1.20 muestra cómo funciona un sistema de control lineal usando la representación en espacio de estados. En él se pueden ver las señales de entrada y salida, así como las variables internas que describen el comportamiento del sistema a lo largo del tiempo.

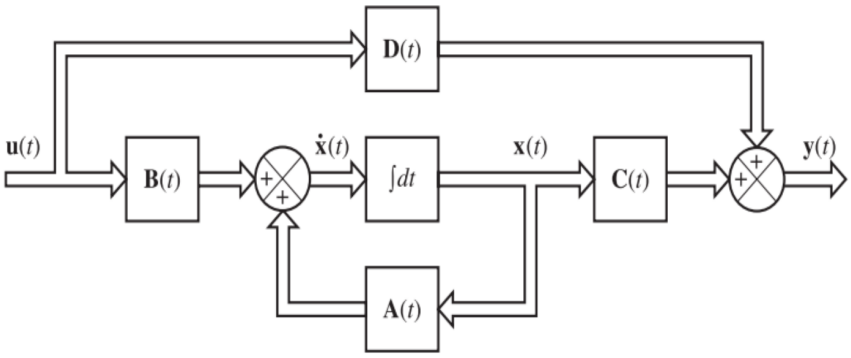


Figura 1.20: Diagrama de bloques de un sistema de control en espacio de estados en tiempo continuo. Fuente:[31]

1.4.15 Control óptimo en sistemas robóticos

El control óptimo es una técnica que busca encontrar la mejor forma de controlar un sistema para lograr un buen desempeño, minimizando al mismo tiempo el esfuerzo o la energía utilizada. En robótica, esta estrategia es muy útil porque permite que los robots realicen sus tareas con precisión, usando la menor cantidad de energía posible y funcionando bien incluso si hay pequeñas fallas o cambios en el modelo del sistema [32].

En el caso del brazo robótico con cinco grados de libertad (5GDL), se busca controlar el movimiento de cada articulación para que el efector final siga una trayectoria deseada. Sin embargo, esto debe hacerse considerando restricciones físicas, consumo energético y ruidos de medición. En este contexto, el uso de controladores óptimos como el Regulador Lineal Cuadrático (LQR) y el Filtro de Kalman, combinados en un controlador LQG (Linear Quadratic Gaussian) [32].

1.4.15.1 Regulador Lineal Cuadrático (LQR)

Antes de explicar el funcionamiento del controlador LQR, es útil observar cómo se organiza el sistema dentro de un lazo de control. La Figura 1.21 muestra un diagrama de bloques que representa la forma en que el controlador LQR actúa sobre un sistema dinámico. En este esquema, el estado del sistema se mide y se utiliza directamente para calcular la señal de control, la cual tiene como objetivo mantener el sistema estable y con buen desempeño ante perturbaciones o cambios en las condiciones de operación.

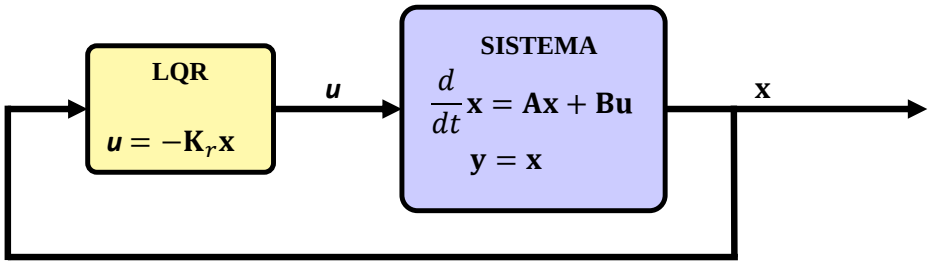


Figura 1.21: Diagrama en lazo cerrado con control óptimo LQR mediante realimentación de estado. Fuente: Elaboración propia.

El LQR es una técnica de control óptimo que se utiliza para sistemas lineales continuos representados en espacio de estados. Su principal objetivo es encontrar una matriz de ganancia K que minimice una función de costo cuadrática del tipo [33] :

$$J(t) = \int_0^t \left[\mathbf{x}^T(\tau)\,\mathbf{Q}\,\mathbf{x}(\tau) + \mathbf{u}^T(\tau)\,\mathbf{R}\,\mathbf{u}(\tau)\right] d\tau \tag{1.18}$$

En esta expresión:

- $\mathbf{x}(\tau)$ es el vector de estado del sistema, que en el caso de un brazo robótico de 5GDL incluye las posiciones y velocidades articulares.
- $\mathbf{u}(\tau)$ representa el vector de control, es decir, los torques aplicados por los actuadores en cada articulación.
- $\mathbf{Q}$ es una matriz de ponderación que penaliza los errores en el estado del sistema.
- $\mathbf{R}$ es una matriz de ponderación que penaliza el esfuerzo de control, promoviendo un uso eficiente de la energía.

El propósito de esta función de costo es encontrar un equilibrio entre mantener el sistema en la trayectoria deseada y minimizar el uso de energía o

esfuerzo excesivo por parte de los motores. Al resolver el problema de optimización asociado a esta función, se obtiene una matriz de ganancia óptima $\mathbf{K}_r$ que define la ley de control como:

$$\mathbf{u}(t) = -\mathbf{K}_r\mathbf{x}(t) \tag{1.19}$$

Esta ley de control implica que la señal de control $\mathbf{u}(t)$ se determina en función del estado actual del sistema $\mathbf{x}(t)$, mediante la multiplicación por la matriz de ganancia negativa $\mathbf{K}_x$. Esta retroalimentación de estado permite al sistema seguir trayectorias con alta precisión, mientras se mantiene la estabilidad y se optimiza el uso de energía.

En sistemas robóticos complejos, como un brazo manipulador de cinco grados de libertad (5GDL), es fundamental contar con un control que permita un movimiento preciso, estable y eficiente. El Regulador Lineal Cuadrático (LQR) es una estrategia óptima que cumple con estos objetivos, ya que calcula una ley de control que minimiza una función de costo compuesta por el error en el estado y el esfuerzo de control.

Para obtener la ley de control, primero se debe resolver una ecuación algebraica conocida como ecuación de Riccati [34], cuya solución es una matriz simétrica positiva definida, denotada por $\mathbf{X}$. Esta matriz se obtiene a partir de la siguiente expresión:

$$\mathbf{A}^T\mathbf{X} + \mathbf{X}\mathbf{A} - \mathbf{X}\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^T\mathbf{X} + \mathbf{Q} = 0 \tag{1.20}$$

Una vez que se conoce $\mathbf{X}$, se puede calcular la matriz de ganancia óptima $\mathbf{K}_x$, que se usa para definir la ley de control como:

$$\mathbf{u}(t) = -\mathbf{K}_x\mathbf{x}(t) \tag{1.21}$$

Esta retroalimentación de estado permite que el sistema siga la trayectoria deseada con precisión, manteniendo al mismo tiempo la estabilidad del manipulador y reduciendo el consumo de energía. Gracias a su formulación matemática, el LQR proporciona un control robusto que responde bien ante perturbaciones y pequeñas inexactitudes en el modelo del sistema.

1.4.15.2 Estimador de Estado: Filtro de Kalman (LQE)

Dado que no siempre es posible medir todos los estados internos de un sistema dinámico como un brazo robótico, resulta necesario contar con un estimador

que proporcione información confiable sobre dichas variables no observables. Para este propósito se emplea el filtro de Kalman, una herramienta fundamental que permite obtener estimaciones óptimas del estado del sistema a partir de las mediciones disponibles, incluso en presencia de ruido e incertidumbre [33, 35].

El filtro de Kalman se fundamenta en un modelo lineal en espacio de estados, al cual se incorporan perturbaciones aleatorias que representan el ruido tanto del proceso como de las mediciones. Este modelo se expresa mediante el siguiente conjunto de ecuaciones:

El modelo representado en las ecuaciones (1.22a) y (1.23a) describe la evolución de un sistema dinámico y la forma en que se obtienen sus mediciones, considerando la presencia de incertidumbre.

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{w}_d(t) \tag{1.22a}$$

En la Ec.(1.22a) se distinguen los siguientes componentes:

- $\frac{d}{dt}\mathbf{x}(t)$ indica la tasa de cambio del estado a lo largo del tiempo.
- $\mathbf{A}$ es la matriz que describe la dinámica interna del sistema, es decir, cómo el estado actual influye en su propia evolución.
- $\mathbf{x}(t)$ es el vector de estado, que contiene las variables que definen el estado actual del sistema, como posiciones o velocidades.
- $\mathbf{B}$ es la matriz que relaciona las entradas o comandos externos con el cambio en el estado.
- $\mathbf{u}(t)$ representa el vector de entradas o controles aplicados al sistema.
- $\mathbf{w}_d(t)$ es el ruido del proceso, que representa perturbaciones o incertidumbres que afectan la evolución del sistema.

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{w}_n(t) \tag{1.23a}$$

La Ec. (1.23a) establece la relación entre el estado del sistema y las mediciones disponibles:

- $\mathbf{y}(t)$ es el vector de mediciones u observaciones.
- $\mathbf{C}$ es la matriz que vincula el estado con las mediciones obtenidas.

- $\mathbf{w}_n(t)$ representa el ruido de medición, es decir, los errores o imprecisiones presentes en las observaciones.

Este modelo facilita el análisis y diseño de sistemas, considerando tanto la dinámica real como las limitaciones derivadas de las mediciones, lo cual resulta esencial para la aplicación de métodos de estimación y control.

Se considera que son tanto $\mathbf{w}_d(t)$ como $\mathbf{w}_n(t)$ procesos estocásticos de ruido blanco gaussiano con media cero. Sus respectivas covarianzas están definidas como sigue:

$$\mathbb{E}\left[\mathbf{w}_d(t)\mathbf{w}_d^T(\tau)\right] = \mathbf{Q}\delta(t-\tau) \tag{1.24a}$$

$$\mathbb{E}\left[\mathbf{w}_n(t)\mathbf{w}_n^T(\tau)\right] = \mathbf{R}\delta(t-\tau) \tag{1.24b}$$

Donde $\mathbf{Q}$ y $\mathbf{R}$ son matrices de covarianza positivas semidefinidas que representan la intensidad de la perturbación y del ruido de medición, respectivamente. La función delta de Dirac $\delta(t-\tau)$ representa la correlación instantánea propia del ruido blanco.

El objetivo es estimar el estado del sistema $\mathbf{x}(t)$ mediante un estimador dinámico que, a partir de las entradas conocidas $\mathbf{u}(t)$ y las salidas medidas $\mathbf{y}(t)$, genere una estimación $\hat{\mathbf{x}}(t)$ del estado real. El estimador propuesto por Kalman tiene la forma:

$$\frac{d}{dt}\hat{\mathbf{x}}(t) = \mathbf{A}\hat{\mathbf{x}}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{K}_f\left[\mathbf{y}(t) - \hat{\mathbf{y}}(t)\right] \tag{1.25a}$$

$$\hat{\mathbf{y}}(t) = \mathbf{C}\hat{\mathbf{x}}(t) \tag{1.25b}$$

La matriz $\mathbf{K}_f$ representa la ganancia del filtro de Kalman, que pondera la corrección basada en la diferencia entre la salida medida y la estimada. Esta estructura permite que el estimador se corrija continuamente frente a las desviaciones producidas por las perturbaciones o errores de medición.

Al sustituir la ecuación de salida estimada (1.25b) en (1.25a), se obtiene la forma explícita del sistema estimador:

$$\frac{d}{dt}\hat{\mathbf{x}}(t) = \left(\mathbf{A} - \mathbf{K}_f\mathbf{C}\right)\hat{\mathbf{x}}(t) + \mathbf{B}\mathbf{u}(t) + \mathbf{K}_f\mathbf{y}(t) \tag{1.26}$$

Este esquema puede visualizarse en la Figura 1.22, donde se representa gráficamente el funcionamiento del filtro de Kalman aplicado a sistemas sin término de alimentación directa $\mathbf{D}$.

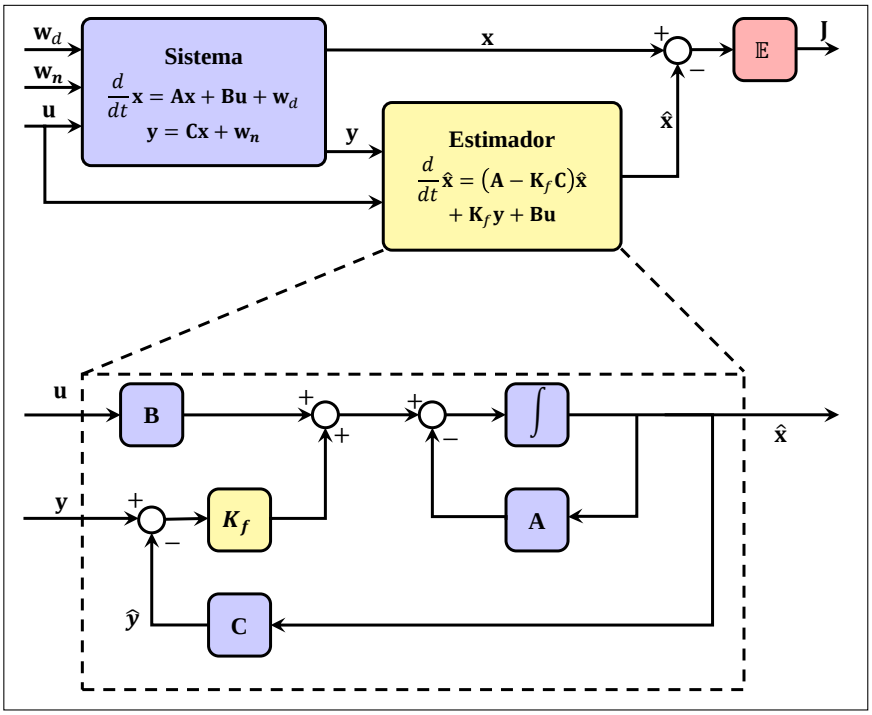


Figura 1.22: Esquema del filtro de Kalman para estimación de estado a partir de medidas ruidosas $\mathbf{s} = \mathbf{Ca} + \mathbf{w}_n$ con ruido de proceso (perturbación) $\mathbf{w}_d$. Obsérvese que en este diagrama no hay ningún término de alimentación $\mathbf{D}$. Fuente: Elaboración propia.

Para analizar la convergencia del estimador, se estudia la evolución del error de estimación $\boldsymbol{\varepsilon}(t) = \mathbf{x}(t) - \hat{\mathbf{x}}(t)$. Derivando respecto al tiempo y usando las ecuaciones del sistema y del estimador [35, 36], se tiene:

$$\frac{d}{dt}\boldsymbol{\varepsilon} = (\mathbf{A} - \mathbf{K}_f\mathbf{C})\,\boldsymbol{\varepsilon} + \mathbf{w}_d(t) - \mathbf{K}_f\mathbf{w}_n(t) \tag{1.27}$$

Esta ecuación muestra que el error de estimación converge a cero en promedio si el sistema es observable y la matriz $\mathbf{A} - \mathbf{K}_f\mathbf{C}$ tiene valores propios con parte real negativa, es decir, si es estable.

La ganancia óptima $\mathbf{K}_f$ del filtro se obtiene resolviendo un problema de optimización que minimiza el error cuadrático medio del estado estimado respecto al real, es decir:

$$J = \lim_{t\to\infty} \mathbb{E}\left[(\mathbf{x}(t) - \hat{\mathbf{x}}(t))^T\,(\mathbf{x}(t) - \hat{\mathbf{x}}(t))\right] \tag{1.28}$$

La solución a este problema, conocida como Estimación Cuadrática Lineal

(LQE), es dual al problema del LQR, y lleva a la siguiente expresión para la ganancia óptima:

$$\mathbf{K}_f = \mathbf{P}\mathbf{C}^T\mathbf{R}^{-1} \tag{1.29}$$

donde $\mathbf{P}$ es la solución de la ecuación algebraica de Riccati asociada:

$$\mathbf{A}\mathbf{P} + \mathbf{P}\mathbf{A}^T - \mathbf{P}\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\mathbf{P} + \mathbf{Q} = \mathbf{0} \tag{1.30}$$

Una vez que se implementa este estimador, es posible conocer con mayor precisión el estado completo del sistema. Con esta información, se puede diseñar un controlador LQG que utilice las estimaciones proporcionadas por el filtro de Kalman.

1.4.15.3 Controlador LQG

El controlador lineal cuadrático gaussiano (LQG) es una estrategia de control óptima que permite tomar decisiones usando la información que proporcionan sensores, incluso cuando hay ruido o incertidumbre en las mediciones y en el modelo del sistema. La idea principal es usar dos herramientas: por un lado, la ganancia de control óptima LQR $\mathbf{K}_r$, que se diseña para tomar las mejores decisiones de control, y, por otro lado, el filtro de Kalman, que estima el estado real del sistema usando las mediciones ruidosas.

Estos dos componentes se diseñan por separado: primero se resuelve una ecuación de Riccati para obtener la ganancia de control $\mathbf{K}_r$, y luego otra ecuación de Riccati para calcular la ganancia del filtro de Kalman $\mathbf{K}_f$. Una vez que se tienen ambas ganancias, se combinan para formar el controlador LQG completo. Esta forma de diseñar el controlador por partes se conoce como el principio de separación [32].

Combinando el control LQR de estado completo con la estimación del filtro Kalman de estado completo, obtenemos un sistema dinámico para el controlador LQG donde $\mathbf{s}$ es la entrada al controlador, $\mathbf{b}$ es la salida del controlador, y el estado interno del controlador es la estimación de estado completo $\hat{\mathbf{a}}$:

$$\frac{d\hat{a}}{dt} = \left(\mathbf{A} - \mathbf{K}_f\mathbf{C} - \mathbf{B}\mathbf{K}_r\right)\hat{a} + \mathbf{K}_f\mathbf{s} \tag{1.31}$$

$$\mathbf{b} = -\mathbf{K}_r\hat{a}. \tag{1.32}$$

La función de coste LQG es el valor medio del conjunto de la Ec. (1.18),

$$J(t) = \int_0^t \left[\mathbf{x}^T(\tau)\,\mathbf{Q}\,\mathbf{x}(\tau) + \mathbf{u}^T(\tau)\,\mathbf{R}\,\mathbf{u}(\tau)\right] d\tau \tag{1.33}$$

Donde los corchetes angulares indican la media de muchas realizaciones de ruido.

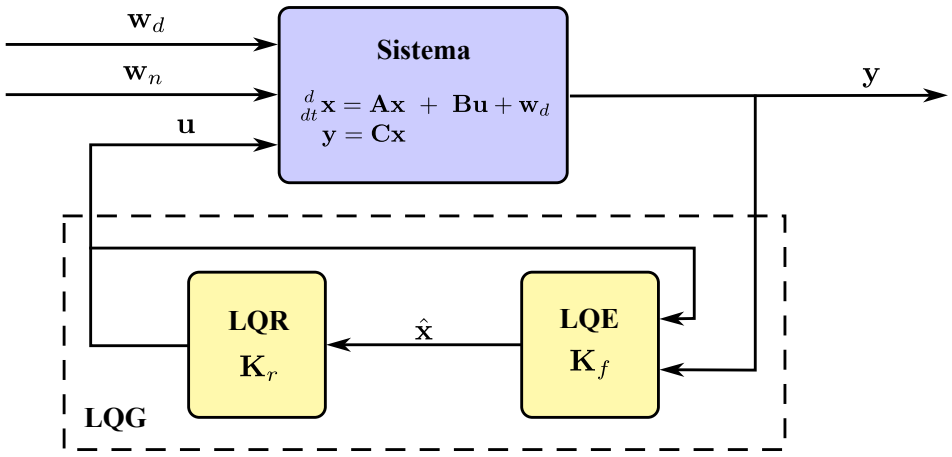


Figura 1.23: Diagrama esquemático del controlador lineal cuadrático gaussiano (LQG). Las matrices óptimas de ganancia LQR y LQE $\mathbf{K}_r$ y $\mathbf{K}_f$ se diseñan por separado basándose en las soluciones de dos ecuaciones algebraicas de Riccati diferentes. Cuando se combinan, la realimentación resultante basada en sensores es óptima. Fuente: Elaboración propia.

Aplicando LQR a la estimación de estado completo, $\hat{\mathbf{a}}$ se obtiene la siguiente dinámica de estado:

$$\begin{aligned}\frac{d}{dt}\mathbf{a} &= \mathbf{A}\mathbf{a} - \mathbf{B}\mathbf{K}_r\hat{\mathbf{a}} + \mathbf{w}_d \\ &= \mathbf{A}\mathbf{a} - \mathbf{B}\mathbf{K}_r\mathbf{a} + \mathbf{B}\mathbf{K}_r(\mathbf{a} - \hat{\mathbf{a}}) + \mathbf{w}_d \\ &= \mathbf{A}\mathbf{a} - \mathbf{B}\mathbf{K}_r\mathbf{a} + \mathbf{B}\mathbf{K}_r\varepsilon + \mathbf{w}_d\end{aligned} \tag{1.34}$$

donde $\varepsilon = \mathbf{a} - \hat{\mathbf{a}}$ como antes. Finalmente, podemos escribir el sistema de lazo cerrado como

$$\frac{d}{dt}\begin{bmatrix}\mathbf{a}\\ \varepsilon\end{bmatrix} = \begin{bmatrix}\mathbf{A} - \mathbf{B}\mathbf{K}_r & \mathbf{B}\mathbf{K}_r\\ \mathbf{0} & \mathbf{A} - \mathbf{K}_f\mathbf{C}\end{bmatrix}\begin{bmatrix}\mathbf{a}\\ \varepsilon\end{bmatrix} + \begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{I} - \mathbf{K}_f & \end{bmatrix}\begin{bmatrix}\mathbf{w}_d\\ \mathbf{w}_n\end{bmatrix} \tag{1.35}$$

Cuando se seleccionan las ganancias $\mathbf{K}_r$ y $\mathbf{K}_f$ para ubicar de forma óptima los valores propios del sistema en lazo cerrado, específicamente mediante las matrices $\mathbf{A} - \mathbf{B}\mathbf{K}_r$ en el diseño LQR y $\mathbf{A} - \mathbf{K}_f\mathbf{C}$ en el filtro de Kalman, estos mismos valores propios determinan el comportamiento dinámico del sistema controlado por LQG utilizando sensores.

El enfoque LQG parte de la idea de que se dispone de un modelo matemático preciso del sistema y de información clara sobre la intensidad del ruido en las mediciones y en el proceso interno del sistema. Además, el término gaussiano indica que estos ruidos se consideran como ruido blanco con distribución normal. Sin embargo, en aplicaciones reales, es común que existan errores en el modelo o que el ruido no siga exactamente esta distribución. Incluso pequeñas imprecisiones pueden afectar el desempeño del controlador LQG e, incluso, generar inestabilidad en el sistema [37].

Este tipo de diseño es conocido como control óptimo $\mathscr{H}_2$, y está orientado a lograr el mejor rendimiento posible bajo condiciones ideales. No obstante, cuando se trabaja con sistemas que tienen incertidumbre en su comportamiento o parámetros, es posible modificar este enfoque para crear controladores más resistentes a esas variaciones. Estos nuevos controladores se conocen como leyes de control robusto $\mathscr{H}_\infty$, cuyo objetivo es mantener el funcionamiento adecuado del sistema incluso bajo las peores condiciones posibles [37].

1.4.16 Lenguajes de programación

En el desarrollo de sistemas de control, automatización y robótica, los lenguajes de programación son herramientas fundamentales. A través de ellos, es posible escribir algoritmos que permiten simular el comportamiento de sistemas dinámicos, diseñar y ajustar controladores, procesar señales, comunicarse con sensores y actuadores, y ejecutar tareas de inteligencia artificial o visión por computadora. Elegir un lenguaje adecuado depende de factores como la facilidad de uso, la disponibilidad de bibliotecas especializadas, la compatibilidad con hardware, la velocidad de ejecución y el costo de la plataforma.

MATLAB

MATLAB es un lenguaje de programación orientado al cálculo numérico y al análisis de sistemas dinámicos. En robótica, se utiliza para resolver problemas relacionados con la planificación de trayectorias, la simulación de modelos y el diseño de controladores. Su estructura facilita el manejo de operaciones matriciales, ecuaciones diferenciales y modelos en espacio de estados [38].

El entorno gráfico Simulink permite construir sistemas mediante bloques interconectados, lo cual facilita la simulación de controladores y modelos robóticos sin necesidad de programar cada componente desde cero. Esto permite verificar el comportamiento del sistema antes de su implementación física [38].

Para trabajar con robots, se puede utilizar la herramienta Robotics Toolbox for MATLAB, desarrollada por Peter Corke. Esta librería permite definir manipuladores mediante estructuras DH, calcular la cinemática directa e inversa, resolver dinámicas y generar trayectorias en el espacio articular o cartesiano. Funciona a través de funciones que describen la estructura del robot y permiten simular su comportamiento en un entorno gráfico tridimensional [39].

Otra herramienta disponible es el ROS Toolbox, que permite la comunicación entre MATLAB y robots reales o simuladores mediante el sistema operativo de robots (ROS). Esta librería incluye funciones para publicar y suscribirse a tópicos, enviar comandos, recibir datos de sensores y sincronizar algoritmos de control desarrollados en MATLAB con entornos de simulación o plataformas físicas [40].

Python

Python es un lenguaje de programación gratuito y de código abierto, ampliamente utilizado para el seguimiento de trayectorias en brazos robóticos. Su popularidad se debe a la gran variedad de librerías especializadas que facilitan el modelado, control y simulación de sistemas robóticos. Librerías como `NumPy` y `SciPy` permiten realizar cálculos matemáticos y resolver ecuaciones diferenciales necesarias para describir el comportamiento dinámico del manipulador. En particular, `scipy.io` facilita la lectura y escritura de archivos de datos, útil para importar trayectorias o parámetros de control. La librería `Control` ayuda en el diseño y análisis de sistemas de control, mientras que `Matplotlib` permite visualizar trayectorias y resultados de simulación. Además, `time` se utiliza para gestionar la temporización y sincronización durante la ejecución de los algoritmos [41–44].

Para la simulación y control en entornos más completos, se emplean librerías como `ROS` (Robot Operating System), que posibilita la comunicación con robots físicos o simuladores. `MoveIt` está orientada a la planificación de movimientos y al seguimiento preciso de trayectorias en manipuladores, y `pybullet` permite simular la dinámica y detectar colisiones en entornos tridimensionales. Por otro lado, la librería `xarm` está diseñada específicamente para controlar manipuladores de la marca xArm, facilitando la conexión, el envío de comandos y la gestión de movimientos en tiempo real. Estas herramientas ofrecen una base sólida para desarrollar algoritmos de seguimiento adaptados a distintas plataformas [45–48].

El trabajo se realizará utilizando el entorno de desarrollo Spyder, que facilita la edición, ejecución y depuración de código Python en un entorno integrado, optimizando el proceso de diseño y prueba de algoritmos para la generación y seguimiento de trayectorias en el brazo robótico [49].

1.4.17 Tipos de comunicación en el brazo robótico

Para el funcionamiento adecuado del brazo robótico, es esencial establecer una comunicación efectiva entre los distintos componentes del sistema. En particular, el microcontrolador (como una tarjeta Arduino) debe enviar instrucciones al controlador de servos, el cual interpreta estas señales para accionar los motores que mueven las articulaciones. Este proceso se realiza utilizando interfaces de comunicación estándar, principalmente **UART** y **USB**.

Comunicación serial UART. La comunicación UART (Universal Asynchronous Receiver-Transmitter) es una forma de transmisión de datos de manera asincrónica entre dos dispositivos. En el brazo robótico, esta interfaz permite que el Arduino se comunique directamente con el controlador de servos mediante los pines RX (recepción) y TX (transmisión). Gracias a esta conexión, es posible enviar comandos en forma de bytes que indican la posición deseada de cada articulación. UART es ampliamente utilizada por su simplicidad, bajo costo y eficiencia en aplicaciones de control embebido [50].

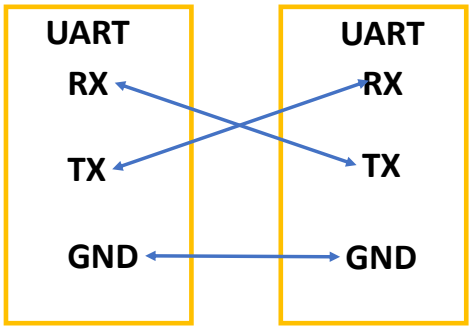


Figura 1.24: Comunicación UART. Fuente: [51]

HID. Durante la etapa de programación o para el monitoreo desde una computadora, el microcontrolador puede conectarse a través de un cable USB. Esta conexión permite cargar el código al Arduino y establecer una interfaz de comunicación con el entorno de desarrollo o software de simulación. Además, el USB puede ser usado para enviar datos de diagnóstico o registrar el comportamiento del sistema en tiempo real, facilitando tareas de prueba y validación

[52].

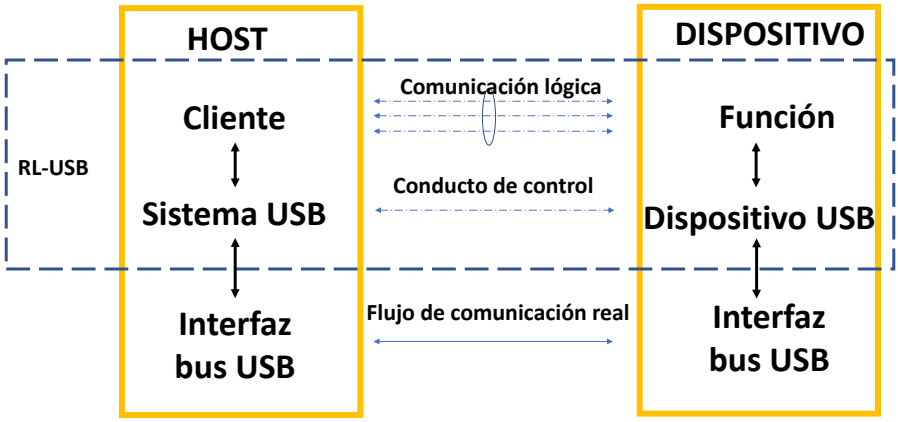


Figura 1.25: Comunicación HID. Fuente: [53]

Ambos tipos de comunicación cumplen funciones complementarias en el sistema del brazo robótico: mientras UART facilita la comunicación directa entre el microcontrolador y el hardware de control de movimiento, el USB se utiliza como canal de interfaz entre el sistema embebido y una estación de trabajo o entorno de simulación.

1.5 Marco contextual

El sistema implementado consta en el estudio y control de un brazo robótico de cinco grados de libertad y una garra diseñada para realizar tareas precisas, su estructura permite ejecutar movimientos complejos en el cual se implementará un controlador LQG el cual combina un regulador cuadrático lineal para optimizar el movimiento y un filtro de Kalman que maneja las incertidumbres del sistema.

Las articulaciones del brazo robótico están controladas mediante servomotores inteligentes de precisión, que reciben comandos de posición, velocidad y torque a través de un sistema de control basado en retroalimentación. Cada servomotor cuenta con sensores integrados que miden el tiempo real el ángulo de rotación y el esfuerzo aplicado. .

Se ha construido un soporte para el manipulador con el objetivo de tener una estabilidad y precisión durante el funcionamiento, la estructura principal está construida con materiales ligeros, pero resistentes, la base cuenta con

puntos de fijación que permiten ajustar y mantener el brazo en la posición deseada. El soporte ha sido diseñado con medidas de seguridad específicas para garantizar un entorno de trabajo seguro, la plataforma contará con distancias mínimas de seguridad alrededor del brazo robótico para prevenir algún tipo de accidente, estas distancias están determinadas en función de las capacidades de movimiento del manipulador, considerando sus cinco grados de libertad y las trayectorias que puede seguir.

La plataforma incluirá una parte como una pizarra o una cartelera, que servirá para mostrar el seguimiento de las trayectorias del brazo robótico, esta superficie será utilizada en tiempo real el desempeño y la ejecución de trayectorias, las cuales serán controladas de manera precisa por el controlador LQG.

El brazo robótico estará alimentado mediante un sistema eléctrico conectado a una fuente de alimentación externa, que transformará la corriente de la red eléctrica en energía adecuada para sus servomotores y los diferentes componentes electrónicos. Este sistema proporcionará un voltaje estable; la alimentación estará integrada de manera segura en la plataforma, asegurando conexiones firmes y confiables durante su operación.

En cuanto a la capacidad en la que cuenta el brazo robótico, tiene un límite de carga útil de aproximadamente de 2 a 3 kilogramos, según las especificaciones técnicas del fabricante. Este peso ayuda a que el manipulador mantenga su precisión y estabilidad para cumplir con las trayectorias; sin embargo, si se llegase a exceder esta capacidad, podría comprometer el funcionamiento y ocasionar daños a los servomotores. Además, el diseño compacto del xArm hace que su rango de alcance y carga deban ser gestionados dentro de sus limitaciones técnicas.

El funcionamiento del brazo robótico será a través de una interfaz que permitirá su control; esta interfaz estará diseñada para ser intuitiva y funcional, la cual permitirá definir las trayectorias, ajustar parámetros de movimiento y monitorear su desempeño en tiempo real.

2 Métodos y diseño experimental

2.1 Metodología

En este proyecto, el proceso de investigación e implementación se basa en una combinación de metodologías cualitativas y cuantitativas. La investigación cualitativa se utiliza para recopilar información sobre el método experimental, la propuesta de inspección y el análisis del comportamiento del sistema. La selección de antecedentes proporciona la creación de una base de datos con la información obtenida.

Metodología cualitativa

En este proyecto se aplicó una metodología cualitativa basada en el análisis documental para estudiar el seguimiento de trayectorias en brazos robóticos utilizando un controlador LQG. Se consultaron fuentes como artículos científicos, libros especializados y documentación técnica reciente. La selección de estos materiales se centró en publicaciones relacionadas con la dinámica de manipuladores, control óptimo y técnicas de seguimiento en entornos con ruido.

El análisis de la información permitió conocer distintas estrategias de planificación y control de movimiento aplicadas a brazos robóticos con múltiples grados de libertad. Se revisaron casos de aplicación práctica que sirvieron como referencia para estructurar la arquitectura del sistema y su lógica de funcionamiento.

A partir de esta revisión, se definieron los componentes esenciales del sistema: la representación matemática del modelo dinámico, el algoritmo de control, los métodos de generación de trayectorias y los criterios de evaluación. La metodología cualitativa permitió establecer una base técnica sólida para guiar el diseño e implementación del proyecto, asegurando coherencia entre los objetivos planteados y las decisiones adoptadas durante el desarrollo.

Además, este enfoque permitió la comprensión de los límites tecnológicos actuales y ayudó a identificar herramientas de software y bibliotecas adecuadas para la simulación, análisis y validación del comportamiento del sistema. Todo este conocimiento fue integrado en el desarrollo del controlador LQG aplicado al brazo robótico.

Metodología experimental

Complementando el enfoque cualitativo, la metodología experimental se utilizará para establecer relaciones entre las variables y el sistema. En esta fase, se diseñarán los experimentos específicos que permitirán evaluar el rendimiento del controlador LQG aplicado a un prototipo de brazo robótico.

Se realizarán pruebas en condiciones controladas para observar cómo el sistema responde a diversas perturbaciones y configuraciones operativas, como variaciones en las cargas y trayectorias a seguir. El diseño experimental incluye la programación y la simulación en MATLAB, donde se implementarán los algoritmos necesarios para evaluar el desempeño del controlador.

2.1.1 Descripción del proyecto

En el proceso de diseño del algoritmo de programación para el microcontrolador, basado en el control LQG, implica la selección de parámetros del controlador, tales como las ganancias del controlador y del observador de estado.

Este proceso necesita un buen ajuste de los parámetros para lograr que el sistema funcione de la mejor manera posible. Una vez que se hayan definido los valores correctos del algoritmo de control LQG, se podrá aplicar en una simulación o en un sistema real, dependiendo de los recursos que se tengan disponibles.

Se realizarán pruebas y ajustes para verificar el rendimiento del controlador y se realizarán mejoras si es necesario [54].

En la Figura. 2.1 muestra un diagrama de flujo, describe las etapas para el diseño e implementación de un controlador para un brazo robótico, incluyendo la identificación del sistema, el modelado, la recopilación de datos, el diseño del controlador y el observador, la implementación en el sistema y las pruebas para verificar su funcionamiento [55].

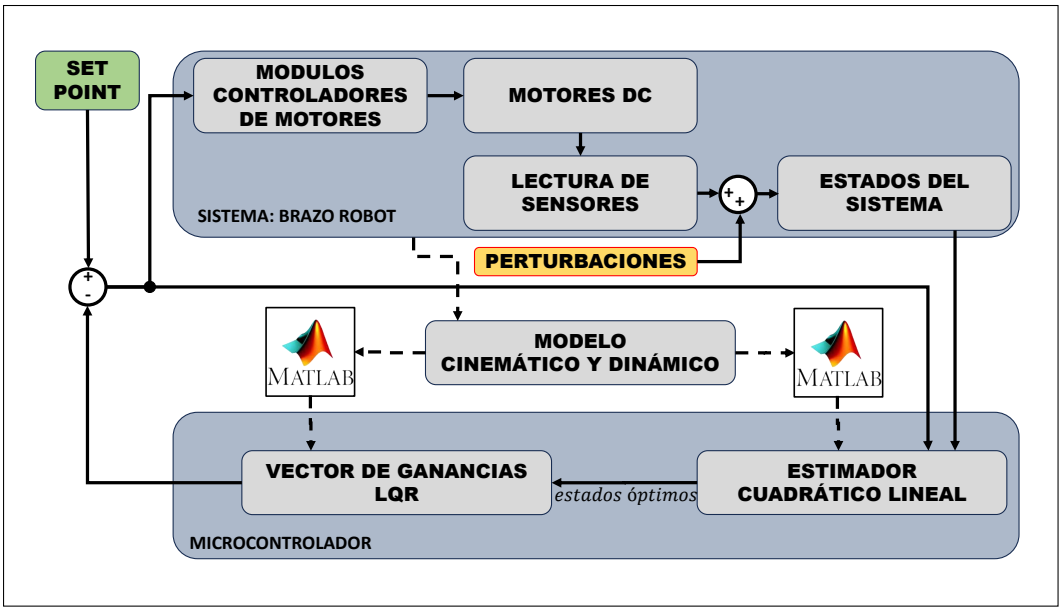


Figura 2.1: Diagrama de bloques del sistema en lazo cerrado: controlador LQG. Fuente: Elaboración propia.

En la Fig.2.2 muestra el diagrama de proceso para el diseño de un algoritmo de control LQG aplicado al seguimiento de trayectoria con un brazo robótico [56].

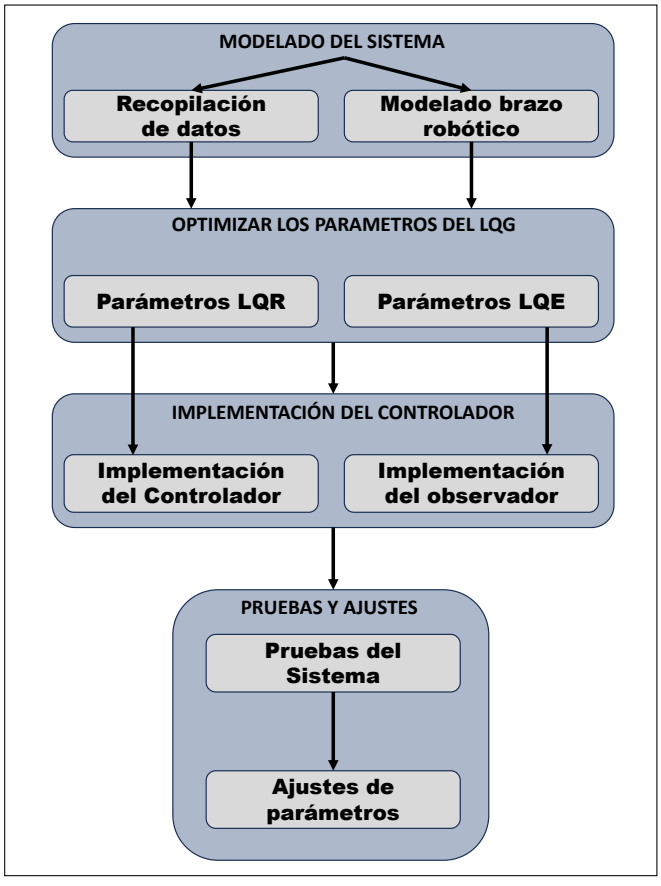


Figura 2.2: Diagrama de flujo del proyecto. Fuente: Elaboración propia.

- **Etapas del proyecto.**

  El proyecto se desarrolla en las siguientes etapas:

  1. Modelado: En esta etapa se obtendrá el modelo dinámico del brazo robótico, que será utilizado para el diseño del controlador. El modelo se obtendrá a partir de las ecuaciones cinemáticas y ecuaciones de movimiento del brazo [57].

  2. Diseño del controlador LQG: En esta etapa se diseñará el controlador LQG utilizando el modelo del sistema obtenido en la etapa anterior. El controlador se diseñará para minimizar el error entre la posición y orientación del brazo y la trayectoria deseada [8].

  3. Implementación del controlador: En esta etapa se implementará el controlador en un simulador para evaluar su desempeño [57].

  4. Pruebas experimentales: En esta etapa se probará el controlador en un brazo robótico real para validar su desempeño en condiciones reales [57].

- **Estructura del algoritmo.**

  El proyecto se desarrollará en las siguientes etapas:

  1. Modelado del sistema: En este paso, se obtiene el modelo cinemático del brazo robótico. El modelo cinemático describe la relación entre las entradas del sistema (ángulos de las articulaciones) y las salidas del sistema (posición y orientación del extremo final del brazo) [56].

  2. Diseño del controlador: En este paso, se calcula la matriz de control óptimo utilizando el algoritmo de Riccati. La matriz de control óptimo es la matriz que minimiza el error entre la trayectoria deseada y la trayectoria real del sistema [56].

  3. Simulación del sistema: En este paso, se simula el sistema utilizando un modelo del sistema real. Esto permite evaluar el desempeño del controlador en diferentes condiciones [56].

  4. Implementación del controlador: En este paso, se implementa el con-

trolador en el sistema real. Esto se puede realizar utilizando un microcontrolador o una tarjeta de desarrollo [56].

2.1.2 Componentes de la propuesta

En esta sección se detallarán los componentes físicos y lógicos que se utilizan en este proyecto, desde el diseño del sistema, la implementación en hardware que contempla la transferencia del controlador LQG a un sistema físico, utilizando microcontroladores. Se describe el proceso de integración del hardware, que incluye actuadores y sensores de retroalimentación.

2.1.2.1 Componentes físicos

En el desarrollo del sistema de control para el seguimiento de trayectoria de un brazo robótico, se emplea diversos componentes físicos donde permite la implementación y validación del algoritmo LQG en un entorno real, esto incluirá la estructura mecánica del brazo robótico, los actuadores responsables del movimiento, la unidad de procesamiento que estará basada en microcontroladores, estos elementos garantizaran la precisión y estabilidad del sistema.

Brazo robótico Xarm 1S

El XArm 1S es un brazo robótico antropomórfico diseñado para aplicaciones educativas, y de investigación, junto con proyectos de automatización, este brazo robótico cuenta con cinco grados de libertad el cual permite realizar movimientos precisos y una de las funciones que tiene es replicar tareas en un entorno controlado, dicho robot se caracteriza por su versatilidad, y facilidad de uso [58].

Los componentes clave de este mecanismo se organizan en función de la estructura mecánica y los elementos electrónicos que permiten un control preciso del robot, tales como soportes en forma de U, discos giratorios y una pinza que permite la manipulación de objetos y la ejecución de tareas.

