



UNIVERSIDAD TÉCNICA DE AMBATO

**FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E
INDUSTRIAL**

**CARRERA DE INGENIERÍA INDUSTRIAL EN PROCESOS DE
AUTOMATIZACIÓN**

Tema:

**DISEÑO E IMPLEMENTACIÓN DE ALGORITMO PARA CONTROL DE
TRAYECTORIA CON COMPENSACIÓN GRAVITATORIA PARA EL
ROBOT KUKA YUBOT**

Trabajo de Titulación Modalidad: Proyecto de Investigación, presentado previo a la
obtención del título de Ingeniero Industrial en Procesos de Automatización

ÁREA: Industrial y Manufactura

LÍNEA DE INVESTIGACIÓN: Tecnología de la información y Sistemas de
control

AUTOR: Christian Andrés Rosero Sánchez

TUTOR: Dr. Marcelo Vladimir Sánchez, Mg

Ambato - Ecuador

agosto – 2024

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Titulación con el tema: DISEÑO E IMPLEMENTACIÓN DE ALGORITMO PARA CONTROL DE TRAYECTORIA CON COMPENSACIÓN GRAVITATORIA PARA EL ROBOT KUKA YOUNBOT, desarrollado bajo la modalidad proyecto de investigación por el señor Christian Andrés Rosero Sánchez, estudiante de la Carrera de Ingeniería Industrial en Proceso de Automatización, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, agosto 2024.

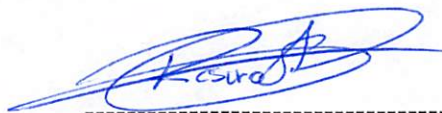


Dr. Marcelo Vladimir Sánchez, Mg
TUTOR

AUTORÍA

El presente Proyecto de Investigación titulado: DISEÑO E IMPLEMENTACIÓN DE ALGORITMO PARA CONTROL DE TRAYECTORIA CON COMPENSACIÓN GRAVITATORIA PARA EL ROBOT KUKA YOUNBOT es absolutamente original, auténtico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, agosto 2024.



Christian Andrés Rosero Sánchez

C.C. 1804046702

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato, para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, agosto 2024.



Christian Andrés Rosero Sánchez

C.C. 1804046702

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Christian Andrés Rosero Sánchez, estudiante de la Carrera de Ingeniería Industrial en Procesos de automatización, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado DISEÑO E IMPLEMENTACIÓN DE ALGORITMO PARA CONTROL DE TRAYECTORIA CON COMPENSACIÓN GRAVITATORIA PARA EL ROBOT KUKA YOUNBOT, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 17 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidente del Tribunal.

Ambato, agosto 2024.



Firmado electrónicamente por:
ELSA PILAR URRUTIA

Ing. Pilar Urrutia, Mg.
PRESIDENTE DEL TRIBUNAL



Firmado electrónicamente por:
SANTIAGO MAURICIO
ALTAMIRANO MELENDEZ

Ing. Mg. Santiago Altamirano Meléndez
PROFESOR CALIFICADOR



Firmado electrónicamente por:
JESUS ISRAEL GUAMAN
MOLINA

Ing. Mg. Jesús Guamán Molina
PROFESOR CALIFICADOR

DEDICATORIA:

*A mis **PADRES**, quienes con amor, dedicación y sacrificio han dado lo mejor de ellos mismo, para sacar adelante a sus 3 hijos, al incansable apoyo y el ejemplo de perseverancia, constancia y dedicación, siendo la motivación en medio de mis tormentas, para seguir adelante y culminar mis estudios. Mi única forma de agradecimiento es no defraudarlos y seguir adelante.*

*A mi querida **ABUELA** Martha, hasta que nos volvamos a ver.*

*A toda mi **FAMILIA***

Christian Andrés Rosero Sánchez

AGRADECIMIENTO:

*Principalmente a **Dios**, por darme la oportunidad de servirle y ayudar a otros a través de Él, por darme la vida y la familia que tengo.*

*A mis **Padres**, por su sacrificio, esfuerzo y sus palabras de ánimo a pesar de la distancia.*

