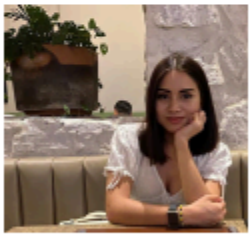
Instituto Tecnológico de Hermosillo 1 de junio de 2025
Materia: Robótica
Profesor: Medina Gil laMadrid Jesus Ivan

REPORTE FINAL
Equipo 3

Madrid Barcelo,
Daniel
danielmadridbarcelo@gmail.com
Teléfono: 6342469087

Camou Mejia,
Josué Román
l21330544@hermosillo.tecnm.mx
Teléfono: 6624622362

Rosas Leyva,
Adrian Ernesto
l21330682@hermosillo.tecnm.mx
Teléfono: 6621503887



Aguiluz Romero,
Blanca Azucena
l19330879@heremosillo.tecnm.mx

Índice

1. Introducción

Este reporte documenta el desarrollo y la implementación de un proyecto robótico integral, llevado a cabo durante el semestre en la asignatura de Robótica de la carrera de Mecatrónica. El objetivo principal de este proyecto fue aplicar los conocimientos teóricos y prácticos adquiridos para diseñar, simular y analizar un sistema robótico, abordando desde su conceptualización en software CAD hasta su manipulación cinemática.

A lo largo del proyecto, el equipo colaboró en el diseño detallado de un robot utilizando SolidWorks, lo que permitió visualizar su estructura y componentes mecánicos. Posteriormente, este diseño fue integrado y simulado en el entorno de Robot Operating System (ROS), un *framework* ampliamente utilizado en el desarrollo robótico que facilitó la validación del comportamiento del robot en un entorno virtual. Para el análisis de movimiento, se empleó MATLAB, una herramienta fundamental para el cálculo y la validación de la cinemática directa e inversa del robot. Este proceso incluyó la derivación de la tabla de Denavit-Hartenberg, una metodología estándar para describir la geometría y las transformaciones espaciales entre los eslabones del robot.

Este informe detalla la metodología utilizada, los resultados obtenidos en cada fase del proyecto y las conclusiones derivadas de esta experiencia multidisciplinaria. El trabajo realizado no solo refuerza la comprensión de los principios de la robótica, sino que también demuestra la capacidad del equipo para integrar diversas herramientas y *software* en la resolución de problemas complejos de ingeniería.

2. Marco Teórico

2.1. Cinemática

La cinemática en el campo de la robótica se refiere al estudio del movimiento de los manipuladores sin considerar las fuerzas o torques que lo causan. Es la base para comprender cómo se posiciona y orienta un robot en el espacio.

En esencia, la cinemática aborda las relaciones geométricas entre las articulaciones de un robot y la posición y orientación de su efector final (la "mano" o herramienta del robot). Para un robot de brazos articulados, esto implica entender cómo los movimientos de cada articulación (rotación o traslación) se combinan para determinar la ubicación final de su punta.

Existen dos problemas principales en la cinemática de robots:

- Cinemática Directa: Como se mencionó anteriormente, este problema busca determinar la posición y orientación del efector final del robot *dadas* las configuraciones de sus articulaciones. Imagina que sabes exactamente cuánto ha girado cada articulación de un brazo robótico; la cinemática directa te diría dónde está la pinza al final del brazo. Es un cálculo relativamente sencillo, ya que se sigue un camino directo desde la base del robot hasta su extremo.

- Cinemática Inversa: Este problema es el opuesto y, a menudo, más complejo. Consiste en encontrar las configuraciones de las articulaciones del robot *necesarias* para que el efector final alcance una posición y orientación deseadas en el espacio. Volviendo al ejemplo, si quieres que la pinza del robot agarre un objeto en un punto específico, la cinemática inversa te diría cuánto debe girar cada articulación para lograrlo. Este problema puede tener múltiples soluciones, una única solución o ninguna, dependiendo de la configuración del robot y del punto deseado.

Para resolver estos problemas cinemáticos, especialmente en robots con múltiples grados de libertad, se utilizan herramientas matemáticas como las matrices de transformación homogéneas y metodologías sistemáticas como la convención de Denavit-Hartenberg (D-H), que permite establecer un sistema de coordenadas consistente para cada eslabón y articulación del robot.

La cinemática es un pilar fundamental en el diseño y control de robots, ya que permite planificar trayectorias, evitar obstáculos y asegurar que el robot pueda interactuar de manera precisa con su entorno.

2.1.1. Cinemática Directa

La **cinemática directa** es uno de los problemas fundamentales en el estudio del movimiento de los manipuladores robóticos. Su objetivo principal es determinar la posición y orientación del efector final (la herramienta o pinza del robot) con respecto a un sistema de coordenadas de referencia fijo, basándose en los valores conocidos de las variables de las articulaciones del robot.

En términos más sencillos, si conocemos los ángulos de rotación de cada articulación rotacional y las longitudes de extensión de cada articulación prismática (también llamadas variables de las articulaciones), la cinemática directa nos permite calcular dónde se encuentra exactamente la punta del robot y hacia dónde está apuntando en el espacio.

Proceso y Metodología:

El cálculo de la cinemática directa se realiza de manera secuencial, desde la base del robot hasta su efector final. Generalmente, este proceso involucra los siguientes pasos:

1. **Definición de Sistemas de Coordenadas:** Se asigna un sistema de coordenadas a cada articulación y eslabón del robot. La convención más utilizada y sistemática para esto es la **convención de Denavit-Hartenberg (D-H)**. Esta convención establece un conjunto de reglas para la asignación de marcos de referencia a lo largo de la cadena cinemática del robot, lo que simplifica la derivación de las ecuaciones de transformación.

2. **Parámetros de Denavit-Hartenberg:** Una vez definidos los sistemas de coordenadas D-H, se extraen cuatro parámetros geométricos para cada eslabón-articulación:

    - ai: Longitud del eslabón (distancia entre los ejes Zi y Zi−1).
    - αi: Giro del eslabón (ángulo entre los ejes Zi y Zi−1 alrededor de Xi).
    - di: Desplazamiento de la articulación (distancia a lo largo de Zi entre Xi−1 y Xi).
    - θi: Ángulo de la articulación (ángulo entre los ejes Xi−1 y Xi alrededor de Zi). Para una articulación rotacional, θi es la variable de la articulación. Para una articulación prismática, di es la variable de la articulación.
3. **Matrices de Transformación Homogénea:** Utilizando los parámetros D-H, se derivan matrices de transformación homogénea (Ai) para cada eslabón. Cada matriz Ai describe la transformación (rotación y traslación) del sistema de coordenadas del eslabón i con respecto al sistema de coordenadas del eslabón i−1. Una matriz de transformación homogénea combina información de rotación y traslación en una sola matriz de 4x4, permitiendo describir la pose (posición y orientación) de un cuerpo rígido en el espacio.

4. La forma general de una matriz de transformación homogénea Ai es: $Ai=\cos(\theta i)\sin(\theta i)00-\sin(\theta i)\cos(\alpha i)\cos(\theta i)\cos(\alpha i)\sin(\alpha i)0\sin(\theta i)\sin(\alpha i)-\cos(\theta i)\sin(\alpha i)\cos(\alpha i)0ai\cos(\theta i)ai\sin(\theta i)di1$

5. **Concatenación de Transformaciones:** La transformación total desde el sistema de coordenadas base del robot (frame 0) hasta el sistema de coordenadas del efector final (frame N, donde N es el número de articulaciones) se obtiene multiplicando sucesivamente las matrices de transformación homogénea de cada eslabón: $TN0=A1A2\cdots AN$ La matriz resultante TN0 es una matriz de transformación homogénea de 4x4 que contiene la posición y orientación del efector final con respecto a la base del robot. La submatriz superior izquierda de 3x3 representa la orientación (matriz de rotación), y las tres primeras entradas de la última columna representan la posición (x,y,z).

Importancia:

La cinemática directa es fundamental para:

- **Simulación:** Permite visualizar la postura del robot en un entorno virtual dadas ciertas configuraciones de articulaciones.
- **Gráficos por Computadora:** Es la base para la representación visual de robots en entornos 3D.
- **Planificación de Trayectorias:** Aunque la cinemática inversa es más directa para la planificación de tareas, la cinemática directa es esencial para verificar si una trayectoria de articulaciones planificada resulta en una trayectoria deseada del efector final.
- **Entrenamiento y Pruebas:** Permite a los programadores y operadores entender el espacio de trabajo del robot y predecir su comportamiento.

2.1.2. Cinemática Diferencial

La cinemática diferencial en robótica estudia la relación entre las velocidades angulares y lineales de las articulaciones de un robot y las velocidades angulares y lineales resultantes del efector final. A diferencia de la cinemática directa e inversa (que tratan con posiciones estáticas), la cinemática diferencial se enfoca en cómo los *cambios instantáneos* en las articulaciones afectan la velocidad y rotación de la herramienta del robot.

En esencia, responde a la pregunta: "Si cada articulación se mueve a cierta velocidad, ¿a qué velocidad y en qué dirección se moverá la punta del robot?".

El Jacobiano

El concepto central de la cinemática diferencial es el Jacobiano del robot, denotado como J. Esta es una matriz que relaciona las velocidades de las articulaciones ($\dot{q}$, un vector que contiene las velocidades angulares/lineales de cada articulación) con las velocidades del efector final ($\dot{x}$, un vector que contiene la velocidad lineal y angular del efector final).

La relación se expresa mediante la siguiente ecuación:

$$\dot{x}=J(q)\dot{q}$$

Donde:

- $\dot{x}$: Vector de velocidades del efector final (velocidad lineal en x,y,z y velocidad angular alrededor de x,y,z).
- J(q): Matriz Jacobiana, que depende de la configuración actual de las articulaciones (q).
- $\dot{q}$: Vector de velocidades de las articulaciones.

Importancia

La cinemática diferencial es crucial para:

- Control de Velocidad: Permite controlar la velocidad y dirección del efector final del robot en tiempo real.
- Evitación de Obstáculos: Facilita la planificación de movimientos para esquivar obstáculos ajustando las velocidades de las articulaciones.
- Control de Fuerza: Es fundamental en aplicaciones donde el robot interactúa con el entorno, permitiendo controlar las fuerzas ejercidas por el efector final.
- Singularidades: Ayuda a identificar y evitar "singularidades" en el espacio de trabajo del robot, que son configuraciones donde el robot pierde uno o más grados de libertad y el Jacobiano se vuelve singular (su determinante es cero).

2.1.3. Cinemática Inversa

La cinemática inversa es el problema fundamental en robótica que consiste en determinar los valores de las variables de las articulaciones (ángulos o desplazamientos) que son necesarios para que el efector final del robot alcance una posición y orientación deseadas en el espacio.

En contraste con la cinemática directa (que va de las articulaciones al efector final), la cinemática inversa "regresa" del objetivo en el espacio de trabajo a la configuración articular requerida.

Complejidad y Soluciones

A diferencia de la cinemática directa, que siempre tiene una solución única para una configuración dada, la cinemática inversa puede ser considerablemente más compleja por varias razones:

- Múltiples Soluciones: Para una posición y orientación deseadas del efector final, un robot con múltiples grados de libertad puede tener varias configuraciones articulares posibles que logren el mismo objetivo. Por ejemplo, un brazo robótico puede alcanzar un punto con el "codo arriba" o el "codo abajo".
- No Soluciones: Algunas posiciones y orientaciones pueden estar fuera del espacio de trabajo alcanzable del robot, lo que significa que no existe una solución para la cinemática inversa.
- Dificultad Matemática: La resolución de la cinemática inversa a menudo implica resolver sistemas de ecuaciones no lineales, lo que puede ser computacionalmente intensivo y no siempre tiene una solución analítica explícita.

Métodos de Resolución

Existen principalmente dos enfoques para resolver la cinemática inversa:

1. Métodos Analíticos: Implican derivar ecuaciones matemáticas explícitas que resuelven directamente para las variables de las articulaciones. Estos métodos son rápidos y precisos cuando se pueden obtener, pero solo son posibles para robots con geometrías simples o un número limitado de grados de libertad (por ejemplo, robots de 3 o 6 GDL con ciertas configuraciones como PUMA o SCARA).
2. Métodos Numéricos/Iterativos: Utilizan algoritmos que, partiendo de una estimación inicial, refinan iterativamente las variables de las articulaciones hasta que la posición del efector final se acerca lo suficiente al objetivo deseado. Estos métodos son más generales y pueden aplicarse a casi cualquier robot, pero son más lentos, pueden quedar atrapados en mínimos locales y requieren una buena estimación inicial. Un ejemplo común es el método basado en el Jacobiano (Pseudo-inversa del Jacobiano).