El movimiento del brazo es controlado por varios servomotores LX-15D y LX-225, que proporcionan un control preciso sobre las articulaciones y los ejes de movimiento. Estos servos están conectados a través de una interfaz de controlador de servo bus, que centraliza la alimentación y la comunicación entre los servos y el sistema de control principal.

Cuenta con un controlador, que incluye múltiples puertos y conectores para la programación y un módulo Bluetooth que permite la operación remota.

Además, el controlador cuenta con indicadores LED; los servomotores están conectados a través de la interfaz de canales, lo que permite controlar con precisión el movimiento de cada articulación.

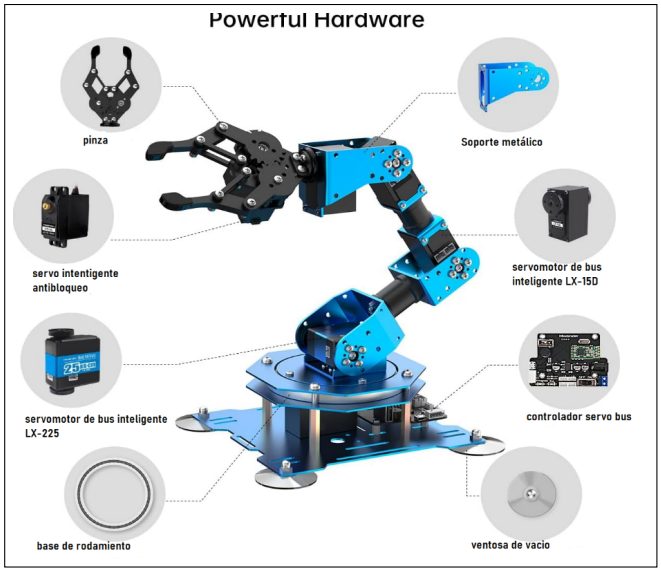


Figura 2.3: XArm 1S: brazo robótico Servo de autobús inteligente Hiwonder para programación Fuente: [34].

Dimensiones del brazo robótico Xarm 1S

En la Fig. 2.4 se proporciona una representación detallada de las dimensiones del brazo robótico en posición vertical. El brazo robótico alcanza $42,9$ cm, mientras que su configuración retraída alcanza $36,4$ cm. El diseño se encuentra de manera segmentada en diversas secciones de longitud variable, el segmento superior es $9,5$ cm, el segmento intermedio cuenta con $10,1$ cm y la base es de $6,5$ cm de altura. La base del brazo robótico cuentas con dimensiones en planta de 14 cm de ancho por 12,7 cm de profundidad, con un ancho total desde el punto de vista frontal, de $15,4$ cm lo que proporciona estabilidad estructural al brazo robótico. La pinza del brazo robótica está diseñada para manipulación de objetos, tiene un ancho máximo de $9,6$ cm y un espacio interno de $5,5$ cm lo que permite el manejo de piezas dentro de estas dimensiones.

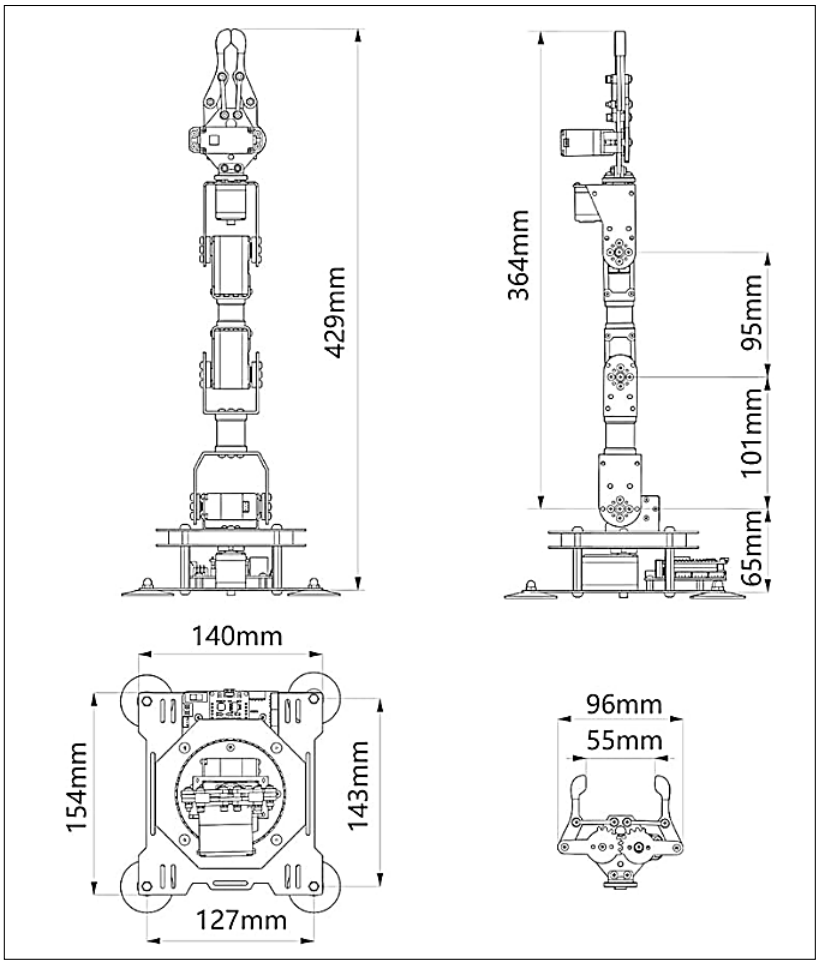


Figura 2.4: Diseño y dimensiones del brazo robótico Fuente: [33]

Módulo controlador LSC-6 para servomotores tipo Servo Bus

La Figura 2.5 muestra el módulo controlador LSC- es el encargado de gestionar la comunicación y el control de los servomotores digitales tipo Servo Bus utilizados en el brazo robótico xArm 1S. Este controlador recibe, interpreta y transmite señales de control, permitiendo el posicionamiento preciso de cada motor. Cuenta con una alimentación externa de 9V y dispone de indicadores LED que permiten verificar visualmente el estado del dispositivo, tales como la activación del sistema y la transmisión de señales de datos. Posee cuatro canales de conexión para servomotores, los cuales pueden ser encadenados en serie gracias a la topología de comunicación del bus

En cuanto a sus interfaces de entrada, el módulo admite comunicación serial a 9600 baudios a través de diversos puertos, incluyendo UART, USB, Bluetooth y conexiones para dispositivos como joystick o mouse. Por su parte, la salida hacia los servomotores se realiza mediante comunicación serial semidúplex a 115200 baudios, lo que garantiza una transmisión rápida y precisa de los comandos de posición y velocidad, asegurando la correcta ejecución de

las trayectorias programadas. Adicionalmente, el LSC-6 incorpora un botón de programación manual que permite ajustar las posiciones articulares directamente desde el controlador, sin necesidad de software externo. Esta función resulta útil para la generación de secuencias de movimiento básicas y para pruebas de funcionamiento de manera autónoma.

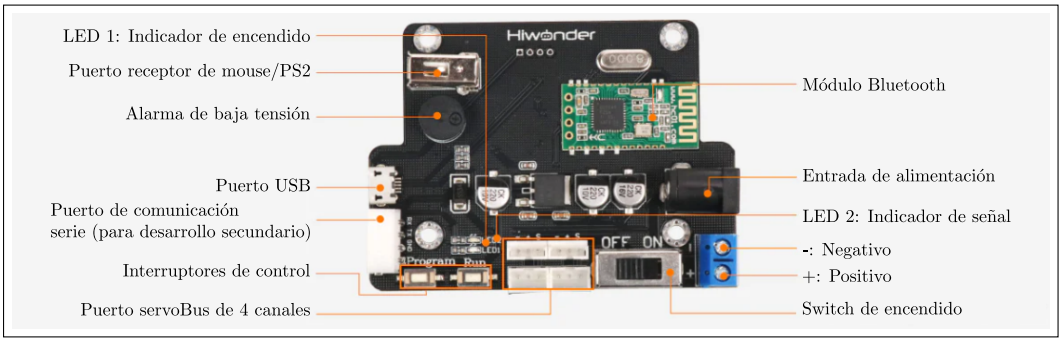


Figura 2.5: Controlador de host de servos [58].

Tabla 2.1: Especificaciones generales del controlador principal del brazo robótico

Parámetro	**Descripción**
Canales	6
Versión Bluetooth	4.0
CPU	STM32 con núcleo ARM Cortex-M3
Capacidad de memoria	16M
Grupos de acción	Hasta 230 grupos
Acciones por grupo	Hasta 1020 acciones
Opciones de control	Aplicación Android, mando de PS2
Interfaz de comunicación	Serial TTL
Protección de sobrecorriente	Si
Interruptor de encendido	Integrado
Voltaje de operación	5 V
Dimensiones	8 cm × 5 cm × 3 cm
Peso	47,8 g

Motores servo bus LX-15D

El LX-15D es un servo bus de alto rendimiento diseñado para proporcionar movimientos precisos y suaves en las articulaciones del brazo robótico. Posee un par máximo de 17 kg·cm a 7,4V, lo que le permite manipular cargas considerables con estabilidad. Además, este diseño incluye engranajes metálicos,

doble rodamiento de bolas y un ángulo de rotación de 0° a 240°, lo que lo hace muy útil para aplicaciones que requieren precisión en el control de posición [58].

Otra característica del LX-15D es su capacidad de retroalimentación en tiempo real, lo que permite monitorear la posición, temperatura y voltaje durante su operación. Esto es fundamental para optimizar el desempeño del brazo robótico y prevenir fallas por sobrecalentamiento o sobrecarga. La integración en el bus serie, el servomotor puede ser controlado de manera eficiente dentro de sistemas robóticos complejos [59]. En la Figura. 2.6 muestra el servomotor LX-15D.

Figura 2.6: Motores LX-15D [58].

Tabla 2.2: Especificaciones técnicas del motor LX-15D.

Item	**Especificaciones**
Peso	43g
Dimensiones	44.02*22.92*35.12mm
Voltaje de trabajo	6–8.4V
Velocidad	0.22sec/60° a 7.4V
Torque	15kg.cm a 6V; 17kg.cm a 7.4V
Rotación	0–240°
Corriente sin carga	100mA
Corriente de parada	2.4∼3A
Precisión del servo	0.3°
Método de control	Comando serie UART
Velocidad de comunicación	115200

Tabla 2.2 continuación de la página anterior	
Item	**Especificaciones**
en baudios	
Almacenamiento	Guarda los datos cuando se apaga
Servo ID	0∼253 para el usuario, ID 1 por defecto
Función de lectura del ángulo	Soporte
Protección	Evita el calado y el sobrecalentamiento
Información de retorno	Temperatura, tensión, posición
Indicador	LED
Modo de trabajo	Modo servo y modo motor de deceleración
Rango del ángulo de control	0–1000, corresponden a 0–240°
Cable	200mm
Engranaje	Engranaje de aleación

Servomotor LX-255

El servomotor LX-255 se caracteriza por su alta durabilidad y resistencia, lo que lo hace adecuado para aplicaciones con exigencias mecánicas considerables. Su carcasa, fabricada en aleación de aluminio anodizado, favorece una mejor disipación térmica, lo que prolonga su vida útil y mejora su fiabilidad en entornos industriales y de investigación. Posee un par máximo de 25 Kg x cm, este servomotor es capaz de controlar cargas moderadas con precisión y estabilidad [59].

Una de sus principales cualidades es su clasificación IP67, que le confiere resistencia al polvo y a la humedad, permitiéndole operar de forma fiable en condiciones ambientales adversas. Al igual que el modelo LX-15D, el LX-255 está equipado con engranajes metálicos y doble rodamiento de esferas, lo que garantiza un movimiento suave y una mayor vida útil del mecanismo [59].

Ambos servomotores emplean señales PWM y comunicación serial, lo que permite realizar ajustes precisos en su posición. Además, ofrecen retroalimentación en tiempo real sobre variables como posición, temperatura y voltaje. Esta característica permite detectar y corregir errores de manera inmediata, asegurando así un control eficiente del robot xArm 1S y mejorando su desempeño en tareas de automatización y robótica.

En la Figura 2.7 se presenta el servomotor LX-255, donde se destacan sus

características principales y detalles constructivos.

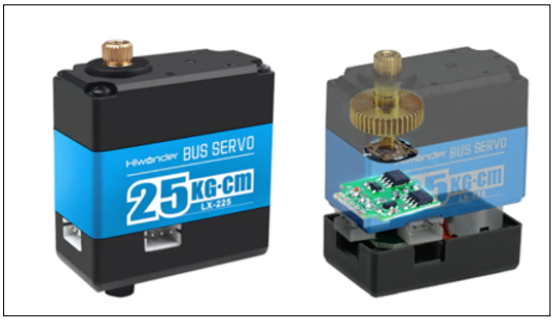


Figura 2.7: Motor LX-255 [58].

Tabla 2.3: Especificaciones técnicas del servomotor LX-255

Especificación	**Valor**
Par máximo	25 kg cm
Velocidad	0,15 s/60° (7,4 V)
Ángulo de rotación	0° a 270°
Retroalimentación,	Posición, temperatura, voltaje
Engranajes	Metálicos
Rodamientos	Doble rodamiento de bolas
Material de carcasa	Aleación de aluminio anodizado
Resistencia	IP67 (resistente al agua y polvo)
Dimensiones	40 mm × 20 mm × 40,5 mm
Peso	68 g

2.1.2.2 Componentes lógicos

En esta sección se describen los programas y herramientas usados para trabajar con el brazo robótico. Se explican los lenguajes y entornos como Matlab, Python, Simulink y código G, que ayudaron a diseñar, controlar y simular el movimiento del robot.

MATLAB

MATLAB es un entorno de desarrollo y lenguaje de programación de alto rendimiento, ampliamente utilizado para el análisis numérico, la simulación matemática y la visualización de datos. Una de sus principales ventajas es la manipulación eficiente de matrices, lo cual permite realizar cálculos complejos de manera sencilla, facilitando el trabajo en áreas como la ingeniería, la ciencia aplicada y la economía. Además, ofrece una gran variedad de funciones matemáticas que ayudan a resolver problemas técnicos de forma rápida y

precisa.

Otra característica destacada de MATLAB es su capacidad para generar gráficos y visualizaciones de datos, lo que permite interpretar los resultados de manera clara. Se pueden crear desde gráficos básicos en 2D hasta visualizaciones más avanzadas en 3D, lo cual es especialmente útil para representar el comportamiento de sistemas dinámicos y analizar su desempeño. Esta función resulta fundamental para comprender el funcionamiento del sistema y tomar decisiones durante el proceso de diseño o simulación.

MATLAB también permite crear algoritmos personalizados, simular sistemas en tiempo continuo o discreto, y desarrollar modelos matemáticos complejos. Además, tiene la capacidad de integrarse con otros lenguajes de programación como C, C++, Java y Python, lo que lo convierte en una herramienta versátil para proyectos multidisciplinarios. Su uso se extiende a la investigación, el análisis de datos, la automatización de procesos y el desarrollo de sistemas de control.

En la Figura 2.8 se presenta la interfaz gráfica de MATLAB, donde se pueden observar los elementos principales utilizados en este proyecto. La ventana de comandos permite ejecutar instrucciones y visualizar resultados de forma inmediata. A la izquierda, se encuentra el espacio de trabajo, donde se listan las variables activas. En la parte superior, se despliegan las pestañas con herramientas para programación, gráficos y ejecución de simulaciones. Esta interfaz facilita el diseño, prueba y ajuste de algoritmos, permitiendo una interacción directa con los datos y modelos utilizados.

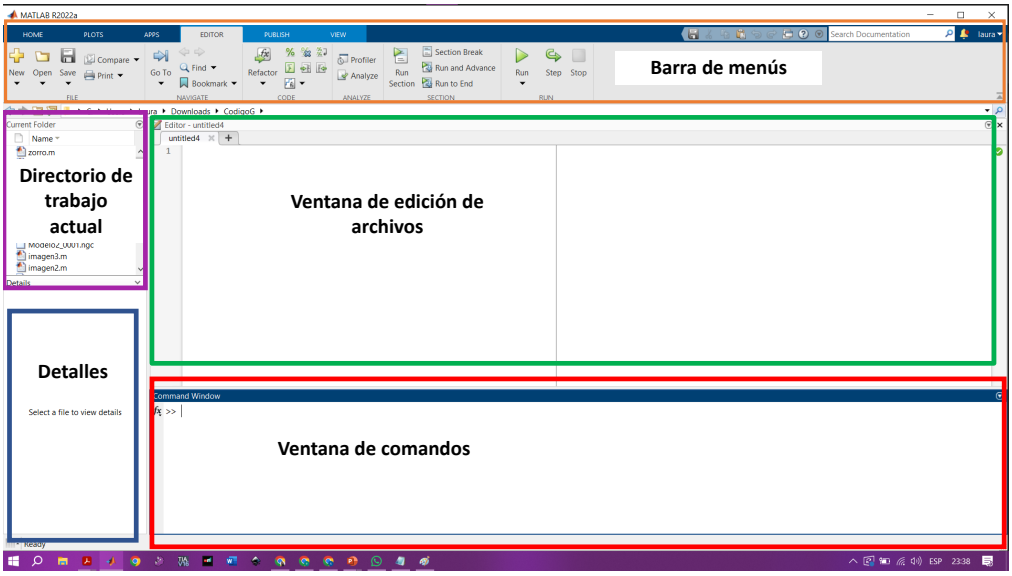


Figura 2.8: Interfaz de MATLAB utilizada en el desarrollo del proyecto

Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general, ampliamente utilizado en entornos científicos y de ingeniería debido a su sintaxis clara y su extenso ecosistema de bibliotecas. En este proyecto, Python cumple un rol fundamental en el diseño experimental, al ser empleado para desarrollar scripts de simulación, generar trayectorias de referencia y procesar los datos obtenidos de las pruebas. Además, se utiliza para la visualización gráfica de resultados y la validación del desempeño del controlador LQG, gracias a herramientas como *NumPy*, *Matplotlib* y *SciPy*. Su uso permite una implementación flexible y eficiente del modelo dinámico del brazo robótico y de los algoritmos de control y estimación.

La Figura 2.9 muestra la interfaz del entorno de desarrollo Spyder, diseñado específicamente para trabajar con el lenguaje de programación Python. En ella se destacan varios elementos clave: el (1) señala la línea activa donde se edita el código, el (2) corresponde a la barra de desplazamiento vertical, y el (3) muestra el menú contextual con funciones como ejecutar celdas o copiar y pegar. El (4) da acceso a configuraciones del editor, el (5) es la barra de herramientas para ejecutar, pausar o guardar código, y el (6) presenta las pestañas de archivos abiertos, facilitando el trabajo con múltiples scripts. Estos componentes permiten una edición y ejecución eficiente dentro del entorno Spyder.

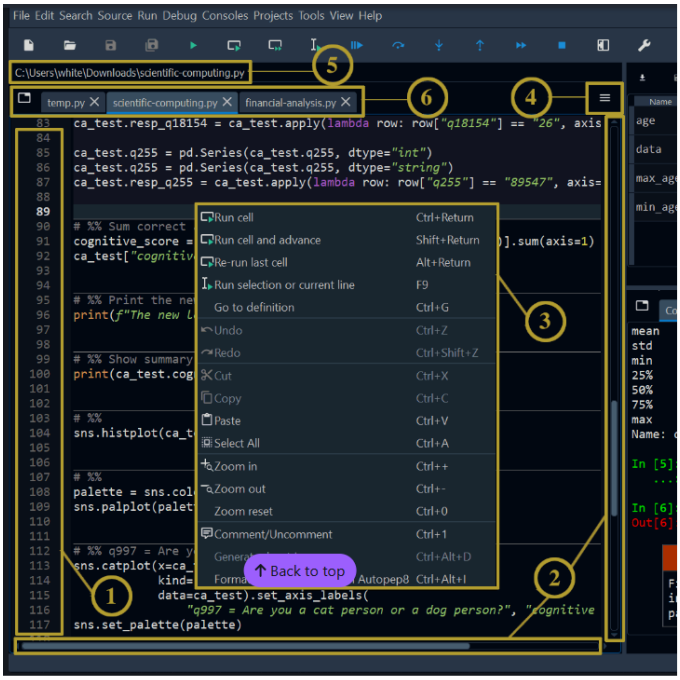


Figura 2.9: Interfaz SYDER Fuente: [60]

Inskecape

Inkscape es una herramienta de software libre utilizada para la edición de gráficos vectoriales en formato SVG. En el contexto del diseño experimental de este proyecto, Inkscape se emplea principalmente para la elaboración de diagramas ilustrativos y esquemáticos del sistema robótico, incluyendo representaciones del modelo cinemático y dinámico, así como la arquitectura de control. Estas representaciones visuales son fundamentales para documentar de manera clara y precisa la estructura del sistema y facilitar la comunicación técnica. La capacidad de Inkscape para generar gráficos escalables y detallados permite una presentación profesional de los elementos visuales integrados en el documento técnico.

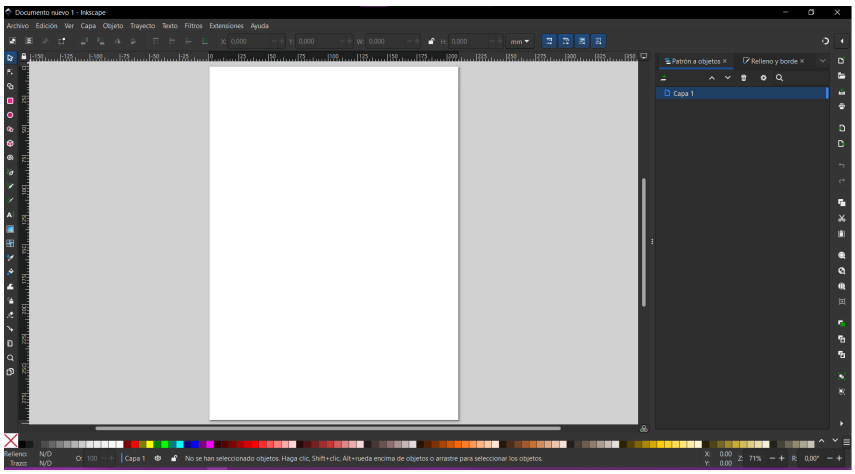


Figura 2.10: Interfaz Inskecape

NC Viewer

NC Viewer es una herramienta en línea diseñada para la visualización y simulación de código G (G-code), ampliamente utilizada en procesos de manufactura asistida por computadora. En el contexto del diseño experimental de este proyecto, NC Viewer se emplea para verificar la correcta interpretación de las trayectorias generadas por el controlador, permitiendo observar de forma gráfica el movimiento planificado del efector final. Su uso facilita la validación previa al envío del código a sistemas físicos, asegurando que las instrucciones de movimiento correspondan con la trayectoria deseada y evitando errores potenciales en la ejecución real.

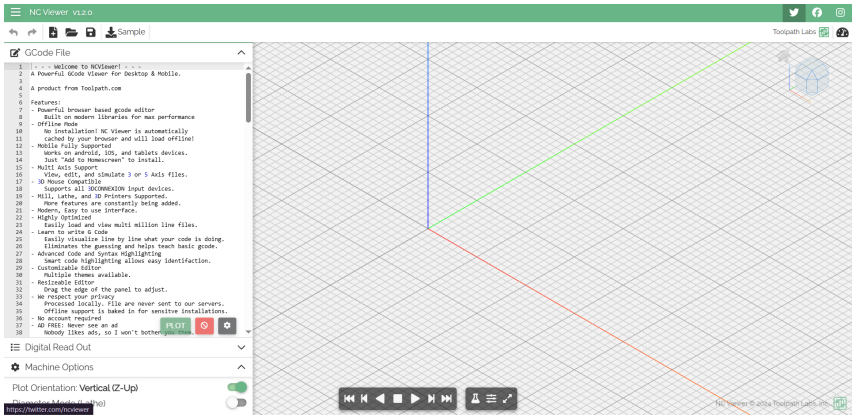


Figura 2.11: Interfaz NC Viewer

App designer

App Designer es una herramienta de MATLAB que permite crear aplicaciones con interfaces gráficas de usuario (GUI) de manera fácil y visual. Con esta herramienta, se pueden diseñar aplicaciones interactivas sin necesidad de escribir mucho código. Es una interfaz que ofrece las opciones de arrastrar y agregar botones, cuadros de texto, gráficos y otros controles.

Además de la parte visual, se puede añadir código en MATLAB para que la aplicación realice tareas específicas, como cálculos o visualización de datos. App Designer también permite crear aplicaciones que interactúan con el usuario en tiempo real. En la Figura. 2.12 se muestra la interfaz de App Designer de MATLAB.

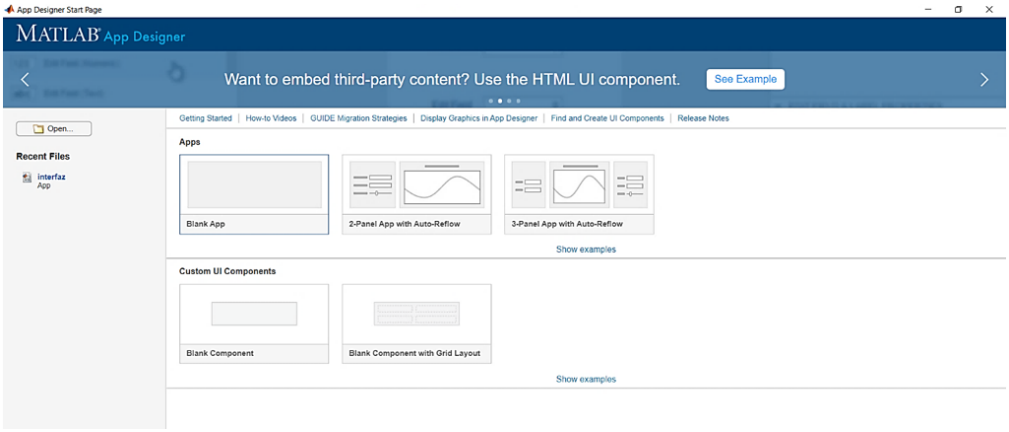


Figura 2.12: Interfaz App Designer

El entorno de trabajo de App Designer está dividido principalmente en dos áreas: el área de diseño y el editor de código. En el área de diseño, el usuario puede arrastrar y soltar diferentes componentes gráficos, como botones, etiquetas, campos de texto, controles deslizantes y gráficos. Esta interfaz intuitiva facilita la creación de interfaces gráficas sin necesidad de conocimientos avanzados en programación.

Por otro lado, el editor de código permite agregar la funcionalidad lógica a cada uno de los componentes insertados. Cada acción o evento generado por el usuario puede vincularse con funciones personalizadas escritas en MATLAB, lo que permite automatizar procesos, realizar cálculos y actualizar visualizaciones dinámicamente. Además, App Designer integra una vista jerárquica de componentes que ayuda a organizar y acceder rápidamente a los elementos de la aplicación.

Este entorno también ofrece herramientas para depurar el código, probar la aplicación en tiempo real y exportarla en diferentes formatos. Gracias a estas funcionalidades, App Designer se convierte en una plataforma completa para el desarrollo de interfaces gráficas eficientes y profesionales en proyectos de ingeniería y simulación.

Simulink

La interfaz de Simulink es una herramienta gráfica integrada en MATLAB, diseñada para modelar, simular y analizar sistemas dinámicos mediante bloques funcionales conectados entre sí. En el desarrollo de un sistema de control para un brazo robótico de 5 grados de libertad, Simulink permite representar tanto el modelo dinámico (derivado mediante el método de Euler-Lagrange) como la implementación del controlador LQG, integrando el control óptimo

LQR y el estimador de estados tipo Kalman.

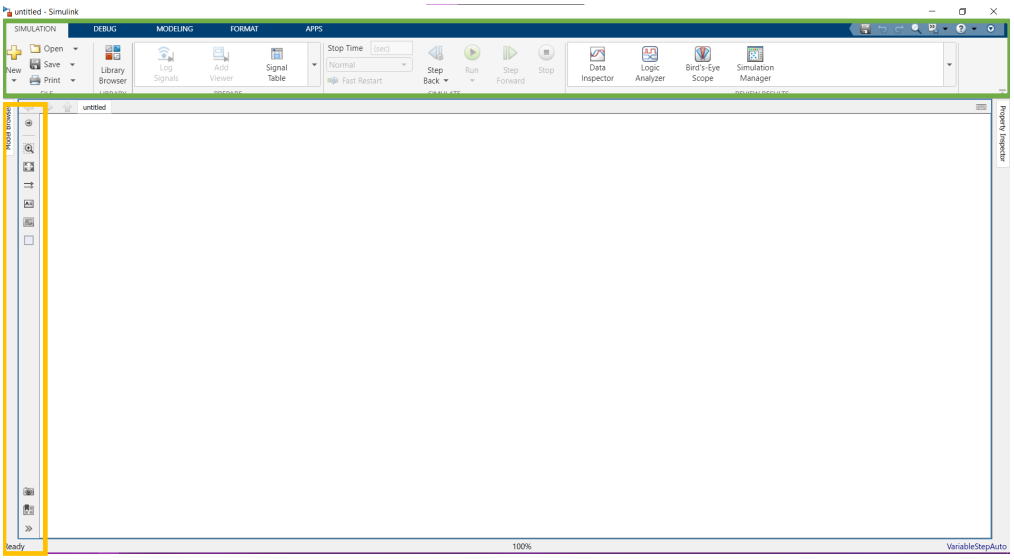


Figura 2.13: Interfaz Simulink

Esta interfaz facilita la organización del modelo en subsistemas, la configuración de parámetros de simulación y la visualización de resultados en tiempo real. También permite el uso de bloques personalizados y la vinculación con código MATLAB, lo que hace posible implementar lógica de control compleja y realizar pruebas antes de aplicar el sistema en hardware físico. En conjunto, Simulink es una plataforma que permite validar, ajustar y optimizar sistemas de control en entornos de robótica y automatización.

Protocolo de comunicación del brazo robótico

La placa controladora del robot Xarm S1 utiliza comunicación por protocolo HID a través de un cable USB, como se muestra en la Figura 2.5. Este método permite la interacción directa con las variables y comandos del motor Servo Bus. Debido a la falta de documentación completa del fabricante Hiwonder sobre las estructuras de las tramas de datos, se llevó a cabo un análisis de tráfico USB mediante la herramienta `Wireshark`, durante la operación del software original `Xarm.exe`. Esta investigación permitió identificar las tramas necesarias para controlar el servo, incluyendo el comando para activar el modo motor en el Servo Bus, sin necesidad de utilizar módulos adicionales para el protocolo LewanSoul.

La comunicación se establece mediante un enlace serie, con una velocidad de transmisión de 9600 baudios. La estructura general de las tramas de datos está compuesta por una cabecera, longitud, comando y parámetros, que se

describen a continuación en la Tabla 2.4.

Tabla 2.4: Formato general de la trama de datos para comunicación ServoBus

Campo	**Descripción**
Cabecera	Dos bytes consecutivos `0x55 0x55` que indican el inicio de una trama válida.
Longitud	Indica la cantidad total de bytes del comando y parámetros. Calculada como número de parámetros N más dos (un byte para el comando y un byte para la longitud), es decir, Longitud = $N + 2$.
Comando	Código que especifica la instrucción o acción a ejecutar.
Parámetros	Información adicional necesaria para el comando, cuya cantidad y significado dependen del tipo de comando.

Entre los comandos relevantes destaca el que activa el modo motor y ajusta la velocidad del servo, identificado como `CMD_MOTOR_MODE_WRITE` con valor 26. Este comando permite controlar la velocidad del motor en un rango de -1000 a 1000, y su estructura se detalla en la Tabla 2.5.

Tabla 2.5: Estructura de la trama para ajuste de velocidad del servo

Campo	**Descripción**
ID del servo	Identificador único del servo a controlar.
Modo	Indica modo motor (1) o modo servo (0). También puede indicar cantidad de servos a controlar (pendiente de confirmación).
Velocidad baja	8 bits menos significativos del valor de velocidad.
Velocidad alta	8 bits más significativos del valor de velocidad.

Como ejemplo, para mover el servo número 2 en modo motor a una velocidad de -1000, la trama enviada es la que se muestra en la Tabla 2.6.

Tabla 2.6: Ejemplo de trama para mover servo en modo motor con velocidad -1000

Cabecera	Longitud	Comando	Parámetros
`0x55 0x55`	`0x06`	`0x1A`	`0x02 0x01 0x18 0xFC`

Otro comando importante es el `CMD_SERVO_MOVE` con valor 3, que permite controlar la posición angular de uno o varios servos, especificando el tiempo de

movimiento. La longitud de la trama depende del número de servos a controlar y su estructura se muestra en la Tabla 2.7.

Tabla 2.7: Estructura de la trama para control de posición de servo.

Campo	**Descripción**
Número de servos	Cantidad de servos a controlar.
Tiempo bajo	8 bits menos significativos del tiempo para alcanzar la posición (ms).
Tiempo alto	8 bits más significativos del tiempo.
ID del servo	Identificador del servo.
Posición baja	8 bits menos significativos del ángulo objetivo.
Posición alta	8 bits más significativos del ángulo objetivo.
El formato de los parámetros para servos adicionales sigue el mismo esquema de ID y valores de posición.	

Como ejemplo práctico, para mover el servo número 1 a la posición 2000 en un tiempo de 1000 ms, la trama correspondiente es la indicada en la Tabla 2.8.

Tabla 2.8: Ejemplo de trama para control de posición del servo

Cabecera	**Longitud**	**Comando**	**Parámetros**
`0x55 0x55`	`0x08`	`0x03`	`0x01 0xE8 0x03 0x01 0xD0 0x07`

2.2 Diseño experimental

Esta sección presenta el diseño experimental llevado a cabo para validar el desempeño del sistema de control propuesto, aplicado a un brazo robótico de 5 grados de libertad. Se abordan los aspectos físicos y lógicos del sistema, así como las condiciones en las que se realizaron las pruebas.

Se describen los componentes físicos involucrados, incluyendo a el brazo robótico, los actuadores, los sensores de posición y el sistema de adquisición de datos. También se detallan los elementos lógicos, tales como los algoritmos de control desarrollados, la estructura de comunicación entre los distintos módulos del sistema y el software utilizado para la supervisión y monitoreo en tiempo real.

Además, se especifican las condiciones de prueba empleadas en los experimentos, entre ellas las trayectorias de referencia utilizadas, los límites ar-

ticulares, las velocidades máximas establecidas y la frecuencia de muestreo. Se describe igualmente la configuración del entorno físico y computacional, incluyendo el hardware utilizado, el sistema operativo, las herramientas de desarrollo implementadas y los protocolos de comunicación adoptados para la interacción entre el controlador y el manipulador.

Como parte integral de la validación, se incorpora la solución matemática de la cinemática y la dinámica del brazo robótico, desarrollada previamente. Este modelo permite predecir el comportamiento del sistema y sirve como base teórica para contrastar los resultados obtenidos de forma experimental. La comparación entre las respuestas del modelo y el comportamiento real del manipulador proporciona una medida de la fidelidad del sistema y de la efectividad del controlador.

Se expone el procedimiento empleado para la evaluación del seguimiento de trayectorias, el cual incluye el análisis comparativo entre las trayectorias deseadas y las realmente ejecutadas por el brazo. Este análisis permite cuantificar el desempeño del sistema de control en términos de precisión, estabilidad ante condiciones reales de operación.

2.2.1 Obtención del modelo del brazo robótico

En esta sección se presenta la solución matemática del modelo de un brazo robótico de 5 grados de libertad, abarcando tanto la cinemática como la dinámica del sistema. Se desarrollan las expresiones que describen el comportamiento del manipulador a partir de su configuración geométrica y propiedades físicas. La finalidad es establecer las bases analíticas necesarias para el análisis, simulación y diseño de estrategias de control.

2.2.1.1 Solución de la cinemática del manipulador Xarm 1S

Considerando que la base teórica sobre cinemática de manipuladores ha sido abordada previamente, en esta sección se presentan las soluciones de los modelos de cinemática directa e inversa.

Solución cinemática directa

El manipulador robótico xArm 1S empleado en este proyecto está montado sobre una base de aluminio que le proporciona estabilidad y rigidez estructural, características fundamentales para garantizar precisión en el posicionamiento. Este manipulador está conformado por una cadena cinemática abierta con cinco grados de libertad, compuesta por articulaciones rotacionales impulsa-

das por servomotores digitales. La solución de la cinemática directa consiste en determinar la posición y orientación del efector final a partir de los valores angulares de las articulaciones. Para este fin, se formulan las ecuaciones correspondientes considerando la estructura geométrica del brazo y las restricciones mecánicas de cada eje. Estas ecuaciones se expresan como funciones de los ángulos articulares, permitiendo calcular la configuración espacial del extremo del manipulador en su espacio de trabajo [1].

La Figura 2.14 presenta el modelo esquemático del brazo robótico, destacando sus cinco articulaciones coordinadas y los actuadores que permiten ejecutar movimientos complejos de forma precisa y continua. Denavit y Hartenberg en Schilling [1] y Crag [56] propusieron una notación sistemática para asignar los marcos de coordenadas ortonormales rectos. Una vez asignados estos marcos de coordenadas vinculados, las transformaciones entre marcos de coordenadas adyacentes pueden representarse mediante una única matriz de transformación de coordenadas homogénea estándar de 4×4.

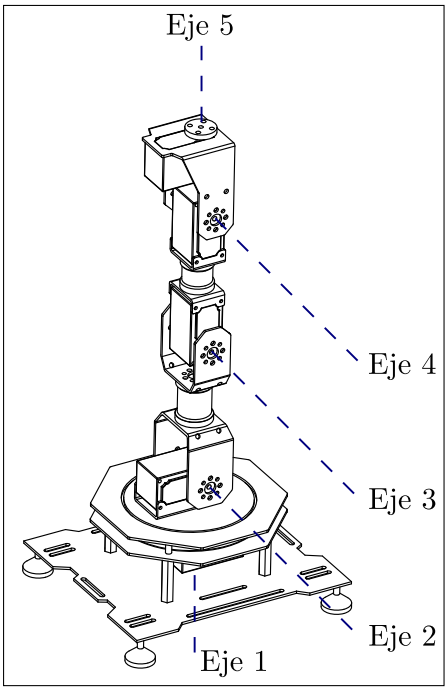


Figura 2.14: Brazo robot Xarm 1S de 5 Grados de libertad.

A continuación aplicamos el algoritmo D-H. Utilizando la Figura 2.15, obtenemos el conjunto de parámetros cinemáticos para el robot xArm 1S que se muestran en la Tabla 2.9. Los θ (ángulo de articulación), d (distancia de articulación), a (longitud de enlace),α (ángulo de torsión de enlace) son parámetros cinemáticos.

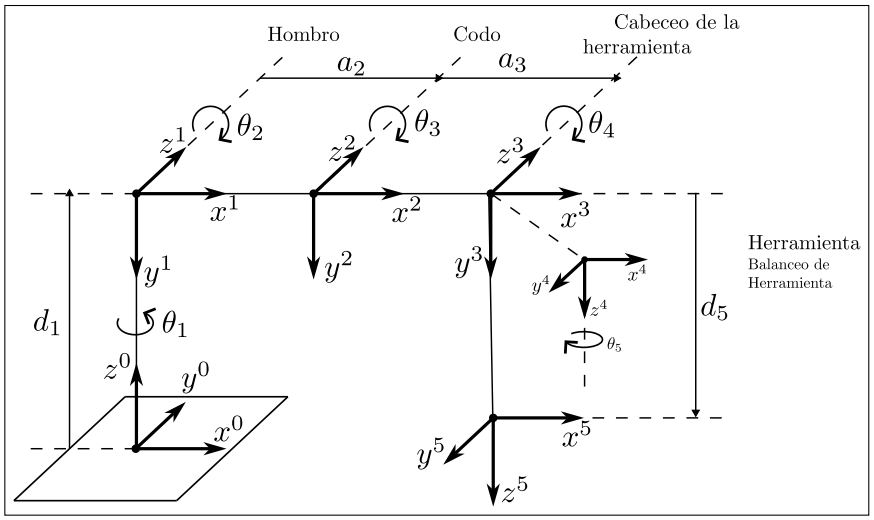


Figura 2.15: Diagrama coordenadas-juntas.

A partir de la Tabla 2.9, utilizamos el algoritmo D − H para establecer las matrices de transformación de enlace. La matriz de transformación de enlace ${}^0T_1$ se define como la expresión de la relación entre el marco de coordenadas 0 y el marco de coordenadas 1, y en los otros es lo mismo.

Tabla 2.9: Parámetros de cinemática.