*A mis **Hermanos**, Josué y Victoria por estar presentes.*

*A mis abuelos **Rodrigo y Joaquín**.*

*A mi abuela **Julia** por su apoyo.*

*A **Patricia y Fredy** por el apoyo brindado.*

*Al **Dr. Marcelo García**, por no rendirse conmigo.*

Jer. 29:11

ÍNDICE DE CONTENIDOS

APROBACIÓN DEL TUTOR	II
AUTORÍA	III
DERECHOS DE AUTOR	IV
APROBACIÓN DEL TRIBUNAL DE GRADO	V
DEDICATORIA:.....	VI
AGRADECIMIENTO:	VII
RESUMEN EJECUTIVO	XIII
ABSTRACT	XIV
CAPÍTULO I	1
MARCO TEÓRICO	1
1.1. TEMA DE INVESTIGACIÓN	1
1.2. PLANTEAMIENTO DEL PROBLEMA.....	1
1.3 ANTECEDENTES INVESTIGATIVOS	4
1.4 FUNDAMENTACIÓN TEÓRICA	6
1.4.1. Modo de control.....	7
1.4.2. Descripción general de los sistemas de control	8
1.4.3. Modelado y compensación de fricción	9
1.4.4. Modelos de control aplicados a la Robótica	11
1.4.5. Robot Operating System (ROS)	13
1.4.6. Automatización de bajo costo.....	14
1.4.7. Cinemática directa de robots.....	15
1.4.8. Robot manipulador KUKA youBot	17
1.4.9. Automatización de Bajo Costo	19
1.3. OBJETIVOS.....	21
1.3.1 Objetivo General.....	21
1.3.2 Objetivo Específico.....	21
CAPÍTULO II.....	22
METODOLOGÍA.....	22
2.1 MATERIALES.....	22
2.2. METODOLOGÍA	24
2.2.1. Esquemas de Control	24
2.2.2. Cálculo de Torque de articulaciones de Robot KUKA youBot	26
2.2.3. Oros KDL (Kinematics and Dynamics Library).....	27
2.2.4. Controlador basado en modelos para un sistema no lineal	29
2.2.5 Control Compensación Gravitatoria	32
2.2.6 Enfoque Cinemático	33
2.2.7. Modelo Dinámico	36
2.2.8. Modelado y Compensación de Fricción en Control Robótico.....	39

2.4 ESTRATEGIA DE CONTROL DE TRAYECTORIA	42
2.4.1 Espacio de Articulaciones.....	43
2.5 ESTRUCTURA DE SOFTWARE.....	45
2.5.1 Estructura de Archivos.....	48
2.6. DESARROLLO DE LA PROPUESTA	48
2.6.1 Creación del Entorno de Comunicación ROS	48
2.6.2 Algoritmo de Planeación de Trayectoria.	52
2.6.3 Algoritmo de Control PD con Compensación de Gravedad.....	54
2.7 CASO DE ESTUDIO.....	59
2.7.1 Preparación del Archivo de Configuración (YAML)	62
2.7.2 Ejecución del algoritmo.....	63
2.8 MÉTODOS	63
2.8.1 Investigación Bibliográfica- Documental.....	64
2.8.2 Investigación de Campo.....	64
CAPÍTULO III	65
RESULTADOS Y DISCUSIÓN.....	65
3.1 ANÁLISIS DE RESULTADOS.....	65
3.1.1 DESVIACIÓN ESTÁNDAR.....	65
3.1.2 <i>Eficacia del Algoritmo</i>	68
3.2 DISCUSIÓN DE LOS RESULTADOS.....	70
CAPÍTULO IV	74
CONCLUSIONES Y RECOMENDACIONES	74
4.1 CONCLUSIONES.....	74
4.2 RECOMENDACIONES	76
REFERENCIAS BIBLIOGRÁFICAS	77
ANEXOS	83

ÍNDICE DE TABLAS

Tabla 1. Materiales usados para el desarrollo del escrito.....	23
Tabla 2: Los torques máximos aplicables de las articulaciones proporcionados por los fabricantes y calculados a partir de la ecuación 7	27
Tabla 3. Parámetros para denavith- hartenberg.....	34
Tabla 4. Características de kuka youbot.....	39
Tabla 5. Registro de los tiempos de ejecución.....	66