2. ROS

ROS (Robot Operating System) es un *framework* de *software* de código abierto diseñado específicamente para el desarrollo de aplicaciones robóticas. Aunque su nombre incluye "Operating System" (Sistema Operativo), ROS no es un sistema operativo en el sentido tradicional (como Windows o Linux), sino más bien un conjunto de herramientas, librerías y convenciones que facilitan la creación de *software* para robots.

Componentes Clave y Funcionalidades:

ROS está estructurado en un modelo de nodos (nodes), donde cada nodo es un proceso ejecutable que realiza una función específica (por ejemplo, un nodo para leer datos de un sensor, otro para controlar los motores, y otro para la planificación de la navegación). Estos nodos se comunican entre sí utilizando un sistema de mensajería de publicación/suscripción (publish/subscribe) y servicios:

- Nodos: Programas ejecutables que realizan tareas específicas. Permiten modularizar el *software* robótico, lo que facilita el desarrollo, la depuración y la reutilización de componentes.
- Temas (Topics): Canales de comunicación por donde los nodos publican y se suscriben a mensajes. Un nodo puede publicar datos (ej. lecturas de sensores) en un tema, y otros nodos pueden suscribirse a ese tema para recibir y procesar esos datos.
- Mensajes (Messages): Estructuras de datos estandarizadas que se envían a través de los temas. Contienen la información que los nodos intercambian (ej. datos de odometría, imágenes, comandos de velocidad).
- Servicios (Services): Mecanismos para la comunicación síncrona de solicitud/respuesta entre nodos. Un nodo cliente envía una solicitud y espera una respuesta de un nodo servidor. Útil para tareas que requieren una acción inmediata y una confirmación.
- Parámetros (Parameters): Un servidor de parámetros que permite almacenar y recuperar datos de configuración para los nodos.
- Bags (Bags): Herramienta para grabar y reproducir datos de temas ROS, útil para depuración, análisis y desarrollo *offline*.

Ventajas y Aplicaciones:

ROS ha ganado una inmensa popularidad en la comunidad robótica debido a sus numerosas ventajas:

- Modularidad y Reusabilidad: Fomenta la creación de componentes de *software* independientes y reutilizables, lo que acelera el desarrollo.
- Interoperabilidad: Permite que componentes escritos en diferentes lenguajes de programación (C++, Python, etc.) se comuniquen entre sí.
- Ecosistema Extenso: Cuenta con una gran cantidad de herramientas, paquetes y librerías preexistentes para una amplia gama de tareas robóticas (navegación, manipulación, visión por computadora, simulación, etc.).
- Comunidad Activa: Una vasta comunidad de desarrolladores contribuye y da soporte, lo que facilita la resolución de problemas y el aprendizaje.
- Simulación: Integra herramientas como Gazebo, que permiten simular robots y entornos complejos para probar algoritmos sin necesidad de *hardware* físico.

Aplicaciones: ROS es utilizado en una variedad de proyectos robóticos, desde robots de investigación y prototipos, hasta aplicaciones industriales y de servicio, incluyendo robots móviles, manipuladores robóticos, drones, vehículos autónomos y sistemas de visión robótica.

2.2.1. Nodo (Node)

En el contexto de Robot Operating System (ROS), un nodo es la unidad de procesamiento fundamental y un componente clave del *framework*. Es esencialmente un programa ejecutable que realiza una función específica dentro del sistema robótico.

Imagina un robot como una orquesta; cada músico sería un nodo. Cada músico (nodo) tiene una tarea definida (tocar un instrumento), y todos trabajan juntos, comunicándose entre sí, para producir una sinfonía (el comportamiento del robot).

Características Clave de un Nodo:

Modularidad: Los nodos permiten descomponer un sistema robótico complejo en componentes más pequeños y manejables. Esto facilita el desarrollo, la depuración y el mantenimiento del *software*. Por ejemplo, un robot puede tener un nodo para controlar sus motores, otro para leer datos de sensores, otro para la planificación de la navegación, etc.

Independencia: Cada nodo es un proceso independiente. Si un nodo falla, no necesariamente derriba todo el sistema, lo que aumenta la robustez del robot.

Comunicación: Los nodos se comunican entre sí principalmente a través de temas (topics) de publicación/suscripción y, en menor medida, a través de servicios (services). Esto permite que la información fluya entre las diferentes partes del sistema. Por ejemplo, un nodo de "visión" podría publicar datos de cámaras en un tema, y un nodo de "control" podría suscribirse a ese tema para recibir esos datos y ajustar el movimiento del robot.

Reusabilidad: Dado que los nodos son módulos independientes, pueden ser reutilizados en diferentes proyectos robóticos o intercambiados por otras implementaciones sin afectar al resto del sistema, siempre y cuando mantengan la misma interfaz de comunicación.

Lenguajes de Programación: Los nodos pueden ser escritos en diversos lenguajes de programación (comúnmente C++ y Python), lo que permite a los desarrolladores utilizar la herramienta más adecuada para cada tarea.

2.2.2. Tema (Topic)

En el contexto de Robot Operating System (ROS), un Tema (Topic) es el principal mecanismo para la comunicación asíncrona (no bloqueante) de datos entre nodos. Piensa en un tema como un canal de difusión donde los nodos pueden publicar información y otros nodos pueden suscribirse para recibir esa información.

Funcionamiento y Propósito:

1. Publicadores (Publishers): Un nodo que desea compartir datos con otros nodos actúa como un "publicador". Envía mensajes a un tema específico. Por ejemplo, un nodo que

lee datos de un sensor LiDAR podría publicar las lecturas de distancia en un tema llamado /lidar_scans.

2. Suscriptores (Subscribers): Un nodo que necesita esos datos actúa como un "suscriptor". Se registra para recibir mensajes de un tema particular. El nodo de navegación, por ejemplo, podría suscribirse al tema /lidar_scans para obtener los datos necesarios para evitar obstáculos.

3. Mensajes (Messages): La información que se envía a través de un tema se empaqueta en mensajes. Estos mensajes tienen un tipo de dato definido (por ejemplo, sensor_msgs/LaserScan para datos de LiDAR, geometry_msgs/Twist para comandos de velocidad). Este tipo de mensaje asegura que tanto el publicador como el suscriptor entiendan el formato de los datos que se están intercambiando.

4. Comunicación Uno a Muchos: Un tema puede tener múltiples publicadores y múltiples suscriptores simultáneamente. Esto permite que la misma información sea compartida con varias partes del sistema de manera eficiente.

2.2.3. Mensaje (Message)

En el contexto de Robot Operating System (ROS), un Mensaje (Message) es el formato de datos estructurado que los nodos de ROS utilizan para comunicarse entre sí a través de los Temas (Topics). Es, en esencia, el contenido real que se transmite por el canal de comunicación.

Piensa en los mensajes como los paquetes de información que se envían por correo. El Tema sería la dirección a la que se envía el paquete, y el Mensaje es lo que va dentro del paquete: los datos específicos que un nodo quiere compartir con otro.

Características y Propósito:

1. Datos Estructurados: Los mensajes no son simplemente cadenas de texto; son estructuras de datos bien definidas. Tienen un formato preestablecido que especifica los tipos y nombres de los campos que contienen. Por ejemplo, un mensaje para un sensor de distancia podría tener campos para la lectura de distancia, la unidad de medida y un *timestamp*.
2. Tipos de Mensajes: ROS define una gran cantidad de "tipos de mensajes" estándar para datos comunes en robótica (como sensor_msgs/LaserScan para datos de láser, geometry_msgs/Twist para comandos de velocidad lineal y angular, nav_msgs/Odometry para datos de odometría, etc.). También puedes crear tus propios tipos de mensajes personalizados si los estándar no se ajustan a tus necesidades.
3. Serialización y Deserialización: Cuando un nodo publica un mensaje, este se "serializa" (se convierte en un formato binario que puede ser transmitido por la red). Cuando un nodo suscriptor recibe el mensaje, este se "deserializa" (se convierte de nuevo a la estructura de datos original) para que el nodo pueda acceder a la información.
4. Consistencia: El uso de tipos de mensajes definidos asegura que tanto el nodo publicador como el nodo suscriptor interpreten los datos de la misma manera, garantizando la consistencia en la comunicación.

2.2.4. Servicio (Service)

En ROS, un Servicio (Service) es un mecanismo de comunicación que permite a los nodos enviar una solicitud (request) y recibir una respuesta (response) de forma síncrona (bloqueante). A diferencia de los Temas, que son para flujos continuos de datos, los Servicios se utilizan para acciones puntuales que requieren un resultado inmediato.

Imagina que necesitas que alguien te traiga un objeto específico. En lugar de gritar repetidamente (como un Tema), simplemente le pides una vez (solicitud) y esperas hasta que te lo traiga (respuesta).

Funcionamiento:

1. Nodo Cliente: Un nodo actúa como "cliente" cuando necesita que otro nodo realice una tarea específica y le devuelva un resultado. Envía una solicitud al Servicio.
2. Nodo Servidor: Otro nodo actúa como "servidor", esperando solicitudes para un Servicio particular. Cuando recibe una solicitud, realiza la tarea y envía una respuesta de vuelta al cliente.
3. Definición de Servicio: Al igual que los Mensajes, los Servicios tienen tipos definidos que especifican la estructura de la solicitud y la respuesta. Esto asegura que cliente y servidor entienden el formato de los datos que intercambian.

Uso e Importancia:

Los Servicios son ideales para:

- Acciones Discretas: Como activar o desactivar un motor, solicitar la posición actual de un actuador, o pedir a un brazo robótico que vaya a una pose específica.
- Consultas de Información: Donde se necesita una respuesta inmediata a una pregunta específica (ej. "¿Estoy cerca de un obstáculo?").
- Configuración: Cambiar parámetros o modos de operación de un nodo en tiempo real.

2.2.5. Gazebo

Gazebo es un simulador 3D de robots de código abierto que permite probar algoritmos complejos, diseñar robots y realizar pruebas de regresión en un entorno virtual realista. Es una herramienta indispensable en el desarrollo de robótica, especialmente cuando se trabaja con ROS, ya que permite simular robots, sensores y entornos con un alto grado de fidelidad física.

Características y Utilidad:

- Simulación Física Realista: Gazebo utiliza un motor de física robusto (como ODE, Bullet, Simbody o DART) para simular la dinámica de cuerpos rígidos, la fricción, la

gravedad y las interacciones entre objetos. Esto permite predecir con precisión cómo se comportará un robot en el mundo real.

- Modelado de Robots y Entornos: Permite importar modelos 3D de robots (comúnmente en formato URDF o SDFormat) y crear entornos complejos con edificios, terreno, obstáculos y otros objetos. Esto es crucial para probar algoritmos de navegación, manipulación y percepción en escenarios variados.
- Simulación de Sensores: Gazebo puede simular una amplia gama de sensores comunes en robótica, como cámaras, láseres (LiDAR), sensores de fuerza/par, GPS, IMUs (Unidades de Medición Inercial) y sensores de contacto. Los datos generados por estos sensores simulados pueden ser directamente integrados con nodos ROS, como si vinieran de *hardware* real.
- Integración con ROS: Gazebo se integra de manera muy estrecha con ROS. Los modelos de robots y sensores en Gazebo pueden publicar datos en Temas ROS y suscribirse a comandos ROS, lo que permite que el *software* desarrollado en ROS para un robot real funcione directamente en la simulación sin modificaciones significativas.
- Depuración y Desarrollo Seguro: Permite a los desarrolladores probar y depurar su *software* en un entorno seguro y reproducible, sin riesgo de dañar *hardware* costoso o poner en peligro a personas. Esto acelera el ciclo de desarrollo y permite experimentar con configuraciones complejas.
- Visualización: Ofrece una interfaz gráfica de usuario para visualizar la simulación en 3D, inspeccionar el estado de los robots, y manipular objetos en el entorno virtual.