Ejes	θ_i	d_i(m)	a_i(m)	α_i
1	θ_1	d_1(0,0650)	0	$-\pi/2$
2	θ_2	0	a_2(0,1010)	0
3	θ_3	0	a_3(0,0950)	0
4	θ_4	0	0	$-\pi/2$
5	θ_5	d_5(0,0510)	0	0

Siguiendo esta convención, la matriz de transformación resultante ${}^{i-1}T_i$ del marco de enlace i al marco de enlace precedente $i-1$ se representa como un producto de cuatro transformaciones básicas, dadas por

$$
\begin{aligned}
{}^{i-1}T_i &= R_{z,\theta_i}\,\mathrm{Trans}_{z,d_i}\,\mathrm{Trans}_{x,a_i}\,R_{x,\alpha_i} \\
&= \begin{bmatrix} \mathrm{c}\theta_i & -\mathrm{s}\theta_i & 0 & 0 \\ \mathrm{s}\theta_i & \mathrm{c}\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathrm{c}\alpha_i & -\mathrm{s}\alpha_i & 0 \\ 0 & \mathrm{s}\alpha_i & \mathrm{c}\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdots \\
\cdots &= \begin{bmatrix} \mathrm{c}\theta_i & -\mathrm{s}\theta_i\mathrm{c}\alpha_i & \mathrm{s}\theta_i\,\mathrm{s}\alpha_i & a_i\mathrm{c}\theta_i \\ \mathrm{s}\theta_i & \mathrm{c}\theta_i\mathrm{c}\alpha_i & -\mathrm{c}\theta_i\,\mathrm{s}\alpha_i & a_i\,\mathrm{s}\theta_i \\ 0 & \mathrm{s}\alpha_i & \mathrm{c}\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.1)
\end{aligned}
$$

Donde θ_i es el ángulo de la articulación, a_i es la longitud del eslabón, d_i es

el desplazamiento del eslabón, y α_i es la torsión del eslabón. cθ_i y sθ_i representan $\cos\theta_i$ y $\sin\theta_i$ en esta matriz de transformación D-H, respectivamente. Esta matriz de transformación ${}^{i-1}T_i$ en la ecuación 2.1 puede obtenerse solo utilizando los parámetros D-H θ_i, a_i, d_i y α_i, lo que significa que esta matriz de transformación es fácilmente re-derivada si se aplica un manipulador robótico diferente.

$$
\begin{aligned}
{}^0T_1 &= \begin{bmatrix} \cos(\theta_1) & 0 & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & 0 & \cos(\theta_1) & 0 \\ 0 & -1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^1T_2 &= \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & a_2\cos(\theta_2) \\ \sin(\theta_2) & \cos(\theta_2) & 0 & a_2\sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^2T_3 &= \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & a_3\cos(\theta_3) \\ \sin(\theta_3) & \cos(\theta_3) & 0 & a_3\sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^3T_4 &= \begin{bmatrix} \cos(\theta_4) & 0 & -\sin(\theta_4) & 0 \\ \sin(\theta_4) & 0 & \cos(\theta_4) & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
{}^4T_5 &= \begin{bmatrix} \cos(\theta_5) & -\sin(\theta_5) & 0 & 0 \\ \sin(\theta_5) & \cos(\theta_5) & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{2.2}
$$

Por lo tanto, la matriz de posición del efector final con respecto al sistema de referencia fijo puede expresarse de la siguiente manera

$$
\begin{aligned}
\mathbf{P} = {}^0T_{\text{end}} &= {}^0T_5 = {}^0T_1 \cdot {}^1T_2 \cdot {}^2T_3 \cdot {}^3T_4 \cdot {}^4T_5 \\
&= \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R & p \\ 000 & 1 \end{bmatrix}
\end{aligned}
\tag{2.3}
$$

donde R es la matriz de rotación del efector final y sus coeficientes son,

$$
\begin{aligned}
r_{11} &= \sin(\theta_1)\sin(\theta_5) + \cos(\theta_2+\theta_3+\theta_4)\cos(\theta_1)\cos(\theta_5), \\
r_{12} &= \cos(\theta_5)\sin(\theta_1) - \cos(\theta_2+\theta_3+\theta_4)\cos(\theta_1)\sin(\theta_5), \\
r_{13} &= -\sin(\theta_2+\theta_3+\theta_4)\cos(\theta_1); \\
r_{21} &= \cos(\theta_2+\theta_3+\theta_4)\cos(\theta_5)\sin(\theta_1) - \cos(\theta_1)\sin(\theta_5), \\
r_{22} &= -\cos(\theta_1)\cos(\theta_5) - \cos(\theta_2+\theta_3+\theta_4)\sin(\theta_1)\sin(\theta_5), \\
r_{23} &= -\sin(\theta_2+\theta_3+\theta_4)\sin(\theta_1); \\
r_{31} &= -\sin(\theta_2+\theta_3+\theta_4)\cos(\theta_5), \\
r_{32} &= \sin(\theta_2+\theta_3+\theta_4)\sin(\theta_5), \\
r_{33} &= -\cos(\theta_2+\theta_3+\theta_4)
\end{aligned}
$$

Y $p = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$ es el vector de posición del efector final en coordenadas cartesianas y sus coeficientes son:

$$\begin{aligned}
p_x =& \cos(\theta_1)\ (a_3\ \cos(\theta_2+\theta_3) + a_2\ \cos(\theta_2) - d_5\ \sin(\theta_2+\theta_3+\theta_4)) \\
=& \cos(\theta_1)\ (0{,}095\ \cos(\theta_2+\theta_3) + 0{,}101\ \cos(\theta_2) - 0{,}168\ \sin(\theta_2+\theta_3+\theta_4)) \\
p_y =& \sin(\theta_1)\ (a_3\ \cos(\theta_2+\theta_3) + a_2\ \cos(\theta_2) - d_5\ \sin(\theta_2+\theta_3+\theta_4)) \\
=& \sin(\theta_1)\ (0{,}095\ \cos(\theta_2+\theta_3) - 0{,}168\ \sin(\theta_2+\theta_3+\theta_4) + 0{,}101\ \cos(\theta_2)) \\
p_z =& d_1 - a_3\ \sin(\theta_2+\theta_3) - a_2\ \sin(\theta_2) - d_5\ \cos(\theta_2+\theta_3+\theta_4) \\
=& 0{,}065 - 0{,}095\ \sin(\theta_2+\theta_3) - 0{,}101\ \sin(\theta_2) - 0{,}168\ \cos(\theta_2+\theta_3+\theta_4)
\end{aligned}$$

La cinemática directa del manipulador robótico de escritorio modelo Xarm 1S se deriva en la Ecuación 2.3, en el que si disponemos del vector de articulación dado $q = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \end{bmatrix}^T$ se puede determinar unívocamente el vector de posición del efector final $p = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T$ en el espacio cartesiano.

Solución cinemática inversa

La cinemática inversa se emplea para calcular los ángulos de las articulaciones que permiten al efector final alcanzar una posición y orientación deseada relativa a la base. Este problema, ampliamente estudiado en robótica, ha dado lugar a diversas técnicas matemáticas, especialmente las desarrolladas por Manuka y Kani [21]. En este estudio, se identifican infinitas soluciones que pueden tener las articulaciones para llegar a un punto ubicado dentro del espacio de trabajo [22].

Para aumentar la precisión del control en tiempo real, reducir el error cinemático inverso y disminuir la carga computacional de cálculo, se realiza un procedimiento geométrico sencillo, desarrollado por [23]. Partimos de una orientación inicial del robot, que son las posiciones estándar de la configuración D-H; y de la estructura de la plataforma. Entonces, conociendo las coordenadas del efector final respecto a la base, determinamos los ángulos de las articulaciones (Base, Hombro, Codo y Muñeca, pitch) mediante un análisis geométrico. La Figura 2.16 ilustra las entradas y salidas del primer paso de la cinemática inversa.

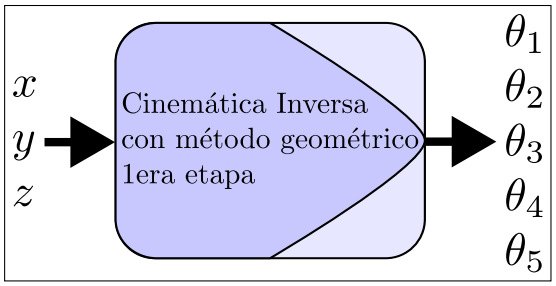


Figura 2.16: Entradas y salidas de la cinemática inversa en la primera etapa

Inicialmente, el cuarto eslabón del robot de brazo se considera hacia abajo, esta suposición impone una limitación en el rendimiento del robot; digamos que el robot no puede cubrir todo el espacio de trabajo. Además, si el robot se utiliza para desplazarse sobre superficies curvas, como la superficie exterior de un depósito, la operación no se realizaría debido a la limitación mencionada. Por lo tanto, aplicando otros dos ángulos de entrada, (Q, P) como se muestra en la Fig. 2.17, como entradas extra de la cinemática inversa, se ha incrementado la capacidad operativa del robot. Q es el ángulo entre la línea normal del efector final relativa a los ejes verticales y P es el ángulo de la pieza desplazada relativa al eje Y_5 en el sistema de coordenadas del efector final. Las entradas de la cinemática inversa pueden variar en el tiempo, lo que mejora el rendimiento del robot para manejar la situación si el destino deseado se actualiza durante la operación. Mediante la introducción de dos nuevas entradas en la cinemática inversa, se solucionan los problemas de tener soluciones múltiples y de no tener soluciones para la cinemática inversa. Estas entradas son definidas por el operador debido a la función esperada del robot de brazo.

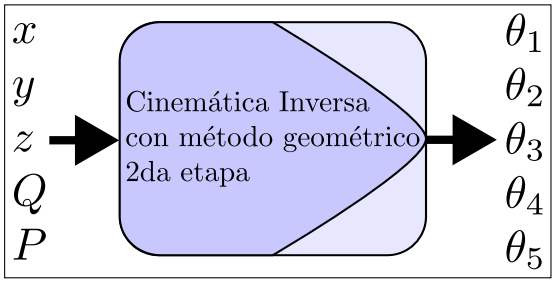


Figura 2.17: Entradas y salidas de la cinemática inversa en la segunda etapa

El problema de la cinemática inversa para el manipulador robótico se ha tratado mediante una solución geométrica. Así, la segunda solución para los ángulos de las articulaciones está dada en las Ecuaciones 2.4-2.10. El ángulo θ_1 se determina con base en la siguiente geometría como:

$$\theta_1 = \arctan\left(\frac{P_y}{P_x}\right) \tag{2.4}$$

Considerando la Figura 2.18, los ángulos Q y P y las coordenadas de destino (P_x, P_y, P_z) como entradas del problema de cinemática inversa, se determinan los ángulos θ_2, θ_3, θ_4.

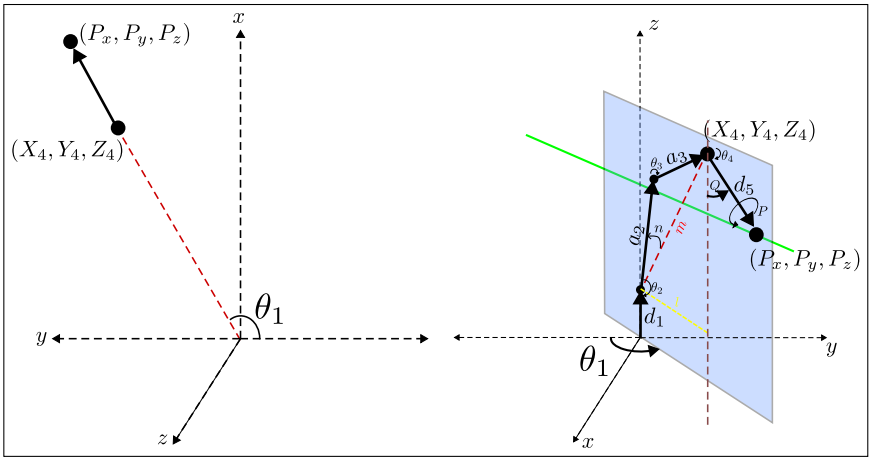


Figura 2.18: Gráfica para el análisis de cinemática inversa.

$$\begin{aligned} X_4 &= P_x - (d_5 \sin(Q)\cos(\theta_1)) \\ Y_4 &= P_y - (d_5 \sin(Q)\sin(\theta_1)) \\ Z_4 &= P_z + (d_5 \cos(Q)) \end{aligned} \tag{2.5}$$

Donde X_4, Y_4, Z_4 son las coordenadas de la posición final del cuarto eslabón. Considerando las siguientes definiciones:

$$\begin{aligned} l &= \sqrt{X_4^2 + Y_4^2} \\ m &= \sqrt{l^2 + Z_4^2} \\ n &= arcos\left(\frac{m^2 + a_2^2 - a_3^2}{2ma_2}\right) \end{aligned} \tag{2.6}$$

Obtenemos:

$$\theta_2 = \begin{cases} -\left(n - \tan^{-1}\left(\dfrac{|Z_4 - d_1|}{l}\right)\right), & \text{si } (Z_4 - d_1) < 0 \\ -\left(\tan^{-1}\left(\dfrac{Z_4 - d_1}{l}\right) + n\right), & \text{en otro caso} \end{cases} \tag{2.7}$$

$$\theta_3 = \pi - \arccos\left(\frac{a_2^2 + a_3^2 - m^2}{2a_2a_3}\right) \tag{2.8}$$

$$\theta_4 = \begin{cases} -\left(\left(\pi - \tan^{-1}\left(\dfrac{l}{|Z_4 - d_1|}\right) + \arccos\left(\dfrac{m^2 + a_3^2 - a_2^2}{2ma_3}\right) + Q\right) - \dfrac{\pi}{2}\right), & \text{si } (Z_4 - d_1) < 0 \\ -\left(\left(\tan^{-1}\left(\dfrac{l}{Z_4 - d_1}\right) + \arccos\left(\dfrac{m^2 + a_3^2 - a_2^2}{2ma_3}\right) + Q\right) - \dfrac{\pi}{2}\right), & \text{en otro caso} \end{cases} \tag{2.9}$$

$$\theta_5 = P \tag{2.10}$$

2.2.1.2 Solución del modelado dinámico del manipulador Xarm 1S

Solución de la dinámica directa

El modelo dinámico del robot utiliza magnitudes tanto cinemáticas como dinámicas. Este puede derivarse a partir de las ecuaciones de Lagrange del segundo tipo, por lo que se empieza obteniendo el lagrangiano del sistema:

$$L = E_{cinetica} - E_{potencial} = (E_{transalacion} + E_{rotacion}) - E_{potencial} \tag{2.11}$$

Sustituyendo cada componente por su expresión:

$$L(q, \dot{q}) = \frac{m}{2}\dot{\xi}^T\dot{\xi} + \frac{1}{2}\gamma^T I\gamma - mgz \tag{2.12}$$

Se encuentra el vector de momentos mediante Euler-Lagrange:

$$F = \tau = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} \tag{2.13}$$

Estas ecuaciones conducen a las ecuaciones de movimiento que pueden escribirse de la siguiente forma

$$M(q, \dot{q})\ddot{q} + C(q, \dot{q}) + G(q) = \tau \tag{2.14}$$

Donde M denota la matriz de inercia, obtenida para el modelo específico del robot y presentada en el Anexo A.4.1; C corresponde a la matriz que agrupa los efectos de Coriolis y centrífugos, detallada en el Anexo A.4.2; y G representa la matriz de términos gravitatorios, mostrada en el Anexo A.4.3.

Solución de la dinámica Inversa

Estas ecuaciones se formulan a partir de matrices de transformación homogénea definidas mediante la cinemática directa, así como de parámetros dinámicos como masas y momentos de inercia, asociados a los marcos de referencia ilustrados. Además, la ecuación de movimiento presentada en 2.14 puede ser reorganizada para despejar la aceleración $\ddot{q}$, obteniéndose la forma expresada en la ecuación 2.15:

$$\ddot{q} = -M^{-1}N\dot{q} - M^{-1}g + M^{-1}\tau \tag{2.15}$$

En estas expresiones, $M = M(q, \dot{q})$ y $N = N(q, \dot{q})$ son matrices que representan los efectos inerciales, mientras que $g = g(q)$ corresponde al vector de fuerzas gravitacionales, cuyos parámetros incluyen longitudes, masas y momentos de inercia del brazo robótico. Por último, τ es el vector de pares aplicados en las articulaciones específicas.

El modelo en 2.15 es conveniente para su transformación en un modelo estándar de espacio de estados lineal, útil para el diseño de controladores. Este modelo puede escribirse como sigue:

$$\begin{aligned}
\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} &= \begin{bmatrix} 0 & I \\ 0 & -M^{-1}N \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} u \\
q &= \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \\
\dot{x} &= A(x)x + Bu \\
y &= Cx + Du
\end{aligned} \tag{2.16}$$

El modelo se complementa con una ecuación algebraica que relaciona los pares articulares τ con un vector auxiliar de acciones de control u, según la ecuación 2.17:

$$\tau = Mu + g \tag{2.17}$$

La introducción del vector de control u y la reorganización del modelo permiten obtener el modelo de espacio de estados lineal presentado anteriormente. La matriz $A(x)$ depende del vector de estado $x = [q, \dot{q}]^T$. Este vector contiene los cinco ángulos articulares y sus velocidades angulares correspondientes al brazo robótico de 5 grados de libertad, mostrado en la Fig. 3, y se define como

$$x = \left[q_1, q_2, q_3, q_4, q_5, \dot{q}_1, \dot{q}_2, \dot{q}_3, \dot{q}_4, \dot{q}_5\right]^T. \tag{2.18}$$

2.2.1.3 Solución del modelo dinámico del actuador

Para facilitar la comprensión del modelado dinámico del actuador, en la Figura 2.19 se presenta el diagrama eléctrico y mecánico simplificado del motor de corriente continua (DC) utilizado en el sistema. Este esquema ilustra los componentes principales del circuito.

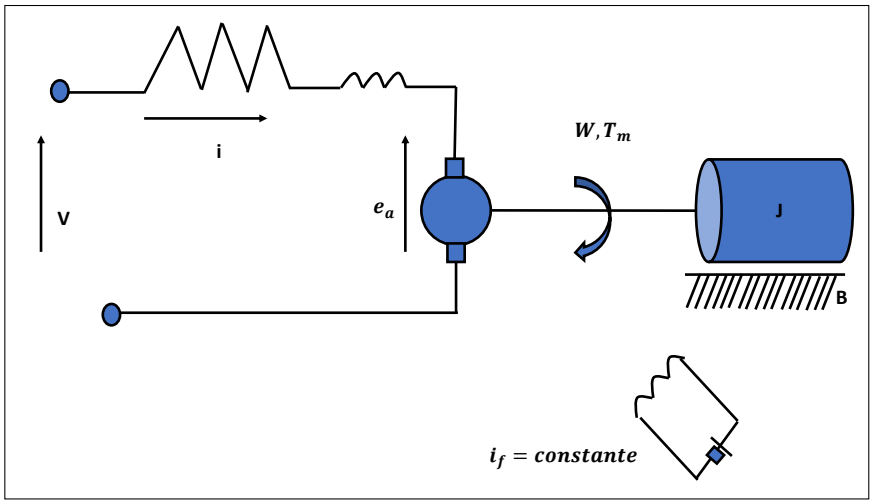


Figura 2.19: Modelado motor DC.

Para obtener el modelo dinámico del motor de corriente continua (DC) utilizado en el brazo robótico, se comienza con el análisis del circuito eléctrico de la armadura. En este circuito se consideran tres elementos principales: la resistencia R, la inductancia L, y la fuerza contraelectromotriz $E_a(t)$ que aparece cuando el rotor está en movimiento. A partir de este análisis se obtiene el siguiente modelo eléctrico:

$$v(t) = Ri(t) + L\frac{di(t)}{dt} + E_a(t) \tag{2.19}$$

$$L\frac{di(t)}{dt} = v(t) - Ri(t) - E_a(t) \tag{2.20}$$

En las ecuaciones (2.19) y (2.20), $v(t)$ representa el voltaje aplicado al motor, $i(t)$ la corriente en la armadura, y $E_a(t)$ la tensión inducida generada por el movimiento del rotor. Estas expresiones describen la dinámica eléctrica del sistema.

A continuación, se modela la parte mecánica del motor. El torque desarrollado se utiliza para vencer las fuerzas de fricción y generar aceleración en la masa giratoria del sistema. Esta dinámica puede representarse de dos formas equivalentes:

$$T_m(t) = J\frac{d\omega(t)}{dt} + B\omega(t) \tag{2.21}$$

$$J\frac{d\omega(t)}{dt} = T_m(t) - B\omega(t) \tag{2.22}$$

La ecuación (2.21) representa la suma de torques aplicada al eje del motor, mientras que la ecuación (2.22) despeja la aceleración angular en función del torque neto disponible. En ambas, $T_m(t)$ es el torque generado por el motor,

J el momento de inercia del sistema, $\omega(t)$ la velocidad angular del motor, y B el coeficiente de fricción viscosa.

Para acoplar los modelos eléctrico y mecánico, se introducen dos relaciones proporcionales. En primer lugar, la fuerza contraelectromotriz se relaciona con la velocidad angular mediante la constante K_a:

$$E_a(t) = K_a\omega(t) \tag{2.23}$$

En segundo lugar, el torque del motor se modela como proporcional a la corriente en la armadura, utilizando la constante de torque K_m:

$$T_m(t) = K_m i(t) \tag{2.24}$$

Las ecuaciones (2.23) y (2.24) permiten acoplar las dinámicas eléctrica y mecánica del motor, para obtener el modelo completo del sistema a controlar. Este modelo es la base para el diseño e implementación del controlador LQG aplicado al seguimiento de trayectoria del brazo robótico.

Funciones de transferencia del motor DC

A continuación, se presentan las ecuaciones del modelo dinámico en el dominio de Laplace, obtenidas a partir de la aplicación de la transformada de Laplace a las ecuaciones diferenciales originales que describen el comportamiento eléctrico y mecánico del motor:

$$Ls \cdot i(s) = v(s) - R \cdot i(s) - E_a(s) \tag{2.25}$$

$$J \cdot s \cdot \omega(s) = T_m(s) - B \cdot \omega(s) \tag{2.26}$$

$$E_a(s) = K_a \cdot \omega(s) \tag{2.27}$$

$$T_m(s) = K_m \cdot i(s) \tag{2.28}$$

Sustituyendo la Ec. (2.24) y la Ec. (2.23) en la Ec.(2.20), se obtiene una nueva relación que permite expresar la tensión de entrada en función del torque:

$$Ls \cdot \left(\frac{T_m(s)}{K_m}\right) = v(s) - R \cdot \left(\frac{T_m(s)}{K_m}\right) - K_a \cdot \omega(s) \tag{2.29}$$

$$v(s) = \left(\frac{R + Ls}{K_m}\right) T_m(s) + K_a \cdot \omega(s) \tag{2.30}$$

Por otro lado, a partir de la ecuación de la dinámica rotacional, es posible obtener una expresión para la velocidad angular del eje del motor como función del par aplicado:

$$\omega(s) = \frac{T_m(s)}{Js + B} \tag{2.31}$$

Reemplazando la ecuación de velocidad angular en la ecuación de voltaje, se obtiene una expresión completa que representa la función de transferencia entre la entrada de voltaje $v(s)$ y el torque desarrollado por el motor:

$$v(s) = \left(\frac{R + Ls}{K_m} + \frac{K_a}{Js + B} \right) T_m(s) \tag{2.32}$$

Multiplicando ambos lados de la ecuación por $(Js+B)$ se obtiene una única expresión racional:

$$v(s) = \frac{(R + Ls)(Js + B) + K_a K_m}{K_m(Js + B)} T_m(s) \tag{2.33}$$

De esta forma se puede despejar la función de transferencia que relaciona la salida en torque con respecto a la entrada en voltaje del motor de corriente continua:

$$\frac{T_m(s)}{v(s)} = \frac{K_m(Js + B)}{LJs^2 + (RJ + LB)s + RB + K_m K_a} \tag{2.34}$$

Esta expresión constituye una de las posibles representaciones del comportamiento dinámico del motor, tomando el torque como variable de salida.

A continuación, se presentan otras funciones de transferencia relevantes del sistema:

- Función de transferencia de la fuerza contraelectromotriz respecto al voltaje de entrada:

$$\frac{E_a(s)}{v(s)} = \frac{K_m K_a}{LJs^2 + (RJ + LB)s + RB + K_m K_a} \tag{2.35}$$

- Función de transferencia de la corriente de armadura respecto al voltaje de entrada:

$$\frac{i(s)}{v(s)} = \frac{Js + B}{LJs^2 + (RJ + LB)s + RB + K_m K_a} \tag{2.36}$$

- Función de transferencia de la velocidad angular respecto al voltaje de entrada:

$$\frac{\omega(s)}{v(s)} = \frac{K_m}{LJs^2 + (RJ + LB)s + RB + K_m K_a} \tag{2.37}$$

Estas expresiones permiten analizar diferentes configuraciones del sistema según la variable de interés en lazo cerrado o en régimen transitorio.

Por otro lado, si se desea obtener la posición angular del eje del motor de corriente continua (DC), basta con considerar que esta es la integral de la velocidad angular. Es decir, puede añadirse un integrador a la función de transferencia de velocidad previamente obtenida. Así, se obtiene la siguiente función de transferencia que relaciona la posición angular $\theta(s)$ con la entrada de voltaje $v(s)$:

$$\frac{\theta(s)}{v(s)} = \frac{K_m}{s\left(LJs^2 + (RJ + LB)s + RB + K_m K_a\right)} \tag{2.38}$$

Una vez obtenidas las funciones de transferencia, es posible formular el modelo dinámico del motor en espacio de estados, lo cual facilita su implementación en simuladores como MATLAB/Simulink y su integración con técnicas modernas de control como el LQG. Se definen los estados del sistema como:

$$\begin{aligned} x_1 &= \omega \quad \text{(velocidad angular)} \\ \dot{x}_1 &= \dot{\omega} \\ x_2 &= i \quad \text{(corriente de armadura)} \\ \dot{x}_2 &= \dot{i} \end{aligned}$$

Reescribiendo el sistema de ecuaciones usando las variables de estado definidas, se tiene:

$$\dot{x}_1 = -\frac{B}{J}x_1 + \frac{K_m}{J}x_2 \tag{2.39}$$

$$\dot{x}_2 = -\frac{K_a}{L}x_1 - \frac{R}{L}x_2 + \frac{1}{L}v \tag{2.40}$$

A partir de las expresiones anteriores, se puede construir la representación en forma matricial del sistema en espacio de estados:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -\frac{B}{J} & \frac{K_m}{J} \\ -\frac{K_a}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} v \tag{2.41}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2.42}$$

La relación de torque con la entrada de control queda de la siguiente manera: $\tau_i(t) = K_m * \mathrm{u}(t)$.

Identificación Dinámica de los Motores DC del Brazo Robótico de 5GDL

Los motores DC son los que generan el movimiento en las articulaciones mediante su salida de torque. Por esta razón, su dinámica tiene un impacto significativo en el modelo dinámico general del brazo robótico de 5 GDL. Para eso se lleva a cabo un proceso de identificación del sistema utilizando las librerías de `Control System Toolbox` de MATLAB, con el objetivo de determinar las constantes dinámicas de los motores DC correspondientes a cada articulación.

La obtención de las señales de salida como la posición angular θ_i se realiza mediante una interfaz de adquisición de datos en Python con el protocolo de comunicación HID (Anexo A.8.5) que incorpora la tarjeta controladora LSC V1.6.

Para obtener la posición a través del tiempo, nos topamos con un problema clásico que existe al momento de medir posición angular con un potenciómetro, resulta que el potenciómetro tiene umbrales de medida, entonces cuando el motor gira a un ángulo fuera de 0° a 240° se desborda la lectura y envía datos erróneos como se observa en la Fig. 2.20. La solución a este problema fue modificar mediante un algoritmo, la lectura de posición, superponiendo los datos válidos hasta obtener una "rampa" que es la señal de salida de la posición angular en motores DC cuando se exita con una señal de entrada $u_6 = 7{,}4\ \mu(t)$ V, que es el voltaje máximo que soportan los motores.

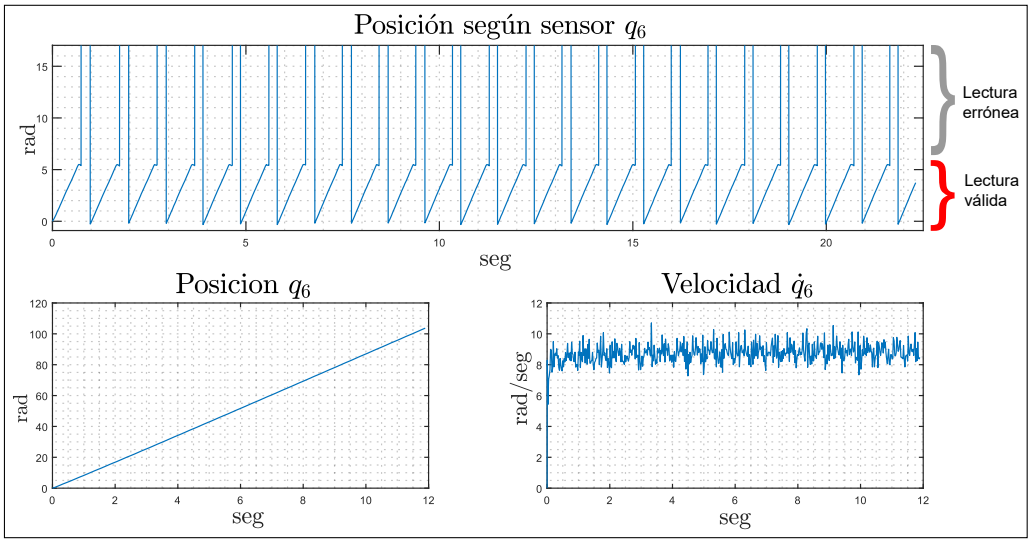


Figura 2.20: Señales de salida del motor q_6, las dos figuras inferiores muestran la señal ajustada.

Para el muestreo de datos se usa un tiempo de $T_s = 21{,}59\ [ms]$, que es

lo que tarda la placa LSC en recoger información del motor. La identificación del sistema en espacio de estados, se realiza utilizando el comando `ssest()` perteneciente a la System Identification Toolbox de MATLAB. Esta función permite estimar un modelo en representación de espacio de estados a partir de datos experimentales en el dominio del tiempo. En este caso, seleccionamos una representación en forma canónica observable, lo que garantiza que todas las salidas del sistema sean observables a partir de los estados internos. En la Figura 2.21 se muestra el ajuste del modelo estimado frente a los datos reales, expresado como un porcentaje que indica el nivel de concordancia entre ambos.

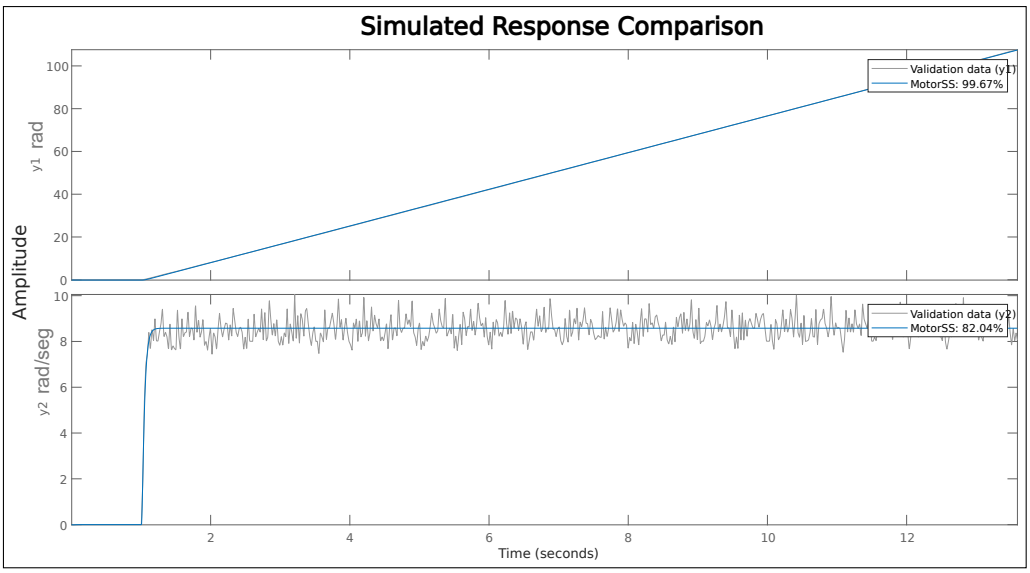


Figura 2.21: Comparación entre q_6 y $\hat{q}_6$.

Este procedimiento de identificación del modelo se aplica de manera uniforme a los motores de todas las articulaciones, considerando condiciones sin carga externa. Aunque durante la identificación se trabaja con articulaciones desacopladas, en operación real las inercias de los demás eslabones generan torques de acoplamiento que actúan como perturbaciones sobre el sistema. Estos efectos dinámicos se modelan mediante el método recursivo de Newton–Euler, el cual permite estimar los torques articulares asociados a los movimientos del robot. A partir de este modelo, un observador de estados se encarga de reconstruir las variables internas del sistema, y el controlador LQR aplica la acción correctiva correspondiente para compensar las perturbaciones e inducir el comportamiento deseado.

Matrices de Espacio Estado de cada uno de los motores

Motor q_6:

$$
\begin{aligned}
A_6 &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 2{,}91 \times 10^{-10} & -500 & 500 & 1 \\ 0 & 0 & 0 & 1 \\ 9{,}403 \times 10{-}12 & -16{,}34 & -2399 & -110{,}7 \end{bmatrix} \\
B_6 &= \begin{bmatrix} -1{,}034 \times 10^{-10} \\ -4{,}748 \\ -4{,}748 \\ 3326 \end{bmatrix} \\
C_6 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}
\end{aligned}
\tag{2.43}
$$

Motores q_1, q_3,q_4, q_5 (Asumimos que tienen el mismo comportamiento):

$$
\begin{aligned}
A_{1,3,4,5} &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ -9{,}403 \times 10^{-11} & -500 & & 5001 \\ 0 & 0 & 0 & 1 \\ -7{,}443 \times 10^{-10} & -3961 & 1546 & -110{,}7 \end{bmatrix} \\
B_{1,3,4,5} &= \begin{bmatrix} 4{,}188 \times 10^{-9} \\ -6{,}909 \times 10^{-9} \\ 9{,}704 \times 10^{-12} \\ 2031 \end{bmatrix} \\
C_{1,3,4,5} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}
\end{aligned}
\tag{2.44}
$$

Motor q_2 :

$$
\begin{aligned}
A_2 &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 4{,}625 \times 10^{-10} & -500 & 500 & 1 \\ 0 & 0 & 0 & 1 \\ -4{,}85 \times 10^{-11} & 48{,}04 & -2464 & -110{,}7 \end{bmatrix} \\
B_2 &= \begin{bmatrix} -1{,}663 \times 10^{-9} \\ -3{,}385 \\ -3{,}385 \\ 2615 \end{bmatrix} \\
C_2 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}
\end{aligned}
\tag{2.45}
$$

2.2.1.4 Metodología para la generación de trayectorias

La generación de trayectorias para el brazo robótico se desarrolló a través de una metodología que combina herramientas de diseño gráfico, simulación CNC y procesamiento matemático. Esta estrategia permite definir trayectorias personalizadas en el espacio cartesiano y traducirlas en coordenadas articulares compatibles con el modelo cinemático del sistema.

Como primera etapa, se utilizó el software *Inkscape* para la creación de figuras vectoriales que representan las trayectorias deseadas. Dicha herramienta permite diseñar contornos precisos sobre un lienzo virtual, facilitando la

edición gráfica de trayectorias en dos dimensiones. Posteriormente, mediante la extensión *Gcodetools*, se procedió a la conversión automática de estos contornos en instrucciones *G-Code*, el cual es un lenguaje de control numérico ampliamente utilizado en sistemas de manufactura.

Una vez generado el archivo *.gcode*, se realizó su validación visual en la plataforma *NC Viewer*. Esta herramienta proporciona una simulación gráfica del movimiento que seguiría una herramienta CNC a partir de las instrucciones contenidas en el código, permitiendo verificar la continuidad, dirección y correcta interpretación de la trayectoria especificada.

La tercera etapa consistió en la conversión del código G en una nube de puntos discretos mediante rutinas desarrolladas en *MATLAB*. Para ello, se empleó una función de lectura de código G que traduce cada instrucción en coordenadas cartesianas tridimensionales. Este conjunto de puntos se almacenó en un vector ordenado que representa la secuencia de posiciones a seguir por el efector final. La densidad de puntos se controló mediante un parámetro de discretización, con el fin de ajustar la resolución espacial de la trayectoria.

La nube de puntos obtenida sirve como entrada para los algoritmos de cinemática inversa desarrollados en etapas posteriores. A partir de estas posiciones cartesianas, se calcularon las configuraciones articulares necesarias para que el brazo robótico siga con precisión la trayectoria planeada. Esta metodología proporciona una herramienta flexible y eficiente para la generación de trayectorias arbitrarias, superando las limitaciones impuestas por trayectorias analíticas convencionales.

2.2.2 Diseño del servo-controlador (LQG)

A continuación, se presenta el desarrollo del procedimiento de diseño para el controlador LQG (Linear Quadratic Gaussian), el cual combina una realimentación óptima de estados mediante la técnica LQR con un estimador de estados basado en el filtro de Kalman.

2.2.2.1 Solución del diseño del Regulador LQR

La dinámica de las articulaciones individuales del sistema lineal discreto en espacio de estados es de dimensión cuatro:

$$x_{k+1} = A\,x_k + B\,u_k, \qquad x_k \in \mathbb{R}^4,\ u_k \in \mathbb{R}^m, \tag{2.46}$$

donde $A \in \mathbb{R}^{4\times4}$ y $B \in \mathbb{R}^{4\times m}$.

Se define la función de costo cuadrático infinita:

$$J = \sum_{k=0}^{\infty} \left(x_k^\top Q x_k + u_k^\top R u_k \right), \tag{2.47}$$

con $Q = Q^\top \succeq 0 \in \mathbb{R}^{4\times4}$ y $R = R^\top \succ 0 \in \mathbb{R}^{m\times m}$.
El problema óptimo consiste en encontrar la ley de realimentación de estado:

$$u_k = -K\, x_k \tag{2.48}$$

que minimiza la función de costo J.

La matriz de ganancias $K \in \mathbb{R}^{m\times4}$ se obtiene mediante la solución $P \in \mathbb{R}^{4\times4}$ de la ecuación algebraica de Riccati discreta:

$$P = A^\top P A - A^\top P B \left(R + B^\top P B \right)^{-1} B^\top P A + Q. \tag{2.49}$$

Para un sistema de orden cuatro, esta ecuación implica resolver un conjunto no lineal de 10 ecuaciones en las componentes de P. En la práctica, se resuelve iterativamente la "ecuación de Riccati inversa":

$$P_{i+1} = A^\top P_i A - A^\top P_i B \left(R + B^\top P_i B \right)^{-1} B^\top P_i A + Q, \tag{2.50}$$

iniciando con $P_0 = Q$ y repitiendo hasta convergencia numérica.

Una vez hallada la solución estacionaria P, la ganancia óptima queda dada por:

$$K = \left(R + B^\top P B \right)^{-1} B^\top P A. \tag{2.51}$$

Bajo este diseño, el sistema en lazo cerrado queda descrito por:

$$x_{k+1} = (A - BK)\, x_k, \tag{2.52}$$

y la elección de Q y R permite ajustar el compromiso entre rapidez de respuesta (modulando los autovalores de $A-BK$) y minimización del esfuerzo de control.

Este enfoque garantiza un balance óptimo, en sentido cuadrático, entre la amortiguación de los estados y el uso de energía de control para sistemas de cuarto orden.

2.2.2.2 Solución del diseño del filtro de Kalman

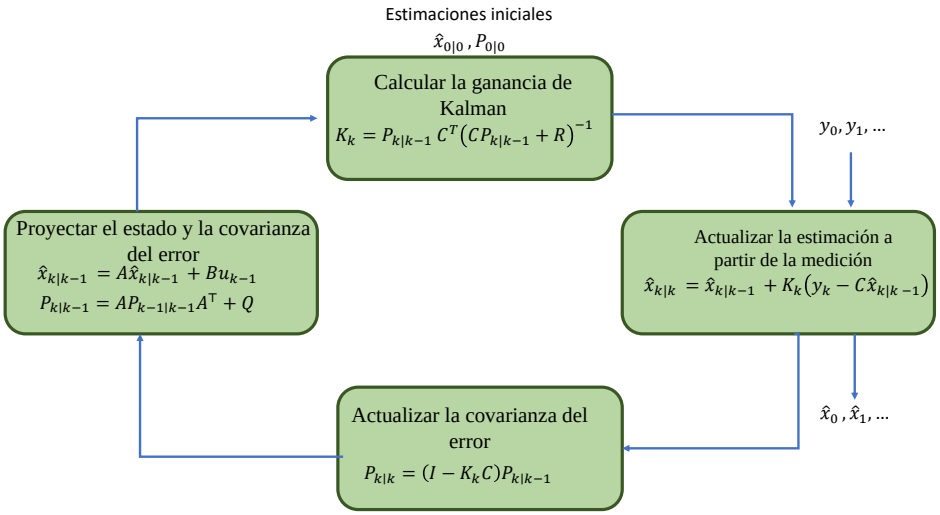


Figura 2.22: Algoritmo de Kalman para la predicción de los estados.

Con el objetivo de estimar el estado de un sistema lineal discreto de cada uno de los motores identificados en el apartado anterior representado en espacio de estados, se implementa el filtro de Kalman como una función, en donde la estructura del sistema considerado es de la forma:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k, \\ y_k &= Cx_k + v_k, \end{aligned} \tag{2.53}$$

donde $x_k \in \mathbb{R}^4$ representa el vector de estados, u_k la entrada de control, y_k la salida medida, w_k el ruido de proceso, y v_k el ruido de medición, con distribuciones $w_k \sim \mathcal{N}(0, Q)$ y $v_k \sim \mathcal{N}(0, R)$.

La función implementada se ejecuta en cada instante de muestreo de $T_s =$ 4 ms menor al retardo de la obtención de datos de 21,59 ms y realiza las etapas clásicas del filtro de Kalman: predicción del estado y la covarianza, cálculo de la ganancia de Kalman, corrección del estado estimado y actualización de la covarianza. Se hace uso de variables persistentes para conservar el estado estimado y la matriz de covarianza entre ejecuciones consecutivas del bloque.

El algoritmo desarrollado para el predictor sigue la siguiente lógica:

Inicialmente, se definen las matrices del sistema (A, B, C), las matrices de covarianza del ruido (Q y R), y las condiciones iniciales del filtro: $\hat{x}_{0|0}$ y $P_{0|0}$. En cada ciclo de simulación, se calcula la predicción del estado como:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_{k-1}, \tag{2.54}$$

y la predicción de la covarianza del error de estimación:

$$P_{k|k-1} = AP_{k-1|k-1}A^\top + Q. \tag{2.55}$$

Posteriormente, se calcula la ganancia de Kalman:

$$K_k = P_{k|k-1}C^\top (CP_{k|k-1}C^\top + R)^{-1}, \tag{2.56}$$

que pondera la confianza entre la predicción del modelo y la medición disponible.

Con esta ganancia se actualiza la estimación del estado:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1}), \tag{2.57}$$

y se corrige la covarianza de la estimación mediante:

$$P_{k|k} = (I - K_kC)P_{k|k-1}. \tag{2.58}$$

Finalmente, se almacenan los valores actualizados de $\hat{x}_{k|k}$ y $P_{k|k}$ en variables persistentes para ser utilizados en la siguiente iteración del filtro.

Este algoritmo proporciona como salidas la estimación actualizada del estado, la covarianza correspondiente, la ganancia de Kalman y la salida estimada del sistema $\hat{y}_k = C\hat{x}_{k|k-1}$. Esta implementación modular permite su integración en lazo cerrado del controlador LQR, además de comparaciones con el sistema real.

En la Fig. 2.23 se muestra este algoritmo que se implementa en todos los motores con el propósito de estimar el estado de la posición y la velocidad angular. Las matrices A_k, B_k, $C_k$deben ser las obtenidas en la linealización. Dado que el tiempo de muestreo puede representar una limitación, especialmente cuando no se dispone de datos actualizados en tiempo real, el algoritmo permite predecir el estado del sistema durante los intervalos en los que no se cuenta con mediciones disponibles. En su implementación en Python, se recurre al uso de hilos de ejecución, donde un núcleo del procesador se encarga de llevar a cabo el muestreo de datos con su correspondiente retardo, mientras que otro núcleo realiza la estimación del estado en tiempos discretos definidos por el instante de muestreo k. Esta estrategia busca mitigar los efectos del fenómeno conocido como "jitter", el cual se refiere a la variabilidad en el

tiempo de ejecución o de adquisición de datos, y que puede afectar la precisión y estabilidad del sistema estimador.

```
function [x_est, P_est, K_out, y_pred] = kalman_step(u, y)
%#codegen
% --- Entradas:
% u : señal de control (m×1)
% y : medida (p×1)
%
% --- Salidas:
% x_est : estado estimado k|k (n×1)
% P_est : covarianza estimada k|k (n×n)
% K_out : ganancia de Kalman K_k (n×p)
% y_pred: salida predicha C*x_{k|k-1} (p×1)

  persistent x_prev P_prev
  if isempty(x_prev)
    % Inicialización (ajusta n, p, valores)
    n = 4; p = size(y,1);
    x_prev = zeros(n,1);          % \hat x_{0|0}
    P_prev = eye(n);              % P_{0|0}
  end

  % --- Definición de parámetros del modelo ( A, B, C, Q, R)
  A = [ 1.000000000000000   0.001729230156138   0.002267289105002   0.000007705802970;
       -0.000000000000110   0.135270910833703   0.862142979268489   0.003780538254553;
       -0.000000000000000  -0.000064372402886   0.997478262511243   0.003780538255806;
       -0.000000000000003  -0.023748697975554  -1.245020749188313   0.891451609989034];   % 4×4

  B = [ -0.000005962927517;
          0.000357330468189;
          0.000357330476269;
          5.008793332656738];   % 4×m

  C =[1     0     0     0;
      0     0     1     0];   % p×4

  Q = 1e-6*eye(4); % 4×4, ruido de proceso
  %R = 1e-6*eye(2);
  R = [ 0.021642153615105  -0.031419400508566;
       -0.031419400508566   0.881067523192092];   % p×p, ruido de medición

  % 1) Predicción
  x_pred = A*x_prev + B*u;
  P_pred = A*P_prev*A' + Q;

  % 2) Ganancia de Kalman
  S = C*P_pred*C' + R;
  K   = P_pred*C' / S;

  % 3) Corrección
  y_pred = C*x_pred;
  innov  = y - y_pred;
  x_est  = x_pred + K*innov;
  P_est  = (eye(size(A)) - K*C) * P_pred;

  % 4) Guardar para el siguiente paso
  x_prev = x_est;
  P_prev = P_est;

  % 5) Salida de la ganancia
  K_out = K;
end
```

Figura 2.23: Algoritmo de Kalman para la predicción de los estados.