ÍNDICE FIGURAS

Figura 1. Modo de control de las articulaciones	7
Figura 2. Sistema robótico de lazo abierto.....	9
Figura 3. Sistema robótico de lazo cerrado.....	9
Figura 4. Modelo de las fricciones.....	10
Figura 5. Arquitectura de ros	14
Figura 6. Kuka youbot.....	15
Figura 7. Cálculo de parámetros dh	16
Figura 8. Vista real del robot kuka youbot.....	18
Figura 9. Rango de movimiento del brazo kuka youbot.	18
Figura 10. Brazo kuka youbot con 5 g.d.l.....	19
Figura 11. Placa sbc raspberry pi.	20
Figura 12. Modelo básico.....	24
Figura 13. Un diagrama que muestra el método alternativo - control de torque computado que incluye el modelo dinámico.....	25
Figura 14. Cadena cinemática en orocos kdl, que es una combinación tanto de la articulación y del eslabón.....	29
Figura 15. Controlador en el espacio articular.....	29
Figura 16. Modo de control de torque.....	30
Figura 17. Diagrama de clases completo para el controlador basado en modelos implementado en este trabajo.....	31
Figura 18. Diagrama de bloque control pd+g.	33
Figura 19. Diagrama cinemático kuka youbot.	34
Figura 20. Modelo simplificado de la fricción de coulomb en sistemas robóticos.....	41
Figura 21. Cálculo de velocidad basada en fricción.....	41
Figura 22. Observador de fricción.....	42
Figura 23. Bloque de control del planificador de trayectoria.....	43
Figura 24. Proceso de aplicación para una planeación de trayectoria óptimo.	44
Figura 25. Diagrama de flujo de la estructura del software.	47
Figura 26. Estructura de datos para el desarrollo.....	48
Figura 27. Estructura de comunicación de nodos ros.....	49

Figura 28. Desarrollo del algoritmo de planeación de trayectoria	51
Figura 29. Diagrama de clases de aplicación implementada	54
Figura 30. Desarrollo del algoritmo del controlador.....	55
Figura 31. Espacio de trabajo, clase principal y suscripción de trayectoria.....	56
Figura 32. Cálculo del erro en el algoritmo pd	57
Figura 33. Aplicación del cálculo de compensación de gravedad al control pd	58
Figura 34. Condición para encontrar los comandos de trayectoria de las articulaciones.....	59
Figura 35. Condición para encontrar las ganancias proporcional y derivativa.	59
Figura 36. Estructura de aplicación de caso de estudio.....	60
Figura 37. Arquitectura del nodo principal de sistema ros implementado.....	61
Figura 38. Flujo de información del sistema ros implementado.	61
Figura 39. Archivo de configuración yaml para el kuka youbot.....	62
Figura 40. Fichero launch del algoritmo de control.	63
Figura 41. Posiciones manipuladas del robot kuka youbot.....	65
Figura 42. Monitoreo del tiempo de ejecución del kuka youbot.....	66
Figura 43. Gráfico de dispersión de los tiempos de ejecución.....	68
Figura 44. Monitoreo de los puntos publicados por cada articulación del kuka youbot.	69
Figura 45. Avance del movimiento que se logra con el algoritmo desarrollado.....	70
Figura 46. Tiempos medidos con y sin observador.....	72

RESUMEN EJECUTIVO

Este trabajo aborda el desafío del control preciso de seguimiento de trayectoria en el manipulador robótico KUKA youBot, centrándose en un método alternativo de control de torque computado. El enfoque busca superar las no linealidades inherentes a estos sistemas, considerando las incertidumbres estructuradas (errores del modelo) y no estructuradas (fricción). El método alternativo implementa un controlador PI en cascada, combinando controladores de posición y velocidad para un mejor rechazo de perturbaciones. Un aspecto crucial es la identificación precisa de los parámetros del modelo dinámico, ya que los proporcionados por los fabricantes suelen ser inexactos. El trabajo incluye un análisis detallado de la semántica utilizada en los algoritmos de cuerpo rígido, contribuyendo a una construcción más precisa de las relaciones geométricas. Además, se incorpora la compensación de fricción para mejorar el rendimiento del controlador.

La implementación del método alternativo se complementa con una capa de control de seguridad que monitorea constantemente el estado de las articulaciones, protegiendo el hardware valioso del sistema. Los experimentos realizados incluyen la validación del modelo dinámico mediante tareas de compensación de gravedad y pruebas del controlador con trayectorias analíticas. Se llevaron a cabo experimentos de modelado y compensación de fricción, y el esquema de control se evaluó en las articulaciones más alejadas de la base. Los resultados demuestran que, a pesar de las inexactitudes en los parámetros del modelo dinámico, el controlador logra un seguimiento de trayectoria preciso con errores aceptables en las articulaciones del manipulador. Este trabajo contribuye significativamente al campo del control robótico, ofreciendo perspectivas valiosas sobre la implementación práctica de controladores avanzados en sistemas robóticos complejos, particularmente en el manejo de incertidumbres y la optimización del rendimiento en tareas de seguimiento de trayectoria.