2.2.6. RViz

RViz (ROS Visualization) es una herramienta de visualización 3D flexible y potente dentro del ecosistema ROS. Su función principal es proporcionar una representación gráfica del estado de un robot, sus sensores y el entorno que percibe, utilizando los datos publicados en Temas ROS. A diferencia de Gazebo, que es un simulador físico, RViz es puramente una herramienta de visualización que no simula la dinámica, sino que "dibuja" lo que el robot está viendo o haciendo.

RViz es capaz de visualizar una amplia variedad de datos ROS, como modelos de robot (URDF) y su configuración actual, nubes de puntos de sensores 3D, mapas 2D y 3D del entorno, trayectorias y odometría, transformaciones (TF) entre sistemas de coordenadas, y *streams* de imágenes de cámaras. Es una herramienta indispensable para la depuración y el diagnóstico del *software* robótico, ya que permite a los desarrolladores ver en tiempo real si los sensores funcionan correctamente, si los algoritmos de localización y mapeo están construyendo el entorno de forma adecuada, y si la planificación de movimientos es coherente. Además, RViz es altamente configurable, permitiendo al usuario decidir qué datos visualizar, cómo representarlos (colores, tamaños, transparencia) y desde qué punto de vista. Es crucial entender que RViz no es un simulador; no simula el comportamiento físico del robot ni del entorno. Solo muestra lo que el robot (sea físico o simulado en Gazebo) está reportando a través de ROS. En resumen, RViz es la "ventana" al mundo interno y percibido de un robot basado en ROS, ofreciendo una representación gráfica esencial para el monitoreo, la depuración y la comprensión del comportamiento del sistema.

2.3. Dinámica

Modelo Dinámico Estándar de un Robot

El modelo dinámico estándar de un robot es una representación matemática que describe cómo las fuerzas y los torques actúan sobre el robot para producir movimiento. A diferencia de la cinemática, que solo se ocupa del movimiento (posición, orientación, velocidad), la dinámica considera las causas de ese movimiento: las fuerzas, los torques, la masa y la inercia. Este modelo es crucial para el diseño de controladores de movimiento que calculan los torques de las articulaciones necesarios para que el robot siga una trayectoria deseada.

La forma general del modelo dinámico de un manipulador robótico con n grados de libertad (GDL), expresado en las coordenadas articulares, es la siguiente:

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

Donde:

- $q \in Rn$: Vector de posiciones articulares (ángulos o desplazamientos).
- $\dot{q} \in Rn$: Vector de velocidades articulares.
- $\ddot{q} \in Rn$: Vector de aceleraciones articulares.
- $\tau \in Rn$: Vector de torques (para articulaciones rotacionales) o fuerzas (para articulaciones prismáticas) aplicadas en las articulaciones. Estos son los comandos de control.
- $M(q) \in Rn \times n$: Matriz de masa o inercia del robot.
- $C(q,\dot{q})\dot{q} \in Rn$: Vector que representa los efectos de las fuerzas de Coriolis y centrífugas.
- $G(q) \in Rn$: Vector que representa los torques o fuerzas debidos a la gravedad.

Matriz de Masa o Inercia (M(q))

La matriz de masa o inercia M(q) es una matriz simétrica y definida positiva que describe la inercia efectiva del robot en una configuración articular dada q. Esta matriz varía con la configuración del robot, ya que la inercia que "siente" el motor de una articulación no solo depende de la masa y geometría de esa articulación, sino también de la masa y geometría de todos los eslabones y objetos aguas abajo que debe mover. Los elementos de la diagonal principal de M(q) representan las inercias de las articulaciones, mientras que los elementos fuera de la diagonal representan el acoplamiento inercial entre las articulaciones; es decir, cómo el movimiento de una articulación afecta la inercia que otra articulación debe superar.

Matriz de Coriolis y Términos Centrífugos ($C(q,\dot{q})\dot{q}$)

La matriz de Coriolis $C(q,\dot{q})$ es un término que, cuando se multiplica por el vector de velocidades articulares $\dot{q}$, produce un vector que representa las fuerzas de Coriolis y centrífugas.

- Las fuerzas centrífugas son fuerzas "hacia afuera" que surgen cuando una masa gira alrededor de un eje. En un robot, se manifiestan cuando los eslabones giran, empujando los componentes del brazo lejos del centro de rotación.
- Las fuerzas de Coriolis son fuerzas que aparecen en sistemas en rotación y actúan perpendicularmente a la dirección del movimiento. En un robot, surgen del acoplamiento entre el movimiento de dos articulaciones cuando ambas se mueven simultáneamente.

Por ejemplo, al mover una articulación, el efecto de Coriolis puede producir un torque inesperado en otra articulación que también se está moviendo. Estos términos son no lineales y dependen tanto de la posición q como de la velocidad $\dot{q}$ de las articulaciones.

Vector de Gravedad (G(q))

El vector de gravedad G(q) representa los torques o fuerzas en las articulaciones que son necesarios para contrarrestar el efecto de la gravedad sobre la masa de cada eslabón del robot. Este vector depende únicamente de la configuración articular q y de las propiedades de masa de los eslabones (masa y centro de masa). Por ejemplo, si un brazo robótico está extendido horizontalmente, la gravedad ejercerá un torque significativo en sus articulaciones para mantener esa postura, mientras que si está completamente vertical, el efecto gravitatorio sobre algunas articulaciones podría ser mínimo.

2.3.1. Matriz de masa o inercia

Matriz de Masa o Inercia (M(q))

La matriz de masa o inercia M(q) representa la "resistencia" del robot al cambio de movimiento en cada una de sus articulaciones, considerando la configuración actual del robot q. Es el equivalente multidimensional de la masa en la ley de Newton F=ma. Una masa mayor implica una mayor fuerza para lograr la misma aceleración. De manera similar, una mayor inercia (elementos más grandes en la matriz M(q)) significa que se requieren mayores torques o fuerzas en las articulaciones para producir una aceleración articular dada.

Relación con la Aceleración Máxima

Para entender cómo M(q) define la aceleración máxima, consideremos el modelo dinámico estándar simplificado, ignorando los términos de Coriolis, centrífugos y gravedad por un momento para enfocarnos en la relación directa entre torque y aceleración:

$\tau = M(q)\ddot{q}$

Despejando la aceleración ($q^{¨}$), obtenemos:

$q^{¨}=M(q)-1\tau$

Esta ecuación revela la relación directa: la aceleración articular ($q^{¨}$) es inversamente proporcional a la matriz de masa M(q) y directamente proporcional al vector de torques aplicados τ.

Los motores de las articulaciones de un robot tienen límites máximos de torque que pueden generar (τmax). Por lo tanto, para una configuración articular dada q, la aceleración máxima que el robot puede alcanzar en una dirección particular de las articulaciones estará limitada por:

1. Los torques máximos disponibles (τmax): Cuanto mayores sean los torques que los motores pueden aplicar, mayor será la aceleración posible.
2. La matriz de masa o inercia M(q):
    - Inversa Proporcionalidad: Un robot con una "masa" o "inercia" efectiva alta (elementos grandes en M(q)) en una configuración particular será más difícil de acelerar. Para un torque máximo dado, una mayor inercia resultará en una *menor* aceleración máxima.
    - Dependencia de la Configuración: M(q) varía con la configuración del robot. Por ejemplo, un brazo robótico extendido tendrá una mayor inercia efectiva para las articulaciones de la base que cuando el brazo está plegado. Esto significa que el robot puede alcanzar diferentes aceleraciones máximas en diferentes configuraciones, incluso con los mismos torques aplicados. Cuando el robot está "extendido", su inercia es mayor, y por ende, su aceleración máxima será menor. Cuando está "recogido", su inercia es menor, permitiendo mayores aceleraciones.

2.3.2. Matriz de coriolis

La matriz de Coriolis $C(q,q^{˙})$ capturan los efectos de las fuerzas de Coriolis y centrífugas en el movimiento de un robot. Estos términos son no lineales y surgen cuando el robot está en movimiento.

- Las fuerzas centrífugas empujan los eslabones "hacia afuera" cuando giran, como la fuerza que sientes al tomar una curva en un coche.
- Las fuerzas de Coriolis aparecen cuando hay movimiento simultáneo en múltiples articulaciones, generando torques en direcciones perpendiculares al movimiento principal. Es un efecto de acoplamiento entre las velocidades de las articulaciones.

Ambas fuerzas dependen de la posición (q) y, crucialmente, de las velocidades ($\dot{q}$) de las articulaciones. Su efecto combinado se traduce en torques que el controlador del robot debe compensar para asegurar un movimiento preciso.

2.3.3. Vector de gravedad

El vector de gravedad G(q) representa los torques o fuerzas en las articulaciones que son necesarios para compensar el peso de los eslabones del robot, es decir, la fuerza de la gravedad. Este vector depende únicamente de la configuración actual del robot (q) y de las propiedades de masa de sus eslabones.

Por qué el Vector de Gravedad en su Valor Máximo debe ser Contrarrestado con la Fuerza de Cada Motor

Cuando un robot está extendido horizontalmente, el vector de gravedad G(q) alcanza su valor máximo en las articulaciones que soportan el peso de los eslabones extendidos. Esto ocurre porque la distancia perpendicular desde el centro de masa de cada eslabón hasta el eje de rotación de la articulación correspondiente es máxima, lo que genera el mayor *brazo de palanca* para la fuerza de la gravedad.

En esta configuración:

1. Máximo Brazo de Palanca: Cada eslabón extendido horizontalmente crea el brazo de palanca más largo posible con respecto a la articulación que lo precede. La fuerza de la gravedad, actuando verticalmente hacia abajo sobre el centro de masa de cada eslabón, genera el torque gravitatorio más grande en esa articulación.

2. Necesidad de Contrarrestar: Para que el robot mantenga esa postura extendida horizontalmente, o para moverlo en esa configuración, cada motor de las articulaciones relevantes (especialmente las más cercanas a la base) debe generar un torque o fuerza igual y opuesto al torque gravitatorio que actúa sobre su eslabón y todos los eslabones subsiguientes. Sin este torque de compensación, el robot simplemente colapsaría debido a su propio peso.
3. Carga Estática y Dinámica: Este torque gravitatorio es una carga estática constante que los motores deben soportar incluso si el robot no se está moviendo. Si el robot está en movimiento, este torque gravitatorio se suma a los torques necesarios para la aceleración y los efectos de Coriolis, aumentando la demanda total sobre los motores.
4. Eficiencia Energética y Diseño: Contrarrestar la gravedad continuamente consume energía y puede ser un factor limitante en el diseño del robot (requiriendo motores más potentes). Por esta razón, algunos robots utilizan contrapesos o mecanismos de equilibrio para reducir la carga de gravedad en los motores.

2.3.4. Fricción

La fricción es una fuerza de resistencia que se opone al movimiento relativo, o al intento de movimiento, entre dos superficies en contacto. Esta fuerza surge de las interacciones a nivel microscópico entre las irregularidades de las superficies.

Existen principalmente dos tipos de fricción:

1. Fricción Estática: Es la fuerza que se opone al inicio del movimiento. Actúa cuando los objetos están en reposo relativo pero hay una fuerza intentando moverlos. La fricción estática puede variar en magnitud, aumentando hasta un valor máximo antes de que el movimiento comience. Si la fuerza aplicada supera este valor máximo, el objeto empezará a moverse.

2. Fricción Cinética (o Dinámica): Es la fuerza que se opone al movimiento una vez que este ha comenzado. Es generalmente menor que la fricción estática máxima para un par

de superficies dadas y se mantiene relativamente constante mientras hay movimiento relativo.

Características Generales de la Fricción:

- Oposición al Movimiento: Siempre actúa en dirección opuesta a la dirección del movimiento (o intento de movimiento).
- Dependencia de la Fuerza Normal: La magnitud de la fricción es directamente proporcional a la fuerza normal (la fuerza perpendicular que presiona las dos superficies juntas). Cuanto mayor sea la fuerza normal, mayor será la fricción.
- Coeficiente de Fricción: La constante de proporcionalidad entre la fuerza de fricción y la fuerza normal se conoce como coeficiente de fricción (μ). Hay un coeficiente de fricción estático (μs) y un coeficiente de fricción cinético (μk). Estos coeficientes dependen de la naturaleza de las superficies en contacto.
- Independencia del Área de Contacto (en primera aproximación): Para superficies secas y limpias, la fricción no depende significativamente del área de contacto aparente, sino de la fuerza normal.
- Disipación de Energía: La fricción convierte la energía mecánica en otras formas de energía, principalmente calor.