2.2.3 Implementación del hadware

En este apartado se describe la arquitectura física del sistema de control implementado para el brazo robótico Xarm 1S. La configuración se basa en el uso de un controlador de servomotores, el cual se encuentra integrado en la base del propio manipulador. Esta arquitectura permite establecer una comunicación directa y eficiente entre el sistema de control y los actuadores, posibilitando un control preciso del manipulador.

La Figura 2.24 muestra la estructura general del sistema, en la que se ob-

serva la conexión entre el sistema de control y el controlador de servomotores, realizada mediante una interfaz serial que utiliza los pines RX, TX, GND y VCC. Esta conexión permite la transmisión continua y confiable de las señales de control hacia los distintos motores que componen el brazo robótico.

El controlador LSC-6 es responsable de distribuir las instrucciones de movimiento a cada uno de los servomotores digitales que conforman el manipulador. Cada servo se encuentra conectado a un puerto específico del controlador y es identificado mediante un ID único, lo cual permite el envío de órdenes individualizadas y sincronizadas. Esta estructura asegura la ejecución precisa y coordinada de los movimientos, manteniendo la alineación adecuada entre todos los grados de libertad del sistema.

Adicionalmente, el controlador es compatible con el protocolo HID, lo que facilita su integración con computadoras personales u otros dispositivos embebidos. Esta característica proporciona mayor flexibilidad para el desarrollo de pruebas, configuraciones y posibles extensiones futuras del sistema de control.

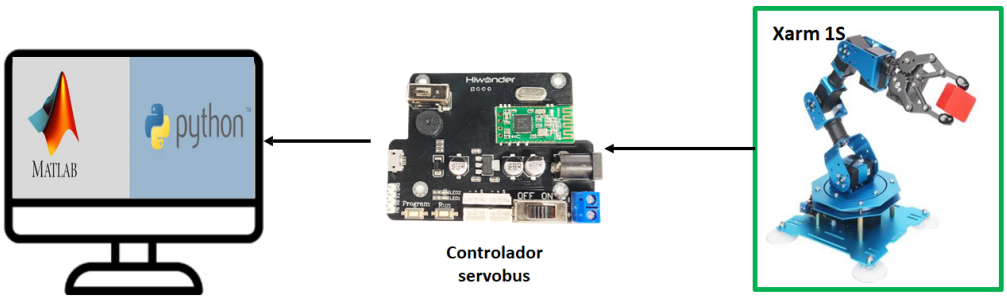


Figura 2.24: Arquitectura del sistema de control del brazo robótico xArm 1S.

2.2.3.1 Conexiones de dispositivos con el manipulador xArm 1S

La Figura 2.25 presenta la distribución física de las conexiones en el sistema. Los servomotores digitales del xArm 1S se conectan directamente a los puertos laterales del controlador LSC-6, lo que permite una administración eficiente tanto de la energía como de las señales de control. El sistema se alimenta mediante una batería Li-poly de 7.4 V, la cual proporciona la potencia necesaria para operar todos los actuadores de manera estable y continua.

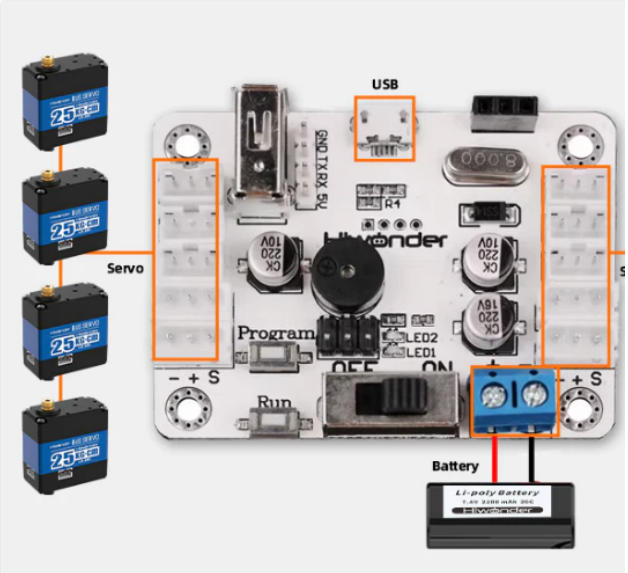


Figura 2.25: Conexión de dispositivos al controlador. Fuente: [61].

2.2.3.2 Área de trabajo

Con el objetivo de garantizar que el manipulador robótico opere de forma eficiente dentro de su entorno físico, se realiza un análisis detallado del espacio de trabajo tridimensional y de las dimensiones de la plataforma donde está instalado. Este análisis permite establecer los límites máximos de alcance del efector final en función de la geometría del sistema, así como verificar la compatibilidad entre la zona operativa del robot y el área disponible en la plataforma de soporte; que para este proyecto es el lienzo donde dibuja la trayectoria deseada.

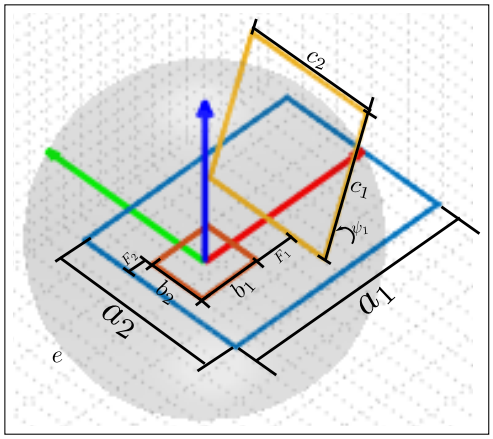
Análisis del espacio de trabajo y de la plataforma física

Los cálculos son desarrollados en MATLAB, mediante el modelo cinemático directo obtenido en la Ecuación. 2.3 donde se determina la posición del efector final en función de las posiciones articulares. A partir de estos datos, se genera una representación volumétrica del espacio de trabajo, lo que permite visualizar las regiones accesibles por el manipulador. Adicionalmente, se establecieron las dimensiones mínimas requeridas para la plataforma física, que se observan en la Figura 2.26, de manera que el efector final alcance el lienzo y pueda mostrar la trayectoria en tiempo real.

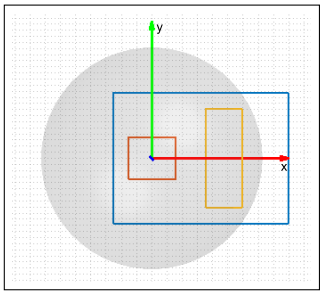
A continuación se muestra en la Tabla. 2.10 las dimensiones del soporte y del área de trabajo:

Tabla 2.10: Dimensiones del espacio de trabajo.

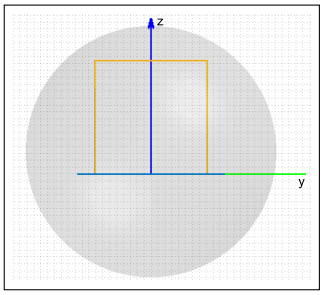
Parámetros	Dimensiones	Objeto
a_1	575 [mm]	Base de Plataforma
a_2	429 [mm]	
b_1	140 [mm]	Base de Robot
b_2	154 [mm]	
c_1	350 [mm]	Lienzo
c_2	325 [mm]	
F_1	100 [mm]	Distancia x entre la base del robot y la base.
F_2	50 [mm]	Distancia x entre la base del robot y el lienzo.
esf	radio= 364 [mm] centro$_{(x,y,z)}$ = $(0, 0, 65)$ [mm]	Espacio de trabajo del robot
ψ	70 [°]	Inclinación del lienzo



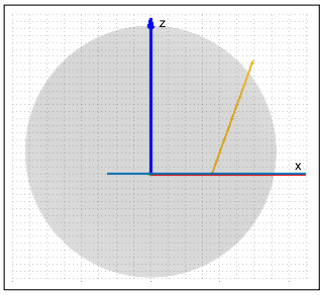
(a) Vista isométrica.



(b) Vistas superior.



(c) Vistas frontal.



(d) Vista lateral.

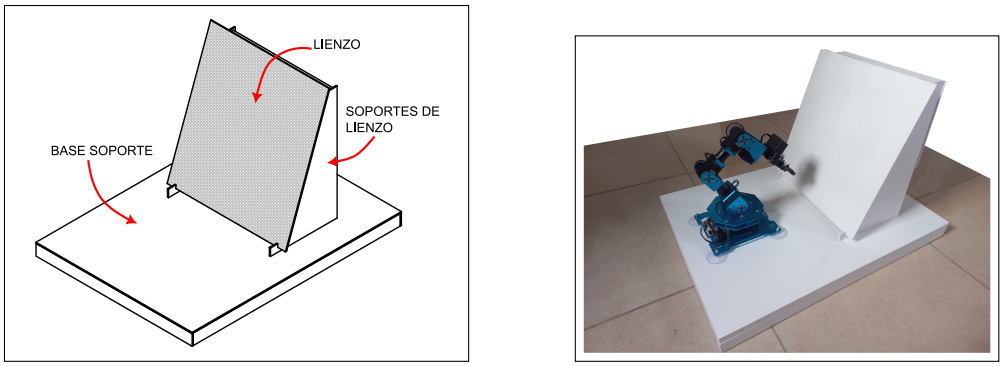
Figura 2.26: Área y plataforma de trabajo.

Diseño de la plataforma soporte

Se diseñó y construyó una plataforma estructural para el entorno de trabajo del manipulador, utilizando el software CAD FreeCAD para modelar con precisión sus componentes cuyas dimensiones significativas son calculadas en MATLAB y se muestran en la Tabla. 2.10. La plataforma consta de tres elementos: una base, un soporte, soportes inclinados para el lienzo y un lienzo de trabajo; este último representa el área efectiva donde el manipulador ejecuta las tareas de trazado de trayectoria o manipulación superficial. La base pro-

porciona estabilidad, los soportes fijan el lienzo en un ángulo óptimo de 70° para el manipulador, y el lienzo es la superficie donde se ejecutan las tareas.

Fabricada en madera contrachapada de 1/8" por su ligereza, rigidez y facilidad de mecanizado, la estructura modular permite transporte, montaje y desmontaje eficiente, ideal para pruebas y validaciones. Su diseño garantiza que el manipulador opere sin interferencias dentro del área activa; además, mantiene el enfoque de este proyecto, al tratarse de un elemento que permite comprobar el seguimiento de trayectoria.



(a) Diseño de FreeCAD. (b) Implementación Real.

Figura 2.27: Plataforma Soporte.

2.2.4 Implementación del software

En este apartado se presentará la implementación del software para el control de trayectoria del brazo robótico utilizando un controlador LQG. Para ello, se generará la trayectoria de referencia que el brazo deberá seguir, y el algoritmo se desarrollará inicialmente en el entorno de MATLAB, lo que permitirá corregir errores y ajustar sus parámetros. Posteriormente, se adaptará a Python para ejecutarse en un entorno computacional adecuado, donde se calcularán las señales de control que se enviarán al Arduino Uno mediante comunicación serial. El Arduino interpretará estas señales y las transmitirá al controlador Servo Bus, encargado de accionar los servomotores del brazo. Esto permitirá aplicar el control de manera precisa y confiable en el sistema físico. Además, se utilizará App Designer, una herramienta que cuenta con funcionalidades para crear una interfaz gráfica que facilitará la visualización del comportamiento del brazo durante el seguimiento de la trayectoria.

La Figura 2.28 muestra el diagrama de bloques del sistema de control implementado para el brazo robótico. Dicho sistema emplea un controlador LQG desarrollado en Python, el cual recibe las posiciones deseadas desde una interfaz remota diseñada en AppDesigner de MATLAB, y genera como salida

señales de control articulares $\mathbf{q}_i$ o señales PWM_i. Estas señales se transmiten al módulo LSC V1.2 mediante una conexión USB, y posteriormente son enviadas al sistema de actuadores del brazo robótico a través de comunicación UART. De este modo, el controlador transforma las coordenadas cartesianas deseadas en comandos precisos que permiten ejecutar los movimientos con exactitud.

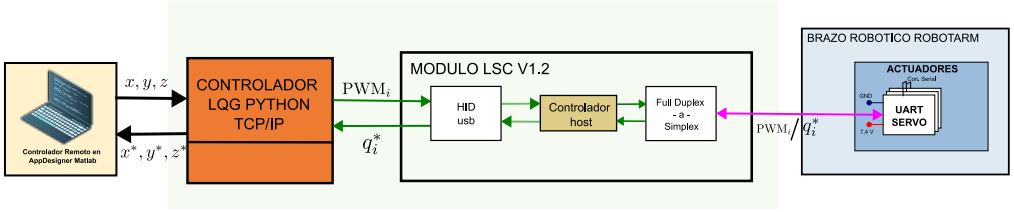


Figura 2.28: Esquema de comunicación.

2.2.4.1 Soluciones matemáticas simuladas en MATLAB

En esta sección se presentan los códigos desarrollados en MATLAB que conforman una parte fundamental del proceso de simulación del brazo robótico de cinco grados de libertad. Estos códigos permiten generar trayectorias tridimensionales, calcular en tiempo real la solución matemática de la cinemática inversa para cada punto de la trayectoria, y visualizar el comportamiento del robot en un entorno gráfico. La implementación se organiza en tres bloques principales que siguen un flujo lógico: generación de puntos en el espacio, cálculo de configuraciones articulares y simulación del movimiento. A través de este enfoque, se traduce el movimiento deseado en coordenadas cartesianas hacia comandos articulares que el modelo del robot puede interpretar y ejecutar, facilitando así el análisis del desempeño cinemático y dinámico del sistema.

Generación de trayectoria y transformación de puntos en el espacio tridimensional

La trayectoria del efector final se define mediante una secuencia de puntos organizados para formar un recorrido cerrado en el plano YZ. Esta trayectoria se genera mediante interpolación lineal (`G01`), que construye segmentos rectilíneos con resolución controlada por el parámetro `N`, garantizando una representación continua del movimiento.

```
clc; clear; close all;
N = 20;
points = [];

points = [points; 0.0, 0, 0.10];
points = G01(0.0, 0.10, 0.10, points, N);
points = G01(0.0, 0.10, 0, points, N);
points = G01(0.0, 0, 0, points, N);
points = G01(0.0, 0, 0.10, points, N);


points = G01(0.0, -0.10, 0.10, points, N);
points = G01(0.0, -0.10, 0, points, N);
points = G01(0.0, 0, 0, points, N);

%%GRAFICADOR

%Gráfica original
%_____________________________
    hold on
    %plot3(points(:,1), points(:,2), points(

%Ubicacion en el lienzo
zl = 0.190;
%zl = 0.02;
yl = 0;

```

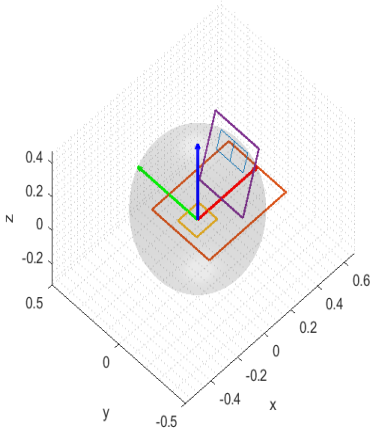


Figura 2.29: Puntos en el espacio tridimensional.

Posteriormente, los puntos definidos en el espacio base son transformados al sistema de referencia del robot mediante una matriz homogénea $T \in \mathbb{R}^{4\times4}$, que incluye una rotación alrededor del eje Y por un ángulo β y una traslación representada por el vector $\mathbf{t}$. Esta transformación asegura la correcta alineación espacial de la trayectoria con el entorno físico del manipulador.

Para validar visualmente la trayectoria, se realiza una representación tridimensional que incluye la estructura física del robot, ejes coordenados y una esfera translúcida que delimita el espacio de trabajo efectivo, definido por los parámetros geométricos del manipulador. Esta visualización facilita la comprobación de que la trayectoria se encuentra dentro de las restricciones físicas del sistema.

La aplicación precisa de la transformación homogénea es fundamental, no solo para la correcta visualización, sino también para permitir la implementación subsecuente de algoritmos de cinemática inversa y simulación dinámica. El código completo correspondiente a esta sección se encuentra disponible en los anexos del presente documento.

Cálculo de ángulos articulares mediante cinemática inversa y análisis temporal

El código implementa el cálculo de las posiciones articulares de un manipulador robótico a partir de una serie de puntos definidos en el espacio tri-

dimensional, aplicando principios de cinemática inversa. Para cada punto, se determina la distancia euclidiana respecto al origen ajustada a la geometría del robot, considerando parámetros físicos d_1, a_2, a_3 y d_5.

Se define una relación lineal entre la distancia al punto y un ángulo Q que actúa como parámetro auxiliar para el cálculo articular, con límites definidos por grados máximos y mínimos. Se verifica la alcanzabilidad de cada punto dentro del rango operacional del robot; en caso contrario, se emite una advertencia y se omite el punto.

El cálculo de los ángulos articulares θ_1, θ_2, θ_3 y θ_4 se realiza mediante funciones trigonométricas inversas, aplicando correcciones para garantizar la validez de los argumentos en las funciones `acos` y adaptando el signo según la posición relativa del punto respecto a la base del manipulador. La rotación de la muñeca se mantiene fija en cero.

```
%% Validación de punto alcanzable
if DISTANCIA_PUNTO > DISTANCIA_MAX || DISTANCIA_PUNTO < DISTANCIA_MIN
    warning('Punto fuera del alcance del robot: [%f, %f, %f]', Px, Py, Pz);
    continue;
end

%% INICIA CALCULO
theta1 = atan2(Py, Px);

X4 = Px - d5 * sin(Q) * cos(theta1);
Y4 = Py - d5 * sin(Q) * sin(theta1);
Z4 = Pz + d5 * cos(Q);

l = sqrt(X4^2 + Y4^2);
m = sqrt(l^2 + (Z4 - d1)^2);

% Corrección de dominio para acos
arg_n = min(max((m^2 + a2^2 - a3^2) / (2 * m * a
n = acos(arg_n);

if (Z4 - d1) < 0
    theta2 = -(n - atan(abs(Z4 - d1) / l));
else
    theta2 = -(atan((Z4 - d1) / l) + n);
end

```

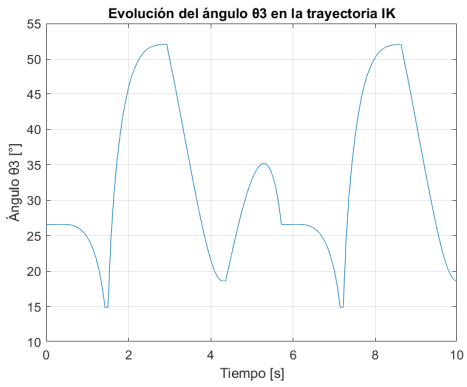


Figura 2.30: Trayectoria de una articulación usando IK.

Finalmente, se grafica la evolución temporal del ángulo θ_3 durante la trayectoria, utilizando un vector de tiempos equiespaciados para visualizar el comportamiento dinámico de esta articulación. Este análisis es esencial para validar el desempeño cinemático y asegurar la factibilidad del movimiento.

El código completo correspondiente a este proceso se encuentra disponible en los anexos del documento para su revisión y aplicación detallada.

Simulación y visualización de la trayectoria articular mediante modelado DH y cinemática directa

El código establece el modelo cinemático de un brazo robótico de cinco grados de libertad utilizando la parametrización Denavit-Hartenberg (DH). Cada eslabón se define mediante sus parámetros DH específicos: desplazamiento d, longitud a y ángulo α. Con estas definiciones se crea un objeto `SerialLink` que representa el manipulador.

Se inicializa una trayectoria articular vacía y se define una posición inicial q_0 con todos los ángulos en cero radianes. A partir de las posiciones articulares calculadas previamente (almacenadas en la matriz `rotaciones`), se generan trayectorias interpoladas mediante la función `jtraj` que calcula transiciones suaves entre puntos articulares en intervalos temporales discretos.

```
% Crear el brazo robótico
robot = SerialLink([L1 L2 L3 L4 L5], 'name', 'Brazo 5DOF X
% Definir posiciones de las articulaciones en radianes
q0 = [0 0 0 0 0];           % Posición inicial
%q1 = [-pi/2 0 0 0 0];

i=1;
q1 = rotaciones(i,:); % Posición intermedia
% % Crear una trayectoria interpolada entre los puntos
  t = 0:0.5:1; % Intervalo de tiempo
 [q, ~, ~] = jtraj(q0, q1, t); % Trayectoria de q0 a q1


% Simulación y visualización del robot
% figure;
robot.plot(q, 'trail', {'r', 'LineWidth', 2});
xlabel("x[m]");ylabel("y[m]");zlabel("z[m]");
% hold on
for i=2:col
% Crear una trayectoria interpolada entre los puntos
 t = 0:1:2; % Intervalo de tiempo
 [q, ~, ~] = jtraj(rotaciones(i-1,:), rotaciones(i,:), t); % Trayectoria de q0 a q1

  q_trayectoria = [q_trayectoria; q];
end
%
% Simulación y visualización del robot
% figure;
robot.plot(q_trayectoria, 'trail', {'r', 'LineWidth', 2});

```

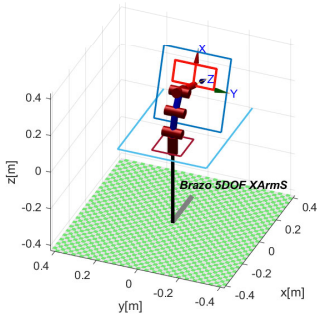


Figura 2.31: Simulación y visualización de la trayectoria articular

La simulación visualiza el movimiento del robot a lo largo de toda la trayectoria interpolada, mostrando la evolución de las articulaciones y el rastro generado por el efector final en el espacio tridimensional. Además, se calcula la posición del efector final mediante la función de cinemática directa `fkine` aplicada a la trayectoria articular completa.

Para contextualizar el movimiento, se grafica la trayectoria espacial deseada junto con la estructura física del manipulador, cargada desde datos externos, y se ajustan las etiquetas y vistas para facilitar el análisis visual. Esta simulación es crucial para validar la factibilidad y precisión del movimiento planificado.

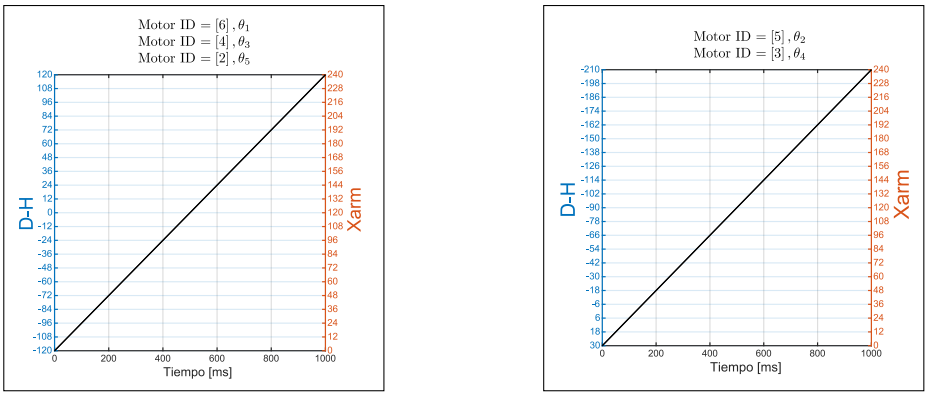
El código fuente completo y detalles adicionales de esta implementación se encuentran disponibles en los anexos del presente documento.

2.2.4.2 Generación y simulación de trayectorias vectoriales

En esta sección se presenta el proceso de generación y simulación de trayectorias para el efector final del manipulador robótico. Inicialmente, se realiza el escalonamiento de los ángulos articulares para asegurar que las posiciones calculadas sean compatibles con las configuraciones físicas de los motores del robot. Posteriormente, se generan las trayectorias vectoriales mediante interpolación entre puntos en el espacio tridimensional, aplicando transformaciones homogéneas que permiten su correcta ubicación en el sistema de referencia del manipulador.

Escalamiento de ángulos

Los motores del brazo robot están configurados con ángulos relativos a dichos motores, en el cálculo de la cinemática directa con las transformaciones de D-H, se obtiene la forma estándar en que se configura el robot, y los puntos iniciales en donde se considera a los ángulos en cero. Entonces, para ir a la par según los cálculos se realiza una conversión y escalamiento de dichos ángulos, Figura. 2.32 .



(a) Para $\theta_{1,3,5}$. (b) Para $\theta_{2,4}$.

Figura 2.32: Gráfica de ángulos equivalentes [*Autoría propia*].

El motor que utiliza el gripper está destinado a las mismas dimensiones, ya que no influyen en el control de la trayectoria.

Procedimiento para la generación de trayectorias

El diseño de las trayectorias se realizó en Inkscape utilizando herramientas de gráficos vectoriales. Para la conversión a instrucciones ejecutables, se empleó

la extensión Gcodetools, que transforma los diseños en código G con comandos G01 para movimientos lineales precisos, que se adaptan a las capacidades de movimiento del brazo robótico.

Antes de enviar las instrucciones al robot, el código se revisa en NC Viewer para corregir posibles fallos o movimientos bruscos. Una vez validado, MATLAB convierte estos comandos en coordenadas que el robot puede seguir punto por punto.

Las trayectorias que se presentarán a continuación comenzarán con patrones simples e irán aumentando su complejidad, siempre dentro de los parámetros seguros de operación del robot.

Los pasos completos y detalles adicionales de esta implementación se encuentran disponibles en el Anexo A.6.

Trayectoria circular

La trayectoria circular constituye un ejercicio fundamental para evaluar la coordinación sincronizada de los ejes del brazo robótico. Este movimiento permite verificar la precisión del sistema, dado que cualquier desviación o error en las articulaciones se manifiesta claramente al intentar completar un ciclo perfecto y regresar al punto inicial.

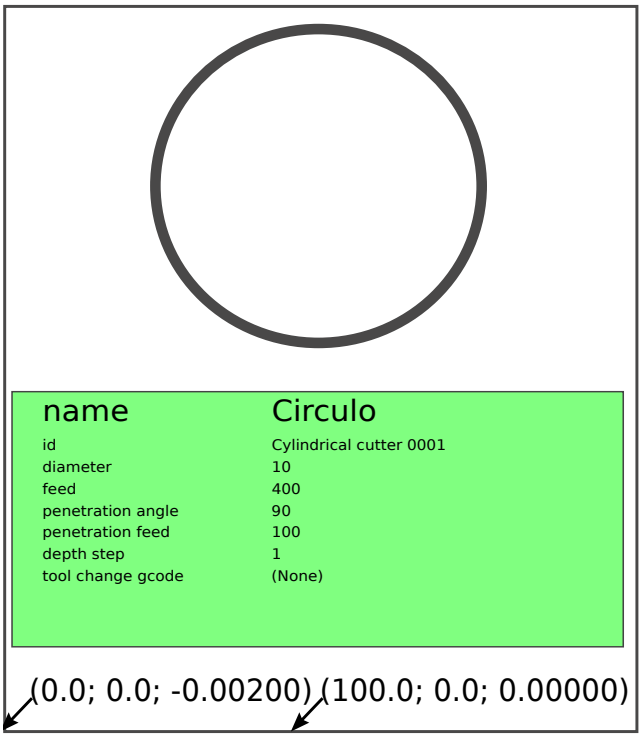


Figura 2.33: Diseño de trayectoria circular en Inkscape .

La generación de esta trayectoria se basa en el diseño gráfico de la figura en un entorno vectorial, seguido de la obtención del archivo G-code mediante herramientas especializadas que traducen dicha figura en instrucciones precisas para el robot.

Posteriormente, el archivo G-code es procesado en MATLAB para extraer y visualizar los puntos de la trayectoria en el espacio tridimensional, asegurando así la correcta forma y continuidad del recorrido antes de su ejecución física.

Este procedimiento garantiza una transición eficiente entre el diseño gráfico, la generación de código y la simulación, facilitando la obtención de trayectorias precisas y reproducibles que respetan las limitaciones cinemáticas y espaciales del manipulador.

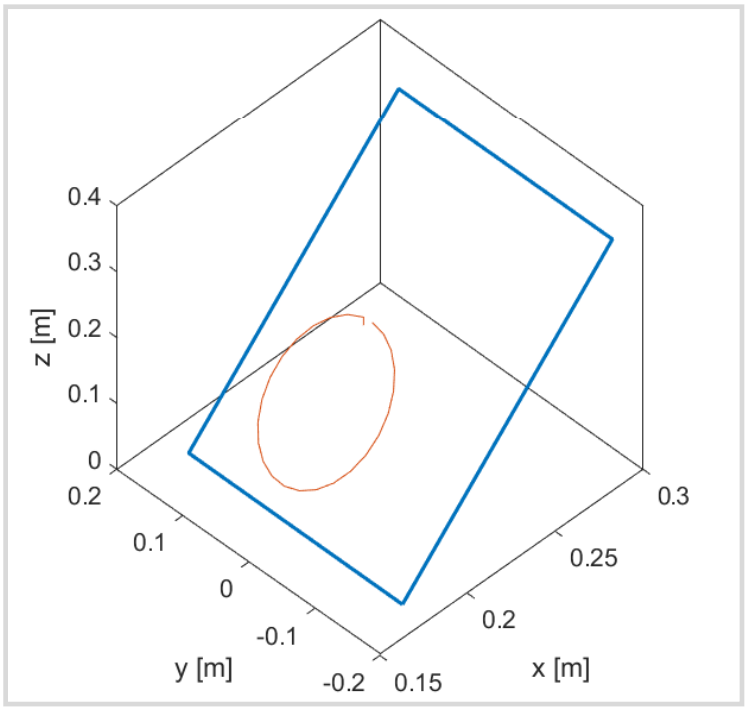


Figura 2.34: Simulación de la trayectoria circular en MATLAB

Las demás trayectorias desarrolladas siguen una metodología similar, adaptando los patrones de movimiento para validar la versatilidad y exactitud del brazo robótico en diversos escenarios operativos.

Trayectoria estrella

La Figura 2.35 presenta la trayectoria diseñada para guiar el efector final del brazo robótico a lo largo del contorno completo de una estrella de cinco puntas. El recorrido comienza en un punto exterior y avanza de manera secuencial

siguiendo secuencial,cada uno de los segmentos que forman la figura estrellada, asegurando que el brazo robótico reproduzca exactamente la forma geométrica.

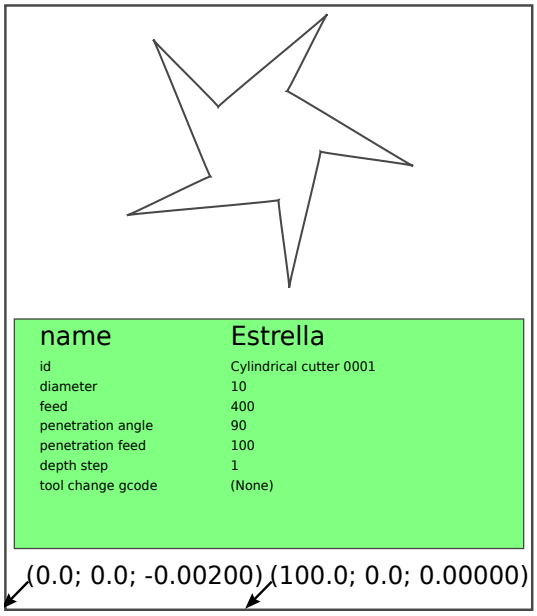


Figura 2.35: Diseño trayectoria estrella en Inkscape

Dado que el método para generar las trayectorias ya ha sido definido, esta ilustración sirve como una guía visual para verificar y comprender el desplazamiento del brazo en el espacio.

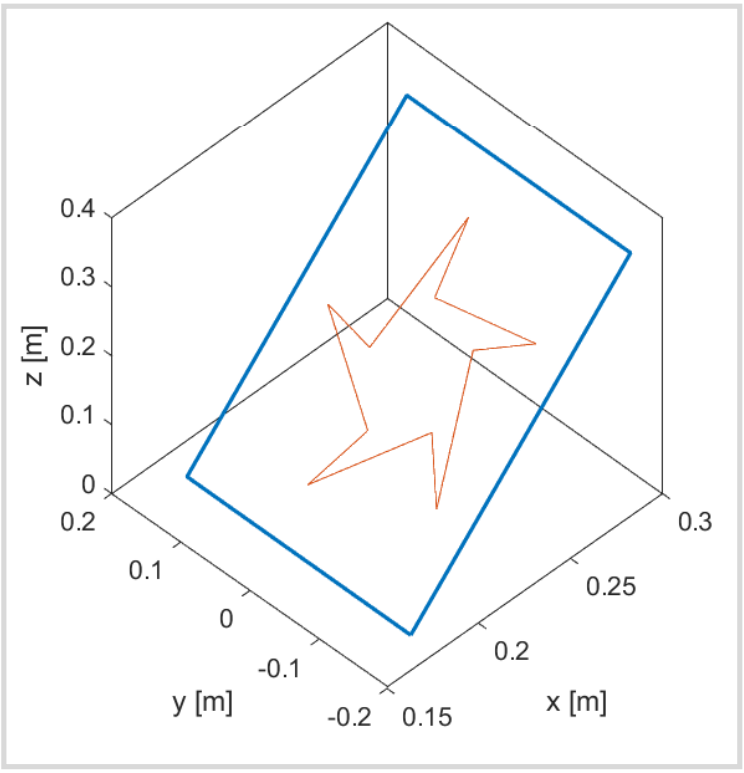


Figura 2.36: Simulación de trayectoria estrella en MATLAB.

Trayectoria pentagonal

En la Figura 2.37, muestra una trayectoria robótica cerrada que describe un pentágono regular. El brazo inicia su recorrido en uno de los vértices y avanza secuencialmente por los cinco lados de igual longitud, regresando al punto de partida, esta trayectoria es ideal para aplicaciones que requieren patrones simétricos.

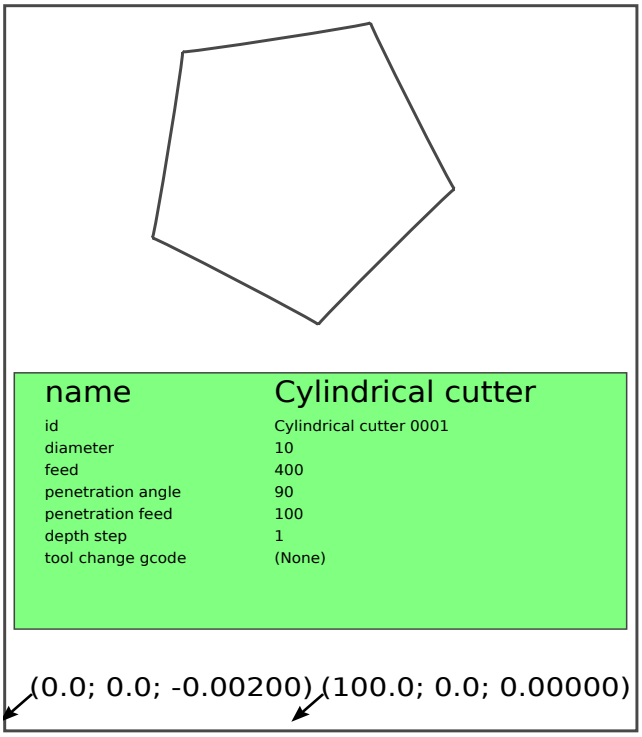


Figura 2.37: Diseño de trayectoria pentagonal en Inkscape.

Además, esta figura permite comprobar el comportamiento del sistema ante trayectorias cerradas y simétricas, lo que es útil para afinar parámetros como velocidad, aceleración, suavizado de trayectoria y errores acumulados. También puede utilizarse como patrón de calibración para validar que el brazo ejecuta correctamente comandos de posición en un entorno de trabajo previamente definido.

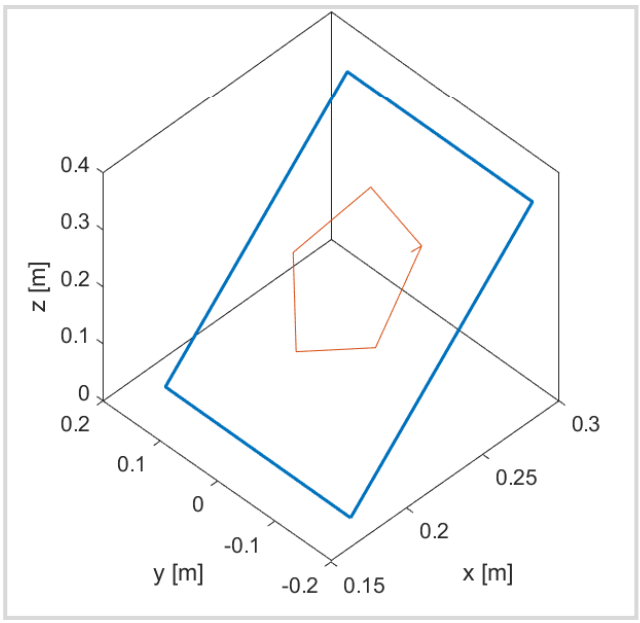


Figura 2.38: Simulación de la trayectoria pentagonal en Matlab

Trayectoria triangular

En la Figura 2.39, se presenta una trayectoria triangular equilátera. El brazo robótico sigue los tres segmentos rectilíneos de idéntica longitud, comenzando y finalizando en el mismo vértice.

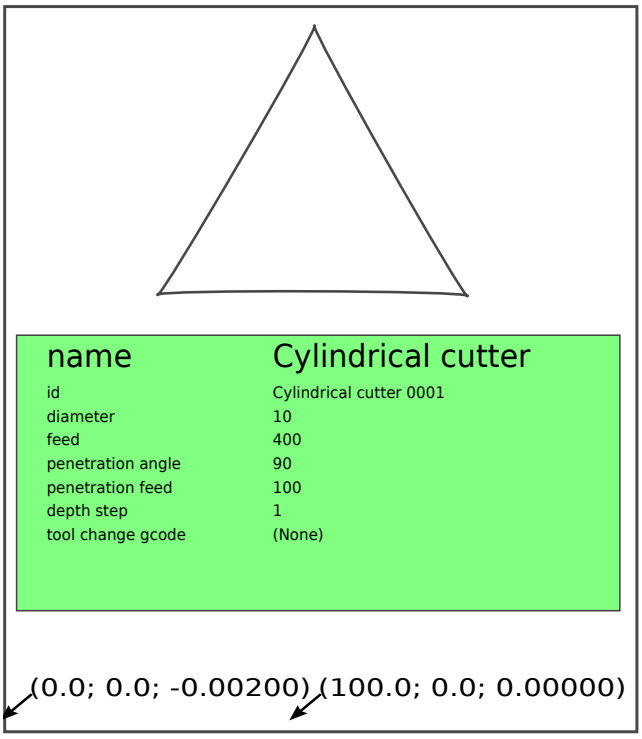


Figura 2.39: Diseño de trayectoria triangular en Inkscape

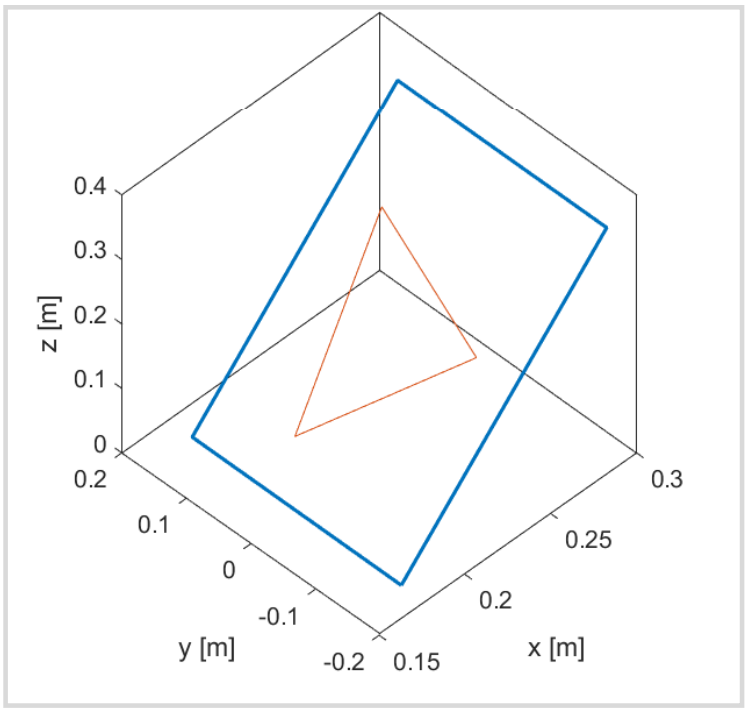


Figura 2.40: Simulación de la trayectoria triangular en Matlab

Trayectoria doble cuadrado

En la Figura 2.41, ilustra una trayectoria compuesta por dos cuadrados adyacentes que comparten un vértice común. El brazo recorre primero el perímetro de un cuadrado, luego se traslada al segundo y completa su contorno, creando una figura unificada. Ideal para patrones complejos que combinan múltiples formas.

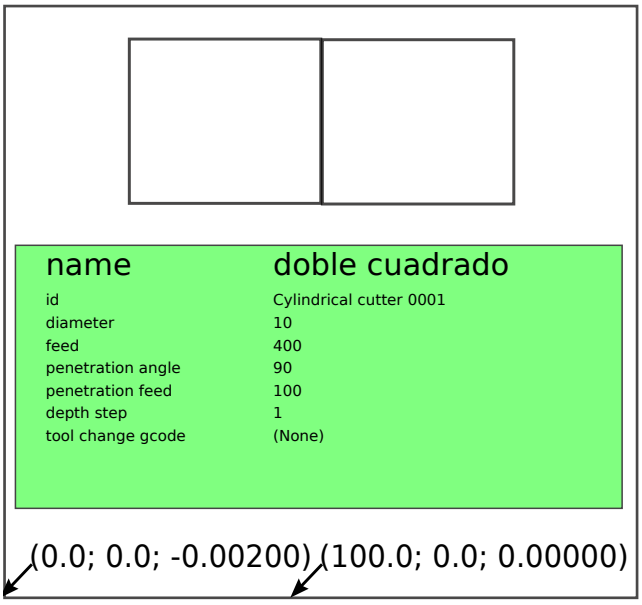


Figura 2.41: Diseño de trayectoria doble cuadrado en Inkscape

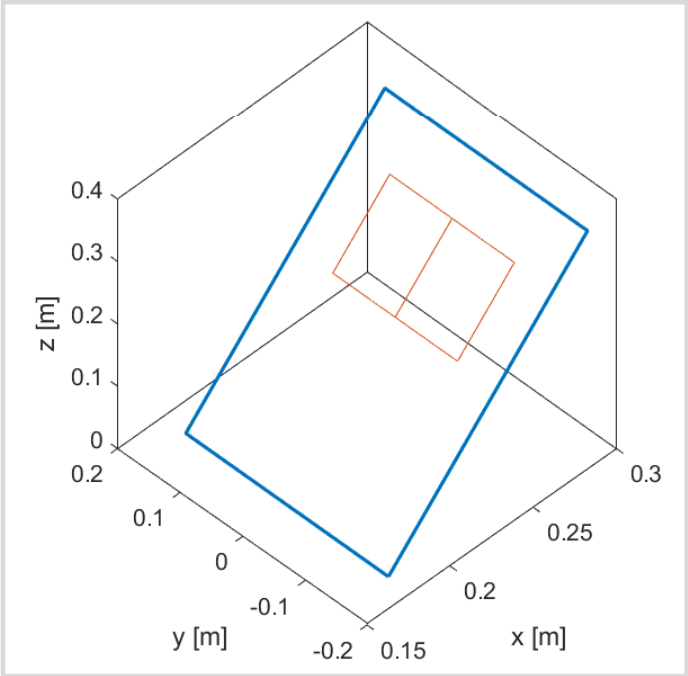


Figura 2.42: Simulación de la trayectoria de doble cuadrado en Matlab

2.2.4.3 Simulación del sistema de control en Simulink

Para llevar a cabo el modelo dinámico y aplicar el controlador LQG en un brazo robótico de cinco grados de libertad, se utilizó la plataforma Simulink. Esta herramienta permite construir y simular sistemas dinámicos de forma modular, visual y flexible.