Palabra clave: KUKA, control, torque, fricción, compensación, articulaciones.

ABSTRACT

This work addresses the challenge of precise trajectory tracking control in the KUKA youBot robotic manipulator, focusing on an alternative computed torque control method. The approach aims to overcome the inherent non-linearities of these systems, considering both structured (model errors) and unstructured (friction) uncertainties. The alternative method implements a cascade PI controller, combining position and velocity controllers for improved disturbance rejection. A crucial aspect is the accurate identification of dynamic model parameters, as those provided by manufacturers are often inaccurate. The work includes a detailed analysis of the semantics used in rigid body algorithms, contributing to a more precise construction of geometric relationships. Additionally, friction compensation is incorporated to enhance controller performance.

The implementation of the alternative method is complemented by a safety control layer that constantly monitors the state of the joints, protecting the system's valuable hardware. The experiments conducted include validation of the dynamic model through gravity compensation tasks and controller tests with analytical trajectories. Friction modeling and compensation experiments were carried out, and the control scheme was evaluated on the joints farthest from the base. The results demonstrate that, despite inaccuracies in the dynamic model parameters, the controller achieves precise trajectory tracking with acceptable errors in the manipulator's joints. This work significantly contributes to the field of robotic control, offering valuable insights into the practical implementation of advanced controllers in complex robotic systems, particularly in handling uncertainties and optimizing performance in trajectory tracking tasks.

Keywords: KUKA, control, torque, friction, compensation, joints

CAPÍTULO I

MARCO TEÓRICO

1.1. Tema de Investigación

DISEÑO E IMPLEMENTACIÓN DE ALGORITMO PARA CONTROL DE TRAYECTORIA CON COMPENSACIÓN GRAVITATORIA PARA EL ROBOT KUKA YOUNBOT

1.2. Planteamiento del Problema

Los manipuladores industriales, como el KUKA youBot, presentan un desafío significativo en el campo del control robótico debido a su naturaleza altamente no lineal. Esta característica inherente complica la tarea de lograr un seguimiento de trayectoria de alto rendimiento, especialmente en operaciones que requieren alta velocidad y precisión[1]. La complejidad se ve aumentada por la presencia de incertidumbres tanto estructuradas como no estructuradas. Las incertidumbres estructuradas se manifiestan como errores en el modelo dinámico del robot, mientras que las no estructuradas, como la fricción, son fenómenos difíciles de modelar con exactitud. Estos factores combinados hacen que el control preciso utilizando métodos lineales convencionales sea particularmente desafiante [2].

Un obstáculo adicional en el control de estos sistemas es la inexactitud de los parámetros del modelo proporcionado por el fabricante. Estos parámetros, cruciales para un control de alto rendimiento, a menudo carecen de la precisión necesaria. En el caso específico del KUKA youBot, la ausencia de sensores de torque en las articulaciones añade otra capa de complejidad al problema de control [3]. Además, la dinámica del sistema varía significativamente dependiendo de la configuración del robot y la carga que maneja, lo que requiere un enfoque de control adaptable y robusto.

En este contexto, el control de torque computado emerge como una solución prometedora para abordar estos desafíos multifacéticos. Este método ofrece la ventaja

de linealizar el sistema no lineal mediante la incorporación del modelo dinámico en el esquema de control. Comparado con métodos de control más simples, el control de torque computado proporciona un mejor rendimiento en el seguimiento de trayectorias, requiere menores ganancias de retroalimentación (lo que se traduce en un menor consumo de energía), y facilita el movimiento compatible, un aspecto crucial en tareas que implican interacción con el entorno [4].

El método de control de torque computado es ventajoso respecto al control PID (Proporcional – Integral - Derivativo) puro en muchos aspectos, como el rendimiento en el control de seguimiento de trayectoria, menores ganancias de retroalimentación, menor consumo de energía y el movimiento compatible [5]. Este trabajo tiene como objetivo lograr el esquema de controlador avanzado basado en el modelo, donde el sistema de control de retroalimentación lineal es obligatorio.