Importancia en Robótica:

En robótica, la fricción es un factor importante a considerar en el diseño y control:

- Articulaciones: La fricción en las articulaciones de un robot (debido a rodamientos, engranajes, etc.) requiere que los motores ejerzan un torque adicional para superarla, especialmente al inicio del movimiento. Esto influye en el consumo de energía y la precisión del control.
- Agarre (End-Effectors): La fricción es deseable en las pinzas o garras del robot para asegurar un agarre firme de los objetos sin que se resbalen.
- Movimiento en Superficies: En robots móviles, la fricción entre las ruedas/orugas y el suelo es crucial para la tracción y el movimiento.

Comprender y modelar la fricción es esencial para diseñar robots eficientes, precisos y que funcionen de manera fiable.

Fricción Estática o Seca

La fricción estática (o seca) es la fuerza de resistencia que se opone al inicio del movimiento entre dos superficies sólidas en contacto sin lubricación. Actúa hasta un valor máximo y debe ser superada para que el movimiento comience.

Fricción Dinámica o Viscosa

La fricción dinámica (o cinética/viscosa) es la fuerza de resistencia que se opone al movimiento una vez que este ha comenzado. En el contexto de robótica, la fricción viscosa es una componente de esta fricción dinámica que es directamente proporcional a la velocidad del movimiento y tiende a cero cuando la velocidad es cero.

2.3.5. Perturbaciones

En el contexto de un sistema, ya sea robótico, mecánico o de control, una perturbación se refiere a cualquier factor o influencia externa no deseada que afecta el comportamiento o el rendimiento esperado del sistema. Estas influencias pueden causar que el sistema se desvíe de su estado o trayectoria deseada.

Las perturbaciones pueden ser de diversa naturaleza:

- Externas al sistema: Como ráfagas de viento inesperadas que empujan a un robot, cambios en la carga útil que manipula un brazo robótico, o vibraciones en el suelo donde se encuentra un robot móvil.
- Internas al sistema: Como la fricción en las articulaciones, la inercia variable de los eslabones, el ruido en los sensores o las inexactitudes en los actuadores.

El objetivo en el diseño de sistemas de control es hacer que el sistema sea robusto, es decir, capaz de mantener su rendimiento deseado a pesar de la presencia de estas perturbaciones. Los

modelos dinámicos, como el que describes para tu robot, son fundamentales para entender cómo estas perturbaciones pueden afectar el sistema y cómo un controlador puede compensarlas.

2.4. Control

Explicación del Diagrama de Bloques: Figura 2.2 - Diagrama de Control de un Robot Industrial

El diagrama de la Figura 2.2 ilustra un sistema de control de robot industrial típico, que opera en un ciclo de realimentación para lograr el movimiento deseado. Este diagrama combina la planificación de la trayectoria en el espacio del efector final con el control dinámico en el espacio articular, incorporando la cinemática inversa y la compensación de perturbaciones.

El flujo principal del control se puede desglosar en las siguientes etapas:

1. Valores del Efector Final Definidos por el Usuario:
    - Aquí es donde el usuario o el programador define la tarea del robot. Se especifican puntos (coordenadas) deseados en el espacio para el efector final, junto con velocidad máxima, aceleración máxima y el tipo de trayectoria (por ejemplo, lineal, circular, *joint-space*). Estas son las especificaciones de alto nivel de lo que el robot debe hacer.
2. Planificación de Trayectoria:
    - Este bloque toma las especificaciones del usuario y genera una trayectoria suave y continua para el efector final en el tiempo. La salida de este bloque son los Valores del efector final deseados en cada instante:
        - p: Posición lineal deseada.
        - v: Velocidad lineal deseada.
        - a: Aceleración lineal deseada.
        - $\eta | q$: Orientación deseada (ángulos de Euler o cuaternión).
        - ω: Velocidad angular deseada.
        - α: Aceleración angular deseada.
    - Esta trayectoria es el camino que el robot debería seguir en el espacio 3D.

3. Cinemática Inversa:

    - La Cinemática Inversa es un bloque crucial que traduce la trayectoria deseada del efector final (en el espacio cartesiano) a la secuencia correspondiente de movimientos de las articulaciones del robot (en el espacio articular).
    - La entrada son los valores de posición (p), velocidad (v), aceleración (a), orientación ($\eta|q$), velocidad angular (ω) y aceleración angular (α) del efector final.
    - La salida son los Valores articulares deseados:
        - qd: Posición articular deseada.
        - $\dot{q}d$: Velocidad articular deseada.
        - $\ddot{q}d$: Aceleración articular deseada.
    - Estos valores articulares qd, $\dot{q}d$, $\ddot{q}d$ son ahora las referencias para el control de las articulaciones.
4. Error y Controlador (Bucle de Realimentación):

    - En esta parte del diagrama, se implementa el control en lazo cerrado.
    - El sistema calcula un Error comparando la posición articular deseada (qd) y la velocidad articular deseada ($\dot{q}d$) con las posiciones articulares reales (q) y velocidades articulares reales ($\dot{q}$) medidas o estimadas del robot. El diagrama muestra xd y x genéricamente, pero en este contexto se refieren a las variables de estado articulares deseadas y reales.
    - El Controlador (ej. PID, control de par) toma este error y genera una señal de control u. Esta señal u representa los torques (τ) o fuerzas (F) que los actuadores del robot deben aplicar a las articulaciones para reducir el error y hacer que el robot siga la trayectoria deseada.
5. Suma de Perturbaciones:

    - La señal de control u generada por el controlador se suma a un término de perturbación (ud). Estas perturbaciones pueden ser fuerzas externas no modeladas, fricción inesperada, errores en la modelización de la dinámica del

robot, etc. Este bloque representa cómo estas fuerzas indeseadas afectan el torque total aplicado al robot.

6. Dinámica del Robot:

    - Este es el bloque más complejo y representa el comportamiento físico del robot.
    - La entrada a este bloque es el torque/fuerza total aplicado a las articulaciones (controlador + perturbaciones).
    - La Dinámica del robot se describe por la ecuación $M(q)\ddot{q}+C(q,\dot{q})\dot{q}+G(q)=\tau$.
    - En el diagrama, se presenta de dos formas:
        - No lineal ($\dot{x}=f(x,u)$): Esta es la representación completa y real de la dinámica del robot, donde x representa el estado (posiciones y velocidades articulares) y u son los torques/fuerzas. Es la forma más precisa pero compleja de manejar.
        - Linealizada ($\dot{x}=Ax+Bu$): A menudo, para simplificar el diseño del controlador, la dinámica se linealiza alrededor de un punto de operación. A y B son matrices que describen el sistema linealizado.
    - El bloque de integración ($\int dt$) transforma las aceleraciones en velocidades y luego las velocidades en posiciones, representando el movimiento físico del robot.
    - Las salidas de este bloque son las posiciones articulares reales (q) y las velocidades articulares reales ($\dot{q}$) del robot, que luego se realimentan al bloque de error para cerrar el lazo de control.

3. Desarrollo

3.1. Características del Robot

El robot diseñado y simulado para este proyecto es un manipulador articulado cuya configuración se describe mediante la tabla de parámetros de Denavit-Hartenberg (D-H) que se presenta a continuación. Esta tabla es fundamental para definir la geometría y las relaciones espaciales entre los eslabones y articulaciones del robot, permitiendo el cálculo de su cinemática. Además, se incluyen las características de movimiento y límites operativos de cada articulación.

Tabla 3.1: Parámetros de Denavit-Hartenberg y Características de las Articulaciones del Robot

Articulación	θ (grados)	d (mm)	a (mm)	α (grados)	Tipo	Mínimo (θ/d)	Máximo (θ/d)	q˙max (grados/s)	q¨max (grados/s^2)
1	0	458	0	90	r	-180	180	180	360
2	90	0	500	0	r	-90	90	180	360
3	90	424	0	90	r	-180	0	180	360
4	90	172	0	90	r	-180	0	180	360

Descripción de Columnas:

- Articulación: Número de la articulación, comenzando desde la base (articulación 1).
- θ (grados): Ángulo de rotación de la articulación i alrededor del eje Zi−1, desde el eje Xi−1 hasta el eje Xi. Es la variable de la articulación para articulaciones rotacionales.
- d (mm): Desplazamiento del eslabón i a lo largo del eje Zi−1, desde el origen del sistema i−1 hasta la intersección de Xi−1 y Zi. Es la variable de la articulación para articulaciones prismáticas.
- a (mm): Longitud del eslabón i, que es la distancia entre el eje Zi−1 y el eje Zi, medida a lo largo del eje Xi.
- α (grados): Ángulo de giro del eslabón i, que es el ángulo entre el eje Zi−1 y el eje Zi, medido alrededor del eje Xi.
- Tipo: Indica el tipo de articulación. 'r' denota una articulación rotacional (revolute).
- Mínimo (θ/d): El límite inferior del rango de movimiento de la articulación (en grados para rotacionales, o mm para prismáticas).
- Máximo (θ/d): El límite superior del rango de movimiento de la articulación (en grados para rotacionales, o mm para prismáticas).
- q˙max (grados/s): Velocidad angular máxima permitida para la articulación (en grados por segundo).

- q¨max (grados/s^2): Aceleración angular máxima permitida para la articulación (en grados por segundo al cuadrado).

Esta tabla proporciona una especificación completa de las propiedades cinemáticas y los límites de movimiento de cada articulación, siendo esencial para la modelización, simulación y control del robot.

3.1.1. Partes

Como parte del desarrollo de este proyecto, hemos diseñado un brazo robótico articulado que se compone de diversas piezas clave, cada una con una función específica que, en conjunto, habilitan su capacidad de movimiento y funcionalidad. A continuación, detallo las principales componentes de nuestro diseño.

Componentes Estructurales y Articulaciones

El punto de partida de nuestro brazo robótico es la base. Esta es la parte fundamentalmente fija del sistema, diseñada para ser atornillada o asegurada firmemente a una superficie de trabajo. Además de proporcionar estabilidad, la base aloja el motor principal que permite el giro horizontal de todo el conjunto del brazo. Hemos contemplado que esta pieza también pueda albergar componentes electrónicos esenciales, como los controladores de motores y la fuente de alimentación, optimizando la integración del sistema.

Sobre la base, hemos dispuesto el Soporte de la Columna o soporte del brazo principal. Esta pieza es el nexo directo entre la base y el primer segmento móvil del brazo. Su diseño permite alojar el motor encargado del movimiento vertical (subida y bajada) de este primer brazo, e integra los rodamientos necesarios para asegurar un movimiento angular suave y preciso en esta articulación.

El Primer Brazo constituye el eslabón de mayor longitud en nuestro diseño. Este componente se eleva desde la base y es crucial para la transmisión del movimiento vertical a la estructura superior del brazo. Su robustez es fundamental, ya que soporta una considerable porción del peso total del manipulador y es responsable de transferir el torque necesario a las articulaciones

subsiguientes. Hemos considerado en su diseño la provisión de espacio interno para el enrutamiento y protección de los cables que conectan a los motores de las secciones superiores.

La conexión entre el primer brazo y el siguiente segmento se materializa en la pieza que denominamos Codo. Funciona como una articulación pivotal, permitiendo el movimiento de extensión y retracción (hacia adelante y atrás) del segundo brazo. En su interior, el codo aloja el motor correspondiente a esta articulación, junto con los mecanismos de transmisión de movimiento que aseguran un control eficaz.

El Segundo Brazo, de una longitud inferior al primero, se extiende desde el codo hasta la muñeca. Su función principal es ajustar la distancia entre la herramienta de trabajo y la base del robot, acercando o alejando el *end-effector* del punto de operación. Similar al primer brazo, este segmento también integra canales para la conducción de cables hacia los motores situados en la muñeca.