Modelado dinámico en Simulink

La Figura 2.43 presenta el modelo dinámico de los motores del brazo robótico, implementado en el entorno de Simulink. Este modelo se enfoca principalmente en representar el comportamiento dinámico de los motores, que constituyen los actuadores principales del sistema, más que en modelar la totalidad del brazo robótico. El esquema incorpora elementos esenciales para simular el funcionamiento real de los motores, incluyendo el modelo matemático del actuador, el estimador de estados, el controlador óptimo, la dinámica inversa, ruido en las mediciones y retardo en la comunicación, con el fin de evaluar el desempeño del control en lazo cerrado bajo condiciones realistas.

El modelado de los motores se basa en la metodología propuesta por Peter Corke, que describe los actuadores eléctricos como sistemas dinámicos de segundo orden, integrando la dinámica eléctrica (resistencia, inductancia y constante electromecánica) y la dinámica mecánica (inercia del rotor y fricción viscosa). Este modelo, expresado en espacio de estados, fue implementado para

cada motor y constituye la base sobre la cual se desarrolla el esquema general [39].

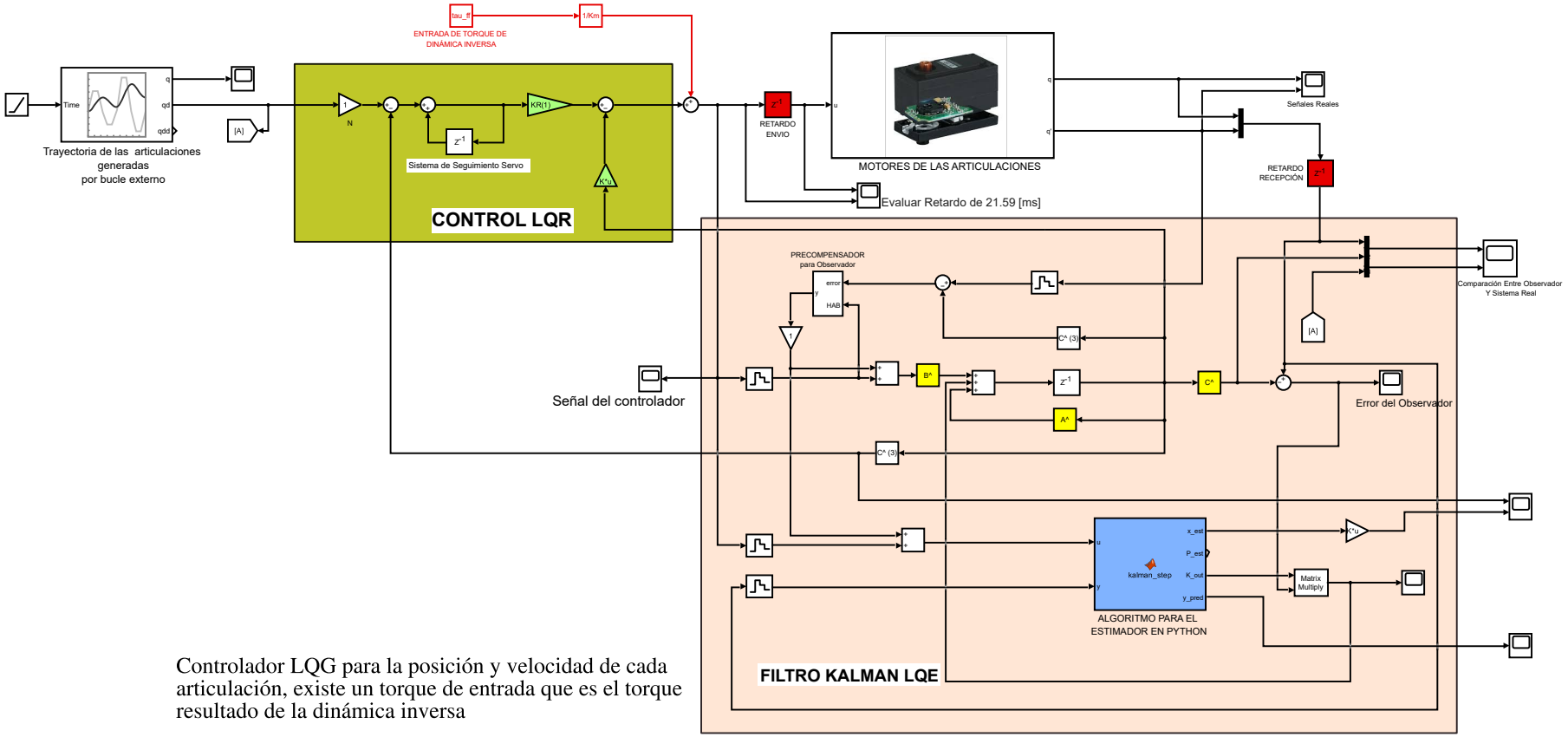
En el modelo se incluyen las señales de torque calculadas mediante el algoritmo recursivo de Newton–Euler, que determina la dinámica inversa necesaria para generar las entradas adecuadas según la trayectoria deseada y el estado actual del sistema. Este torque se introduce como una perturbación estructurada que refleja el esfuerzo dinámico requerido por los motores para seguir trayectorias específicas, considerando aceleraciones, gravedad y fuerzas de Coriolis.

Para estimar los estados internos de los motores ante mediciones afectadas por ruido, se implementó un estimador LQE (Linear Quadratic Estimator). El ruido de medición se modeló como ruido blanco gaussiano, representado mediante matrices de covarianza Q y R, ajustadas experimentalmente para optimizar la precisión de la estimación. En el esquema, el bloque resaltado en amarillo corresponde al modelo en espacio de estados utilizado por el observador.

El controlador óptimo LQR (Linear Quadratic Regulator) genera la señal de control a partir de los estados estimados, minimizando un criterio cuadrático que penaliza el error de seguimiento y el esfuerzo de control aplicado. La interacción entre el controlador, el estimador y el modelo del motor conforma el núcleo del sistema en lazo cerrado, permitiendo evaluar estabilidad y robustez frente a perturbaciones.

Adicionalmente, se incorporó la simulación del retardo de comunicación, medido en el entorno de planificación desarrollado en Python. Este retardo se implementa mediante bloques discretos en Simulink que simulan la latencia en la transmisión de señales, permitiendo analizar su efecto sobre la estabilidad y precisión del sistema.

En conjunto, este esquema ofrece una simulación integral del comportamiento dinámico de los motores del brazo robótico, abarcando las acciones de control y estimación, así como condiciones reales de operación como ruido en sensores y retardo en comunicaciones. Este modelo es fundamental para validar y ajustar el diseño del controlador y estimador antes de su implementación práctica en el sistema físico.



Controlador LQG para la posición y velocidad de cada articulación, existe un torque de entrada que es el torque resultado de la dinámica inversa

Figura 2.43: Esquema general del sistema de control implementado en Simulink.

2.2.4.4 Simulación e implementación del controlador en Python

En esta sección se detalla la implementación del controlador LQG empleando el lenguaje de programación Python. Se desarrollan las funciones correspondientes al modelo dinámico del brazo robótico, así como los componentes del observador de estados y del regulador óptimo. La estructura del código permite simular el seguimiento de trayectorias planificadas, evaluando la precisión del sistema y su capacidad de rechazo ante posibles perturbaciones. El código fuente completo correspondiente al desarrollo del controlador LQG propuesto se encuentra detallado en la sección de anexos A.8.

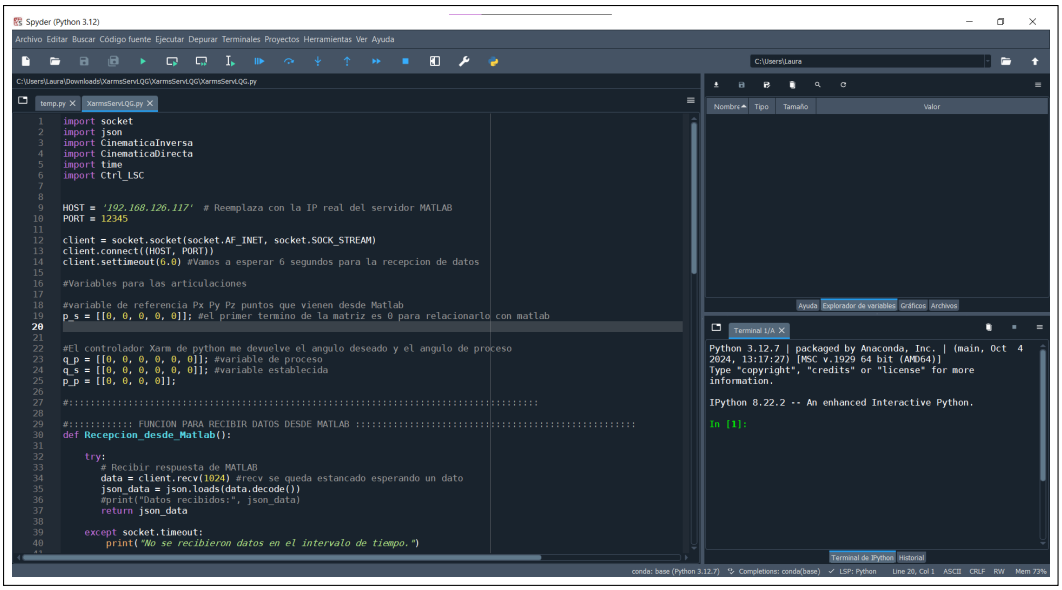


Figura 2.44: Programación en python.

El diagrama de flujo representa el funcionamiento del controlador LQG en el seguimiento de trayectorias de un brazo robótico. Tras verificar las comunicaciones TCP/IP y HID, se reciben desde MATLAB los puntos de trayectoria y la posición actual del robot. Luego, se calcula la cinemática inversa, se aplica el control LQG y se generan las señales PWM. Finalmente, la cinemática directa estima la nueva posición, que es enviada de vuelta a MATLAB, cerrando el lazo de control en tiempo real.

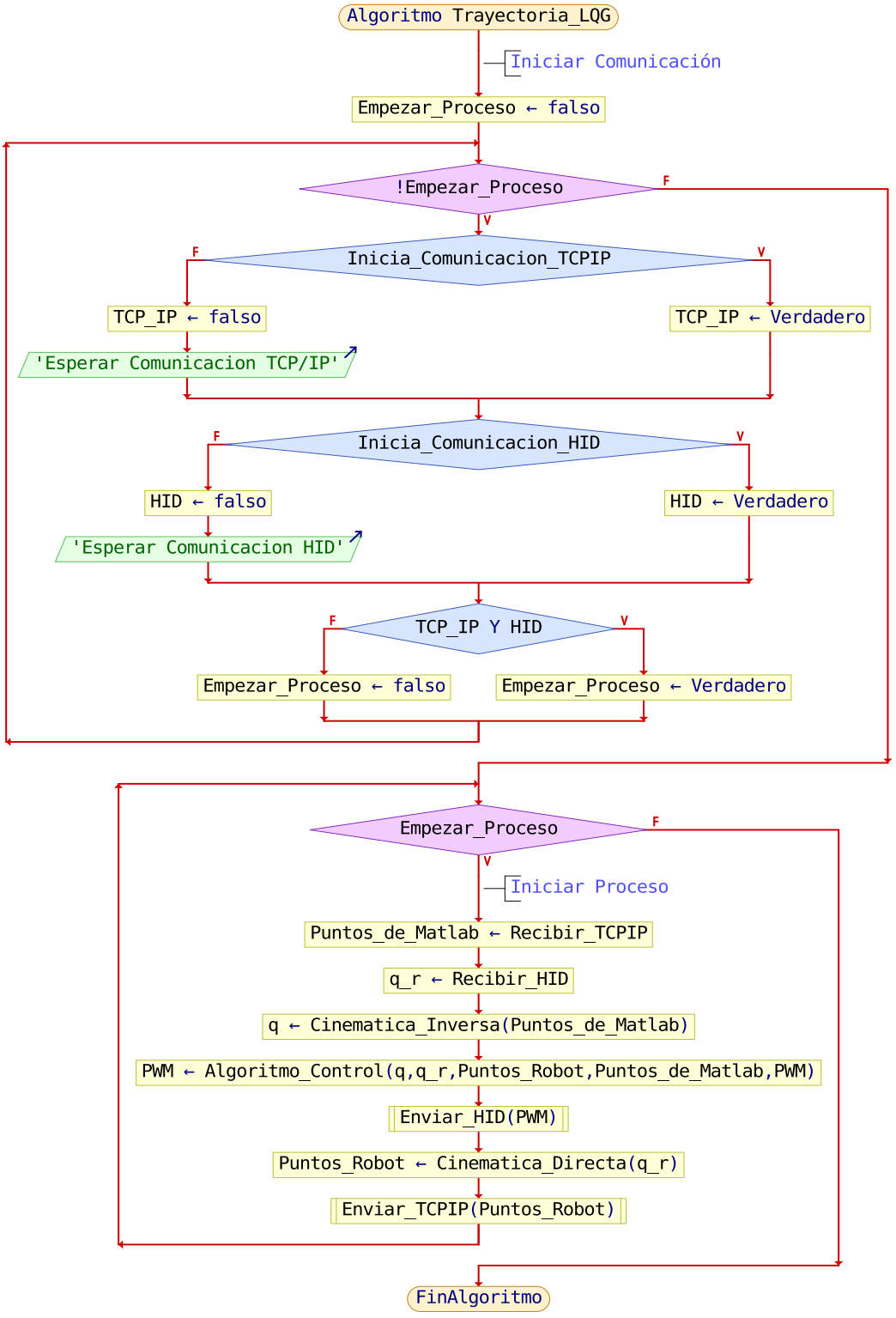


Figura 2.45: Diagrama de flujo del controlador LQG

2.2.4.5 Interfaz gráfica de usuario para control del brazo robótico

Como parte del desarrollo del sistema de control, se implementó una interfaz gráfica en App Designer de *MATLAB* con el fin de integrar las funciones de operación, monitoreo y configuración del brazo robótico. Su desarrollo responde a la necesidad de gestionar trayectorias, supervisar variables dinámicas y facilitar la interacción con el hardware desde un entorno intuitivo y estructurado. A continuación, se presentan las secciones que componen dicha aplicación.

En la Figura 2.46 se muestra la pantalla inicial de la interfaz, correspondiente a la pestaña *Inicio*. Esta sección funciona como punto de acceso general al sistema, presentando una interfaz simplificada donde destaca un botón principal que permite ingresar al *Panel de control*, el cual habilita todas las funciones de control y monitoreo del brazo robótico.

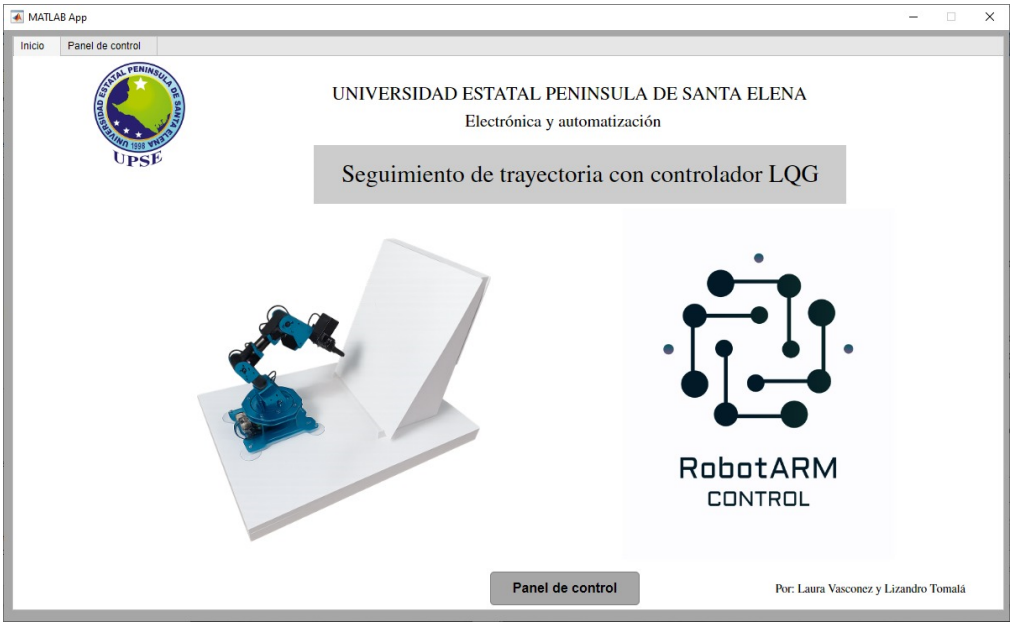


Figura 2.46: Pantalla principal de RobotARM Control

La Figura 2.47 muestra la pestaña del panel de control, diseñada para supervisar y ejecutar el seguimiento de trayectorias en un brazo robótico mediante un controlador LQG. La interfaz gráfica está organizada en secciones funcionales que permiten una interacción intuitiva con el sistema.

1. **Selección de trayectoria:** En la parte izquierda, se encuentra el módulo para seleccionar el tipo de trayectoria, el cual ofrece distintas opciones predefinidas a través de un menú desplegable. Además, es posible incorporar trayectorias adicionales previamente diseñadas, siempre que se sigan los procedimientos descritos anteriormente.

2. **Control de ejecución:** Justo debajo, se ubican los botones para iniciar o cancelar la ejecución del seguimiento, brindando control manual al usuario.

3. **Interruptor de comunicación:** Más arriba, se localiza el interruptor de comunicación, que permite habilitar o deshabilitar la conexión con los dispositivos involucrados.

4. **Indicadores de conexión:** Inmediatamente a continuación, se visualizan dos indicadores tipo LED que informan sobre el estado de la conexión con el servidor MATLAB y con el brazo robótico.

5. **Selector de vista:** También en la parte izquierda, se integra un selector de vista que permite cambiar la perspectiva de la trayectoria mostrada (isométrica, lateral o tipo lienzo), facilitando la interpretación espacial del movimiento.

6. **Gráfica principal de trayectoria:** Al centro de la interfaz, se encuentra la gráfica principal donde se representa la trayectoria deseada (en azul) y la trayectoria real (en rojo), permitiendo comparar visualmente el rendimiento del sistema.

7. **Visualización de métricas:** Debajo de esta gráfica, se incluyen campos que muestran en tiempo real el tiempo de ejecución y el error cometido durante el seguimiento.

8. **Gráficas angulares:** A la derecha, se presentan cinco gráficas individuales que muestran el comportamiento angular de cada articulación (θ_1 a θ_5) en función del tiempo.

9. **Panel de selección de articulaciones:** Encima de estas gráficas, se ubica un panel de selección que permite elegir qué articulación visualizar o bien observar todas simultáneamente.

10. **Botón de almacenamiento:** Finalmente, en la parte inferior derecha, se dispone de un botón para guardar los resultados de la ejecución, lo cual permite almacenar datos útiles para análisis posteriores.

Esta interfaz facilita una supervisión completa y eficiente del sistema, integrando elementos de visualización, control y monitoreo en una plataforma.

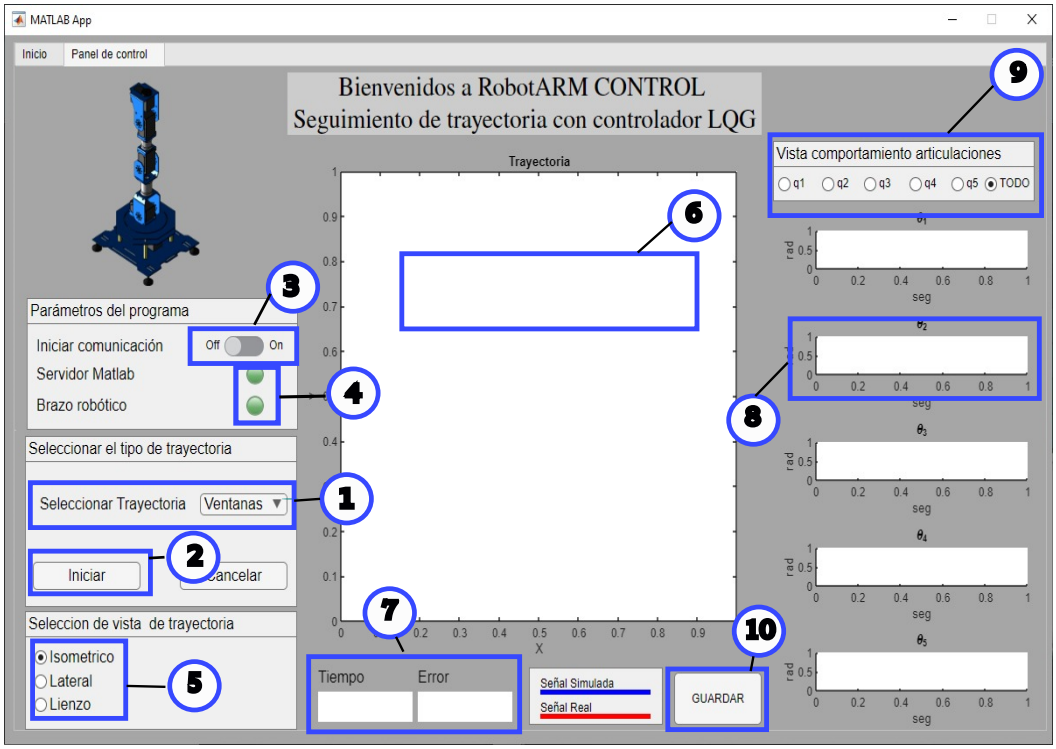


Figura 2.47: Pestaña de programación

El diseño de esta aplicación se alinea con los principios establecidos en la norma ISO 9241-210:2019, la cual promueve el diseño centrado en el usuario mediante la mejora de la eficiencia, accesibilidad y facilidad de uso. Asimismo, se basó en la norma ISO/IEC 25010:2011, que define un modelo de calidad del software basado en características como funcionalidad, fiabilidad, usabilidad, eficiencia. La implementación de una estructura modular, el uso de codificación por colores y la incorporación de elementos gráficos interactivos evidencian buenas prácticas en ingeniería de interfaces, permitiendo que incluso usuarios con conocimientos limitados en programación puedan interactuar con el sistema de manera segura y precisa.

3 Resultados

En este capítulo se presentan los resultados obtenidos tras la implementación de un sistema de control para el seguimiento de trayectorias en un brazo robótico de cinco grados de libertad empleando el algoritmo LQG, el cual fue implementado y programado utilizando el lenguaje de programación Python en el entorno de desarrollo Spyder. La función principal de este controlador consistió en minimizar el error cuadrático esperado, considerando tanto la incertidumbre del modelo como el ruido de las mediciones sensoriales.

El análisis matemático del modelo dinámico del brazo robótico fue desarrollado en MATLAB, utilizando herramientas simbólicas para obtener las ecuaciones de movimiento por el método de Euler-Lagrange. Posteriormente, se procedió a linealizar el modelo en torno a un punto de operación para su implementación en el esquema de control LQG. Este modelo permitió obtener las matrices de estado necesarias para diseñar el estimador de estados (filtro de Kalman) y el regulador óptimo (LQR), componentes fundamentales del controlador LQG.

Las trayectorias a seguir por el brazo robótico fueron generadas a partir de instrucciones G-code exportadas desde Inkscape. Se emplearon los comandos G00, G01 y G02, correspondientes a desplazamientos rápidos, interpolación lineal e interpolación circular respectivamente. Estos datos fueron procesados mediante un script en MATLAB para su conversión a coordenadas temporizadas, compatibles con el sistema de control. Las trayectorias convertidas fueron integradas al entorno de simulación para evaluar el rendimiento del controlador.

Para facilitar la interacción con el sistema y permitir la ejecución de pruebas, se desarrolló una interfaz gráfica mediante MATLAB App Designer. En la Figura 2.47 se observa la pestaña de control manual, que permite enviar señales directas a cada articulación del brazo, así como encender y apagar la pinza, verificar la conexión serial y ajustar parámetros como posición angular y velocidad. En la Figura 2.47 se muestra la pestaña de programación, que permite ingresar posiciones cartesianas y ángulos articulares, activar tareas específicas, y gestionar la secuencia de movimientos mediante botones de añadir, guardar y cerrar.

Para la validación del desempeño del sistema de control, se definieron tres

trayectorias de prueba con diferentes características geométricas. Cada trayectoria fue ejecutada cinco veces de manera independiente con el objetivo de evaluar la precisión, estabilidad y repetibilidad del sistema de control. La métrica principal utilizada fue el error promedio de posición, calculado como la media de los errores absolutos entre la trayectoria de referencia y la trayectoria obtenida. Adicionalmente, se determinó la desviación estándar del error para cada trayectoria, como medida de consistencia del desempeño del controlador LQG ante perturbaciones simuladas.

A continuación, se presentan los resultados obtenidos para cada una de las trayectorias implementadas, mostrando tablas comparativas de error, gráficos de desempeño temporal y análisis de estabilidad del sistema. Estos resultados permitirán evaluar la eficacia del controlador LQG en el seguimiento preciso de trayectorias generadas a partir de G-code.

3.1 Validación de Ik mediante Fk

Para verificar la validez de (Px, Py, Pz) en el cálculo de cinemática inversa, se aplica el método de cinemática directa al modelo del brazo robótico, en donde, con una configuración específica de q_i, se calcula la posición del efector final. En la tab. 3.1 se muestran los resultados obtenidos al comparar con el valor de I_k, previamente determinado con las Ecs.2.4-2.10. La coincidencia entre ambas posiciones confirmó que I_k representa con precisión la ubicación del efector final para la configuración articular dada. Este procedimiento no solo válida la precisión del modelo cinemático, sino también la correcta determinación de I_k, lo cual es esencial para la siguiente etapa que es el análisis de control.

Posición	Entrada de cinemática Inversa					Salida de cinemática inversa					salida de cinemática directa			Error(%)
	X	Y	Z	Q	P	θ_1	θ_2	θ_3	θ_4	θ_5	X	Y	Z	
1	0,265	-0,040	0,284	110,564	0,000	-8,584	-62,064	13,920	-62,420	0,000	0,265	-0,040	0,284	0
2	0,253	0,009	0,200	79,358	0,000	-2,037	-78,050	33,017	-34,325	0,000	0,253	0,009	0,200	0
3	0,249	0,020	0,191	75,602	0,000	4,592	-77,425	30,672	-28,849	0,000	0,249	0,020	0,191	0
4	0,261	0,000	0,222	88,676	0,000	0,000	-77,898	37,058	-47,836	0,000	0,261	0,000	0,222	0
5	0,227	0,031	0,129	53,512	0,000	7,776	-75,078	30,813	-9,247	0,000	0,227	0,031	0,129	0
6	0,228	-0,128	0,131	70,274	0,000	29,310	-83,755	70,176	-56,695	0,000	0,228	-0,128	0,131	0
7	0,224	-0,134	0,122	68,955	0,000	-30,888	-83,894	73,646	-58,706	0,000	0,224	-0,134	0,122	0

Tabla 3.1: Resultados del modelo geométrico cuantitativo inverso y error entre el resultado de la cinemática directa y el resultado de la cinemática inversa

3.2 Análisis de la respuesta dinámica del sistema simulado

Las gráficas obtenidas permiten analizar la evolución temporal de variables clave como la posición angular (θ_i), la velocidad angular ($\dot{\theta}_i$) y el torque aplicado (τ_i) en cada una de las articulaciones del robot.

La Figura 3.1 muestra la respuesta dinámica de las posiciones articulares (θ_1, θ_2, θ_3 y θ_4) a lo largo del tiempo. En dichas gráficas, la línea azul representa la trayectoria de referencia, mientras que la línea roja indica la trayectoria real seguida por el robot. Se observa un seguimiento adecuado en todas las articulaciones, con pequeñas discrepancias en ciertos intervalos, particularmente en θ_2 y θ_3, donde se presentan leves desviaciones respecto a la referencia. Estas diferencias pueden deberse a limitaciones de los actuadores, retardos en el sistema o ajustes subóptimos en los parámetros del controlador.

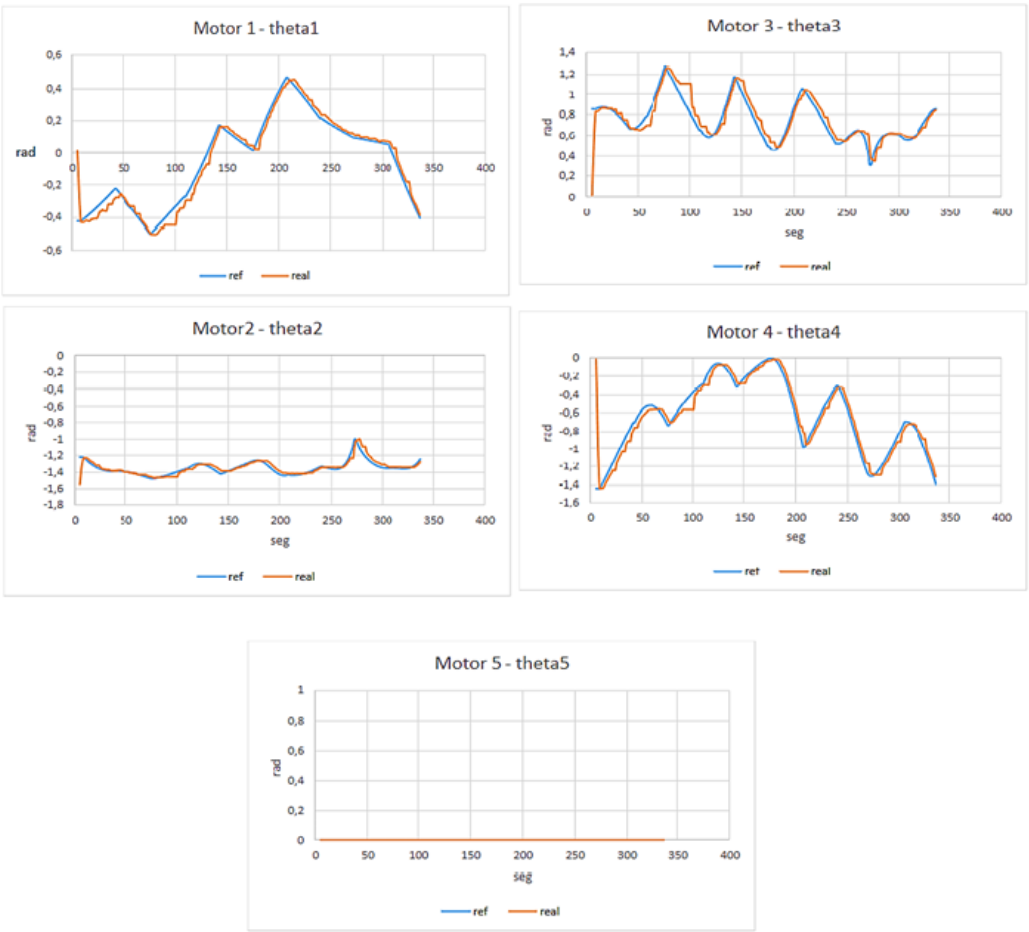


Figura 3.1: Gráficas de respuesta de dinámica.

En conjunto, las gráficas permiten verificar que el sistema controlado logra un seguimiento estable y preciso de las trayectorias deseadas, lo cual evidencia el correcto desempeño del controlador LQG implementado. Asimismo, se puede inferir que el sistema responde de manera robusta frente a perturbaciones o condiciones iniciales variables, lo que resulta fundamental para validar la funcionalidad del diseño de control y garantizar el cumplimiento de los objetivos de precisión y estabilidad requeridos por la aplicación robótica.

3.3 Visualización del modelo robótico en el espacio tridimensional

La Figura 3.2 muestra una representación tridimensional del modelo simulado del brazo robótico *XArmS* con 5 grados de libertad. En la imagen se observa claramente la estructura del manipulador, compuesta por eslabones articulados y representada mediante cuerpos geométricos de color rojo. Además, se muestran los sistemas de referencia asociados al espacio base (ejes X, Y, Z) y al efector final, lo que permite visualizar la orientación espacial del robot durante la ejecución de la trayectoria.

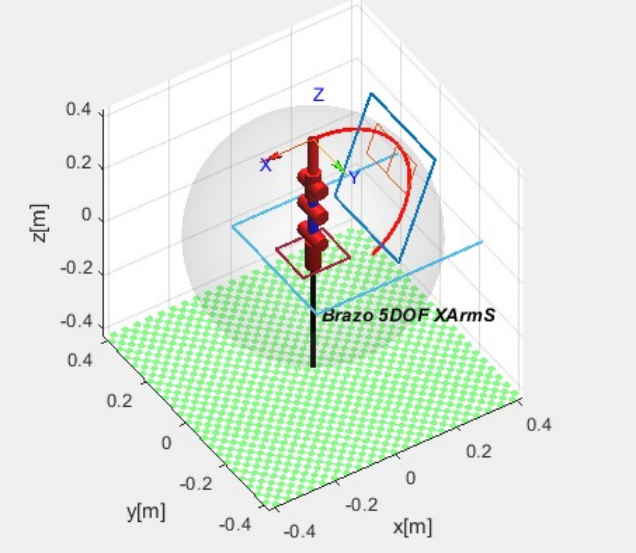


Figura 3.2: Representación tridimensional del brazo robótico

La trayectoria deseada está representada mediante una curva de color rojo en el espacio cartesiano, mientras que la trayectoria real seguida por el efector final se encuentra superpuesta, lo cual facilita la comparación visual del desempeño del sistema de control. Se aprecia que el efector final del brazo sigue con precisión la trayectoria planificada, manteniéndose dentro del área de trabajo definida (sombreada en verde). Esta capacidad de seguimiento evidencia un

diseño de control eficaz y una correcta parametrización del modelo cinemático y dinámico del robot.

En conjunto, esta visualización permite validar tanto la implementación geométrica del brazo robótico como la precisión del controlador en el seguimiento de trayectorias en el espacio tridimensional. Esta etapa de verificación es crucial para aplicaciones donde se requiere alta exactitud en la manipulación de objetos o interacción con el entorno.

3.4 Validación del modelo de control mediante simulación en Simulink

En esta sección se presenta el análisis de la respuesta del sistema de control aplicado al brazo robótico, obtenido a partir de una simulación desarrollada en el entorno de Simulink. La gráfica muestra la evolución de la posición y la velocidad angular de una articulación del robot bajo la acción del controlador LQG, comparando la señal obtenida por los sensores con la estimación generada por el observador.

Análisis de la gráfica del comportamiento dinámico - primera respuesta

A continuación, se presenta la Figura. 3.3 y el análisis del comportamiento dinámico del sistema de control LQG implementado en el brazo robótico.

Comportamiento de la gráfica:

- **Respuesta en posición angular:** El sistema alcanza la consigna de aproximadamente 4,26 rad con un sobreimpulso leve cercano a los 4,7 rad, estabilizándose antes de los 10 segundos. Esta respuesta indica un comportamiento ágil y controlado durante la fase transitoria.
- **Comparación entre señales:** La señal del sensor (naranja) presenta una mayor cantidad de ruido visible, mientras que la señal estimada por el observador (azul) sigue una trayectoria suave. Esto resalta la efectividad del observador en la reconstrucción de estados precisos a pesar de las perturbaciones externas.
- **Respuesta en velocidad angular:** Se observa un pico inicial de aproximadamente 1,5 rad/s seguido de una estabilización rápida alrededor de 0 rad/s, cumpliendo con el punto de consigna. Sin embargo, el ruido en la señal del sensor es más pronunciado, lo que influye en la oscilación aparente del sistema a lo largo del tiempo.

Análisis cuantitativo de los valores:

- **Intervalo analizado:** Se consideran los tiempos $t_1 = 19{,}323$ s y $t_2 = 57{,}968$ s, lo que da un intervalo de $\Delta t = 38{,}645$ s.
- **Variación de posición según sensor:** La posición cambia de 4,059 rad a 4,268 rad, con una variación de $\Delta Y = 0{,}209$ rad.
- **Velocidad angular promedio:**

$$v = \frac{\Delta Y}{\Delta t} = \frac{0{,}209}{38{,}645} \approx 5{,}404 \times 10^{-3} \text{ rad/s} = 5{,}404 \text{ mrad/s}$$

  Este valor, aunque bajo, representa una variación mayor respecto a otras pruebas, posiblemente influida por el ruido en la medición del sensor.

- **Frecuencia equivalente del intervalo:**

$$f = \frac{1}{\Delta t} = \frac{1}{38{,}645} \approx 25{,}877 \text{ mHz}$$

  La frecuencia se mantiene constante y sirve como parámetro de referencia temporal para caracterizar el estado estacionario.

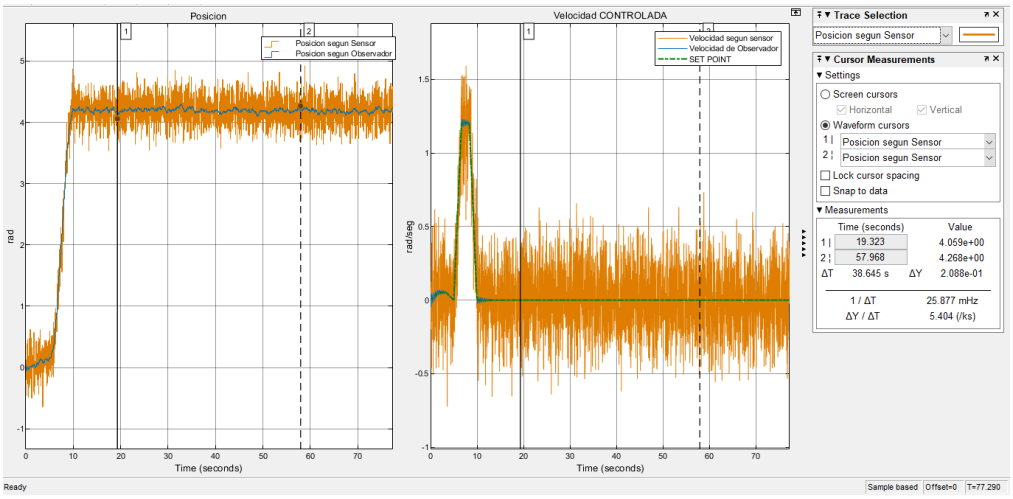


Figura 3.3: Comportamiento del sistema controlado: comparación entre sensor y observador

Análisis de la gráfica del comportamiento dinámico - segunda respuesta

En esta sección se examina una segunda respuesta experimental del sistema bajo condiciones similares, permitiendo observar con mayor detalle la evolución del sistema controlado y la efectividad del observador. Se evalúa el seguimiento de la trayectoria, la precisión en la estimación de estados y el nivel de oscilación presente en las señales, en el comportamiento en estado estacionario y los efectos del ruido en la señal sensada.

Comportamiento de la gráfica:

- **Respuesta en posición angular:** El sistema alcanza la posición deseada de aproximadamente 4,2 rad con un sobreimpulso leve cercano a los 5 rad y se estabiliza antes de los 10 segundos. Esto refleja un desempeño transitorio eficiente por parte del controlador LQG.
- **Comparación entre señales:** La señal del sensor (naranja) presenta ruido evidente, mientras que la señal estimada por el observador (azul) es suave y precisa. Esto demuestra que el observador implementado filtra las perturbaciones, mejorando la calidad de la estimación del estado.
- **Respuesta en velocidad angular:** Se observa un pico inicial de aproximadamente 1,5 rad/s seguido de una rápida estabilización hacia 0 rad/s, coincidiendo con el punto de consigna. La velocidad estimada se mantiene estable y con bajo nivel de oscilaciones, a diferencia de la señal medida, que contiene ruido.

Análisis cuantitativo de los valores:

- **Intervalo analizado:** Se analizaron dos puntos en régimen permanente: $t_1 = 19{,}323$ s y $t_2 = 57{,}968$ s, con un intervalo de tiempo de $\Delta t = 38{,}645$ s.
- **Variación de posición estimada:** En dicho intervalo, la posición estimada cambió de 4,161 rad a 4,200 rad, dando una variación de $\Delta Y = 0{,}039$ rad.
- **Velocidad angular promedio:**

$$v = \frac{\Delta Y}{\Delta t} = \frac{0{,}039}{38{,}645} \approx 1{,}008 \times 10^{-3} \text{ rad/s} = 1{,}008 \text{ mrad/s}$$

  Este valor, casi nulo, confirma la estabilidad del sistema durante el régimen permanente.

- **Frecuencia equivalente del intervalo:**

$$f = \frac{1}{\Delta t} \approx \frac{1}{38{,}645} \approx 25{,}877 \text{ mHz}$$

  Aunque no representa una frecuencia operativa, es útil para caracterizar la periodicidad en estado estable.

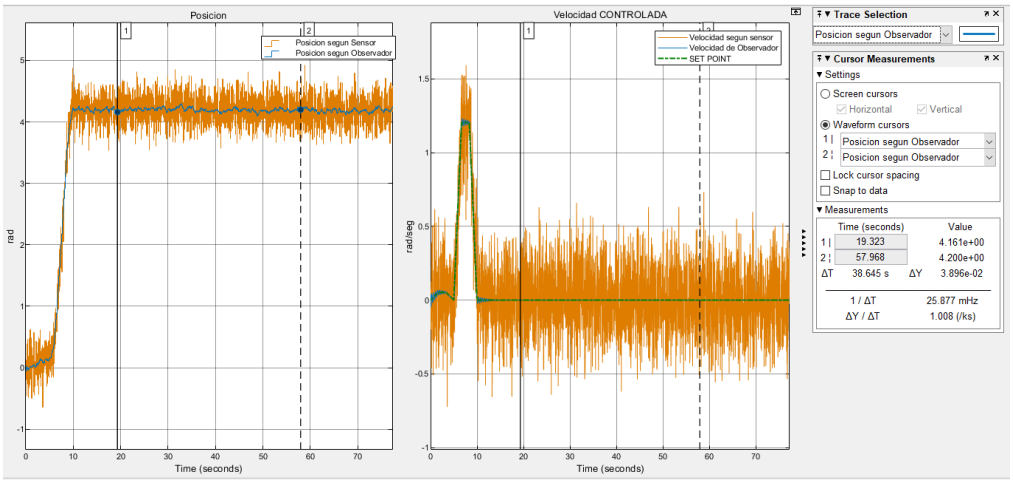


Figura 3.4: Comportamiento del sistema controlado: comparación entre sensor y observador

3.5 Monitoreo de trayectorias desde la interfaz gráfica

A continuación se presentan los resultados de las pruebas de seguimiento de trayectorias realizadas en la interfaz del sistema, utilizando un brazo robótico de cinco grados de libertad (GDL) controlado mediante LQG. Se analizan las gráficas mostradas en la interfaz, comparando las señales de referencia con las señales reales para evaluar el desempeño del seguimiento.