Hay dos aspectos importantes considerados en el control de torque computado: la ley de control y el modelo dinámico. Aunque este método es efectivo para el seguimiento preciso de trayectorias, el manipulador está sujeto a incertidumbres incluso en un entorno industrial bien definido, como la precisión de los parámetros del modelo y las dinámicas no modeladas [6]. La primera incertidumbre en el manipulador es que los parámetros del modelo proporcionados por los fabricantes no son lo suficientemente precisos, y los hallazgos al respecto son presentados por muchos investigadores [7]. La segunda incertidumbre en el manipulador es la dinámica no modelada, es decir, la fricción. El modelo preciso del robot es obligatorio, como se discutió en el párrafo anterior, y la falta de disponibilidad de sensores de torque en las articulaciones del manipulador youBot hace que el problema de control de seguimiento sea aún más difícil [8].

Aparentemente, hay solo ventajas limitadas de la estrategia de control de torque computado cuando la precisión de los parámetros del modelo no es lo suficientemente buena. Por lo tanto, los parámetros precisos del modelo de los enlaces son obligatorios para resolver este problema y ha sido ampliamente investigado por muchos investigadores [9], [10], donde los autores proponen el procedimiento de identificación

que se puede utilizar para estimar los parámetros del modelo del sistema, pero hay muchos desafíos involucrados en lograrlo debido a la presencia de convenciones implícitas, problemas semánticos y criterios de selección de trayectoria. La formulación recursiva de Newton-Euler modificada se utiliza para este propósito y luego se pueden usar métodos de ajuste de modelo para identificar los parámetros del modelo de los enlaces [11]. Este trabajo intenta implementar los parámetros del modelo estimados junto con la compensación de fricción, lo que nos acerca a obtener la dinámica completa del sistema, que a su vez proporciona un seguimiento de trayectoria robusto en operaciones de alta velocidad.

El control se puede considerar en dos espacios diferentes:

- **Control en el espacio de articulaciones:** El controlador de articulaciones contiene el sistema de control de retroalimentación que asegura que la ejecución del movimiento esperado sea seguida de cerca por las coordenadas de las articulaciones.
- **Control del espacio de tarea/operacional:** Es importante lograr el control preciso del movimiento del efector final, para lo cual es necesario el control de movimiento compatible. El control compatible no es más que tener un algoritmo adecuado que relacione la interacción del efector final y el entorno de manera apropiada.

Este trabajo utiliza el controlador en el espacio de articulaciones, por lo que se pueden comandar torques a las articulaciones del manipulador para ejecutar el movimiento deseado y no regula las fuerzas [12]. Los torques comandados pasan por una capa adicional de control de seguridad que se implementa alrededor del esquema de control de torque computado. Esta capa realiza la verificación en los tres modos: posición, velocidad y torque, que depende particularmente de los datos del codificador antes de aplicar los torques comandados a las articulaciones del manipulador.

Este enfoque holístico aspira a lograr un control preciso y robusto del manipulador KUKA youBot, superando las limitaciones de los métodos convencionales. Al abordar las múltiples facetas del problema de control, desde la modelización precisa hasta la

seguridad operativa, se busca aprovechar al máximo las capacidades del sistema, incluso en presencia de incertidumbres y dinámicas complejas. El objetivo final es establecer un marco de control que no solo mejore el rendimiento en tareas de seguimiento de trayectoria, sino que también abra nuevas posibilidades en aplicaciones robóticas avanzadas que requieren alta precisión y adaptabilidad.

1.3 Antecedentes Investigativos

La industria y la tecnología se encuentran en un proceso de transformación continua, donde las operaciones, procesos y métodos evolucionan constantemente para adaptarse a las demandas y la competitividad de los mercados globales. Esta evolución ha dado paso a la consolidación de la Industria 4.0 y al surgimiento de la Industria 5.0, cuya principal diferencia radica en el grado de interacción y colaboración entre humanos y máquinas. En el sector industrial, se observa una tendencia creciente hacia la automatización de procesos y la búsqueda de nuevas metodologías operativas. Entre estas innovaciones, destaca el uso de la robótica para realizar tareas programables y monitoreables, anticipando que el avance tecnológico continuará proporcionando herramientas que simplifiquen tanto las actividades cotidianas como los procesos industriales complejos [13].