La Muñeca representa una articulación compleja y multifuncional en nuestro diseño. Permite los movimientos de giro tanto vertical como horizontal de la herramienta o actuador final. Generalmente, hemos previsto que integre dos motores de menor tamaño dedicados a gestionar estos movimientos, complementados en algunos casos con engranajes o poleas para optimizar la transmisión de torque y la precisión posicional.

Finalmente, en el extremo distal del brazo se ubica la Herramienta o Actuador Final (End-effector). Esta pieza es personalizable y su forma y función varían ampliamente según la aplicación específica del robot. Puede ser una pinza para agarre, un destornillador, una cámara de inspección, un dispensador o cualquier otro accesorio necesario para la tarea. Este componente suele incorporar un motor adicional que facilita sus funciones operativas, como la apertura, cierre o rotación.

Materiales y Elementos Complementarios

Complementando las piezas estructurales, el diseño de nuestro brazo robótico incorpora diversos elementos esenciales para su operación. Los motores se seleccionan entre opciones NEMA o servomotores, dimensionados según el torque requerido en cada articulación para asegurar un rendimiento óptimo. Para la transmisión eficiente del movimiento, se utilizan engranajes o poleas

donde sea necesario. La reducción de la fricción en cada junta se logra mediante la inclusión de rodamientos. El ensamblaje de todas las piezas se realiza con tornillería adecuada, y el cableado interno se mantiene organizado y protegido mediante cubiertas protectoras. Todo el sistema es alimentado por una fuente de alimentación y gestionado por controladores de motor.

En lo que respecta a la selección de materiales, para un prototipo funcional y con miras a la accesibilidad, sugiero la utilización de PETG impreso en 3D para los eslabones. Este material ofrece ventajas como su economía, facilidad de fabricación y modificación, ligereza (lo que reduce significativamente el torque necesario en los motores), y un buen acabado estético. Aunque el PLA también es una opción económica y fácil de fabricar, el PETG es preferible debido a su mayor resistencia a la temperatura y flexibilidad sin fractura, lo que lo hace más robusto para piezas estructurales. Sin embargo, para una versión definitiva que demande mayor resistencia y durabilidad, consideraríamos la transición a un material más robusto como el aluminio para los eslabones. Para la base del robot y la punta se optó por considerar un material reforzado como el acero debido a su mayor concentración de energía mecánica

Motores:
Para dar vida al movimiento de nuestro brazo robótico, hemos diseñado una configuración que requiere un motor para cada uno de sus seis grados de libertad, que son los movimientos típicos de un brazo industrial. La elección de cada motor se pensó cuidadosamente para que tuviera la fuerza (torque) necesaria para mover la parte que le corresponde, sin gastar más energía de la cuenta.

- 1. El Motor de la Base (Giro Horizontal): Este motor es el que permite que todo el brazo gire de izquierda a derecha, como cuando giras sobre tu propio eje. Es el que soporta el peso de todo el conjunto del robot, así que necesita ser bastante fuerte. Por eso, lo ideal es usar un motor NEMA 23 o un servomotor con un buen torque, entre 10 y 20 kg·cm. Es como el "cuello" robusto del robot.

- 2. El Motor del Brazo Principal (Arriba y Abajo): Este es el "hombro" del robot. Tiene la tarea pesada de levantar y bajar una gran parte de la estructura del brazo. Por eso, necesitamos que sea potente. Aquí, otro NEMA 23 o un servomotor son la mejor opción,

pero con un torque aún mayor, de al menos 20 a 30 kg·cm.

- 3. El Motor del Brazo Secundario (Extensión/Retracción): Este motor controla el "codo" del robot, permitiendo que el brazo se estire o se encoja. Su fuerza necesaria dependerá de cuánto pese esa sección y la herramienta final. Un NEMA 17 o un servomotor con un torque de entre 10 y 20 kg·cm sería adecuado aquí.

- 4. El Motor de la Muñeca (Movimiento Vertical): Llegamos a la "muñeca" del robot. Este motor controla el movimiento hacia arriba y abajo de la muñeca. Aquí, la carga es menor, así que un NEMA 17 o un servomotor con un torque de 5 a 10 kg·cm es suficiente.

- 5. El Motor de la Muñeca (Giro Horizontal): Este motor permite que la muñeca gire sobre su propio eje, lo que es vital para orientar la herramienta o la pinza. Al igual que el anterior, un NEMA 17 o un servomotor con un torque de 5 a 10 kg·cm es ideal.

- 6. El Motor de la Herramienta Final (Pinza/Garra): Finalmente, este motor se encarga de las acciones de la "mano" del robot, como abrir y cerrar una pinza, o girar una herramienta. Si la herramienta es ligera, un servomotor más pequeño, como un MG996R o incluso un SG90, podría funcionar, con un torque de entre 2 y 5 kg·cm.

Eslabones

El brazo robótico está compuesto por varios eslabones, cada uno con características específicas de masa, inercia, longitud y material, que determinan el comportamiento dinámico del sistema.

El primer eslabón corresponde a la base y soporte de columna. Aunque su función es principalmente estructural y de soporte, puede considerarse con una masa aproximada de 0.6 kg si está fabricado en PETG y de 1.2 kg en aluminio. Su inercia es relativamente baja dado que su movimiento es un giro en plano horizontal y su centro de masa se encuentra cercano al eje de rotación. Su longitud efectiva puede considerarse de 10 cm, al ser una base compacta que permite sostener el brazo sin desbalance.

El segundo eslabón es el primer brazo o eslabón largo. Este tiene una longitud de aproximadamente 25 cm. Su masa, utilizando PETG, estaría en torno a 0.5 kg, y en aluminio alcanzaría cerca de 0.9 kg. Su inercia respecto al eje de rotación que lo une con la base es significativa, ya que se mueve verticalmente y soporta el peso del resto del brazo. Para este tipo de pieza, la inercia se calcula considerando su masa y la distancia de su centro de masa al eje, resultando en un valor aproximado de 0.015 kg·m² para PETG y 0.028 kg·m² para aluminio.

El tercer eslabón es el segundo brazo o eslabón corto, con una longitud estimada de 18 cm. Su masa en PETG rondaría los 0.35 kg, y en aluminio cerca de 0.7 kg. La inercia de este eslabón es menor que la del primero, ya que su longitud y masa son inferiores. Puede estimarse una inercia de 0.007 kg·m² para PETG y 0.014 kg·m² para aluminio.

El cuarto eslabón corresponde a la muñeca, que funciona como una unión entre el segundo brazo y la herramienta. Es una pieza más compacta, de aproximadamente 8 cm de longitud. Su masa en PETG sería de 0.25 kg, y en aluminio 0.5 kg. Debido a su cercanía al extremo y su menor masa, su inercia es reducida: alrededor de 0.003 kg·m² en PETG y 0.006 kg·m² en aluminio.

El quinto eslabón es la herramienta o actuador final, cuya masa depende del accesorio que se utilice. Si es una pinza impresa en PETG, podría tener una masa de 0.15 kg, mientras que una versión en aluminio rondaría 0.3 kg. Su longitud estimada es de 5 cm, y su inercia sería de aproximadamente 0.001 kg·m² en PETG y 0.002 kg·m² en aluminio.

Finalmente, es importante considerar que todas estas masas y momentos de inercia son estimaciones que dependen de la forma específica de cada pieza, la densidad del material y la distancia de su centro de masa al eje de rotación correspondiente.

Para las piezas que no cargan mucho peso ni sufren grandes esfuerzos, como las cubiertas, las carcasas de los motores o elementos de protección, hemos pensado en usar impresión 3D con materiales como PLA o PETG. Esto tiene varias ventajas:

- ¡Producir estas piezas con impresión 3D es bastante económico en comparación con otros métodos de fabricación.
- Fabricación y Modificación Sencilla: Si necesitamos hacer algún ajuste en el diseño o cambiar una pieza, No hay complicaciones.

- Súper Ligero: Estos materiales son muy livianos, lo cual es una gran ventaja. Al reducir el peso total del robot, los motores no tienen que esforzarse tanto, lo que significa que necesitan menos torque y consumen menos energía.
- Estéticamente agradable: Podemos obtener un acabado visual muy bueno, lo que le da un aspecto profesional y limpio al robot.

Ahora, hay que tener en cuenta algunas desventajas:

- No Son para Cargas Pesadas: No son los materiales más resistentes si el robot va a recibir impactos fuertes o cargar mucho peso. Por eso los usamos solo en las cubiertas y carcasas, no en la estructura principal.
- Sensibilidad al Calor: Especialmente el PLA, puede deformarse si se expone a temperaturas elevadas. Imagina dejarlo al sol por mucho tiempo.

3.1.2. Límites y propiedades dinámicas de las articulaciones

En esta sección, nos enfocamos en las características operativas de cada una de las articulaciones del robot, tal como se detalló en la Tabla 3.1. Estas propiedades fueron cruciales para entender el rango de movimiento y las capacidades de velocidad y aceleración de nuestro manipulador. Aunque la tarea de modelado dinámico completo no fue profundizada en este semestre, los límites que se presentaron fueron fundamentales para cualquier análisis futuro y para la planificación de trayectorias.

La Tabla 3.1, además de los parámetros de Denavit-Hartenberg que definieron la geometría del robot, especificó los siguientes límites para cada articulación:

- Mínimo (θ/d): Esta columna indica el valor angular o de desplazamiento más bajo que cada articulación podía alcanzar. Fue el límite inferior de su rango de operación, expresado en grados para articulaciones rotacionales. Por ejemplo, la articulación 1 podía

girar hasta -180 grados, mientras que la articulación 2 tuvo un límite inferior de -90 grados.

- Máximo (θ/d): Complementando el límite mínimo, esta columna definió el valor angular o de desplazamiento más alto que cada articulación podía alcanzar. Fue el límite superior de su rango de operación, también en grados para articulaciones rotacionales. Así, la articulación 1 pudo llegar hasta 180 grados, mientras que las articulaciones 3 y 4 estuvieron restringidas a un máximo de 0 grados.
- q˙max (grados/s): Esta fue la velocidad máxima de la articulación, indicando la rapidez con la que una articulación podía cambiar su posición. Para todas las articulaciones de nuestro robot, se estableció una velocidad angular máxima de 180 grados por segundo. Este valor fue clave para determinar la velocidad máxima a la que el efector final podía moverse.
- q¨max (grados/s^2): Esta columna representó la aceleración máxima de la articulación, es decir, la tasa máxima a la que la velocidad de una articulación podía cambiar. Para todas las articulaciones, la aceleración angular máxima fue de 360 grados por segundo al cuadrado. Este parámetro fue vital para la planificación de trayectorias, asegurando que el robot pudiera iniciar o detener sus movimientos de manera eficiente y controlada sin exceder las capacidades de sus actuadores.

Estos límites fueron intrínsecos al diseño y a la selección de los actuadores y mecanismos de cada articulación. Si bien no se completó un análisis dinámico profundo en este proyecto, comprender estos límites fue esencial. Por ejemplo, la capacidad de un motor para alcanzar la q¨max depende directamente de su torque máximo y de la inercia efectiva de la articulación en esa configuración, aspectos que serían clave en un estudio dinámico completo. La integración de estos valores en la simulación de ROS y en el cálculo de la cinemática en MATLAB permitió generar movimientos que respetaron las restricciones físicas del robot, asegurando un comportamiento realista en el entorno virtual.

3.2. Proceso de Cinemática

Para la implementación y simulación de nuestro brazo robótico, nuestro equipo desarrolló un *script* en MATLAB que abarcó desde la definición inicial del robot hasta la animación de su movimiento y la generación de gráficos de sus trayectorias. Este proceso lo estructuramos siguiendo los principios de la cinemática directa e inversa, así como la planificación de trayectorias.

Definición Inicial y Estructura del Robot

El primer paso consistió en establecer los parámetros fundamentales del robot y su configuración inicial.

Inicialmente, definimos la posición (x, y, z) y la orientación (phi, theta, psi) del efector final del robot en su estado inicial. Esto lo realizamos mediante variables que posteriormente agrupamos en un vector p_robot para la posición y utilizamos para calcular una matriz de rotación inicial (R_inicial) a partir de ángulos de Euler. Estos valores establecieron el punto de partida para todas las operaciones cinemáticas. La estructura completa del robot la creamos basándonos en la tabla de Denavit-Hartenberg (DH) que fue cargada desde un archivo CSV (robot-10 - robot-7.csv). Esta tabla es la base geométrica del robot y fue fundamental para los cálculos cinemáticos. La función crear_robot utilizó esta tabla DH y la matriz de transformación homogénea inicial (A0, construida a partir de R_inicial y p_robot) para generar el modelo matemático del robot en MATLAB.