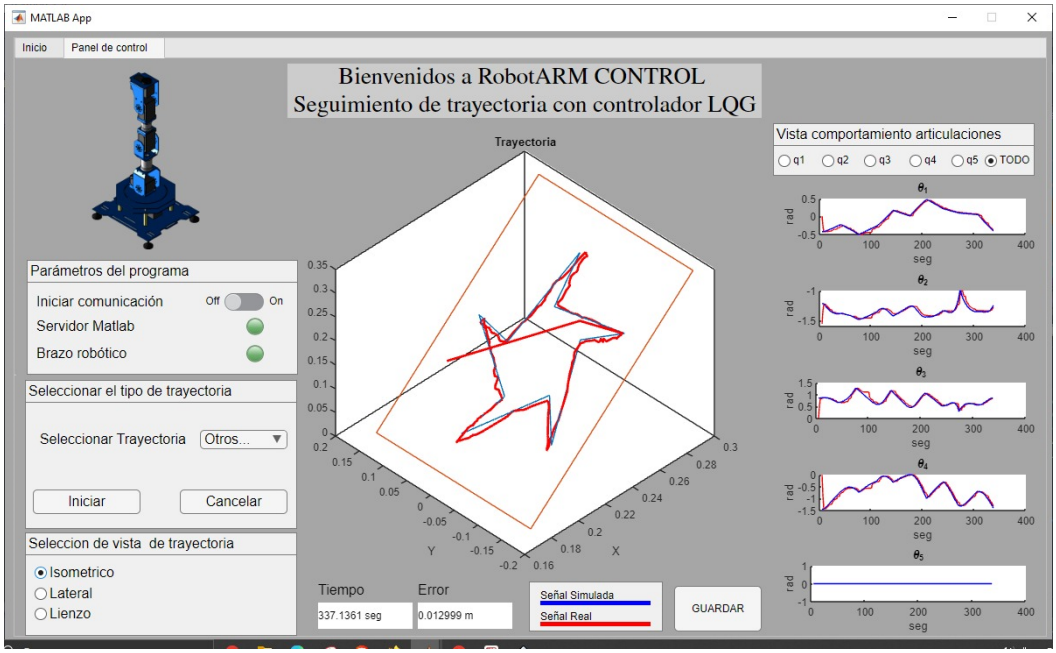


Figura 3.5: Seguimiento de trayectoria en la interfaz del sistema con controlador LQG

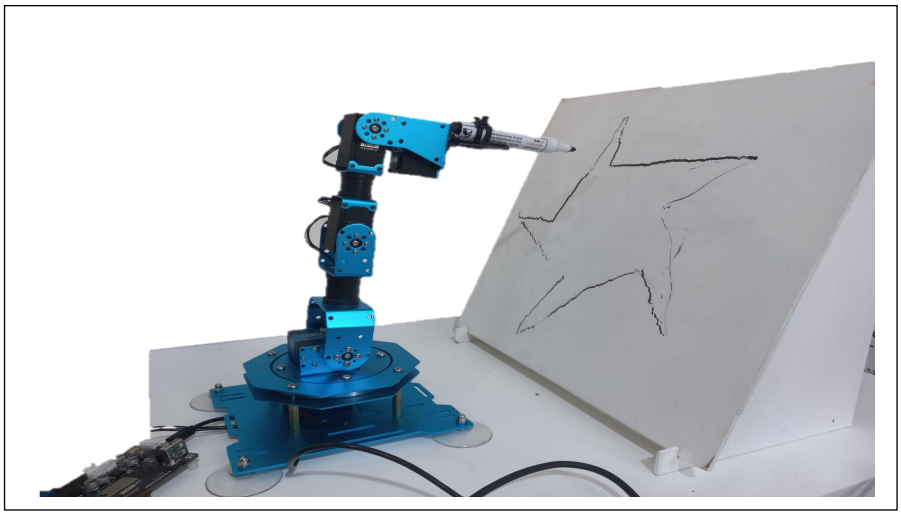


Figura 3.6: Seguimiento de trayectoria de manera física

Articulación θ_1: Se observa una coincidencia notable entre la señal deseada (azul) y la señal real (roja), lo cual indica que el controlador LQG logra un seguimiento preciso en esta articulación. Las diferencias son mínimas, lo que sugiere que la dinámica del sistema en esta articulación está bien modelada y que la acción del controlador compensa adecuadamente las perturbaciones y el ruido.

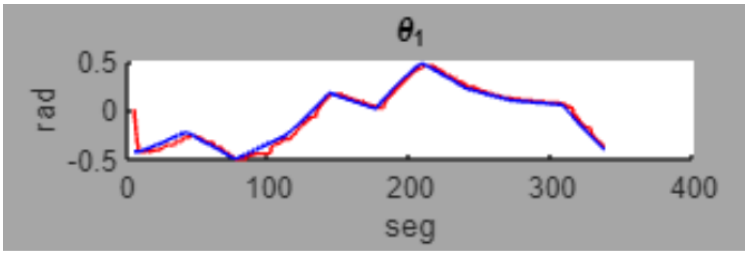


Figura 3.7: Resultado articulación 1

Articulación θ_2: En este caso se observa una mayor diferencia entre la señal de referencia y la señal real. Aunque el perfil de la trayectoria es seguido en términos generales, la señal real presenta oscilaciones o picos pronunciados, especialmente al inicio del movimiento. Esto indica un comportamiento transitorio brusco, posiblemente causado por condiciones iniciales no ideales, alta ganancia en la retroalimentación o ruido en la medición. Posteriormente, la trayectoria se estabiliza y sigue el patrón de la referencia con una precisión aceptable, aunque persisten algunas diferencias.

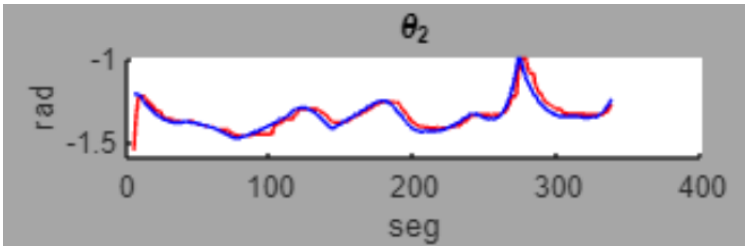


Figura 3.8: Resultado articulación 2

Articulación θ_3: Las señales de referencia y real presentan un comportamiento muy similar, con un seguimiento más preciso en comparación con la articulación θ_2. Ambas curvas tienen variaciones suaves, y el controlador LQG logra mantener la trayectoria real bien alineada con la deseada. Se observan ligeras desviaciones en los puntos de cambio de dirección, pero sin efectos significativos. Este buen desempeño sugiere un ajuste adecuado del observador y el controlador para esta articulación.

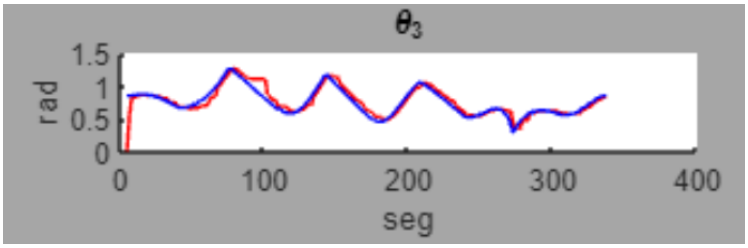


Figura 3.9: Resultado articulación 3

Articulación θ_4: Se destaca una discrepancia notable en los primeros segundos de la simulación. La señal real presenta un comportamiento brusco y discontinuo. Sin embargo, posteriormente la señal se ajusta a la trayectoria deseada con mayor precisión.

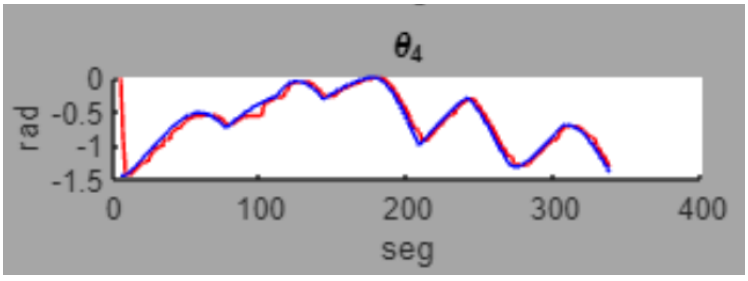


Figura 3.10: Resultado articulación 4

Articulación θ_5: Esta articulación corresponde a la parte final del brazo robótico, donde se encuentra el marcador que dibuja la trayectoria. En la gráfica se observa que la señal es constante a lo largo del tiempo y la única presente es la trayectoria azul (deseada). Esta condición es intencional, para simplificar el análisis, ya que la tarea no requería movimientos en ese eje.

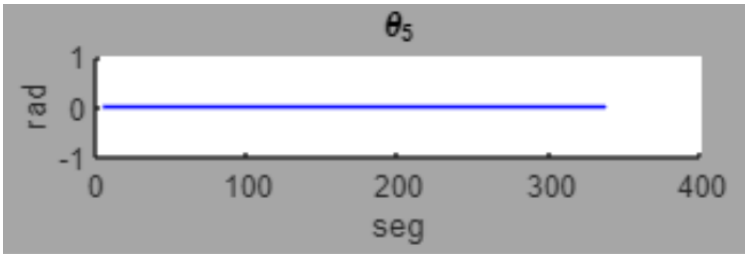


Figura 3.11: Resultado articulación 5

Verificación de la trayectoria pentagonal de manera simulada y real

En esta sección se presentan dos evidencias complementarias del seguimiento de trayectoria ejecutado por el brazo robótico. La Figura 3.12 muestra la representación simulada del movimiento dentro de la interfaz del sistema, donde se visualiza la trayectoria planificada y su ejecución controlada mediante el algoritmo LQG. Por su parte, la Figura 3.13 evidencia la realización física de dicha trayectoria en el entorno real, confirmando que el comportamiento del manipulador en la práctica se ajusta al esperado en simulación. Esta comparación respalda la validez del modelo de control implementado.

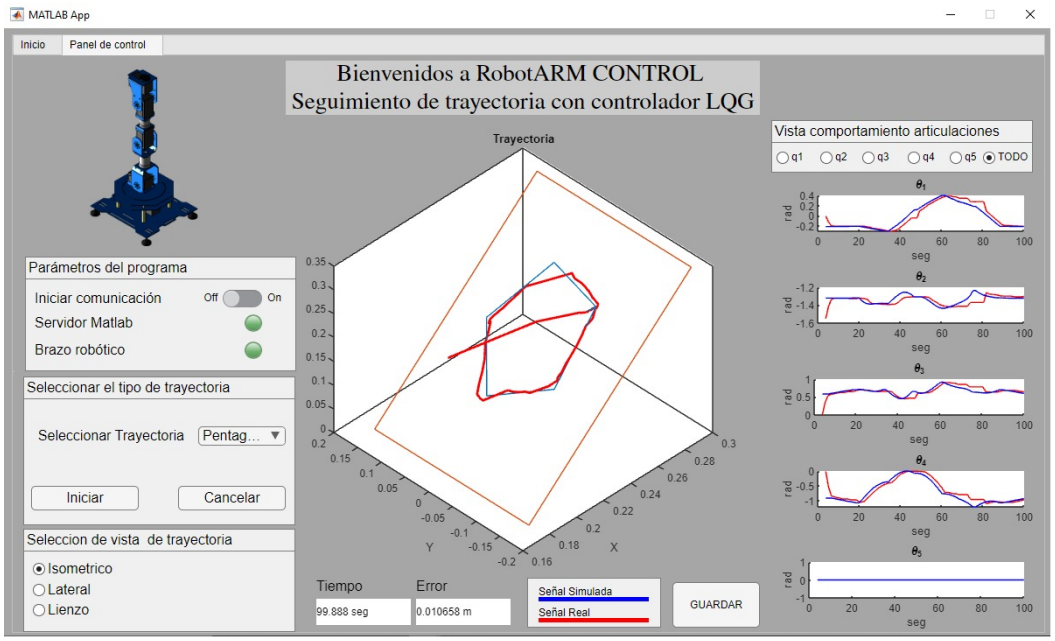


Figura 3.12: Seguimiento de trayectoria pentagonal en la interfaz del sistema con controlador LQG

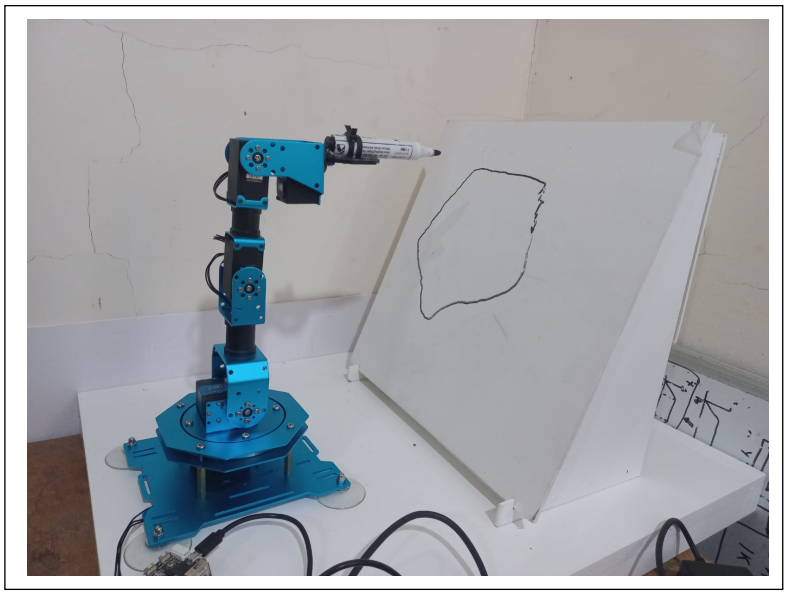


Figura 3.13: Seguimiento de trayectoria pentagonal de manera física

Validación de la trayectoria de doble cuadrado de manera simulada y real

Para la trayectoria correspondiente a la figura de doble cuadrado, se realizó una verificación integral del desempeño del sistema en dos entornos distintos. En primer lugar, la Figura 3.14 presenta el seguimiento realizado en la interfaz gráfica, lo cual permite observar el control y la respuesta del sistema en el entorno virtual. Posteriormente, en la Figura 3.15, se documenta la ejecución física del movimiento, evidenciando la capacidad del manipulador para reproducir con precisión la trayectoria compleja bajo condiciones reales. Para la validación del controlador y la coherencia entre simulación y práctica.

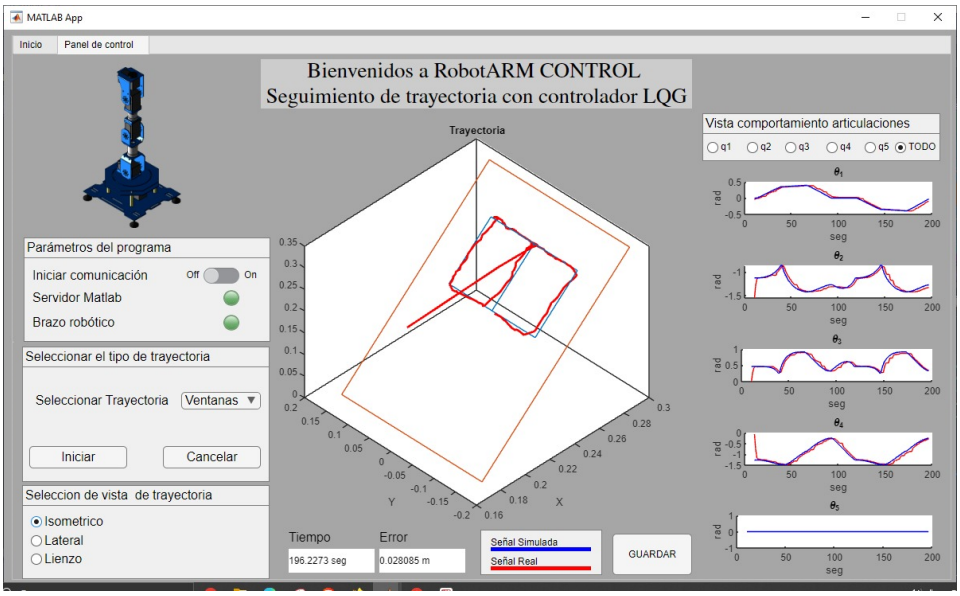


Figura 3.14: Seguimiento de trayectoria de cuadrado doble en la interfaz del sistema con controlador LQG

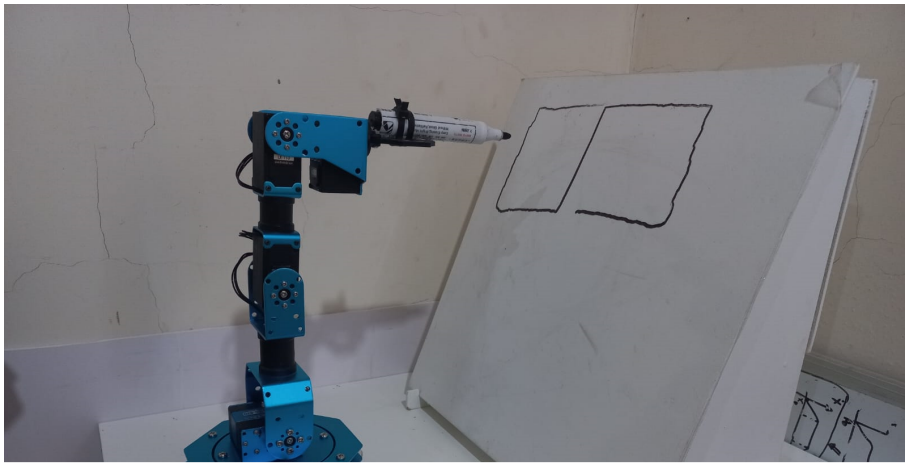


Figura 3.15: Seguimiento de trayectoria doble cuadrado de manera física

3.6 Análisis de datos de las trayectorias

Los datos tabulados representan mediciones en tiempo real exportadas desde la interfaz de control del sistema robótico. Cada registro contiene: Tiempo de muestreo ($t(s)$), Posiciones articulares reales (X_p, Y_p , Z_p), Valores de referencia (x_p, y_p, z_p), y Error (e). La adquisición se realizó a 50 Hz mediante el módulo de adquisición de datos del sistema de control LQG implementado.

Datos numéricos de la trayectoria triangular

La siguiente Tabla 3.2 presenta los datos numéricos correspondientes a la trayectoria triangular diseñada para el sistema. Estos valores incluyen parámetros relevantes que describen los puntos clave del recorrido, los cuales serán utilizados para evaluar y analizar el desempeño del controlador en el seguimiento de dicha trayectoria.

Tabla 3.2: Resultados del comportamiento de la trayectoria triangular.

$t(s)$	X_p	Y_p	Z_p	x_p	y_p	z_p	e
6,6334	0,2440	−0,1006	0,1830	0,1736	−0,0015	0,2560	0,1418
8,0401	0,2440	−0,1006	0,1830	0,2285	−0,0556	0,2180	0,0591
9,4103	0,2440	−0,1006	0,1830	0,2410	−0,0919	0,1904	0,0118
10,7774	0,2440	−0,1006	0,1830	0,2415	−0,0980	0,1888	0,0068
12,1556	0,2440	−0,1006	0,1830	0,2415	−0,0991	0,1898	0,0074
13,5060	0,2440	−0,1006	0,1830	0,2413	−0,1002	0,1895	0,0071
14,8377	0,2440	−0,1006	0,1830	0,2417	−0,0992	0,1895	0,0070
16,1485	0,2440	−0,1006	0,1830	0,2413	−0,1002	0,1895	0,0071
17,4974	0,2440	−0,1006	0,1830	0,2413	−0,1002	0,1895	0,0071
18,7725	0,2440	−0,1006	0,1830	0,2413	−0,1002	0,1895	0,0071

Tabla 3.2 continuación de la página anterior							
20,1423	0,2440	−0,1006	0,1830	0,2417	−0,1004	0,1884	0,0059
21,4844	0,2440	−0,1006	0,1830	0,2413	−0,1003	0,1885	0,0061
22,8065	0,2385	−0,0839	0,1679	0,2418	−0,0993	0,1885	0,0258
24,0878	0,2330	−0,0672	0,1529	0,2413	−0,1003	0,1902	0,0506
25,3447	0,2275	−0,0505	0,1378	0,2363	−0,0959	0,1782	0,0614
26,6036	0,2220	−0,0337	0,1227	0,2308	−0,0815	0,1655	0,0647
27,9480	0,2166	−0,0170	0,1076	0,2248	−0,0597	0,1461	0,0581
29,2479	0,2111	−0,0003	0,0926	0,2217	−0,0462	0,1339	0,0627
30,5792	0,2056	0,0164	0,0775	0,2217	−0,0462	0,1339	0,0858
31,9194	0,2001	0,0332	0,0624	0,2217	−0,0462	0,1339	0,1089
33,2127	0,1946	0,0499	0,0474	0,2217	−0,0462	0,1339	0,1321
34,5336	0,1946	0,0499	0,0474	0,1983	0,0217	0,0636	0,0327
35,8286	0,2023	0,0554	0,0685	0,1891	0,0402	0,0420	0,0333
37,2002	0,2100	0,0609	0,0897	0,1863	0,0429	0,0360	0,0614
38,5319	0,2177	0,0665	0,1108	0,1956	0,0502	0,0550	0,0622
39,8587	0,2254	0,0720	0,1320	0,2047	0,0572	0,0755	0,0619
41,1165	0,2331	0,0775	0,1531	0,2136	0,0616	0,0997	0,0591
42,4491	0,2408	0,0830	0,1743	0,2167	0,0654	0,1077	0,0729
43,8000	0,2485	0,0886	0,1954	0,2278	0,0730	0,1284	0,0718
45,1473	0,2562	0,0941	0,2165	0,2412	0,0784	0,1563	0,0640
46,4710	0,2639	0,0996	0,2377	0,2510	0,0850	0,1844	0,0567
47,8358	0,2639	0,0996	0,2377	0,2554	0,0901	0,2087	0,0317
49,1106	0,2617	0,0774	0,2316	0,2554	0,0901	0,2087	0,0270
50,3768	0,2595	0,0551	0,2255	0,2676	0,0982	0,2243	0,0439
51,7003	0,2573	0,0329	0,2195	0,2676	0,0982	0,2243	0,0664
52,9830	0,2550	0,0106	0,2134	0,2668	0,0709	0,2124	0,0614
54,3286	0,2528	−0,0116	0,2073	0,2651	0,0460	0,2067	0,0589
55,6057	0,2506	−0,0339	0,2012	0,2620	0,0154	0,2013	0,0506
56,9485	0,2484	−0,0561	0,1951	0,2587	−0,0033	0,1968	0,0539
58,2314	0,2462	−0,0784	0,1891	0,2560	−0,0291	0,1951	0,0506
59,5230	0,2440	−0,1006	0,1830	0,2530	−0,0505	0,1860	0,0511

Análisis cuantitativo:

Métrica	**Valor**
Error máximo	0,3069 m
Error mínimo	0,0027 m
Error promedio	0,0553 m
Desviación estándar	0,0635 m
Tiempo de estabilización (a 5 %)	3,21 s

Los percentiles del error muestran que:

- 50 % de las muestras $< 0{,}0508\,\text{m}$
- 95 % de las muestras $< 0{,}0970\,\text{m}$
- 5 % de las muestras $> 0{,}0029\,\text{m}$ (transitorios iniciales)

Análisis cualitativo:

Los resultados obtenidos muestran que el sistema controlado con LQG tiene un comportamiento bastante bueno durante el seguimiento de la trayectoria. El error promedio fue de aproximadamente 5,5 cm, lo que indica que, en general, el brazo robótico siguió la trayectoria de forma aceptable. Aunque hubo cierta variación en el error, esta se mantuvo dentro de límites manejables, y solo se observaron errores más altos en momentos puntuales, principalmente al inicio del movimiento.

Además, los valores del error permiten comprender mejor cómo se comportó el sistema a lo largo del tiempo: la mitad de las mediciones presentó errores menores a 5,08 cm, y el 95 % de las mediciones se mantuvo por debajo de 9,7 cm. Solo un pequeño porcentaje (5 %) mostró errores superiores a 2,9 cm, lo cual se relaciona con los primeros instantes de funcionamiento. También se observó que el sistema tarda aproximadamente 3,21 segundos en estabilizarse, lo cual representa un tiempo razonable para este tipo de aplicación. En conjunto, estos datos confirman que el controlador LQG es eficaz, aunque existe margen de mejora en la respuesta inicial y en la reducción de la variabilidad del error.

Datos numéricos de la trayectoria pentagonal

Se presentan a continuación los análisis cuantitativo y cualitativo correspondiente al seguimiento de la trayectoria pentagonal diseñada para el sistema. Los datos numéricos recopilados describen parámetros relevantes de los puntos clave del recorrido, y permiten evaluar el desempeño del controlador implementado. A partir de estos registros, se analiza la precisión del seguimiento, la estabilidad del sistema y la calidad de la respuesta dinámica frente a dicha trayectoria.

Análisis cuantitativo:

Métrica	Valor
Error máximo	0,0342 m
Error mínimo	0,0028 m
Error promedio	0,0297 m
Desviación estándar	0,0430 m
Tiempo de estabilización (a 5 %)	6,89 s

Los percentiles del error muestran que:

- 50 % de las muestras $<$ 0,0035 m
- 95 % de las muestras $<$ 0,0330 m
- 5 % de las muestras $>$ 0,0030 m (transitorios iniciales)

Análisis cualitativo:

El desempeño del sistema controlado con LQG fue adecuado en el seguimiento de la trayectoria deseada. El error promedio fue de aproximadamente 2.97 cm, mientras que el error máximo se mantuvo en torno a 3.42 cm, lo cual indica que el sistema operó dentro de márgenes aceptables en todo momento.

Se observó que, tras un tiempo de estabilización de aproximadamente 6.89 segundos, el error se mantuvo estable y reducido. La desviación estándar indica que las variaciones fueron contenidas y no representaron un riesgo significativo para el seguimiento de trayectoria.

Datos numéricos de la trayectoria en forma de estrella

Se presentan los análisis cuantitativo y cualitativo correspondiente al seguimiento de la trayectoria con forma de estrella. Los datos recopilados contienen información de los puntos de referencia trazados durante el movimiento, lo cual permite evaluar el comportamiento del controlador y analizar la estabilidad y capacidad de respuesta.

Análisis cuantitativo

Métrica	Valor
Error máximo	0,0342 m
Error mínimo	0,0028 m
Error promedio	0,0184 m
Desviación estándar	0,0085 m
Tiempo de estabilización (a 5 %)	5,7 s

Los percentiles del error muestran que:

- 50 % de las muestras $<$ 0,016 m
- 95 % de las muestras $<$ 0,033 m

- 5 % de las muestras > 0,030 m (transitorios iniciales)

Análisis cualitativo:

El desempeño del sistema controlado con LQG para el seguimiento de la trayectoria en forma de estrella tuvo un error promedio que fue de aproximadamente 1.84 cm, mientras que el error máximo alcanzó 3.42 cm, lo que indica que el sistema mantuvo un seguimiento dentro de límites.

El tiempo de estabilización estimado en 5.7 segundos refleja una respuesta rápida y efectiva del controlador para minimizar desviaciones. La desviación estándar baja muestra que el error se mantuvo bastante constante, sin grandes oscilaciones.

3.7 Análisis gráfico del desempeño del sistema basado en datos numéricos

A continuación, se realiza un análisis del desempeño del sistema de seguimiento de trayectoria controlado mediante un algoritmo LQG a través de gráficas que comparan la trayectoria deseada con la trayectoria ejecutada, tanto en simulación como en condiciones reales. El análisis gráfico complementa los resultados numéricos y proporciona una visión más clara del desempeño general del controlador.

Análisis gráfico de la trayectoria triangular a partir de registros numéricos

La Figura. 3.16 muestra la comparación entre el recorrido que debería seguir el sistema (trayectoria ideal, en color azul) y el que realmente siguió (trayectoria real, en color amarillo). Sin embargo, al observar la trayectoria real, se notan diferencias importantes. La más evidente ocurre en la base del triángulo, donde en lugar de seguir una línea recta como se esperaba, el recorrido real se curva. Esto sugiere que el sistema tuvo dificultades para seguir con precisión la trayectoria programada.

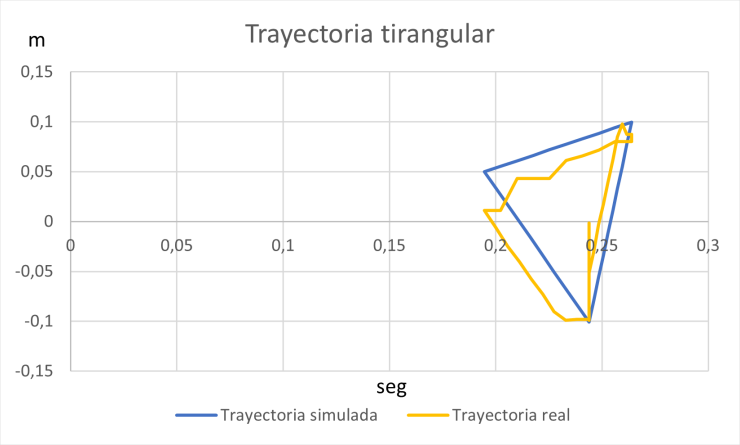


Figura 3.16: Trayectoria real comportamiento del sistema real frente a la trayectoria de referencia triangular.

Análisis gráfico de la trayectoria pentagonal a partir de registros numéricos

En la Figura 3.17 se puede observar el comportamiento del sistema al intentar seguir un patrón geométrico definido. En ella, la línea azul representa el recorrido ideal con forma de pentágono, conformado por segmentos rectos. Esta trayectoria sirve como referencia del desplazamiento esperado.

Por su parte, la trayectoria real, marcada en color amarillo, muestra un seguimiento general del contorno, aunque con claras diferencias en varios tramos. Se evidencian pequeñas ondulaciones en los lados rectos, y en las esquinas, el sistema no logra reproducir adecuadamente los vértices, lo que da lugar a trayectorias redondeadas o desviadas. El sistema logra mantener una forma general reconocible.

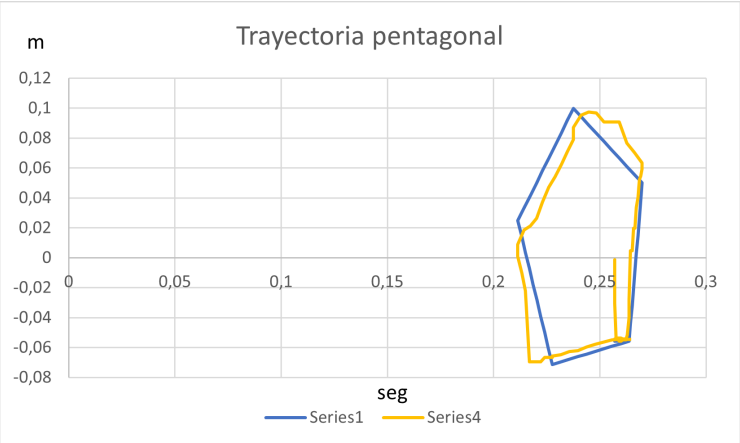


Figura 3.17: Trayectoria real comportamiento del sistema real frente a la trayectoria de referencia pentagonal.

Análisis gráfico de la trayectoria estrella a partir de registros numéricos

La Figura. 3.18 muestra que, por la parte de la trayectoria real, sigue la forma de la estrella, pero presenta pequeñas desviaciones a lo largo del trayecto. Estas diferencias son más evidentes en los puntos donde hay cambios bruscos de dirección, como en los picos y valles de la figura, donde el sistema tiende a suavizar las esquinas o desviarse ligeramente del trazo deseado.

A pesar de estas imperfecciones, existe una buena correspondencia global entre ambas trayectorias, lo que indica que el sistema realiza un seguimiento aceptable.

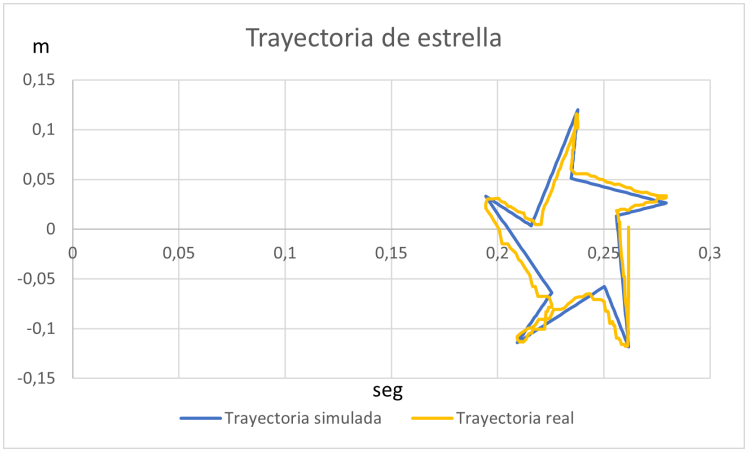


Figura 3.18: Trayectoria real comportamiento del sistema real frente a la trayectoria de referencia estrella.

4 Conclusiones y recomendaciones

En este capítulo se presentan las conclusiones generales del desarrollo del sistema de control basado en el algoritmo LQG, aplicado al seguimiento de trayectoria de un brazo robótico de cinco grados de libertad. Las conclusiones se fundamentan en el cumplimiento de los objetivos establecidos, así como en el análisis de los resultados obtenidos durante las etapas de modelado, simulación e implementación.

4.1 Conclusiones

A lo largo del desarrollo del sistema de control basado en el algoritmo LQG, se logró cumplir con el objetivo principal de implementar una estrategia de control óptimo para el seguimiento de trayectorias en un brazo robótico de cinco grados de libertad. Las pruebas en simulación demostraron un comportamiento estable y preciso del sistema. Sin embargo, la transición al entorno físico evidenció ciertas limitaciones técnicas que influyeron en el rendimiento real del sistema.

La formulación del modelo dinámico y cinemático mediante el método de Euler-Lagrange fue importante en el diseño del controlador. Este modelo permitió expresar las ecuaciones del sistema en espacio de estados, facilitando la aplicación del regulador LQR junto con el filtro de Kalman. No obstante, se identificó que la falta de consideración de ciertos factores reales, como la fricción y el peso específico de los componentes, limitó la exactitud de las simulaciones al momento de replicarlas en el entorno físico.

En las simulaciones realizadas en MATLAB/Simulink, el controlador LQG mostró un desempeño eficaz al seguir trayectorias suaves, manteniendo errores mínimos en la posición de las articulaciones. A pesar de ligeras desviaciones en θ_2 y θ_3, el sistema demostró capacidad de corrección y estabilidad. Sin embargo, en el entorno real, se observaron dificultades para mantener esa precisión debido a la latencia en la comunicación con los actuadores.

Los resultados obtenidos en el análisis de la cinemática directa e inversa confirmaron la validez del modelo geométrico del brazo robótico. Las posiciones alcanzadas por el efector final coincidieron con las trayectorias deseadas en las simulaciones. Sin embargo, en la implementación física, los errores de posicionamiento fueron mayores debido a retrasos en la respuesta de los servomotores y a las limitaciones en la retroalimentación de los sensores.

En las pruebas experimentales indican que el controlador LQG permitió un seguimiento preciso y estable de trayectorias en el brazo robótico, con errores promedio de 0,0553 m, 0,0297 m y 0,0184 m para las trayectorias circular, pentagonal y en forma de estrella, respectivamente. La menor desviación estándar se obtuvo en la trayectoria estrellada (0,0085 m), lo que evidencia una alta consistencia en el seguimiento.

El error máximo fue de 0,3069 m en la trayectoria circular, asociado a los transitorios iniciales, mientras que en las otras dos trayectorias no superó los 0,0342 m. Además, el 95 % de los errores se mantuvo por debajo de 0,0970 m, 0,0330 m y 0,0330 m respectivamente.

Respecto al tiempo de estabilización, el sistema requirió 3,21 s, 6,89 s y 5,7 s para las trayectorias circular, pentagonal y estrella, lo que indica una respuesta dinámica aceptable.

En conclusión, el controlador LQG mostró un desempeño robusto y eficaz en escenarios con distintas exigencias geométricas, siendo especialmente destacable su precisión en trayectorias complejas. No obstante, se identifican oportunidades de mejora en la etapa inicial del movimiento, con el fin de reducir el error máximo y acelerar la estabilización.

4.2 Recomendaciones

Se recomienda considerar el uso de servomotores con mejores características de realimentación y mayor precisión que los integrados en el robot XArm 1S. Este robot emplea servomotores digitales diseñados para aplicaciones educativas, con sensores integrados de resolución y velocidad limitadas, además de estar conectados en serie mediante un bus de comunicación digital. Esta configuración genera retrasos acumulativos en la transmisión de señales que afectan significativamente la respuesta del sistema. Estas limitaciones impactan el desempeño del controlador LQG, que requiere retroalimentación precisa y de baja latencia para optimizar el seguimiento de trayectoria.

Como sugerencia para mejorar la arquitectura actual, se recomienda considerar el uso de aislamiento eléctrico entre la Raspberry Pi y los motores, a fin de proteger los pines GPIO y evitar posibles interferencias. También sería conveniente separar las fuentes de alimentación de lógica y potencia, y, en caso de utilizar varios canales de comunicación, integrar un sistema de multiplexa-

do o un microcontrolador intermedio para facilitar la gestión. Finalmente, una mejor organización del cableado puede contribuir a una mayor estabilidad y fiabilidad del sistema.

Se sugiere una posible arquitectura en la que una unidad central, como una Raspberry Pi, se encargue de enviar comandos de control a múltiples actuadores o motores a través de líneas de comunicación serial. Para ello, se podrían utilizar resistencias limitadoras de $2\,\mathrm{k}\Omega$ en las líneas de señal, con el fin de proteger los pines de salida de la unidad de control ante posibles picos de voltaje o interferencias.

Cada motor podría recibir instrucciones codificadas en un protocolo serial simple, ejecutando acciones específicas conforme a las órdenes recibidas. Esta arquitectura permitiría controlar varios actuadores desde un solo nodo maestro, facilitando aplicaciones de sincronización de movimientos o seguimiento de trayectorias.

Asimismo, se recomienda mantener separadas las fuentes de alimentación de la lógica de control y de los motores, para evitar interferencias eléctricas y mejorar la estabilidad del sistema. En caso de requerir una mayor cantidad de actuadores, se podría considerar el uso de un multiplexor UART o de microcontroladores intermedios que actúen como nodos secundarios, reduciendo la carga de procesamiento sobre la unidad central.

Esta propuesta podría servir como base para futuras implementaciones, ofreciendo una solución flexible y escalable para sistemas de control distribuido en aplicaciones educativas, experimentales o de automatización básica.

Bibliografía

[1] R. Schilling, *Fundamentals of Robotics*. Prentice-Hall Inc, 1997.

[2] A. Emilov Goranov, "Control de un brazo robot con articulaciones elásticas." Tesis de titulación, Universitat Politècnica de València, 2018.

[3] Z. CARREÑO and J. Jorge, "Control robusto de la dinámica de un robot," *Computing*, pp. 173–176, 2009.

[4] L. E. García Jaimes and D. Piedrahíta Flórez, "Diseño de un controlador lineal cuadrático gaussiano y un controlador autosintonizado por asignación de polos para el control del voltaje de un generador cd," *Ingeniería*, vol. 26, no. 1, pp. 25–40, 2021.

[5] A. M. A. Ghiet and A. Baba, "Robot arm control with arduino," *Univ. Turkish Aeronaut. Assoc*, 2017.

[6] JulioH.Braslavsky, "Control automático 2." Automatizacion y Control Industrial, Universidad Nacional de Quilmes, 2001.

[7] R. I. G. Ricardo Garibay Jiménez, "Controlador lqg experimental de propÓsito didÁctico," *AMCA*, 2007.

[8] G. Singh Rani, "Integración de sensores de proximidad en brazo robot comercial," B.S. thesis, Universitat Politècnica de Catalunya, 2022.

[9] I. C. Villa Escudero, "Modelamiento y simulación de un algoritmo para el control del brazo robótico.," Master's thesis, Escuela Superior Politécnica de Chimborazo, 2017.

[10] A. P. G, ed., *Cinemática y Dinámica de Robots Móviles con Ruedas*. ILADI, 2023.

[11] O. R. Ponce, "Conceptos básicos - fundamentos de robótica," 2020. Accedido el 5 de abril de 2025.

[12] A. Fernández-Pacheco, *Robótica educativa.* RA-MA, 2015.

[13] I. The MathWorks, *Simscape Multibody User's Guide*. The MathWorks, Inc., 2025.

[14] S. Plan, "Tipos de articulaciones robóticas — dorna robotics," 5 2024. [Online; accessed 2025-03-31].

[15] N. P. Navarro Narváez, "Modelado cinemático y dinámico de un manipulador de 5 grados de libertad articulado verticalmente," *Pontificia Universidad Católica del Perú*, 2011.

[16] S. K. Saha, *Introducción a la Robótica*. McGraw-Hill España, 1 ed., 2011.

[17] J. J. Craig, "Introduction to robotics: Mechanics and control," 2005.

[18] C. R. Daza, "Repositorio de tesis ucsm," 2014.

[19] B. Siciliano, "Force control," 2009.

[20] G. A. Y. Torres, "Diseño e implementación de una interfaz de comunicación para el control de posición de un brazo robótico de forma local y remota," 2024.

[21] J. J. Graig, "Robótica," 2010.

[22] B. A, "Fundamentos de robótica," 2012.

[23] R. Balasubramanian, "The denavit hartenberg convention," *USA: Robotics Insitute Carnegie Mellon University*, 2011.

[24] C. D. D. Bravo, "Diseño mecánico y generación de trayectoria para un robot antropomórfico de 6 grados de libertad," 2022.

[25] K. Ogata, *Ingeniería de Control Moderna*. Pearson, 5 ed., 2010.

[26] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Boston, MA, USA: Pearson, 7 ed., 2015.

[27] W. J. Rugh, *Linear System Theory*. Upper Saddle River, NJ, USA: Prentice Hall, 2 ed., 1996.

[28] R. F. Stengel, *Optimal Control and Estimation*. Mineola, NY, USA: Dover Publications, 1994.

[29] A. E. Bryson and Y.-C. Ho, *Applied Optimal Control: Optimization, Estimation and Control*. New York, NY, USA: Hemisphere Publishing, 1975.

[30] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*. Pearson, 7 ed., 2014.

[31] L. A. V. Mellado, "Diseño de un sistema de control basado en linealización por realimentación para un robot móvil tipo ackerman con velocidad variable y movimiento en doble sentido describiendo trayectorias óptimas," 2017.

[32] B. D. O. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Courier Corporation, 2007.

[33] D. E. Kirk, *Optimal Control Theory: An Introduction*. Dover Publications, 2012.

[34] W. M. Wonham, "On a matrix riccati equation of stochastic control," *SIAM Journal on Control*, vol. 6, no. 4, pp. 681–697, 1968.

[35] R. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[36] U. of Toronto, "Kalman filter lecture notes." Available at: `https://www.control.utoronto.ca/~broucke/ece1505/kalman.pdf`.

[37] K. Zhou, J. C. Doyle, and K. Glover, *Essentials of Robust Control*. Upper Saddle River, NJ: Prentice Hall, 1998.

[38] I. The MathWorks, "Matlab documentation," 2023.

[39] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, 2nd ed., 2017.

[40] I. The MathWorks, "Ros toolbox documentation," 2023.

[41] N. Developers, "Numpy documentation," 2023.

[42] S. Community, "Scipy documentation," 2023.

[43] M. Developers, "Matplotlib documentation," 2023.

[44] P. C. S. L. Contributors, "Python control systems library," 2023.

[45] O. Robotics, "Robot operating system (ros)," 2023.

[46] M. Community, "Moveit! documentation," 2023.

[47] B. P. Library, "Pybullet physics simulation," 2023.

[48] UFactory, "xarm python sdk documentation," 2023.

[49] S. P. Contributors, "Spyder ide documentation," 2023.

[50] S. Electronics, "Understanding uart communication," 2021. Accedido en mayo de 2025.