El impacto de la robótica se extiende más allá del ámbito industrial, alcanzando sectores críticos como la medicina. En este campo, se han desarrollado métodos revolucionarios que permiten realizar cirugías de alta precisión utilizando sistemas robóticos avanzados, como el renombrado Sistema Quirúrgico Da Vinci. Además, la aplicación de robots y brazos robóticos en la rehabilitación de pacientes con daño cerebral o parálisis ha mostrado resultados prometedores. Estos dispositivos, programados para ejecutar movimientos repetitivos y trayectorias simples, como las de punto a punto, se basan en estudios exhaustivos para mejorar la movilidad de los usuarios. La integración de tecnologías como el aprendizaje automático, la inteligencia artificial y las redes neuronales en estos sistemas robóticos ha abierto nuevas posibilidades en el diagnóstico y tratamiento médico, ampliando los horizontes de la medicina moderna [14], [15].

En el ámbito internacional, se han realizado diversos estudios sobre el control de trayectoria, con un énfasis particular en el control punto a punto. Un ejemplo destacado es la investigación de Deng et al [16], que se centra en mejorar un brazo robótico portátil diseñado para usuarios con discapacidad motriz. Su trabajo logra avances significativos en la seguridad del movimiento, eliminando colisiones entre articulaciones y la necesidad de iniciar en una posición neutral, mediante la implementación de un algoritmo de compensación de gravedad que garantiza movimientos estables, fluidos y precisos.

En el contexto latinoamericano, Alan Fernando[17] propone un enfoque innovador para el control cinemático en el seguimiento de trayectoria de un manipulador 3R. Su investigación se centra en diseñar una estrategia de control para el seguimiento preciso del eslabón final de un robot manipulador. Aunque logra un error inferior a 1 cm, este se atribuye a dinámicas no modeladas y errores en los servomotores, destacando la importancia de considerar todos los factores en el diseño de sistemas de control robótico.

Adicionalmente, es importante mencionar los avances en el control de fuerza y cumplimiento en robots manipuladores. Li et al [18] introdujo el concepto de control de impedancia, que ha sido fundamental para mejorar la interacción de los robots con su entorno. Este enfoque permite a los robots ajustar su rigidez y comportamiento dinámico, facilitando tareas que requieren contacto físico y manipulación delicada.

En el campo de la robótica colaborativa, los trabajos de Luo et al. han sido pioneros en el desarrollo de estrategias de control que permiten una interacción segura y eficiente entre humanos y robots. [19] Sus investigaciones han contribuido significativamente a la implementación de robots en entornos de trabajo compartidos, mejorando la seguridad y la eficiencia en la colaboración humano-robot.

Otra área de investigación relevante es la aplicación de técnicas de aprendizaje por refuerzo en el control de robots. Los estudios de Mohammadi et al. han demostrado cómo estos métodos pueden ser utilizados para que los robots aprendan tareas

complejas de manipulación a través de la experiencia, mejorando continuamente su rendimiento y adaptabilidad a nuevas situaciones. [20]

En Ecuador, el trabajo de Cristian Fuentes [21] se enfoca en el desarrollo de un prototipo de brazo robótico soldador para superficies planas, principalmente orientado a la industria automotriz. Este proyecto destaca por su eficiencia y bajo costo en comparación con la mano de obra humana. La programación, realizada en Matlab, demuestra la capacidad del brazo robótico para simular con gran precisión el proceso de soldadura en diversos patrones de movimiento, evidenciando un funcionamiento óptimo de los componentes del brazo robótico.

Los antecedentes investigativos en el campo del control de trayectoria y la robótica aplicada muestran un panorama diverso y en rápida evolución. Desde aplicaciones industriales hasta médicas, pasando por desarrollos en robótica colaborativa y técnicas de aprendizaje avanzado, se observa una tendencia hacia sistemas más precisos, adaptables y seguros. Estos avances no solo mejoran la eficiencia y la calidad en diversos sectores, sino que también abren nuevas posibilidades para la interacción entre humanos y máquinas.

1.4 Fundamentación Teórica

La fundamentación teórica en el campo del control de robots manipuladores constituye la base esencial para comprender y desarrollar sistemas de control avanzados y eficientes. En esta sección, exploraremos los principios fundamentales que sustentan el control de trayectoria preciso en robots como el KUKA youBot. Se abordará conceptos clave como la cinemática y dinámica de robots, los principios del control de torque computado, las técnicas de identificación de parámetros dinámicos, y los métodos de compensación de fricción. Estos elementos teóricos son cruciales para diseñar controladores robustos capaces de manejar las no linealidades inherentes a los sistemas robóticos y las incertidumbres en los parámetros del modelo. Además, se examinará las bases del control en el espacio de articulaciones y su relación con el control en el espacio operacional. La comprensión profunda de estos conceptos permitirá desarrollar estrategias de control avanzadas que mejoren significativamente

el rendimiento y la precisión en el seguimiento de trayectorias, aspectos fundamentales para la aplicación exitosa de robots manipuladores en entornos industriales y de investigación cada vez más exigentes.