Trayectoria Cartesiana y Cinemática Inversa

Una vez definida la estructura del robot, el siguiente paso fue la planificación del movimiento en el espacio de trabajo.

La trayectoria que el efector final debía seguir la cargamos desde un archivo CSV (cargar_trayectoria_1.csv). Este archivo contenía una secuencia de puntos deseados (posición y orientación) que el robot debía alcanzar. Las posiciones se almacenaron en pos_puntos. Para

cada punto de la trayectoria cartesiana, calculamos la configuración articular correspondiente mediante un algoritmo de cinemática inversa. Este algoritmo se basó en el descenso del gradiente (o Jacobiano), que a través de varias iteraciones y una velocidad alpha, buscó el valor objetivo reduciendo el error entre la posición deseada y la actual del efector final. Utilizamos parámetros como la tolerancia (tol), el número máximo de iteraciones (max_iter), el factor de corrección alpha y el número de muestras (numSamples) para manejar posibles mínimos locales, asegurando que se exploraran múltiples candidatos para encontrar una solución óptima. La solución para cada punto de la trayectoria se almacenó en q_sol.

Planificación de Trayectorias Articulares y Cinemática Directa

Con las configuraciones articulares obtenidas de la cinemática inversa, procedimos a planificar el movimiento suave de las articulaciones y a verificar las trayectorias resultantes en el espacio cartesiano.

Planificamos una trayectoria entre los puntos articulares calculados por la cinemática inversa. Determinamos el tiempo final (t_final) necesario para el movimiento, utilizando el valor más alto de los tiempos mínimos requeridos para cada articulación, garantizando un perfil de velocidad triangular. La función trapveltraj la utilizamos para generar las trayectorias de posición (q), velocidad ($\dot{q}$) y aceleración ($\ddot{q}$) de las articulaciones de forma suave. Finalmente, aplicamos el cálculo de cinemática directa. Utilizando las trayectorias de posición, velocidad y aceleración articulares generadas, calculamos la posición, velocidad y aceleración lineal del efector final, así como su orientación, velocidad y aceleración angular. Esto permitió verificar que la trayectoria planificada en el espacio articular resultara en el movimiento deseado en el espacio cartesiano.

Animación y Visualización de Resultados

Para visualizar el comportamiento del robot y analizar sus movimientos, implementamos etapas de animación y generación de gráficos.

Configuramos la animación del robot para ser reproducida a 30 cuadros por segundo (fps). Creamos una estructura gráfica del robot, y generamos un archivo de video (.avi) con la fecha y hora actuales para registrar la simulación del movimiento. Dentro de un bucle, actualizamos la

posición del robot en cada instante de tiempo de la animación y lo dibujamos. También incluimos la opción de capturar cada fotograma y escribirlo en el video. Para un análisis más profundo, generamos gráficas que mostraban la posición cartesiana (X, Y, Z), la velocidad lineal, la aceleración lineal, la orientación (ángulos de Euler ϕ,θ,ψ), la velocidad angular y la aceleración angular del efector final a lo largo del lazo de tiempo. Estas gráficas fueron esenciales para validar el rendimiento del robot y asegurar que los movimientos cumplían con las especificaciones.

Este proceso integral en MATLAB nos permitió simular y analizar el comportamiento cinemático del brazo robótico, validando el diseño y la implementación de las funciones de cinemática directa e inversa.

3.2.1. Cinemática Directa

En esta sección, explicaremos el proceso que seguimos para implementar la cinemática directa de nuestro brazo robótico en MATLAB. La cinemática directa es crucial porque nos permite predecir la posición y orientación del efector final del robot, dadas las configuraciones de sus articulaciones. Esencialmente, es como si el robot nos dijera "dado cómo están mis brazos, mi mano está en este punto".

Nuestro código en MATLAB se estructuró para simular este comportamiento de forma precisa.

Valores Iniciales y Configuración del Robot

Comenzamos definiendo un punto de partida para nuestro robot. Esto implicó establecer una posición inicial (X, Y, Z) en el espacio cartesiano y una orientación inicial (Phi, Theta, Psi, que corresponden a rotaciones en los ejes X, Y y Z respectivamente). Estas variables, como se muestra en nuestro código, se agruparon en un vector posicion_inicial y se utilizaron para calcular una matriz de rotación inicial, R_inicial. Esta R_inicial y posicion_inicial conformaron la matriz de transformación homogénea inicial (A0), que representa la pose del sistema de coordenadas base del robot en el mundo.

Luego, procedimos a construir la estructura matemática de nuestro robot. Para ello, cargamos los parámetros de Denavit-Hartenberg (DH) desde un archivo CSV (robot-10 - robot-7.csv). Esta tabla DH es como el "ADN" geométrico de nuestro robot, ya que define la longitud de cada eslabón y cómo se conectan las articulaciones. Con la tabla DH y nuestra matriz de transformación inicial A0, utilizamos una función crear_robot para generar el modelo matemático completo del robot en MATLAB, lo que nos permitió realizar cálculos cinemáticos con él.

Trayectorias de las Articulaciones y Cálculo de Cinemática Directa

Una vez que teníamos nuestro robot definido, el siguiente paso fue determinar cómo se moverían sus articulaciones a lo largo del tiempo. Para esto, generamos las trayectorias de las articulaciones.

Definimos un periodo para el movimiento cíclico del robot, que en nuestro caso fue de 2 segundos. Luego, utilizamos la función trayectoria_q junto con el modelo del robot y el intervalo de tiempo definido (t_inicial a t_final con un t_paso de 0.1 segundos) para calcular las posiciones (q), velocidades (dq) y aceleraciones (ddq) de cada articulación a lo largo del tiempo. Esto nos proporcionó un perfil de movimiento suave y coordinado para todas las articulaciones.

Finalmente, el corazón de esta sección fue la aplicación de la cinemática directa. Utilizamos la función cinematica_dir con las trayectorias de posición, velocidad y aceleración articulares que habíamos calculado (q, dq, ddq) y la secuencia de orientación (XYZ). Esta función nos devolvió la posicion, vel_linear, acel_linear, orientacion, vel_angular y acel_angular del efector final en el espacio cartesiano a lo largo de toda la trayectoria. Esto nos permitió ver exactamente dónde y cómo se movía la "mano" del robot a medida que sus articulaciones seguían las trayectorias planificadas.

Animación y Visualización de Resultados

Para poder visualizar el movimiento del robot y analizar los resultados de la cinemática, implementamos funcionalidades de animación y generación de gráficos.

Configuramos la animación para que se reprodujera a 30 cuadros por segundo (fps). Creamos una representación gráfica del robot y, si hubiéramos querido, podríamos haber generado un archivo de video (.avi) para guardar la simulación. En un bucle, actualizábamos la posición de las articulaciones del robot en cada instante de tiempo y lo dibujábamos, creando la ilusión de movimiento.

Para un análisis más cuantitativo, generamos gráficas que mostraron la evolución de la posición cartesiana (X, Y, Z), la velocidad lineal (Vx, Vy, Vz), la aceleración lineal (Ax, Ay, Az), la orientación (ángulos de Euler Phi, Theta, Psi), la velocidad angular y la aceleración angular del efector final a lo largo del tiempo. Estas gráficas fueron esenciales para validar que el robot se movía como esperábamos y que las trayectorias generadas eran correctas.

Los resultados visuales y gráficos de este proceso se pueden apreciar en el Capítulo 4 de nuestro documento, donde se muestran las animaciones y las trayectorias resultantes.

3.2.2. Cinemática Diferencial

En esta sección, nuestro equipo abordó la implementación de la cinemática diferencial en MATLAB. Esta parte del código es fundamental para entender cómo las velocidades y aceleraciones de las articulaciones de un robot se relacionan con las velocidades y aceleraciones de su efector final. En esencia, nos permite comprender el movimiento instantáneo del robot en el espacio de trabajo.

Inicialización de Variables

Para asegurar un rendimiento eficiente de nuestro *script*, realizamos una preasignación de variables. Esto significa que antes de entrar en los bucles de cálculo, reservamos el espacio en memoria necesario para todas las matrices y vectores que íbamos a utilizar (por ejemplo, posicion, orientacion, vel_linear, vel_angular, acel_linear, acel_angular, y las matrices Jacobianas Jv, Jw y sus derivadas dJv, dJw). Al hacer esto, evitamos que MATLAB tuviera que redimensionar estas variables en cada iteración, lo que hubiera ralentizado significativamente la

ejecución del código, especialmente en simulaciones largas. Definimos un paso de tiempo dt a partir de t_paso para los cálculos discretos.

Cálculo de Velocidad y Aceleración

El corazón de la cinemática diferencial reside en el cálculo de las velocidades y aceleraciones. Realizamos estos cálculos de forma discreta, es decir, no utilizamos una función continua del tiempo f(t), sino que nos referimos a los índices k del vector de tiempo que corresponden al tiempo actual, al tiempo anterior (k-1) o al tiempo siguiente (k+1).

Dentro de un bucle que recorrió cada instante de tiempo (for k = 1:length(t)), nuestro equipo ejecutó los siguientes pasos:

1. Actualización de la configuración del robot: Primero, actualizamos la pose del robot con los valores articulares q(::,k) correspondientes al instante actual. Esto se realizó mediante la función actualizar_robot.
2. Extracción de la posición y orientación del efector final: Una vez que el robot estuvo en la configuración correcta, extrajimos la matriz de transformación homogénea (robot.T) y de ahí obtuvimos la posicion y la matriz de rotación R del efector final para el instante k.
3. Obtención de la orientación en ángulos de Euler: Convertimos la matriz de rotación R a ángulos de Euler (orientacion(::,k)) utilizando la función rotMat2euler y la secuencia de ángulos especificada (secuencia).
4. Cálculo del Jacobiano geométrico: Este fue un paso fundamental. Calculamos el Jacobiano geométrico del robot (Jv, Jw) utilizando la función jac_geometrico(robot). El Jacobiano nos permitió relacionar las velocidades articulares con las velocidades lineal y angular del efector final.
5. Cálculo de la velocidad lineal y angular: Aplicando la fórmula $v=Jq^{\cdot}$, calculamos las velocidades lineal (vel_linear(::,k)) y angular (vel_angular(::,k)) del efector final. Esto se hizo multiplicando las submatrices del Jacobiano (Jv y Jw) por las velocidades articulares (dq(::,k)).
6. Cálculo de la derivada temporal del Jacobiano: Para poder calcular las aceleraciones, necesitamos la derivada del Jacobiano ($J^{\cdot}$). La calculamos usando diferencias finitas. Si k era mayor que 1 (es decir, a partir del segundo instante de tiempo), calculamos dJv y dJw

como la diferencia entre el Jacobiano actual y el anterior, dividido por el paso de tiempo dt.

7. Cálculo de la aceleración lineal y angular: Finalmente, calculamos las aceleraciones lineal (acel_linear(::,k)) y angular (acel_angular(::,k)) del efector final utilizando la fórmula $a=\frac{d}{dt}(Jq^{\cdot})=Jq^{\cdot\cdot}+J^{\cdot}q^{\cdot}$. Esto implicó sumar el producto del Jacobiano por la aceleración articular (ddq(::,k)) con el producto de la derivada del Jacobiano por la velocidad articular.

Animación y Visualización de Resultados

Para visualizar los resultados de la cinemática diferencial y analizar el comportamiento del robot, implementamos secciones para la animación y la generación de gráficos.

Configuramos la animación para ser reproducida a 30 fotogramas por segundo (fps). Generamos las trayectorias de las articulaciones para la animación (q_anim) utilizando la función trayectoria_q. Dentro de un bucle, en cada instante de tiempo de la animación, actualizamos la configuración del robot y lo dibujamos en la figura. También incluimos el código para capturar cada fotograma y guardarlo en un archivo de video, aunque esta función estuvo comentada por defecto.