[51] Udaytdf, "Protocolos de comunicación spi, i2c, uart, uart." https://udaytdf.wixsite.com/udaytdf/post/protocolos-de-comunicacin-spi-i2c-uart-uart. Accessed: 2025-06-01.

[52] U. I. Forum, "Universal serial bus (usb) overview," 2020. Accedido en mayo de 2025.

[53] Keil, "USB Communication Device Class - Communication Model." https://www.keil.com/support/man/docs/rlarm/rlarm_usb_com_model.htm. Accessed: 2025-06-01.

[54] ELECTROINDUSTRIA, "Introducción histórica del control automático."

[55] I. K. Mohammed and M. N. Noaman, "Optimal control approach for robot system using lqg technique," *Journal Européen des Systèmes Automatisés*, vol. 55, no. 5, p. 671, 2022.

[56] J. Craig, *Introduction to Robotics: Mechanics and Control*. Addison-Wesley Publishing Company Inc, 1989.

[57] J. C. Grandas Franco and C. Borrás Pinilla, "On the dynamic and control for a three degree-of-freedom robotic arm used for rehabilitation purposes in medicine," in *ASME International Mechanical Engineering Congress and Exposition*, vol. 85628, p. V07BT07A044, American Society of Mechanical Engineers, 2021.

[58] Hiwonder, "Lx-15d."

[59] Hiwonder, "Lx-225."

[60] Spyder Development Team, "Editor — Spyder 5 Documentation," 2025. Accedido: 6 de junio de 2025.

[61] Hiwonder, "Serial bus servo controller," 2025. Accessed: Jun. 4, 2025.

A Anexos

A.1 Cinemática Directa

```
clear
clc
format long
%Calculo de la matriz de transformación directa.

% syms    theta_1 theta_2 theta_3 theta_4 theta_5 ...
%         d_1 d_2 d_3 d_4 d_5 ...
%         a_1 a_2 a_3 a_4 a_5 ...
%         alpha_1 alpha_2 alpha_3 alpha_4 alpha_5

% Encabezado
joint   = {'1'; '2'; '3';'4'; '5';};

% Con todas las variables y parametros
%_____________________________________
% theta_i = [theta_1; theta_2; theta_3; theta_4; theta_5];
% d_i     = [d_1; d_2; d_3; d_4; d_5];
% a_i     = [a_1; a_2; a_3; a_4; a_5];
% alpha_i = [alpha_1; alpha_2; alpha_3; alpha_4; alpha_5];

% Solo con parámetros del robot Xarm y variables
%_____________________________________________
% v_theta_i = [theta_1; theta_2; theta_3; theta_4; theta_5];
% v_d_i     = [d_1; 0; 0; 0; d_5];
% v_a_i     = [0; a_2; a_3; 0; 0];
% v_alpha_i = [-pi/2; 0; 0; -pi/2; 0];

% Ubicar valor a los parámetros y variables
%__________________________________________
 q1 = [0 0 0 0 0]; %Especificar posiciones angulares
 v_theta_i = q1';

 %Ubicar los parámetros
 %_____________________________________

 v_d_i     = [0.0650; 0; 0; 0; 0.168];
 v_a_i     = [0; 0.1010; 0.0950; 0; 0];
 v_alpha_i = [-pi/2; 0; 0; -pi/2; 0];

% Crear la tabla con nombres de columnas
T = table(joint, v_theta_i, v_d_i, v_a_i, v_alpha_i, ...
    'VariableNames', {'Joint', 'theta_i', 'd_i', 'a_i', 'alpha_i'});
% Número de articulaciones
n = 5;

% Crear una celda para almacenar las matrices
T_i = cell(1, n);

for i = 1:n
    theta_i = v_theta_i(i);
    d_i = v_d_i(i);
    a_i = v_a_i(i);
    alpha_i = v_alpha_i(i);
    %MATRICES DE TRANSFORMARCION SEGUN (3.1)
    % Matriz de Rotación alrededor de z por theta_i
    Rz_theta = [cos(theta_i), -sin(theta_i), 0, 0;
                sin(theta_i), cos(theta_i), 0, 0;
                0, 0, 1, 0;
                0, 0, 0, 1];

    % Matriz de Traslación a lo largo de z por d_i
    Tz_d = [1, 0, 0, 0;
            0, 1, 0, 0;
            0, 0, 1, d_i;
            0, 0, 0, 1];

    % Matriz de Traslación a lo largo de x por a_i
    Tx_a = [1, 0, 0, a_i;
            0, 1, 0, 0;
            0, 0, 1, 0;
            0, 0, 0, 1];

    % Matriz de Rotación alrededor de x por alpha_i
    Rx_alpha = [1, 0, 0, 0;
                0, cos(alpha_i), -sin(alpha_i), 0;
                0, sin(alpha_i), cos(alpha_i), 0;
                0, 0, 0, 1];

    % Calcular la transformación homogénea i-1 A_i
    T_i{i} = Rz_theta * Tz_d * Tx_a * Rx_alpha;

    T_i{1}(1,2) = 0;T_i{1}(2,2) = 0;T_i{1}(3,3) = 0;
    T_i{4}(1,2) = 0;T_i{4}(2,2) = 0;T_i{4}(3,3) = 0;

    fprintf('Trasnformadas para theta %s:\n\n', string(joint(i)));
    disp(T_i{i})
end


    fprintf('Matriz de Trasnformada Homogénea\n\n');
    T_final = T_i{1}*T_i{2}*T_i{3}*T_i{4}*T_i{5};
    %T_final = simplify(T_final);
    disp(T_final)
```

Código Cod1: Código para cinemática directa en Matlab.

A.2 Cinemática Inversa

```
close all
clc


[col, fil] = size(points);
rotaciones = zeros(col,5);

for i=1:col
    %%VALORES PREDEFINIDOS
    Px = points(i, 1);
    Py = points(i, 2);
    Pz = points(i, 3);


    d1 = 0.0650;
    a2 = 0.1010;
    a3 = 0.0950;
    d5 = 0.168;

    DISTANCIA_PUNTO = sqrt((0-Px)^2 + (0-Py)^2 + (d1-Pz)^2);
    DISTANCIA_MAX = a2+a3+d5;   DISTANCIA_MIN = 0.180;

    GRADOS_MAX = 120;   GRADOS_MIN = 23;

    M =  (GRADOS_MAX - GRADOS_MIN)/(DISTANCIA_MAX - DISTANCIA_MIN);

    dQ = M * (DISTANCIA_PUNTO-DISTANCIA_MIN)+GRADOS_MIN;

    Q = deg2rad(dQ); %120 angulo maximo


    %% INICIA_CALCULO
    theta1 = atan(Py/Px);

    X4 = Px - d5 * sin(Q) * cos(theta1);

    Y4 = Py - d5 * sin(Q) * sin(theta1);

    Z4 = Pz + d5*cos(Q);

    comp = sqrt((X4-Px)^2 + (Y4-Py)^2 + (Z4-Pz)^2);

    l = sqrt(X4^2 + Y4^2);

    m = sqrt(l^2 + (Z4 - d1)^2);

    n = acos((m^2 + a2^2 - a3^2)/(2*m*a2));

    if  (Z4 - d1)<0
    theta2 = -(n - atan( abs(Z4 - d1)/l ) );
    else
        theta2 = -(atan((Z4 - d1)/l) + n);
    end


    theta3 = pi - acos((a2^2 + a3^2 - m^2)/(2*a2*a3));

    if  (Z4 - d1)<0
        theta4 = - ((pi-atan(l/abs(Z4 - d1)) + acos((m^2 + a3^2 - a2^2)/(2*m*a3)) + Q) - pi/2);
    else
        theta4 = - ((atan(l/(Z4 - d1)) + acos((m^2 + a3^2 - a2^2)/(2*m*a3)) + Q) - pi/2);
    end

    %Guardamos todas las posiciones de las articulaciones
    rotaciones(i, 1) = theta1;
    rotaciones(i, 2) = theta2;
    rotaciones(i, 3) = theta3;
    rotaciones(i, 4) = theta4;
    rotaciones(i, 5) = 0;

end
```

Código Cod2: Código para cinemática inversa en Matlab.

A.3 Peso de Eslabones

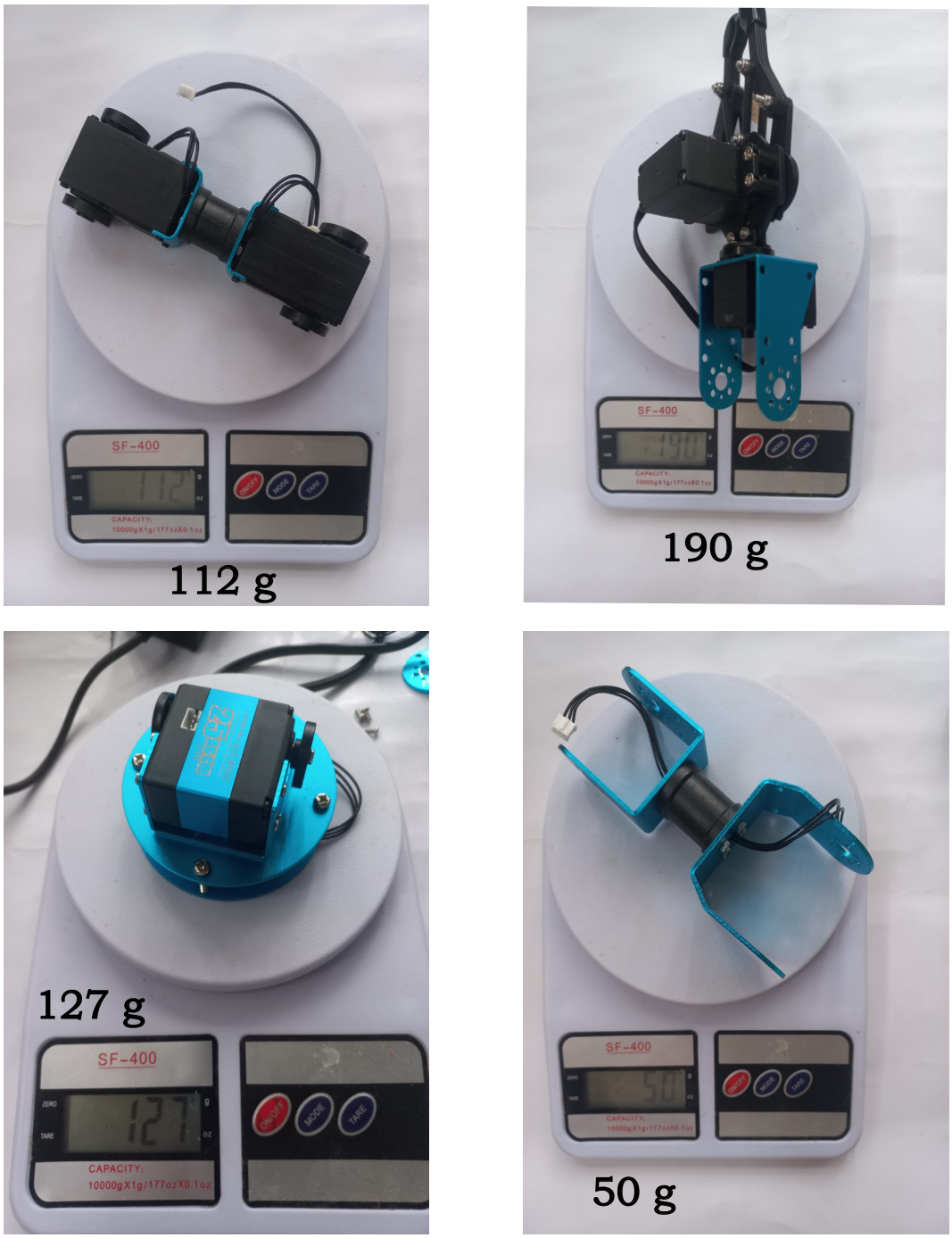


Figura Im1: Imágenes correspondientes al pesaje de las partes principales del brazo robótico utilizado en el presente proyecto.

A.4 Matrices de Modelo Dinámico

A.4.1 Matriz de inercia

$$
\begin{aligned}
M(1,1) =& \frac{I_{2xx}}{2} + I_{1yy} + \frac{I_{3xx}}{2} + \frac{I_{2yy}}{2} + \frac{I_{3yy}}{2} + \frac{I_{5xx}}{4} + \frac{I_{5yy}}{4} + \frac{I_{5zz}}{2} \quad \cdots \\
& - \frac{I_{5xx}\,\cos(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4)}{4} - \frac{I_{5yy}\,\cos(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4)}{4} + \frac{I_{5zz}\,\cos(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4)}{2} \quad \cdots \\
& - \frac{I_{2xx}\,\cos(2\,\theta_2)}{2} + \frac{I_{2yy}\,\cos(2\,\theta_2)}{2} + \frac{I_{5xx}\,\cos(2\,\theta_5)}{4} - \frac{I_{5yy}\,\cos(2\,\theta_5)}{4} \quad \cdots \\
& - \frac{I_{5xx}\,\cos(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4 - 2\,\theta_5)}{8} - \frac{I_{5xx}\,\cos(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4 + 2\,\theta_5)}{8} \quad \cdots \\
& + \frac{I_{5yy}\,\cos(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4 - 2\,\theta_5)}{8} + \frac{I_{5yy}\,\cos(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4 + 2\,\theta_5)}{8} \quad \cdots \\
& + \frac{{l_2}^2\,m_3}{2} + \frac{{l_2}^2\,m_5}{2} + \frac{{l_3}^2\,m_5}{2} + \frac{{l_{c2}}^2\,m_2}{2} + \frac{{l_{c3}}^2\,m_3}{2} + \frac{{l_{c5}}^2\,m_5}{2} - \frac{I_{3xx}\,\cos(2\,\theta_2 + 2\,\theta_3)}{2} \quad \cdots \\
& + \frac{I_{3yy}\,\cos(2\,\theta_2 + 2\,\theta_3)}{2} - \frac{{l_{c5}}^2\,m_5\,\cos(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4)}{2} + \frac{{l_2}^2\,m_3\,\cos(2\,\theta_2)}{2} \quad \cdots \\
& + \frac{{l_2}^2\,m_5\,\cos(2\,\theta_2)}{2} + \frac{{l_{c2}}^2\,m_2\,\cos(2\,\theta_2)}{2} + \frac{{l_3}^2\,m_5\,\cos(2\,\theta_2 + 2\,\theta_3)}{2} \quad \cdots \\
& + \frac{{l_{c3}}^2\,m_3\,\cos(2\,\theta_2 + 2\,\theta_3)}{2} - l_3\,l_{c5}\,m_5\,\sin(\theta_4) - l_2\,l_{c5}\,m_5\,\sin(\theta_3 + \theta_4) \quad \cdots \\
& - l_2\,l_{c5}\,m_5\,\sin(2\,\theta_2 + \theta_3 + \theta_4) + l_2\,l_3\,m_5\,\cos(2\,\theta_2 + \theta_3) + l_2\,l_{c3}\,m_3\,\cos(2\,\theta_2 + \theta_3) \quad \cdots \\
& - l_3\,l_{c5}\,m_5\,\sin(2\,\theta_2 + 2\,\theta_3 + \theta_4) + l_2\,l_3\,m_5\,\cos(\theta_3) + l_2\,l_{c3}\,m_3\,\cos(\theta_3) \\
M(1,2) =& \frac{1}{4}\,\left(\cos(\theta_2 + \theta_3 + \theta_4 - 2\,\theta_5) - \cos(\theta_2 + \theta_3 + \theta_4 + 2\,\theta_5)\right)\,(I_{5xx} - I_{5yy}) \\
M(1,3) =& \frac{1}{4}\,\left(\cos(\theta_2 + \theta_3 + \theta_4 - 2\,\theta_5) - \cos(\theta_2 + \theta_3 + \theta_4 + 2\,\theta_5)\right)\,(I_{5xx} - I_{5yy}) \\
M(1,4) =& \frac{1}{4}\,\left(\cos(\theta_2 + \theta_3 + \theta_4 - 2\,\theta_5) - \cos(\theta_2 + \theta_3 + \theta_4 + 2\,\theta_5)\right)\,(I_{5xx} - I_{5yy}) \\
M(1,5) =& -\cos(\theta_2 + \theta_3 + \theta_4)\,I_{5zz}
\end{aligned}
\tag{Ec1}
$$

$$
M(2,1) = \frac{\left(\cos(\theta_2 + \theta_3 + \theta_4 - 2\,\theta_5) - \cos(\theta_2 + \theta_3 + \theta_4 + 2\,\theta_5)\right)\,(I_{5xx} - I_{5yy})}{4}
\tag{Ec2}
$$

$$
\begin{aligned}
M(2,2) =& \frac{I_{5xx}}{2} + I_{2zz} + I_{3zz} + \frac{I_{5yy}}{2} - \frac{I_{5xx}\,\cos(2\,\theta_5)}{2} + \frac{I_{5yy}\,\cos(2\,\theta_5)}{2} + {l_2}^2\,m_3 + {l_2}^2\,m_5 + {l_3}^2\,m_5 + \\
& {l_{c2}}^2\,m_2 + {l_{c3}}^2\,m_3 + {l_{c5}}^2\,m_5 - 2\,l_3\,l_{c5}\,m_5\,\sin(\theta_4) - 2\,l_2\,l_{c5}\,m_5\,\sin(\theta_3 + \theta_4) + \\
& 2\,l_2\,l_3\,m_5\,\cos(\theta_3) + 2\,l_2\,l_{c3}\,m_3\,\cos(\theta_3)
\end{aligned}
\tag{Ec3}
$$

$$
\begin{aligned}
M(2,3) =& m_5\,{l_3}^2 - 2\,m_5\,\sin(\theta_4)\,l_3\,l_{c5} + l_2\,m_5\,\cos(\theta_3)\,l_3 + m_3\,{l_{c3}}^2 + l_2\,m_3\,\cos(\theta_3)\,l_{c3} + m_5\,{l_{c5}}^2 - \\
& l_2\,m_5\,\sin(\theta_3 + \theta_4)\,l_{c5} + \frac{I_{5xx}}{2} + I_{3zz} + \frac{I_{5yy}}{2} - \frac{I_{5xx}\,\cos(2\,\theta_5)}{2} + \frac{I_{5yy}\,\cos(2\,\theta_5)}{2}
\end{aligned}
\tag{Ec4}
$$

$$
M(2,4) = I_{5xx} - I_{5xx}\,\cos(\theta_5)^2 + I_{5yy}\,\cos(\theta_5)^2 + {l_{c5}}^2\,m_5 - l_3\,l_{c5}\,m_5\,\sin(\theta_4) - l_2\,l_{c5}\,m_5\,\sin(\theta_3 + \theta_4)
\tag{Ec5}
$$

$$
M(2,5) = 0
\tag{Ec6}
$$

$$
M(3,1) = \frac{\left(\cos(\theta_2 + \theta_3 + \theta_4 - 2\,\theta_5) - \cos(\theta_2 + \theta_3 + \theta_4 + 2\,\theta_5)\right)\,(I_{5xx} - I_{5yy})}{4}
\tag{Ec7}
$$

$$
\begin{aligned}
M(3,2) =& m_5\,{l_3}^2 - 2\,m_5\,\sin(\theta_4)\,l_3\,l_{c5} + l_2\,m_5\,\cos(\theta_3)\,l_3 + m_3\,{l_{c3}}^2 + l_2\,m_3\,\cos(\theta_3)\,l_{c3} + m_5\,{l_{c5}}^2 - \\
& l_2\,m_5\,\sin(\theta_3 + \theta_4)\,l_{c5} + \frac{I_{5xx}}{2} + I_{3zz} + \frac{I_{5yy}}{2} - \frac{I_{5xx}\,\cos(2\,\theta_5)}{2} + \frac{I_{5yy}\,\cos(2\,\theta_5)}{2}
\end{aligned}
\tag{Ec8}
$$

$$
M(3,3) = I_{5xx} + I_{3zz} - I_{5xx}\,\cos(\theta_5)^2 + I_{5yy}\,\cos(\theta_5)^2 + {l_3}^2\,m_5 + {l_{c3}}^2\,m_3 + {l_{c5}}^2\,m_5 - 2\,l_3\,l_{c5}\,m_5\,\sin(\theta_4)
\tag{Ec9}
$$

$$
M(3,4) = I_{5yy} + I_{5xx}\,\sin(\theta_5)^2 - I_{5yy}\,\sin(\theta_5)^2 + {l_{c5}}^2\,m_5 - l_3\,l_{c5}\,m_5\,\sin(\theta_4)
\tag{Ec10}
$$

$$M(3,5) = 0 \tag{Ec11}$$

$$M(4,1) = \frac{(\cos(\theta_2 + \theta_3 + \theta_4 - 2\,\theta_5) - \cos(\theta_2 + \theta_3 + \theta_4 + 2\,\theta_5))\ (I_{5\mathrm{xx}} - I_{5\mathrm{yy}})}{4} \tag{Ec12}$$

$$M(4,2) = I_{5\mathrm{xx}} - I_{5\mathrm{xx}} \cos(\theta_5)^2 + I_{5\mathrm{yy}} \cos(\theta_5)^2 + {l_{\mathrm{c}5}}^2\, m_5 - l_3\, l_{\mathrm{c}5}\, m_5\ \sin(\theta_4) - l_2\, l_{\mathrm{c}5}\, m_5\ \sin(\theta_3 + \theta_4) \tag{Ec13}$$

$$M(4,3) = I_{5\mathrm{yy}} + I_{5\mathrm{xx}} \sin(\theta_5)^2 - I_{5\mathrm{yy}} \sin(\theta_5)^2 + {l_{\mathrm{c}5}}^2\, m_5 - l_3\, l_{\mathrm{c}5}\, m_5\ \sin(\theta_4) \tag{Ec14}$$

$$M(4,5) = 0 \tag{Ec15}$$

$$M(5,1) = -I_{5\mathrm{zz}}\ \cos(\theta_2 + \theta_3 + \theta_4) \tag{Ec16}$$

$$M(5,2) = 0 \tag{Ec17}$$

$$M(5,3) = 0 \tag{Ec18}$$

$$M(5,4) = 0 \tag{Ec19}$$

$$M(5,5) = I_{5\mathrm{zz}} \tag{Ec20}$$

A.4.2 Matriz de Coriolis y fuerzas centrífugas

$$
\begin{aligned}
C(1,1) = {} & \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\left(\dot{\theta_2}\right)^2}{4} - \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\left(\dot{\theta_2}\right)^2}{4} - \\
& \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\left(\dot{\theta_3}\right)^2}{4} + \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\left(\dot{\theta_3}\right)^2}{4} - \\
& \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\left(\dot{\theta_4}\right)^2}{4} + \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\left(\dot{\theta_4}\right)^2}{4} + \\
& \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\left(\dot{\theta_2}\right)^2}{4} - \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\left(\dot{\theta_2}\right)^2}{4} + \\
& \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\left(\dot{\theta_3}\right)^2}{4} - \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\left(\dot{\theta_3}\right)^2}{4} + \\
& \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\left(\dot{\theta_4}\right)^2}{4} - \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\left(\dot{\theta_4}\right)^2}{4} + \\
& \frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4-2\,\theta_5\right)\,\dot{\theta_2}\,\dot{\theta_1}}{4} + \frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4+2\,\theta_5\right)\,\dot{\theta_2}\,\dot{\theta_1}}{4} + \\
& \frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4-2\,\theta_5\right)\,\dot{\theta_3}\,\dot{\theta_1}}{4} + \frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4+2\,\theta_5\right)\,\dot{\theta_3}\,\dot{\theta_1}}{4} + \\
& \frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4-2\,\theta_5\right)\,\dot{\theta_4}\,\dot{\theta_1}}{4} + \frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4+2\,\theta_5\right)\,\dot{\theta_4}\,\dot{\theta_1}}{4} - \\
& \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4-2\,\theta_5\right)\,\dot{\theta_2}\,\dot{\theta_1}}{4} - \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4+2\,\theta_5\right)\,\dot{\theta_2}\,\dot{\theta_1}}{4} - \\
& \frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4-2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{4} + \frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4+2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{4} - \\
& \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4-2\,\theta_5\right)\,\dot{\theta_3}\,\dot{\theta_1}}{4} - \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4+2\,\theta_5\right)\,\dot{\theta_3}\,\dot{\theta_1}}{4} - \\
& \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4-2\,\theta_5\right)\,\dot{\theta_4}\,\dot{\theta_1}}{4} - \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4+2\,\theta_5\right)\,\dot{\theta_4}\,\dot{\theta_1}}{4} + \\
& \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4-2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{4} - \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4+2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{4} + \\
& I_{3xx}\,\sin\left(2\,\theta_2+2\,\theta_3\right)\,\dot{\theta_2}\,\dot{\theta_1} + I_{3xx}\,\sin\left(2\,\theta_2+2\,\theta_3\right)\,\dot{\theta_3}\,\dot{\theta_1} - I_{3yy}\,\sin\left(2\,\theta_2+2\,\theta_3\right)\,\dot{\theta_2}\,\dot{\theta_1} - \\
& I_{3yy}\,\sin\left(2\,\theta_2+2\,\theta_3\right)\,\dot{\theta_3}\,\dot{\theta_1} + I_{5zz}\,\sin\left(\theta_2+\theta_3+\theta_4\right)\,\dot{\theta_5}\,\dot{\theta_2} + I_{5zz}\,\sin\left(\theta_2+\theta_3+\theta_4\right)\,\dot{\theta_5}\,\dot{\theta_3} + \\
& I_{5zz}\,\sin\left(\theta_2+\theta_3+\theta_4\right)\,\dot{\theta_5}\,\dot{\theta_4} + \frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\dot{\theta_2}\,\dot{\theta_1}}{2} + \\
& \frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\dot{\theta_3}\,\dot{\theta_1}}{2} + \frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_1}}{2} + \\
& \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\dot{\theta_2}\,\dot{\theta_1}}{2} + \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\dot{\theta_3}\,\dot{\theta_1}}{2} + \\
& \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_1}}{2} - I_{5zz}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\dot{\theta_2}\,\dot{\theta_1} - \\
& I_{5zz}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\dot{\theta_3}\,\dot{\theta_1} - I_{5zz}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_1} + I_{2xx}\,\sin\left(2\,\theta_2\right)\,\dot{\theta_2}\,\dot{\theta_1} - \\
& I_{2yy}\,\sin\left(2\,\theta_2\right)\,\dot{\theta_2}\,\dot{\theta_1} - \frac{I_{5xx}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} + \frac{I_{5yy}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} - \\
& \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_3}\,\dot{\theta_2}}{2} + \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_3}\,\dot{\theta_2}}{2} - \\
& \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_4}\,\dot{\theta_2}}{2} + \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_4}\,\dot{\theta_2}}{2} + \\
& \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_2}}{2} - \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_4}\,\dot{\theta_3}}{2} + \\
& \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_2}}{2} + \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_4}\,\dot{\theta_3}}{2} + \\
& \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_3}\,\dot{\theta_2}}{2} - \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_3}\,\dot{\theta_2}}{2} + \\
& \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_3}}{2} + \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_3}}{2} + \\
& \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_4}\,\dot{\theta_2}}{2} - \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_4}\,\dot{\theta_2}}{2} + \\
& \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_4}}{2} + \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_4}}{2} - \\
& \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_2}}{2}
\end{aligned}
\tag{Ec21}
$$

$$\cdots + \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_4}\,\dot{\theta_3}}{2} - \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_2}}{2} -$$
$$\frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_4}\,\dot{\theta_3}}{2} - \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_3}}{2} -$$
$$\frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_3}}{2} - \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_4}}{2} -$$
$$\frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_4}}{2} - {l_3}^2\,m_5\,\sin\left(2\,\theta_2+2\,\theta_3\right)\,\dot{\theta_2}\,\dot{\theta_1} - {l_3}^2\,m_5\,\sin\left(2\,\theta_2+2\,\theta_3\right)\,\dot{\theta_3}\,\dot{\theta_1} -$$
$${l_{c3}}^2\,m_3\,\sin\left(2\,\theta_2+2\,\theta_3\right)\,\dot{\theta_2}\,\dot{\theta_1} - {l_{c3}}^2\,m_3\,\sin\left(2\,\theta_2+2\,\theta_3\right)\,\dot{\theta_3}\,\dot{\theta_1} +$$
$${l_{c5}}^2\,m_5\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\dot{\theta_2}\,\dot{\theta_1} + {l_{c5}}^2\,m_5\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\dot{\theta_3}\,\dot{\theta_1} +$$
$${l_{c5}}^2\,m_5\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_1} - {l_2}^2\,m_3\,\sin\left(2\,\theta_2\right)\,\dot{\theta_2}\,\dot{\theta_1} - {l_2}^2\,m_5\,\sin\left(2\,\theta_2\right)\,\dot{\theta_2}\,\dot{\theta_1} -$$
$${l_{c2}}^2\,m_2\,\sin\left(2\,\theta_2\right)\,\dot{\theta_2}\,\dot{\theta_1} - 2\,l_2\,l_{c5}\,m_5\,\cos\left(2\,\theta_2+\theta_3+\theta_4\right)\,\dot{\theta_2}\,\dot{\theta_1} - l_2\,l_{c5}\,m_5\,\cos\left(2\,\theta_2+\theta_3+\theta_4\right)\,\dot{\theta_3}\,\dot{\theta_1} -$$
$$l_2\,l_{c5}\,m_5\,\cos\left(2\,\theta_2+\theta_3+\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_1} - 2\,l_2\,l_3\,m_5\,\sin\left(2\,\theta_2+\theta_3\right)\,\dot{\theta_2}\,\dot{\theta_1} - l_2\,l_3\,m_5\,\sin\left(2\,\theta_2+\theta_3\right)\,\dot{\theta_3}\,\dot{\theta_1} -$$
$$2\,l_2\,l_{c3}\,m_3\,\sin\left(2\,\theta_2+\theta_3\right)\,\dot{\theta_2}\,\dot{\theta_1} - l_2\,l_{c3}\,m_3\,\sin\left(2\,\theta_2+\theta_3\right)\,\dot{\theta_3}\,\dot{\theta_1} -$$
$$2\,l_3\,l_{c5}\,m_5\,\cos\left(2\,\theta_2+2\,\theta_3+\theta_4\right)\,\dot{\theta_2}\,\dot{\theta_1} - 2\,l_3\,l_{c5}\,m_5\,\cos\left(2\,\theta_2+2\,\theta_3+\theta_4\right)\,\dot{\theta_3}\,\dot{\theta_1} -$$
$$l_3\,l_{c5}\,m_5\,\cos\left(2\,\theta_2+2\,\theta_3+\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_1} - l_3\,l_{c5}\,m_5\,\cos\left(\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_1} - l_2\,l_3\,m_5\,\sin\left(\theta_3\right)\,\dot{\theta_3}\,\dot{\theta_1} -$$
$$l_2\,l_{c3}\,m_3\,\sin\left(\theta_3\right)\,\dot{\theta_3}\,\dot{\theta_1} - l_2\,l_{c5}\,m_5\,\cos\left(\theta_3+\theta_4\right)\,\dot{\theta_3}\,\dot{\theta_1} - l_2\,l_{c5}\,m_5\,\cos\left(\theta_3+\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_1}$$

(Ec22)

$$C(2,2) = \frac{I_{3yy}\,\sin\left(2\,\theta_2+2\,\theta_3\right)\,\left(\dot{\theta_1}\right)^2}{2} - \frac{I_{3xx}\,\sin\left(2\,\theta_2+2\,\theta_3\right)\,\left(\dot{\theta_1}\right)^2}{2} -$$
$$\frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{4} - \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{4} +$$
$$\frac{I_{5zz}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{2} - \frac{I_{2xx}\,\sin\left(2\,\theta_2\right)\,\left(\dot{\theta_1}\right)^2}{2} + \frac{I_{2yy}\,\sin\left(2\,\theta_2\right)\,\left(\dot{\theta_1}\right)^2}{2} -$$
$$\frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4-2\,\theta_5\right)\,\left(\dot{\theta_1}\right)^2}{8} - \frac{I_{5xx}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4+2\,\theta_5\right)\,\left(\dot{\theta_1}\right)^2}{8} +$$
$$\frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4-2\,\theta_5\right)\,\left(\dot{\theta_1}\right)^2}{8} + \frac{I_{5yy}\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4+2\,\theta_5\right)\,\left(\dot{\theta_1}\right)^2}{8} +$$
$$\frac{{l_3}^2\,m_5\,\sin\left(2\,\theta_2+2\,\theta_3\right)\,\left(\dot{\theta_1}\right)^2}{2} + \frac{{l_{c3}}^2\,m_3\,\sin\left(2\,\theta_2+2\,\theta_3\right)\,\left(\dot{\theta_1}\right)^2}{2} -$$
$$\frac{{l_{c5}}^2\,m_5\,\sin\left(2\,\theta_2+2\,\theta_3+2\,\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{2} + \frac{{l_2}^2\,m_3\,\sin\left(2\,\theta_2\right)\,\left(\dot{\theta_1}\right)^2}{2} +$$
$$\frac{{l_2}^2\,m_5\,\sin\left(2\,\theta_2\right)\,\left(\dot{\theta_1}\right)^2}{2} + \frac{{l_{c2}}^2\,m_2\,\sin\left(2\,\theta_2\right)\,\left(\dot{\theta_1}\right)^2}{2} - I_{5zz}\,\sin\left(\theta_2+\theta_3+\theta_4\right)\,\dot{\theta_5}\,\dot{\theta_1} +$$
$$I_{5xx}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_2} + I_{5xx}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_3} + I_{5xx}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_4} - I_{5yy}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_2} -$$
$$I_{5yy}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_3} - I_{5yy}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_4} + \frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} +$$
$$\frac{I_{5xx}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} - \frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4-2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} -$$
$$\frac{I_{5yy}\,\sin\left(\theta_2+\theta_3+\theta_4+2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} + l_3\,l_{c5}\,m_5\,\cos\left(2\,\theta_2+2\,\theta_3+\theta_4\right)\,\left(\dot{\theta_1}\right)^2 -$$
$$l_3\,l_{c5}\,m_5\,\cos\left(\theta_4\right)\,\left(\dot{\theta_4}\right)^2 - l_2\,l_3\,m_5\,\sin\left(\theta_3\right)\,\left(\dot{\theta_3}\right)^2 - l_2\,l_{c3}\,m_3\,\sin\left(\theta_3\right)\,\left(\dot{\theta_3}\right)^2 -$$
$$l_2\,l_{c5}\,m_5\,\cos\left(\theta_3+\theta_4\right)\,\left(\dot{\theta_3}\right)^2 - l_2\,l_{c5}\,m_5\,\cos\left(\theta_3+\theta_4\right)\,\left(\dot{\theta_4}\right)^2 +$$
$$l_2\,l_{c5}\,m_5\,\cos\left(2\,\theta_2+\theta_3+\theta_4\right)\,\left(\dot{\theta_1}\right)^2 + l_2\,l_3\,m_5\,\sin\left(2\,\theta_2+\theta_3\right)\,\left(\dot{\theta_1}\right)^2 +$$
$$l_2\,l_{c3}\,m_3\,\sin\left(2\,\theta_2+\theta_3\right)\,\left(\dot{\theta_1}\right)^2 - 2\,l_3\,l_{c5}\,m_5\,\cos\left(\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_2} - 2\,l_3\,l_{c5}\,m_5\,\cos\left(\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_3} -$$
$$2\,l_2\,l_3\,m_5\,\sin\left(\theta_3\right)\,\dot{\theta_3}\,\dot{\theta_2} - 2\,l_2\,l_{c3}\,m_3\,\sin\left(\theta_3\right)\,\dot{\theta_3}\,\dot{\theta_2} - 2\,l_2\,l_{c5}\,m_5\,\cos\left(\theta_3+\theta_4\right)\,\dot{\theta_3}\,\dot{\theta_2} -$$
$$2\,l_2\,l_{c5}\,m_5\,\cos\left(\theta_3+\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_2} - 2\,l_2\,l_{c5}\,m_5\,\cos\left(\theta_3+\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_3}$$

(Ec23)

$$C(3,1) = \frac{I_{3yy}\,\sin\left(2\,\theta_2 + 2\,\theta_3\right)\,\left(\dot{\theta_1}\right)^2}{2} - \frac{I_{3xx}\,\sin\left(2\,\theta_2 + 2\,\theta_3\right)\,\left(\dot{\theta_1}\right)^2}{2} - \frac{I_{5xx}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{4} - \frac{I_{5yy}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{4} + \frac{I_{5zz}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{2} - \frac{I_{5xx}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4 - 2\,\theta_5\right)\,\left(\dot{\theta_1}\right)^2}{8} - \frac{I_{5xx}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4 + 2\,\theta_5\right)\,\left(\dot{\theta_1}\right)^2}{8} + \frac{I_{5yy}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4 - 2\,\theta_5\right)\,\left(\dot{\theta_1}\right)^2}{8} + \frac{I_{5yy}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4 + 2\,\theta_5\right)\,\left(\dot{\theta_1}\right)^2}{8} + \frac{{l_3}^2\,m_5\,\sin\left(2\,\theta_2 + 2\,\theta_3\right)\,\left(\dot{\theta_1}\right)^2}{2} + \frac{{l_{c3}}^2\,m_3\,\sin\left(2\,\theta_2 + 2\,\theta_3\right)\,\left(\dot{\theta_1}\right)^2}{2} - \frac{{l_{c5}}^2\,m_5\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{2} - I_{5zz}\,\sin\left(\theta_2 + \theta_3 + \theta_4\right)\,\dot{\theta_5}\,\dot{\theta_1} + I_{5xx}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_2} + I_{5xx}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_3} + I_{5xx}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_4} - I_{5yy}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_2} - I_{5yy}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_3} - I_{5yy}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_4} + \frac{I_{5xx}\,\sin\left(\theta_2 + \theta_3 + \theta_4 - 2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} + \frac{I_{5xx}\,\sin\left(\theta_2 + \theta_3 + \theta_4 + 2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} - \frac{I_{5yy}\,\sin\left(\theta_2 + \theta_3 + \theta_4 - 2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} - \frac{I_{5yy}\,\sin\left(\theta_2 + \theta_3 + \theta_4 + 2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} + l_3\,l_{c5}\,m_5\,\cos\left(2\,\theta_2 + 2\,\theta_3 + \theta_4\right)\,\left(\dot{\theta_1}\right)^2 - l_3\,l_{c5}\,m_5\,\cos\left(\theta_4\right)\,\left(\dot{\theta_4}\right)^2 + \frac{l_2\,l_3\,m_5\,\sin\left(\theta_3\right)\,\left(\dot{\theta_1}\right)^2}{2} + l_2\,l_3\,m_5\,\sin\left(\theta_3\right)\,\left(\dot{\theta_2}\right)^2 + \frac{l_2\,l_{c3}\,m_3\,\sin\left(\theta_3\right)\,\left(\dot{\theta_1}\right)^2}{2} + l_2\,l_{c3}\,m_3\,\sin\left(\theta_3\right)\,\left(\dot{\theta_2}\right)^2 + \frac{l_2\,l_{c5}\,m_5\,\cos\left(\theta_3 + \theta_4\right)\,\left(\dot{\theta_1}\right)^2}{2} + l_2\,l_{c5}\,m_5\,\cos\left(\theta_3 + \theta_4\right)\,\left(\dot{\theta_2}\right)^2 + \frac{l_2\,l_{c5}\,m_5\,\cos\left(2\,\theta_2 + \theta_3 + \theta_4\right)\,\left(\dot{\theta_1}\right)^2}{2} + \frac{l_2\,l_3\,m_5\,\sin\left(2\,\theta_2 + \theta_3\right)\,\left(\dot{\theta_1}\right)^2}{2} + \frac{l_2\,l_{c3}\,m_3\,\sin\left(2\,\theta_2 + \theta_3\right)\,\left(\dot{\theta_1}\right)^2}{2} - 2\,l_3\,l_{c5}\,m_5\,\cos\left(\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_2} - 2\,l_3\,l_{c5}\,m_5\,\cos\left(\theta_4\right)\,\dot{\theta_4}\,\dot{\theta_3} \tag{Ec24}$$

$$C(4,1) = \frac{I_{5zz}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{2} - \frac{I_{5yy}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{4} - \frac{I_{5xx}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{4} - \frac{I_{5xx}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4 - 2\,\theta_5\right)\,\left(\dot{\theta_1}\right)^2}{8} - \frac{I_{5xx}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4 + 2\,\theta_5\right)\,\left(\dot{\theta_1}\right)^2}{8} + \frac{I_{5yy}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4 - 2\,\theta_5\right)\,\left(\dot{\theta_1}\right)^2}{8} + \frac{I_{5yy}\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4 + 2\,\theta_5\right)\,\left(\dot{\theta_1}\right)^2}{8} - \frac{{l_{c5}}^2\,m_5\,\sin\left(2\,\theta_2 + 2\,\theta_3 + 2\,\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{2} - I_{5zz}\,\sin\left(\theta_2 + \theta_3 + \theta_4\right)\,\dot{\theta_5}\,\dot{\theta_1} + I_{5xx}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_2} + I_{5xx}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_3} + I_{5xx}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_4} - I_{5yy}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_2} - I_{5yy}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_3} - I_{5yy}\,\sin\left(2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_4} + \frac{I_{5xx}\,\sin\left(\theta_2 + \theta_3 + \theta_4 - 2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} + \frac{I_{5xx}\,\sin\left(\theta_2 + \theta_3 + \theta_4 + 2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} - \frac{I_{5yy}\,\sin\left(\theta_2 + \theta_3 + \theta_4 - 2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} - \frac{I_{5yy}\,\sin\left(\theta_2 + \theta_3 + \theta_4 + 2\,\theta_5\right)\,\dot{\theta_5}\,\dot{\theta_1}}{2} + \frac{l_3\,l_{c5}\,m_5\,\cos\left(2\,\theta_2 + 2\,\theta_3 + \theta_4\right)\,\left(\dot{\theta_1}\right)^2}{2} + \frac{l_3\,l_{c5}\,m_5\,\cos\left(\theta_4\right)\,\left(\dot{\theta_1}\right)^2}{2} + l_3\,l_{c5}\,m_5\,\cos\left(\theta_4\right)\,\left(\dot{\theta_2}\right)^2 + l_3\,l_{c5}\,m_5\,\cos\left(\theta_4\right)\,\left(\dot{\theta_3}\right)^2 + \frac{l_2\,l_{c5}\,m_5\,\cos\left(\theta_3 + \theta_4\right)\,\left(\dot{\theta_1}\right)^2}{2} + l_2\,l_{c5}\,m_5\,\cos\left(\theta_3 + \theta_4\right)\,\left(\dot{\theta_2}\right)^2 + \frac{l_2\,l_{c5}\,m_5\,\cos\left(2\,\theta_2 + \theta_3 + \theta_4\right)\,\left(\dot{\theta_1}\right)^2}{2} + 2\,l_3\,l_{c5}\,m_5\,\cos\left(\theta_4\right)\,\dot{\theta_3}\,\dot{\theta_2} \tag{Ec25}$$

$$C(5,1) = \frac{I_{5xx}\,\sin(2\,\theta_5)\,\left(\dot{\theta_1}\right)^2}{4} - \frac{I_{5xx}\,\sin(2\,\theta_5)\,\left(\dot{\theta_2}\right)^2}{2} - \frac{I_{5xx}\,\sin(2\,\theta_5)\,\left(\dot{\theta_3}\right)^2}{2} - \frac{I_{5yy}\,\sin(2\,\theta_5)\,\left(\dot{\theta_1}\right)^2}{4} - \frac{I_{5xx}\,\sin(2\,\theta_5)\,\left(\dot{\theta_4}\right)^2}{2} + \frac{I_{5yy}\,\sin(2\,\theta_5)\,\left(\dot{\theta_2}\right)^2}{2} + \frac{I_{5yy}\,\sin(2\,\theta_5)\,\left(\dot{\theta_3}\right)^2}{2} + \frac{I_{5yy}\,\sin(2\,\theta_5)\,\left(\dot{\theta_4}\right)^2}{2} + \frac{I_{5xx}\,\sin(2\,\theta_2+2\,\theta_3+2\,\theta_4-2\,\theta_5)\,\left(\dot{\theta_1}\right)^2}{8} - \frac{I_{5xx}\,\sin(2\,\theta_2+2\,\theta_3+2\,\theta_4+2\,\theta_5)\,\left(\dot{\theta_1}\right)^2}{8} - \frac{I_{5yy}\,\sin(2\,\theta_2+2\,\theta_3+2\,\theta_4-2\,\theta_5)\,\left(\dot{\theta_1}\right)^2}{8} + \frac{I_{5yy}\,\sin(2\,\theta_2+2\,\theta_3+2\,\theta_4+2\,\theta_5)\,\left(\dot{\theta_1}\right)^2}{8} + I_{5zz}\,\sin(\theta_2+\theta_3+\theta_4)\,\dot{\theta_2}\,\dot{\theta_1} + I_{5zz}\,\sin(\theta_2+\theta_3+\theta_4)\,\dot{\theta_3}\,\dot{\theta_1} + I_{5zz}\,\sin(\theta_2+\theta_3+\theta_4)\,\dot{\theta_4}\,\dot{\theta_1} - I_{5xx}\,\sin(2\,\theta_5)\,\dot{\theta_3}\,\dot{\theta_2} - I_{5xx}\,\sin(2\,\theta_5)\,\dot{\theta_4}\,\dot{\theta_2} - I_{5xx}\,\sin(2\,\theta_5)\,\dot{\theta_4}\,\dot{\theta_3} + I_{5yy}\,\sin(2\,\theta_5)\,\dot{\theta_3}\,\dot{\theta_2} + I_{5yy}\,\sin(2\,\theta_5)\,\dot{\theta_4}\,\dot{\theta_2} + I_{5yy}\,\sin(2\,\theta_5)\,\dot{\theta_4}\,\dot{\theta_3} - \frac{I_{5xx}\,\sin(\theta_2+\theta_3+\theta_4-2\,\theta_5)\,\dot{\theta_2}\,\dot{\theta_1}}{2} - \frac{I_{5xx}\,\sin(\theta_2+\theta_3+\theta_4+2\,\theta_5)\,\dot{\theta_2}\,\dot{\theta_1}}{2} - \frac{I_{5xx}\,\sin(\theta_2+\theta_3+\theta_4-2\,\theta_5)\,\dot{\theta_3}\,\dot{\theta_1}}{2} - \frac{I_{5xx}\,\sin(\theta_2+\theta_3+\theta_4+2\,\theta_5)\,\dot{\theta_3}\,\dot{\theta_1}}{2} - \frac{I_{5xx}\,\sin(\theta_2+\theta_3+\theta_4-2\,\theta_5)\,\dot{\theta_4}\,\dot{\theta_1}}{2} - \frac{I_{5xx}\,\sin(\theta_2+\theta_3+\theta_4+2\,\theta_5)\,\dot{\theta_4}\,\dot{\theta_1}}{2} + \frac{I_{5yy}\,\sin(\theta_2+\theta_3+\theta_4-2\,\theta_5)\,\dot{\theta_2}\,\dot{\theta_1}}{2} + \frac{I_{5yy}\,\sin(\theta_2+\theta_3+\theta_4+2\,\theta_5)\,\dot{\theta_2}\,\dot{\theta_1}}{2} + \frac{I_{5yy}\,\sin(\theta_2+\theta_3+\theta_4-2\,\theta_5)\,\dot{\theta_3}\,\dot{\theta_1}}{2} + \frac{I_{5yy}\,\sin(\theta_2+\theta_3+\theta_4+2\,\theta_5)\,\dot{\theta_3}\,\dot{\theta_1}}{2} + \frac{I_{5yy}\,\sin(\theta_2+\theta_3+\theta_4-2\,\theta_5)\,\dot{\theta_4}\,\dot{\theta_1}}{2} + \frac{I_{5yy}\,\sin(\theta_2+\theta_3+\theta_4+2\,\theta_5)\,\dot{\theta_4}\,\dot{\theta_1}}{2} \tag{Ec26}$$