Para un análisis más detallado, creamos una serie de gráficas organizadas en subplots para visualizar las diferentes componentes del movimiento del efector final. Mostramos la posición cartesiana (X, Y, Z), la velocidad lineal (Vx, Vy, Vz) y la aceleración lineal (Ax, Ay, Az). Adicionalmente, graficamos la orientación (ϕ,θ,ψ), la velocidad angular ($\dot{\phi},\dot{\theta},\dot{\psi}$) y la aceleración angular ($\ddot{\phi},\ddot{\theta},\ddot{\psi}$) del efector final. Estas gráficas nos permitieron validar visualmente y cuantitativamente la coherencia de los cálculos de cinemática diferencial con las trayectorias articulares de entrada.

3.2.3. Cinemática Inversa

Nuestro equipo implementó la cinemática inversa del brazo robótico en MATLAB, un proceso crucial que nos permitió determinar las configuraciones articulares necesarias para que el efector final del robot alcanzara una posición y orientación deseadas en el espacio. Esencialmente, esta

parte del código responde a la pregunta: "Si quiero que mi mano esté aquí, ¿cómo deben estar mis brazos y codos?".

Abordaje del Problema y Metodología

La cinemática inversa puede ser compleja debido a la posibilidad de múltiples soluciones o de ninguna solución alcanzable. Nuestro enfoque se basó en un método iterativo de descenso del gradiente o Jacobiano. Este método busca gradualmente reducir el error entre la posición actual del efector final y la posición deseada, ajustando las configuraciones articulares en cada paso.

Como se muestra en nuestras imágenes de código, la cinemática inversa se basó en el descenso del gradiente o Jacobiano, donde a través de varias iteraciones y una velocidad α, se obtiene un valor objetivo reduciendo el error.

Reconocimos que, en caso de la presencia de límites o de funciones de error con múltiples mínimos, el descenso del gradiente podría quedarse "atrapado" en un mínimo local, sin llegar a la solución óptima. Para mitigar esto, implementamos una estrategia que consistió en usar varios candidatos que se distribuyen uniformemente en todo el rango de las articulaciones. Esto aumentó la probabilidad de que al menos uno de ellos llegara al objetivo deseado.

Parámetros Clave del Algoritmo

Para controlar la precisión y la eficiencia del algoritmo, definimos varios parámetros importantes:

- tol = 1e-6: Esta fue nuestra tolerancia o el error máximo que estábamos dispuestos a aceptar. Cuando el error entre la posición deseada y la actual del efector final caía por debajo de este valor, considerábamos que habíamos encontrado una solución.
- max_iter = 100: Establecimos un número máximo de iteraciones de 100. Esto nos sirvió como una medida de seguridad para evitar que el algoritmo corriera indefinidamente si no encontraba una solución o si el error no convergía lo suficientemente rápido.
- alpha = 1.0: Este parámetro representó el factor de corrección del descenso del gradiente. Controló el tamaño de los pasos que el algoritmo tomaba en cada iteración para ajustar

las articulaciones. Un alpha adecuado es crucial para la convergencia: si es muy pequeño, la convergencia es lenta; si es muy grande, puede haber oscilaciones o divergencia.

- numSamples = 9: Este fue el número de muestras que utilizamos para buscar diferentes puntos en caso de llegar a un mínimo local. Esta técnica nos permitió explorar distintas configuraciones articulares iniciales para aumentar las posibilidades de encontrar una solución válida, incluso si el algoritmo se estancaba en un mínimo local con una configuración inicial.

Proceso de Cálculo

Nuestro *script* cargó una trayectoria cartesiana deseada (cargar_trayectoria_1.csv), que contenía una serie de puntos (posición y orientación) que el efector final del robot debía alcanzar. Para cada uno de estos puntos de la trayectoria, aplicamos la función de cinemática inversa (cinematica_inv). Esta función recibió como entradas el modelo del robot, el punto deseado de la trayectoria (pos_puntos(::,i)), la tolerancia, el número máximo de iteraciones, el factor alpha y el número de muestras. La salida de esta función fue la configuración articular (q_sol(::,i)) que correspondía a cada punto deseado.

La q_sol fue un arreglo donde almacenamos la solución articular (los ángulos de cada articulación) para cada punto de la trayectoria cartesiana que el robot debía seguir. Esta información fue vital para la siguiente etapa, la planificación de trayectorias articulares, donde se generaron perfiles suaves de movimiento para estas articulaciones.

3.3. Control

Nuestro equipo reconoce que el control de un robot es un campo complejo y fundamental para su operación autónoma y precisa. Aunque este semestre no profundizamos en la implementación de algoritmos de control avanzados, es crucial entender el propósito de esta etapa en el sistema robótico que hemos estado desarrollando.

En términos generales, el objetivo del control es asegurar que el robot siga fielmente la trayectoria deseada, incluso ante la presencia de perturbaciones o incertidumbres. Para lograr esto, un sistema de control de robot típicamente requiere las siguientes entradas:

- Valores de posición y velocidad deseados de las articulaciones ($q_d, \dot{q}_d$): Estos son los "objetivos" que el robot debe alcanzar. Provienen directamente de la etapa de cinemática inversa y planificación de trayectoria. Indican la posición y la velocidad que cada articulación *debería* tener en cada instante de tiempo para que el efector final siga su trayectoria planificada.
- Valores de posición y velocidad reales de las articulaciones ($q, \dot{q}$): Estos son los "datos de realimentación" que el robot obtiene de sus sensores (por ejemplo, encoders en los motores). Representan la posición y la velocidad *actuales* de cada articulación. Son fundamentales para que el sistema de control sepa dónde está el robot en todo momento.
- El modelo dinámico del robot: Aunque no lo implementamos en detalle, un controlador avanzado necesitaría conocer la dinámica del robot. Esto incluye la masa de los eslabones, su inercia, las fuerzas de Coriolis, la gravedad y la fricción. Con este modelo, el controlador puede predecir cómo se moverá el robot en respuesta a los torques aplicados y cómo compensar fuerzas como la gravedad o la fricción.

Con estas entradas, un controlador avanzado (como un controlador de torque basado en modelo o un control adaptativo) calcularía los torques o fuerzas (u o τ) que los actuadores de cada articulación deben aplicar. El objetivo es minimizar el "error" entre la trayectoria deseada y la trayectoria real, haciendo que el robot se comporte de manera robusta y precisa.

A pesar de no haber profundizado en el diseño e implementación de un controlador dinámico en este semestre, la comprensión de estas entradas y la importancia del bucle de realimentación (como el que se muestra en el Diagrama de Control de un Robot Industrial) son pasos fundamentales para futuras etapas de desarrollo de nuestro robot. Esperamos en proyectos venideros poder sumergirnos en la fascinante tarea de diseñar y probar algoritmos de control que permitan a nuestro robot ejecutar tareas complejas con mayor autonomía y eficiencia.

3.4. Simulación

Proceso de Simulación en ROS

Para llevar a cabo la simulación de nuestro brazo robótico en ROS , en el equipo seguimos una serie de pasos esenciales que abarcan desde la obtención del modelo, su exportación, la configuración del entorno de desarrollo hasta la simulación y visualización del movimiento. Para una revisión más detallada del código y los archivos de configuración, se puede consultar nuestro repositorio en GitHub.

Obtención y Exportación del Modelo del Robot

El modelo base de nuestro brazo robótico lo obtuvimos a través de GrabCAD, una plataforma en línea que nos proporcionó los archivos CAD 3D iniciales. Una vez que tuvimos el modelo en SolidWorks, el siguiente paso crucial fue exportarlo a un formato compatible con ROS. Utilizamos las herramientas de SolidWorks para generar un archivo URDF. Este archivo es fundamental en ROS, ya que describe la cinemática, la dinámica y la apariencia visual del robot (eslabones, articulaciones, masas, etc.).

Configuración del Entorno Ubuntu

Para trabajar con ROS, que se ejecuta de forma nativa en Linux, optamos por utilizar Ubuntu. Nuestro equipo exploró dos enfoques para esto: algunos miembros utilizamos máquinas virtuales (como VirtualBox) para instalar Ubuntu, mientras que otros aprovechamos el Windows Subystem for Linux (WSL) en Windows. Ambas opciones nos permitieron tener un entorno Ubuntu funcional donde pudimos instalar ROS y sus paquetes necesarios.

Simulación de las Pinzas y Actuador Final

En cuanto a la simulación de nuestro efector final, que incluía pinzas, nuestro objetivo era que estas se movieran de forma coordinada. Esto lo logramos configurando sus controladores en el URDF para que respondieran a un solo comando de forma síncrona, simulando un movimiento de agarre. Aunque exploramos la posibilidad de integrar un electroimán para tareas específicas de manipulación, en esta etapa nos centramos en el control del movimiento de las pinzas como tal.

Modificación de Controladores y Ejecución en ROS

Un paso clave fue la modificación de los controladores en los archivos de configuración de ROS. Ajustamos los parámetros de los controladores de las articulaciones (por ejemplo, ganancias PID) para asegurar que el robot respondiera de manera estable y siguiera las trayectorias deseadas.

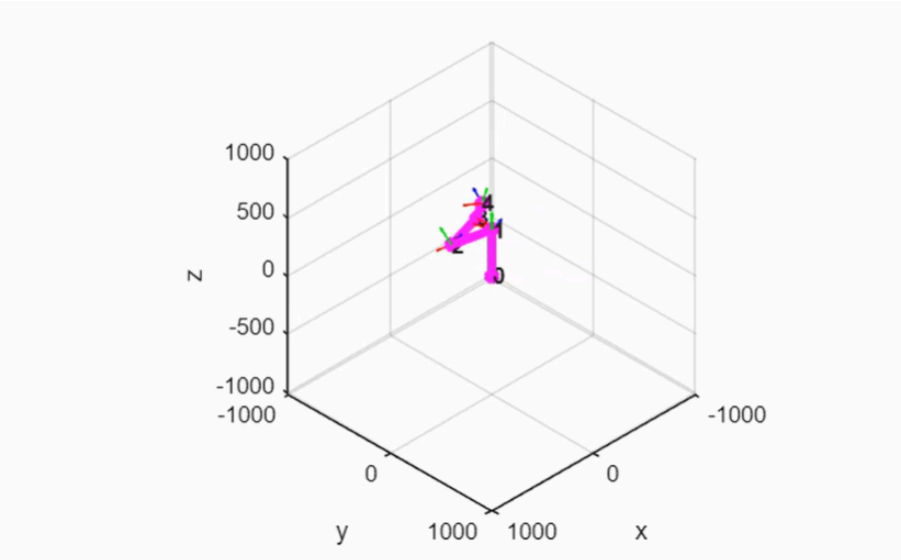
Para ejecutar la simulación y visualizar el comportamiento del robot, utilizamos varias herramientas de ROS:

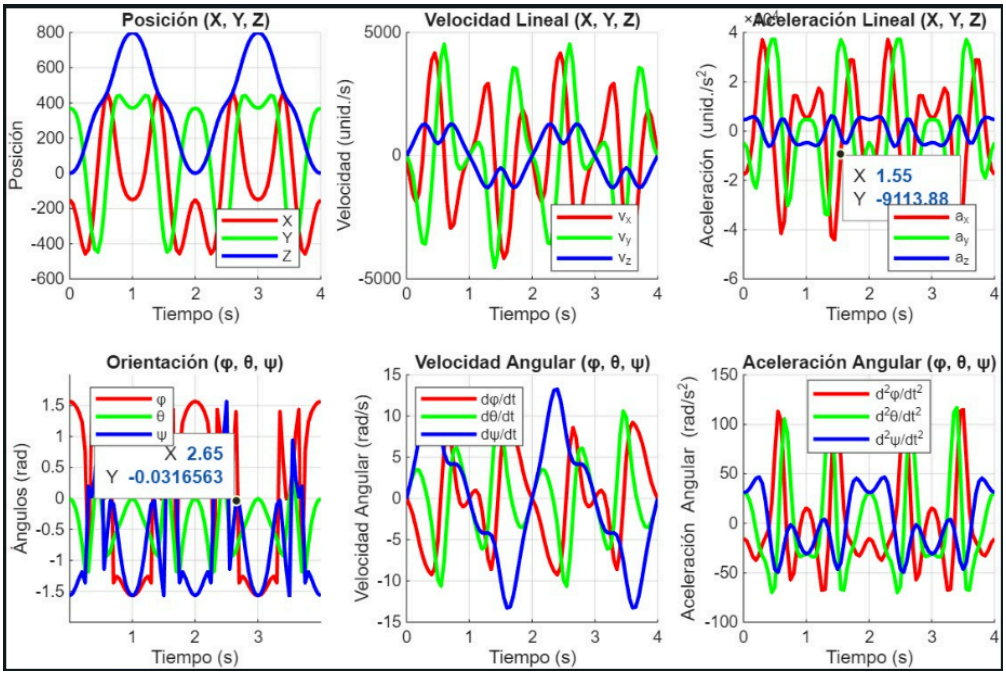
- RViz: Esta herramienta nos permitió visualizar el modelo 3D del robot y sus movimientos en tiempo real. Fue indispensable para verificar que el robot se movía según lo esperado y para depurar posibles errores en la cinemática o en los controladores.
- Gazebo: Para una simulación más realista, empleamos Gazebo. Este simulador de física nos permitió modelar las interacciones del robot con su entorno, incluyendo gravedad y colisiones, ofreciendo una representación más fidedigna del comportamiento del brazo.
- MoveIt!: Integrar MoveIt! nos proporcionó capacidades avanzadas de planificación de movimiento. Con MoveIt!, pudimos planificar trayectorias complejas para el brazo robótico, evitando obstáculos y calculando las configuraciones articulares óptimas para alcanzar puntos deseados en el espacio de trabajo.

Mediante estos pasos y el uso de estas herramientas, nuestro equipo logró simular de manera efectiva el comportamiento del brazo robótico en un entorno virtual, lo que nos permitió validar su diseño y sus capacidades de movimiento antes de cualquier implementación física.

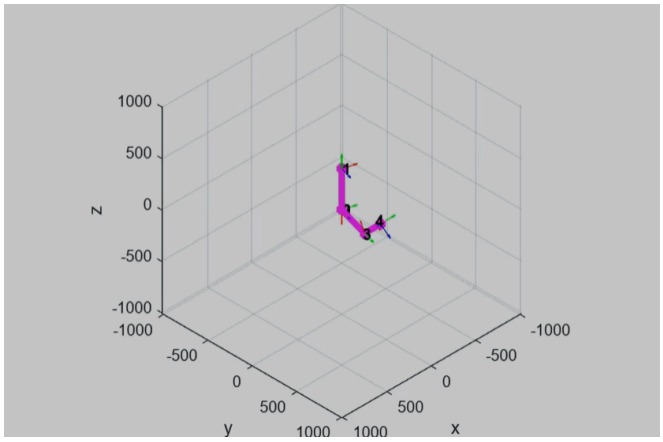
4. Resultados

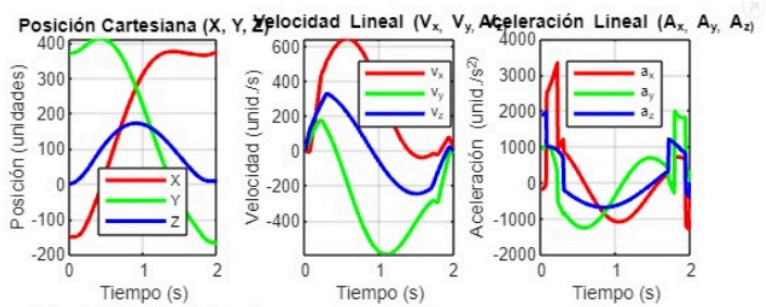
Cinemática directa:

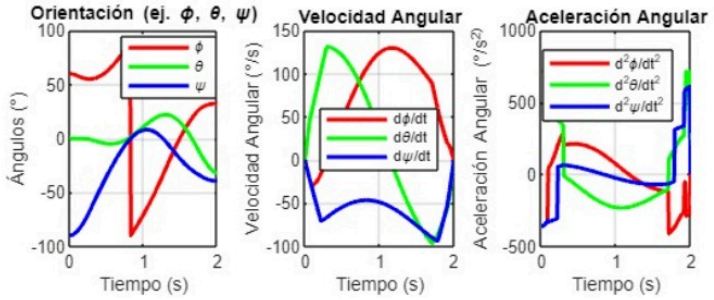


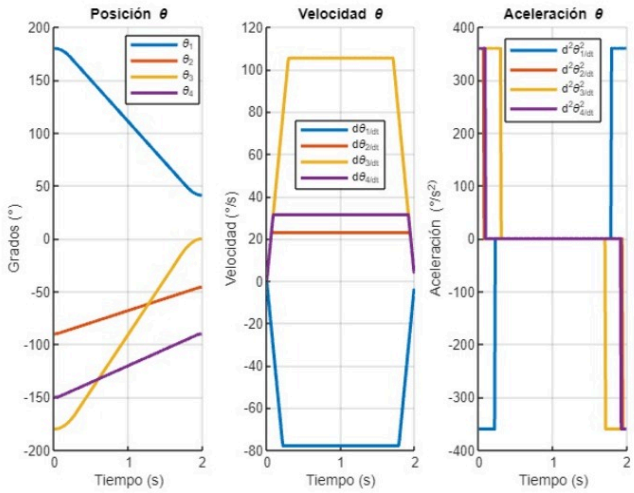


Cinemática Inversa





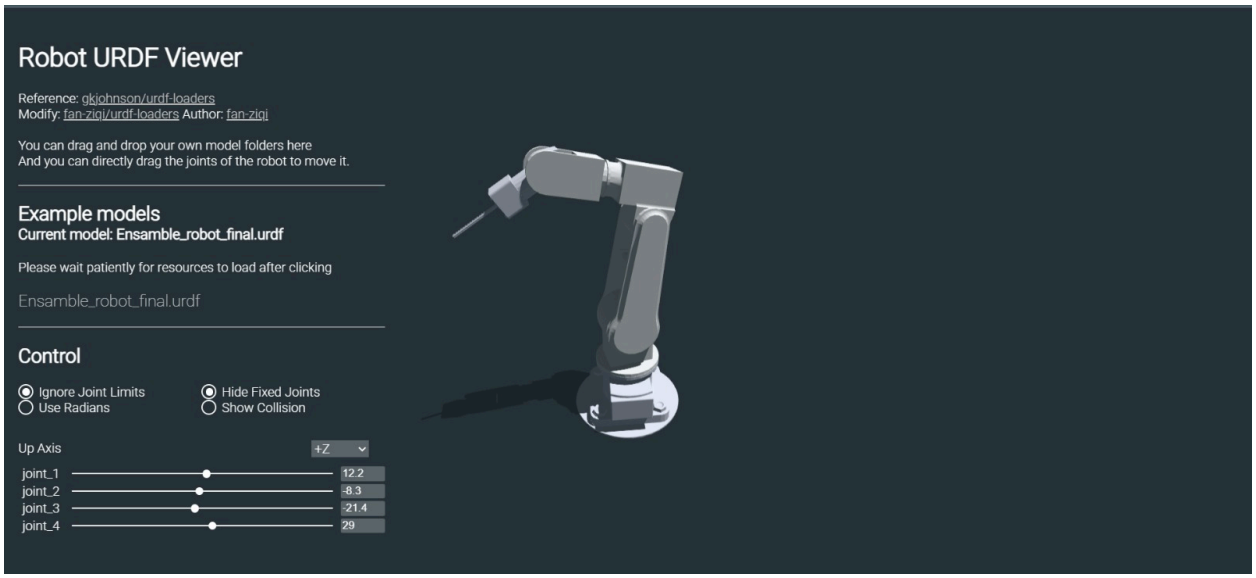




Simulación

En nuestro caso no pudimos realizar la simulación con GAZEBO ya que al momento de querer ejecutar la simulación nos percatamos que unos archivos no estaban en sus carpetas correspondientes y eso causaba el atraso en la simulación.

A cambio les mostramos lo que es nuestro robot en "Robot URDF Viewer" y esto nos ayuda a corroborar que nuestros cálculos, piezas y robot es **100% funcional**



5. Conclusiones

5.1. Daniel Madrid Barcelo

Para mí, diseñar y simular este robot en ROS fue un reto grande pero muy valioso. Al principio, juntar el modelo mecánico del robot con los números de cómo se mueve (parámetros cinemáticos) y hacerlo funcionar en un simulador como ROS, fue complicado. Tuvimos que asegurarnos de que todo "encajara" bien. Un problema que se repitió fue arreglar los archivos de descripción del robot (URDF) y entender bien los números de Denavit-Hartenberg en el simulador, porque si no, el robot se movía de formas raras o la simulación fallaba. Pero esta parte práctica me enseñó lo importante que es probar todo en una computadora antes de construir el robot de verdad. Aprendí que, para ahorrar tiempo y dinero en proyectos de robótica, no basta con un buen diseño inicial, sino que hay que simular y arreglar los errores con mucho cuidado.

5.2. Josue Camou

Hacer este robot nos permitió poner en práctica los conceptos básicos de cómo se mueve un robot, tanto "hacia adelante" (cinemática directa) como "hacia atrás" (cinemática inversa), empezando por crear su tabla de Denavit-Hartenberg. Tuvimos problemas para que las soluciones de la cinemática inversa fueran siempre buenas, porque a veces nos daban varias opciones o se quedaban "atoradas" en un lugar que no nos servía para un movimiento suave. Arreglar los fallos en el código de MATLAB, sobre todo en la parte del Jacobiano, nos tomó mucho trabajo y ajustes. Pero cuando logramos simular con éxito su movimiento en ROS, fue algo muy gratificante. Pudimos ver cómo nuestras fórmulas se convertían en movimientos reales en la pantalla. Este proyecto nos hizo entender mucho mejor los sistemas robóticos y cómo controlarlos, mostrando que las matemáticas son tan importantes como el programa de la computadora.

5.3. Adrian Rosas Leyva

Este proyecto fue clave para mí para juntar lo que habíamos aprendido en teoría y en la práctica sobre robótica. Diseñar el brazo robótico y definir sus números DH fue uno de los primeros retos; entender cómo cada número cambiaba la forma real del robot nos llevó varias pruebas y ajustes. Después, simular el movimiento en ROS tuvo sus propias dificultades, como configurar los programas y hacer que MATLAB y ROS "hablaran" bien para enviar las órdenes de movimiento. A veces, errores en los datos o que las cosas no estaban a tiempo nos hacían ver movimientos extraños en la simulación. Pero al superar estos problemas, entendimos muchísimo mejor cómo se conectan y trabajan juntos las piezas físicas y los programas de una máquina automática. Aprendimos de primera mano lo importante que es diseñar las partes por separado y que todo el *software* se comunique bien para que el robot se mueva como debe ser.

Bibliografía

[1] ROS 2 Documentation Team. (2025) Understanding ros2 topics. ROS 2 Documentation. [Online; accessed 21-May-2025]. https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Topics/Understanding-ROS2-Topics.html

[2] J. I. Medina Gil Lamadrid. (2025) Robotica: Código fuente del proyecto de robótica. GitHub. [código en línea; consultado 21-May-2025]. [Online]. Available: https://github.com/IvanMedinaGL/Robotica.git

Beer, F. P., Johnston, E. R., Mazurek, D. F., & Cornell, P. P. J. (2019). *Mecánica vectorial para ingenieros: Dinámica* (12a ed.). McGraw-Hill.

Craig, J. J. (2018). *Introduction to Robotics: Mechanics and Control* (4a ed.). Pearson Education.

Equipo 3. (s.f.). *Diagrama de control de un robot industrial* [Diagrama de bloques]. Material proporcionado para el análisis del brazo robótico.

Equipo 3. (s.f.). *Tabla de Parámetros de Denavit-Hartenberg y Características de las Articulaciones del Robot* [Tabla de datos]. Material proporcionado para el análisis del brazo robótico.

Franklin, G. F., Powell, J. D., & Emami-Naeini, A. (2015). *Feedback Control of Dynamic Systems* (7a ed.). Pearson.

Ogata, K. (2010). *Ingeniería de control moderna* (5a ed.). Pearson Education.

Siciliano, B., Sciavicco, L., Villani, L., & Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*. Springer.

Spong, M. W., Hutchinson, S., & Vidyasagar, M. (2006). *Robot Modeling and Control*. John Wiley & Sons.