A.4.3 Vector de fuerzas o momentos debidos a la gravedad

$$G(1,1) = 0 \tag{Ec27}$$

$$G(2,1) = g\,l_{c5}\,m_5\,\sin(\theta_2+\theta_3+\theta_4) - g\,l_3\,m_5\,\cos(\theta_2+\theta_3) - g\,l_{c3}\,m_3\,\cos(\theta_2+\theta_3) - g\,l_2\,m_3\,\cos(\theta_2) - g\,l_2\,m_5\,\cos(\theta_2) - g\,l_{c2}\,m_2\,\cos(\theta_2) \tag{Ec28}$$

$$G(3,1) = g\,l_{c5}\,m_5\,\sin(\theta_2+\theta_3+\theta_4) - g\,l_3\,m_5\,\cos(\theta_2+\theta_3) - g\,l_{c3}\,m_3\,\cos(\theta_2+\theta_3) \tag{Ec29}$$

$$G(4,1) = g\,l_{c5}\,m_5\,\sin(\theta_2+\theta_3+\theta_4) \tag{Ec30}$$

$$G(5,1) = 0 \tag{Ec31}$$

A.5 Código dibujar líneas MATLAB

```m
clc; clear; close all;
N = 20;
points = [];

points = [points; 0.0, 0, 0.10];
points = G01(0.0, 0.10, 0.10, points, N);
points = G01(0.0, 0.10, 0, points, N);
points = G01(0.0, 0, 0, points, N);
points = G01(0.0, 0, 0.10, points, N);


points = G01(0.0, -0.10, 0.10, points, N);
points = G01(0.0, -0.10, 0, points, N);
points = G01(0.0, 0, 0, points, N);

%%GRAFICADOR

%Gráfica original
%_____________________________
    hold on
    %plot3(points(:,1), points(:,2), points(:,3));

%Ubicacion en el lienzo
zl = 0.190;
%zl = 0.02;
yl = 0;

%Transformación de puntos
%_____________________________
xp = 0.177 + zl*cos((7/18) * pi);
yp = yl;
zp = zl*sin((7/18) * pi);
% Matriz de rotación en el eje X
beta = pi/2-(7/18)*pi;


% Vector de traslación
t = [xp; yp; zp];

% Rx = [1, 0, 0;
%       0, cos(beta), -sin(beta);
%       0, sin(beta),  cos(beta)];

Ry = [cos(beta) 0 sin(beta);
               0 1         0;
     -sin(beta) 0 cos(beta)];


% Matriz de transformación homogénea 4x4
T = [Ry, t;
     0 0 0 1];

% Convertir a coordenadas homogéneas (añadir columna de 1s)
P_hom = [points, ones(size(points,1),1)];  % Resultado: N x 4

% Aplicar la transformación
P_trans_hom = (T * P_hom')';  % Resultado: N x 4

% Extraer los nuevos puntos (sin la cuarta coordenada)
points = P_trans_hom(:,1:3);


%Graficamos puntos transformados
%_____________________________________
    plot3(points(:,1), points(:,2), points(:,3));
    xlabel('x');
    ylabel('y');
    zlabel('z');

%Cargamos la plataforma de trabajo
%_____________________________________
    load('PuntosEstructuraFisica.mat')
    plot3(points1(:,1), points1(:,2), points1(:,3),'LineWidth', 1.5);
    plot3(points2(:,1), points2(:,2), points2(:,3),'LineWidth', 1.5);
    plot3(points3(:,1), points3(:,2), points3(:,3),'LineWidth', 1.5);

%Dibujamos los ejes
%_____________________

    % Longitud de los ejes
    L = 0.5;

    % Dibujar los ejes con quiver3
    quiver3(0,0,0, L,0,0, 'r', 'LineWidth', 2); % Eje X (rojo)
```

```
    quiver3(0,0,0, 0,L,0, 'g', 'LineWidth', 2); % Eje Y (verde)
    quiver3(0,0,0, 0,0,L, 'b', 'LineWidth', 2); % Eje Z (azul)
grid minor

%Dibujamos la esfera
%___________________________________
d1 = 0.065;
a2 = 0.101;
a3 = 0.095;
d5 = 0.168;


R = a2 + a3 + d5;

% Crear datos para la esfera
[X, Y, Z] = sphere(50); % 50 define la resolución

s = surf(R*X, R*Y, R*Z+d1); % Dibujar la esfera

% Color verde
s.FaceColor = [0.6 0.6 0.6]; % RGB del verde puro

% Quitar las líneas de la malla
s.EdgeColor = 'none';

% Hacerla transparente (0 = totalmente transparente, 1 = opaca)
s.FaceAlpha = 0.1; % Puedes ajustar este valor a tu gusto

% Mejorar la visualización
%axis equal
camlight
 lighting gouraud
%

ylim([-0.5 0.5]);
axis equal

view(-45, 45);
```

A.6 Desarrollo del archivo G-code para trayectoria circular

A continuación se presentan los pasos detallados para la obtención del archivo G-code, desde el diseño inicial en Inkscape hasta la verificación visual y simulación en MATLAB.

Paso 1: Diseño de la figura

La trayectoria se diseña en Inkscape mediante un círculo que representa el recorrido del efector final dentro del espacio operativo.

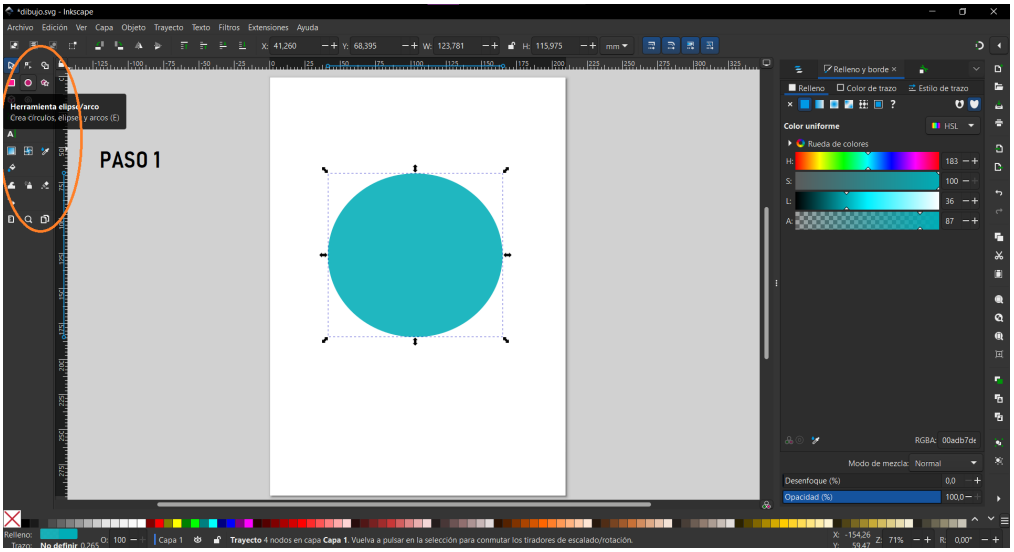


Figura Im2: Diseño de la trayectoria circular en Inkscape

El objeto se reconoce automáticamente como un trayecto (path), simplificando su procesamiento en Gcodetools. Se recomienda ubicar la figura en el primer cuadrante para facilitar la interpretación de coordenadas positivas compatibles con el sistema de referencia del robot.

Paso 2: Definición de puntos de orientación

Mediante la extensión Gcodetools se accede a la función “Puntos de orientación” para establecer el sistema de coordenadas desde el cual se generará el G-code.

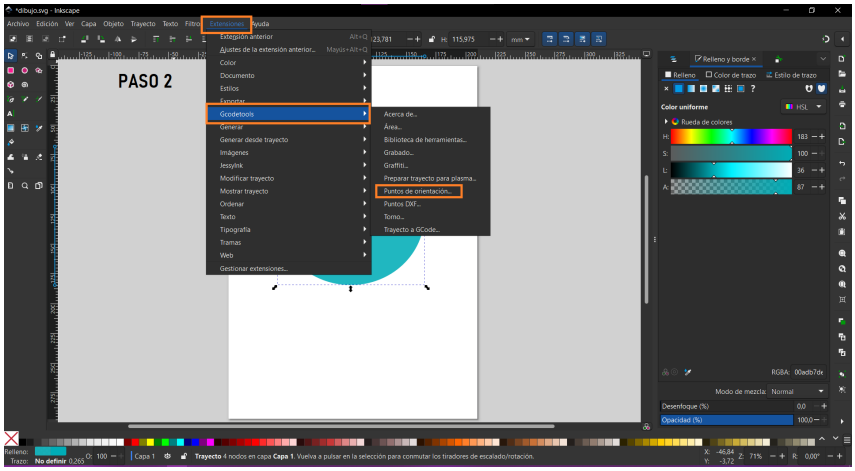


Figura Im3: Configuración de puntos de orientación en Gcodetools

Se define un modo de dos puntos para precisar el origen y la dirección del eje X, ajustando además la altura Z a cero milímetros para trabajar sobre un plano horizontal.

Paso 3: Confirmación y fijación de puntos de orientación

Una vez establecidos los parámetros, se confirma la acción y los puntos de orientación se visualizan en el lienzo. Es imprescindible que estos permanezcan en la misma capa que la trayectoria y que no se modifiquen para garantizar la consistencia del código generado.

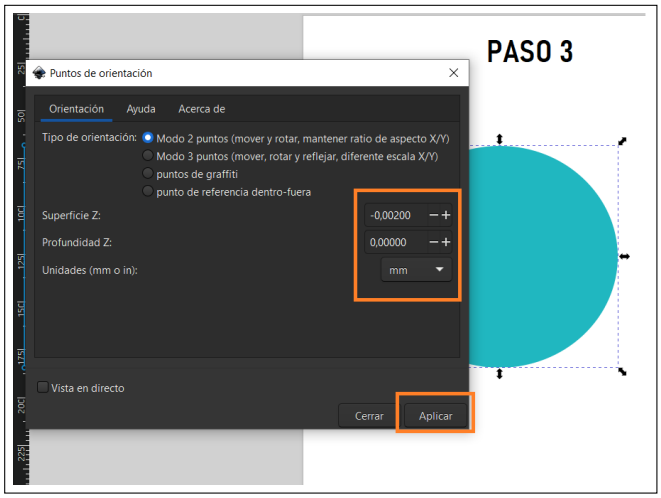


Figura Im4: Visualización y selección de puntos de orientación

Paso 4: Configuración de la herramienta virtual

Se accede a la “Biblioteca de herramientas” para definir las características del efector final simulado, tales como tipo, dimensiones y velocidad de avance.

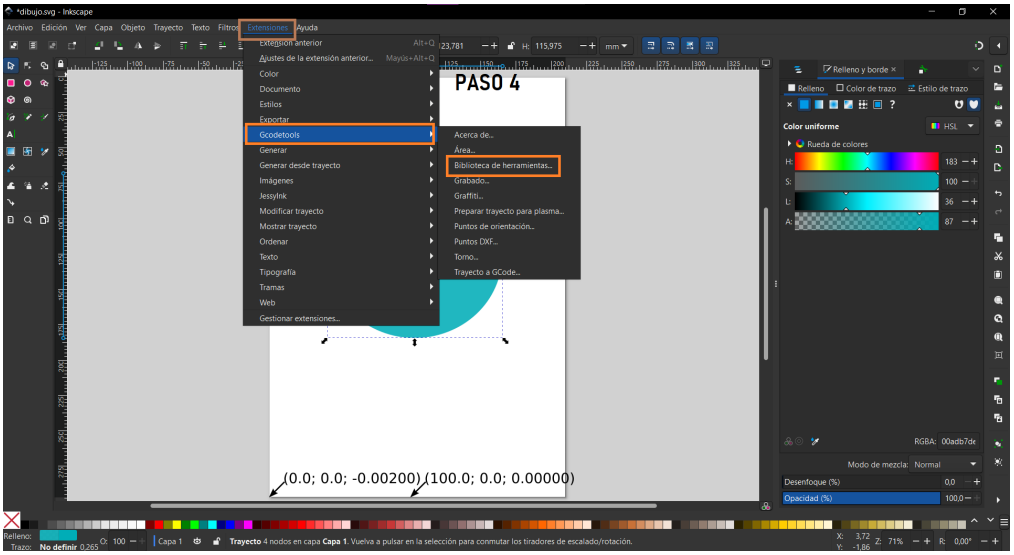


Figura Im5: Acceso y configuración de la biblioteca de herramientas

Paso 5: Selección y parametrización de la herramienta

Se selecciona una herramienta cilíndrica, adecuada para una trayectoria plana, y se definen parámetros como diámetro, profundidad y velocidad, importantes para una correcta simulación, aunque no se realice mecanizado físico.

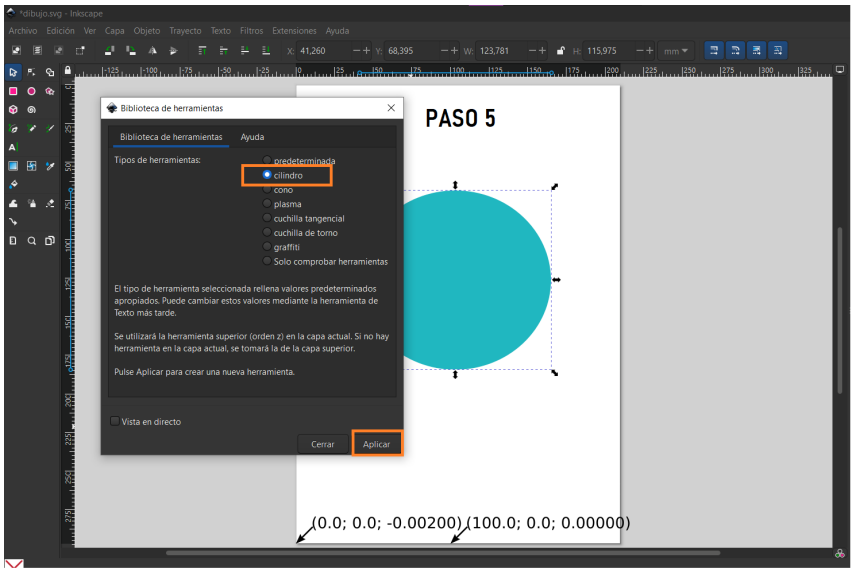


Figura Im6: Configuración detallada de la herramienta virtual

Paso 6: Generación del archivo G-code

Finalmente, se genera el archivo G-code mediante la opción “Trayecto a Gcode”, configurando parámetros finales como nombre, ubicación, profundidad

en Z, cantidad de pasadas y velocidad de avance.

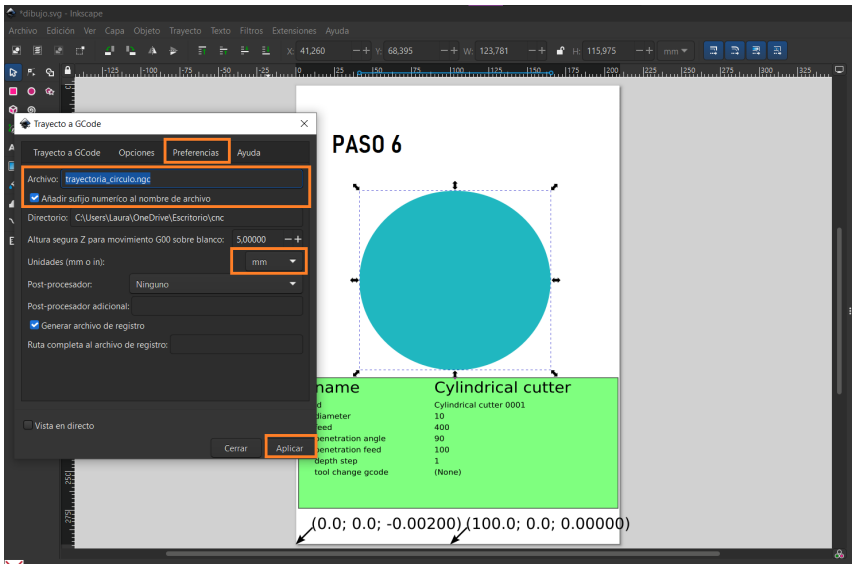


Figura Im7: Generación del archivo G-code.

Paso 7: Confirmación y verificación visual

Se confirma la correcta generación del código, lo cual se puede corroborar mediante visualizadores especializados que muestran gráficamente la trayectoria programada.

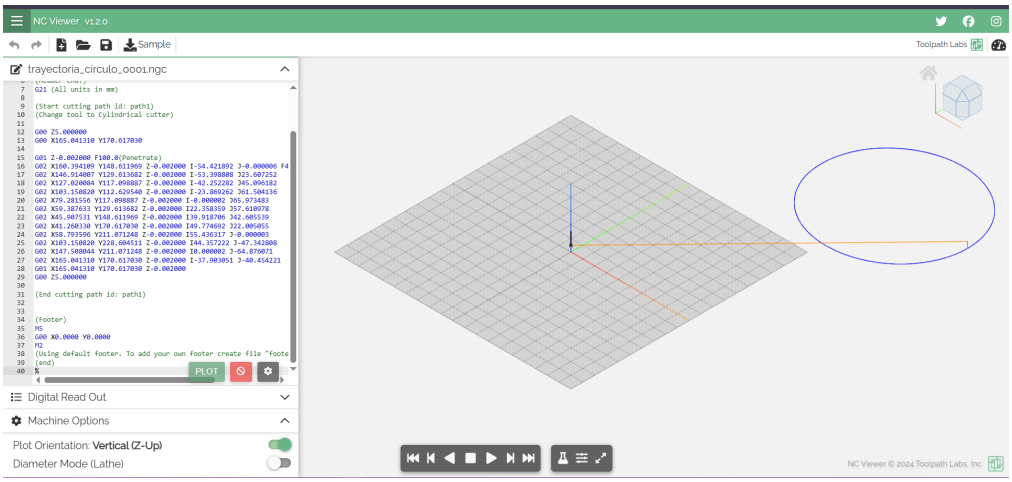


Figura Im8: Verificación visual del código G generado

Procesamiento y simulación en MATLAB

El archivo G-code se procesa en MATLAB mediante funciones que leen línea a línea, extraen comandos y coordenadas, y almacenan los puntos en una matriz para su posterior visualización tridimensional. Este paso es fundamental

para validar la trayectoria antes de su ejecución en el brazo robótico.

```
clc; clear; close all;
N = 20;
points = [];

points = [points; 0, 0, 5];

% Movimiento lineal
points = G01(165.041310, 170.617030, -0.002, points, N);

% Arcos circulares en sentido horario (G02)
points = G02(160.394109, 148.611969, -0.002, -54.421892, -0.000006, points, N);
points = G02(146.914007, 129.613682, -0.002, -53.398808, 23.607252, points, N);
points = G02(127.020084, 117.098887, -0.002, -42.252282, 45.096182, points, N);
points = G02(103.150820, 112.629540, -0.002, -23.869262, 61.504136, points, N);
points = G02(79.281556, 117.098887, -0.002, -0.000002, 65.973483, points, N);
points = G02(59.387633, 129.613682, -0.002, 22.358359, 57.610978, points, N);
points = G02(45.907531, 148.611969, -0.002, 39.918706, 42.605539, points, N);
points = G02(41.260330, 170.617030, -0.002, 49.774692, 22.005055, points, N);
points = G02(58.793596, 211.071248, -0.002, 55.436317, -0.000003, points, N);
points = G02(103.150820, 228.604511, -0.002, 44.357222, -47.342808, points, N);
points = G02(147.508044, 211.071248, -0.002, 0.000002, -64.876071, points, N);
points = G02(165.041310, 170.617030, -0.002, -37.903051, -40.454221, points, N);

% Cierre de la trayectoria
points = G01(165.041310, 170.617030, -0.002, points, N);

plot3(points(:,1), points(:,2), points(:,3));
title('Trayectoria circular');
xlabel('x');
ylabel('y');
zlabel('z');
grid on
% view(90,0)
```

Código Cod3: Código MATLAB para visualización de puntos de trayectoria

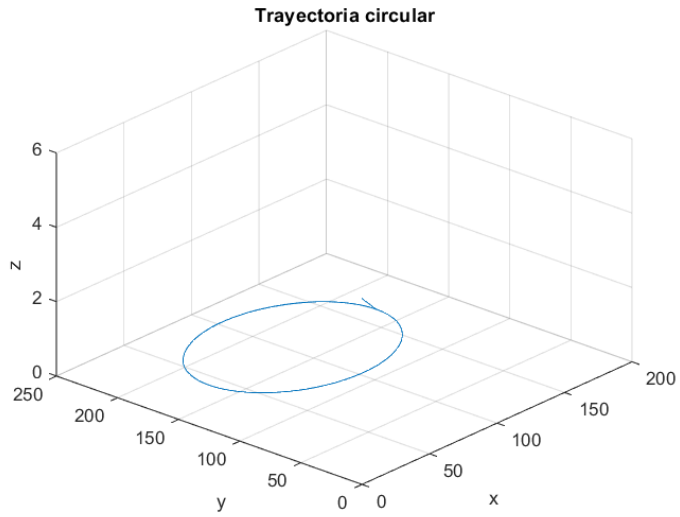


Figura Im9: Simulación de trayectoria circular en MATLAB

Este método garantiza la generación de trayectorias complejas de forma visual e intuitiva, integrando eficazmente el diseño gráfico con la simulación y el control del robot.

A.7 Área de Trabajo

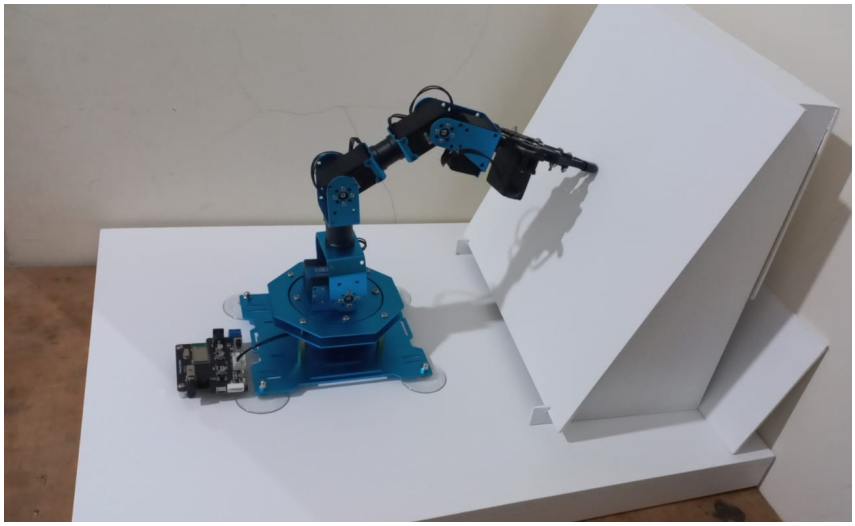


Figura Im10: Área de trabajo del brazo robótico

A.8 Código del controlador en python

A.8.1 MAIN

```
import socket
import json
import CinematicaInversa
import CinematicaDirecta
import time
import Ctrl_LSC


HOST = '192.168.126.117'  # Reemplaza con la IP real del servidor MATLAB
PORT = 12345

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect((HOST, PORT))
client.settimeout(6.0) #Vamos a esperar 6 segundos para la recepcion de datos

#Variables para las articulaciones

#variable de referencia Px Py Pz puntos que vienen desde Matlab
p_s = [[0, 0, 0, 0, 0]]; #el primer termino de la matriz es 0 para relacionarlo con matlab


#El controlador Xarm de python me devuelve el angulo deseado y el angulo de proceso
q_p = [[0, 0, 0, 0, 0, 0]]; #variable de proceso
q_s = [[0, 0, 0, 0, 0, 0]]; #variable establecida
p_p = [[0, 0, 0, 0]];

#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

#:::::::::::::: FUNCION PARA RECIBIR DATOS DESDE MATLAB ::::::::::::::::::::::::::::::::::::::::::::::::::::::::
def Recepcion_desde_Matlab():

    try:
       # Recibir respuesta de MATLAB
       data = client.recv(1024) #recv se queda estancado esperando un dato
       json_data = json.loads(data.decode())
       #print("Datos recibidos:", json_data)
       return json_data

    except socket.timeout:
        print("No se recibieron datos en el intervalo de tiempo.")

#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

#:::::::::::::: FUNCION PARA ENVIAR DATOS A MATLAB ::::::::::::::::::::::::::::::::::::::::::::::::::::::::
def Enviar_a_Matlab(q_s, q_p, p_p):
    # Diccionario a enviar
    Pqt_datos = {
        "q_s1": q_s[0][1],
        "q_s2": q_s[0][2],
        "q_s3": q_s[0][3],
        "q_s4": q_s[0][4],
        "q_s5": q_s[0][5],
        "q_p1": q_p[0][1],
        "q_p2": q_p[0][2],
        "q_p3": q_p[0][3],
        "q_p4": q_p[0][4],
        "q_p5": q_p[0][5],
        "p_px": p_p[0][1],
        "p_py": p_p[0][2],
        "p_pz": p_p[0][3],
    }

    # Convertir a JSON
    json_data = json.dumps(Pqt_datos)

    # Enviar como bytes
    client.sendall((json_data+'\n').encode('utf-8'))

    #print("Enviado:", json_data)
    #time.sleep(1)  # Esperar 1 segundo
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
pausa = 0.1  # segundos

#:::::::::::::::::::::::: VOID LOOP ::::::::::::::::::::::::::::::::::::::::::::::::::::
try:
    while True:
        Recibidos = Recepcion_desde_Matlab() #Bucle esta esperando la escritura de mas datos

        p_s[0][1] =  Recibidos.get("p_sx")
        p_s[0][2] =  Recibidos.get("p_sy")
        p_s[0][3] =  Recibidos.get("p_sz")
        p_s[0][4] =  Recibidos.get("P")

        q_s[0][1],q_s[0][2],q_s[0][3],q_s[0][4],q_s[0][5] = CinematicaInversa.Ik(p_s[0][1], p_s[0][2], p_s[0][3], p_s[0][4])
        t_inicio = time.time()

        while True:
            if time.time() - t_inicio >= pausa:
                q_p[0][1],q_p[0][2],q_p[0][3],q_p[0][4],q_p[0][5] =  Ctrl_LSC.obtener_posicion(q_p[0][1],q_p[0][2],q_p[0][3],q_p[0][4],q_p[0][5])

                Ctrl_LSC.enviar_posicion(q_s[0][1],q_s[0][2],q_s[0][3],q_s[0][4],q_s[0][5], 1500)
                p_p[0][1], p_p[0][2], p_p[0][3]= CinematicaDirecta.FK(q_p[0][1],q_p[0][2],q_p[0][3],q_p[0][4],q_p[0][5])
                break

        Enviar_a_Matlab(q_s, q_p, p_p)

except Exception as M:
    print("Error al conectar o recibir datos.", M)

finally:
    client.close()
    print("Cliente cerrado.")
    print("Proceso Finalizado.")
    Ctrl_LSC.enviar_posicion(0, -3.14/2, 0, 0, 0, 1500)
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

Código Cod4: Control Principal.

A.8.2 Función de Control LQG

```
1. import numpy as np
2. import hid
3. from scipy.linalg import solve_discrete_are
4. from scipy.signal import dlti
5.
6. # Dimensiones del sistema
7. n = 5  # número de estados
8. m = 5  # número de entradas (motores)
9. p = 5  # número de salidas (medidas)
10.
11. # Simulación de modelo (A, B, C)
12. A = np.eye(n) * 0.9
13. B = np.eye(n) * 0.1
14. C = np.eye(n)
15. D = np.zeros((p, m))
16.
17. # Pesos del LQR
18. Q = np.eye(n) * 10  # penaliza los errores de estado
19. R = np.eye(m)       # penaliza el esfuerzo de control
20.
21. # Matriz de covarianzas para el estimador de Kalman
22. W = np.eye(n) * 0.01  # ruido del proceso
23. V = np.eye(p) * 0.05  # ruido de medición
24.
25. # Resolver ecuación de Riccati para LQR
26. P = solve_discrete_are(A, B, Q, R)
27. K = np.linalg.inv(B.T @ P @ B + R) @ (B.T @ P @ A)
28.
29. # Ganancia del observador de Kalman (estimador de estado)
30. P_kalman = solve_discrete_are(A.T, C.T, W, V)
31. L = P_kalman @ C.T @ np.linalg.inv(C @ P_kalman @ C.T + V)
32.
33. # Inicialización
34. x_hat = np.zeros((n, 1))  # estimación del estado
35. u = np.zeros((m, 1))      # señal de control
36.
37. # Configuración HID (esto depende del dispositivo específico)
38. VENDOR_ID = 0x1234
39. PRODUCT_ID = 0x5678
40.
41. device = hid.device()
42. device.open(VENDOR_ID, PRODUCT_ID)
43.
44. def read_hid_input():
45.     # Aquí deberías leer los datos reales del HID (sensores)
46.     data = device.read(64)
47.     return np.array(data[:5]).reshape((5, 1)) / 255.0  # ejemplo normalizado
48.
49. def send_hid_output(u):
50.     u_bytes = (u.flatten() * 255).astype(int).clip(0, 255).tolist()
51.     device.write([0x00] + u_bytes + [0] * (63 - len(u_bytes)))  # relleno a 64 bytes
52.
53. # Simulación/control en lazo cerrado
54. for t in range(1000):
55.     y = read_hid_input()
56.
57.     # Estimador de estados (Kalman)
58.     x_hat = A @ x_hat + B @ u + L @ (y - C @ x_hat)
59.
60.     # Control LQR
61.     u = -K @ x_hat
62.
63.     # Enviar a motores por HID
64.     send_hid_output(u)
65.
66. device.close()
```

Código Cod5: Función de Control LQG.

A.8.3 Función de cinemática directa

```
import numpy as np


def FK(q1, q2, q3, q4, q5):
    # Posiciones angulares (q1)
    q1 = [q1, q2, q3, q4, q5]
    v_theta_i = np.array(q1).reshape(-1, 1)

    # Parámetros DH
     v_d_i = np.array([0.0650, 0, 0, 0, 0.168])
     v_a_i = np.array([0, 0.1010, 0.0950, 0, 0])
     v_alpha_i = np.array([-np.pi / 2, 0, 0, -np.pi / 2, 0])

     n = 5
     T_i = []

     for i in range(n):
         theta_i = v_theta_i[i, 0]
         d_i = v_d_i[i]
         a_i = v_a_i[i]
         alpha_i = v_alpha_i[i]

         Rz_theta = np.array([
             [np.cos(theta_i), -np.sin(theta_i), 0, 0],
             [np.sin(theta_i),  np.cos(theta_i), 0, 0],
             [0,                0,               1, 0],
             [0,                0,               0, 1]
         ])

         Tz_d = np.array([
             [1, 0, 0, 0],
             [0, 1, 0, 0],
             [0, 0, 1, d_i],
             [0, 0, 0, 1]
         ])

         Tx_a = np.array([
             [1, 0, 0, a_i],
             [0, 1, 0, 0],
             [0, 0, 1, 0],
             [0, 0, 0, 1]
         ])

         Rx_alpha = np.array([
             [1, 0,              0,             0],
             [0, np.cos(alpha_i), -np.sin(alpha_i), 0],
             [0, np.sin(alpha_i),  np.cos(alpha_i), 0],
             [0, 0,              0,             1]
         ])

         T = Rz_theta @ Tz_d @ Tx_a @ Rx_alpha

         # Cambios específicos en T_i[0] y T_i[3] como en el código MATLAB
         if i == 0 or i == 3:
             T[0, 1] = 0
             T[1, 1] = 0
             T[2, 2] = 0

         T_i.append(T)

     # Calcular la transformación homogénea final
     T_final = T_i[0] @ T_i[1] @ T_i[2] @ T_i[3] @ T_i[4]

     # Mostrar resultado
     np.set_printoptions(precision=8, suppress=True)
     #print("Matriz de Transformada Homogénea\n")
     #print(T_final)
     return T_final[0][3], T_final[1][3], T_final[2][3]

# Ejecutar la cinemática directa
if __name__ == "__main__":
     try:
         print(FK(1,2,3,4,5))
     except KeyboardInterrupt:
         print("Algo fallo.")
```

Código Cod6: Cinemática Directa.

A.8.4 Función de cinemática inversa

```
import math

def Ik(Px, Py, Pz, P):

    d1 = 0.0650;
    a2 = 0.1010;
    a3 = 0.0950;
    d5 = 0.168;

    DISTANCIA_PUNTO = math.sqrt((0-Px)**2 + (0-Py)**2 + (d1-Pz)**2);
    DISTANCIA_MAX = a2+a3+d5;   DISTANCIA_MIN = 0.180;

    GRADOS_MAX = 120;   GRADOS_MIN = 23;

    M =  (GRADOS_MAX - GRADOS_MIN)/(DISTANCIA_MAX - DISTANCIA_MIN);

    dQ = M * (DISTANCIA_PUNTO-DISTANCIA_MIN)+GRADOS_MIN;

    Q = math.radians(dQ); #120 angulo maximo


    #INICIA_CALCULO
    theta1 = math.atan(Py/Px);

    X4 = Px - d5 * math.sin(Q) * math.cos(theta1);

    Y4 = Py - d5 * math.sin(Q) * math.sin(theta1);

    Z4 = Pz + d5*math.cos(Q);



    l = math.sqrt(X4**2 + Y4**2);

    m = math.sqrt(l**2 + (Z4 - d1)**2);

    n = math.acos((m**2 + a2**2 - a3**2)/(2*m*a2));

    if  (Z4 - d1)<0:
        theta2 = -(n - math.atan( abs(Z4 - d1)/l ) );
    else:
        theta2 = -(math.atan((Z4 - d1)/l) + n);


    theta3 = math.pi - math.acos((a2**2 + a3**2 - m**2)/(2*a2*a3));

    if  (Z4 - d1)<0:
        theta4 = - ((math.pi-math.atan(l/abs(Z4 - d1)) + math.acos((m**2 + a3**2 - a2**2)/(2*m*a3)) + Q) - math.pi/2);
    else:
        theta4 = - ((math.atan(l/(Z4 - d1)) + math.acos((m**2 + a3**2 - a2**2)/(2*m*a3)) + Q) - math.pi/2);


    theta5 = P
    return theta1,theta2,theta3,theta4,theta5



px =  0.253
py = -0.009
pz =  0.200
P = 0;

# Ejecutar el xinemática inversa
if __name__ == "__main__":
    try:
        theta1,theta2,theta3,theta4,theta5 =  Ik(px, py, pz, P)
        print(theta1,theta2,theta3,theta4,theta5)
    except KeyboardInterrupt:
        print("La IK no se puede resolver.")
```

Código Cod7: Cinemática Inversa.

A.8.5 Función de comunicación HID

```
1. import hid
2. import time
3. import numpy as np
4. import math
5.
6.
7. dev = hid.device()
8. dev.open(0x0483, 0x5750)
9.
10. dev.set_nonblocking(1)# Activa el modo no bloqueante
11. theta1=theta2=theta3=theta4=theta5 = 0
12.
13.
14. #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
15.
16. #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
17. #Funcion para igualar las columnas de datos
18. def pad_or_truncate(some_list, target_len):
19.     return some_list[:target_len] + [0]*(target_len - len(some_list))
20. #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
21.
22. #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
23. #Funcion para igualar las columnas de datos
24. def convertidor(q1, q2, q3, q4, q5):
25.     q1 = q1 * 240/1000
26.     q1 = q1-120
27.     q1 = math.radians(q1)
28.
29.     q2 = q2 * 240/1000
30.     q2 = 30-q2;
31.     q2 = math.radians(q2)
32.
33.     q3 = q3 * 240/1000
34.     q3 = q3-120
35.     q3 = math.radians(q3)
36.
37.     q4 = q4 * 240/1000;
38.     q4 = 30-q4;
39.     q4 = math.radians(q4)
40.
41.     q5 = q5
42.     return q1, q2, q3, q4, q5
43.
44.
45. #::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
46. def obtener_posicion(theta1,theta2,theta3,theta4,theta5):
47.     response = []
48.     data = [0, 0x55, 0x55, 0x07, 0x15, 0x04, 3, 4, 5, 6,]
49.     data = pad_or_truncate(data, 65)
50.     dev.write(data)
51.     while True:
52.         d = dev.read(65, 50)
53.         if d:
54.             response.extend(d)
55.             if response[0] == 0x55 and response[1] == 0x55 and response[2] == 0x0f and response[3] == 0x15:
56.                 #q6 = 0 #este no responde
57.                 q5 = 0
58.                 q4 = response[7] * 256 + response[6]
59.                 q3 = response[10] * 256 + response[9]
60.                 q2 = response[13] * 256 + response[12]
61.                 q1 = response[16] * 256 + response[15]
```

Código Cod8: Función de Comunicacion HID parte 1.

```
                response = []
                theta1, theta2, theta3, theta4, theta5 = convertidor(q1, q2, q3, q4, q5)
                return theta1, theta2, theta3, theta4, theta5
        else:
            return theta1,theta2,theta3,theta4,theta5
            break
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
def enviar_posicion(p1,p2,p3,p4,p5, tiempo):
    p1 =  (math.degrees(p1)+120) * (1000/240);
    p2 = -(math.degrees(p2)-30) * (1000/240);
    p3 =  (math.degrees(p3)+120) * (1000/240);
    p4 = -(math.degrees(p4)-30) * (1000/240);
    p5 =  (math.degrees(p5)+120) * (1000/240);

    p1 = np.int16(p1)
    p2 = np.int16(p2)
    p3 = np.int16(p3)
    p4 = np.int16(p4)
    p5 = np.int16(p5)
    p6 = np.int16(0)
    tiempo = np.int16(tiempo)


    data = [0,
            0x55, #Encabezado
            0x55, #Encabezado
            0x17, #Longitud
            0x03, #Comando
            0x06, #Numero de servos
            np.uint8(tiempo & 0xFF), #Tiempo bajo
            np.uint8(tiempo >> 8), #Tiempo alto
            0x01, #ID del servo1-----
            np.uint8(p6 & 0xFF), #Posicion bajo
            np.uint8(p6 >> 8), #Posicion alto
            0x02, #ID del servo2-----
            np.uint8(p5 & 0xFF), #Posicion bajo
             np.uint8(p5 >> 8), #Posicion alto
             0x03, #ID del servo3-----
             np.uint8(p4 & 0xFF), #Posicion bajo
             np.uint8(p4 >> 8), #Posicion alto
             0x04, #ID del servo4-----
             np.uint8(p3 & 0xFF), #Posicion bajo
             np.uint8(p3 >> 8), #Posicion alto
             0x05, #ID del servo5-----
             np.uint8(p2 & 0xFF), #Posicion bajo
             np.uint8(p2 >> 8), #Posicion alto
             0x06, #ID del servo6-----
             np.uint8(p1 & 0xFF), #Posicion bajo
             np.uint8(p1 >> 8), #Posicion alto
             ]
     data = pad_or_truncate(data, 65)
     dev.write(data)
#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

#::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
t_inicio = time.time()
pausa = 2  # segundos
# Ejecutar el cinemática inversa
if __name__ == "__main__":
    try:
        while True:
            #if time.time() - t_inicio >= pausa:


            #time.sleep(0.5)
            theta1,theta2,theta3,theta4,theta5 =  obtener_posicion(theta1,theta2,theta3,theta4,theta5)
            #time.sleep(0.5)
            #enviar_posicion(0.3841,-1.407,0.90638,-0.9682, 0, 1200)
            #

            print(theta1,theta2,theta3,theta4,theta5)
                #break
    except KeyboardInterrupt:
        print("Algo fallo.")
```

Código Cod9: Función de Comunicacion HID parte 2.