1.4.1. Modo de control

La articulación puede ser comandada con tres tipos de entradas: posición, velocidad y torque. Por lo tanto, una articulación puede ser controlada en tres modos diferentes, como se muestra en la Figura 1

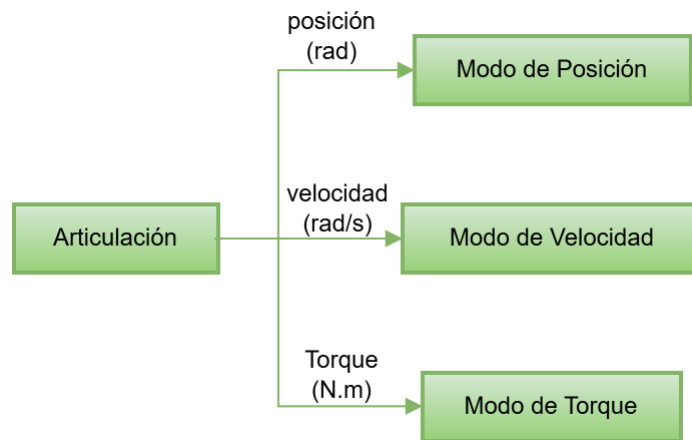


Figura 1. Modo de control de las articulaciones

En el modo de posición, se comanda una posición (ángulo de la articulación en radianes) a una articulación para moverla de una posición particular a otra. Mientras que el modo de velocidad comanda la velocidad a las articulaciones para lograr el movimiento deseado. La clasificación final es el modo de control de torque que comanda el torque a las articulaciones del manipulador. Los errores de seguimiento de trayectoria se pueden minimizar eficientemente a través del modo de torque ya que se consideran la dinámica y la configuración de la articulación [3]. Dado que el motor toma la corriente como entrada, el torque de la articulación comandado debe convertirse en corrientes objetivo utilizando un modelo de motor basado en la siguiente ecuación (1)

$$\text{Corriente} = \frac{\text{Torque comandado} \times \text{Relación de engranaje}}{\text{Constante de torque}} \quad (\text{amperios}) \quad (1).$$

El inverso de la ecuación anterior convierte la corriente que se mide de los motores en torque de la articulación.

$$\text{Torque} = \frac{\text{Corriente} \times \text{Constante de torque}}{\text{Relación de engranaje}} \quad (\text{Newton metros}) \quad (2)$$

La conversión de corriente a torque como se especifica en la ecuación (2) es particularmente útil para interpretar los resultados del sistema fácilmente. Las ecuaciones (1) y (2) se dan basadas en el controlador youbot que se utiliza en este trabajo. El control de torque es bastante robusto y sigue las trayectorias de manera precisa. Este trabajo utiliza el controlador de torque avanzado basado en modelo y se implementa en el espacio de articulaciones en lugar del espacio de tareas. Esto nos ayuda a ejecutar/seguir las trayectorias con precisión. Este trabajo asume que las ganancias existentes del controlador a nivel de articulación son óptimas, por lo que la sintonización de las ganancias del controlador a nivel de articulación no es el enfoque principal.

1.4.2. Descripción general de los sistemas de control

Las definiciones básicas de esta sección se refieren a [22], [23]. Un sistema se considera lineal cuando un cambio en la salida es directamente proporcional al cambio en la entrada. Dado que los sistemas robóticos son altamente no lineales, es importante tener una estrategia de control lineal. Esto se logra mediante el uso de técnicas de linealización que resuelven el problema de los sistemas no lineales. Hay dos tipos de sistemas de control disponibles: sistemas de lazo abierto y sistemas de lazo cerrado. Los sistemas de lazo abierto y lazo cerrado se representan en las Figuras 2 y 3 respectivamente. En los sistemas de lazo abierto, se dan entradas al sistema y se acepta el resultado independientemente de las desviaciones en el comportamiento deseado (no se considera la retroalimentación).

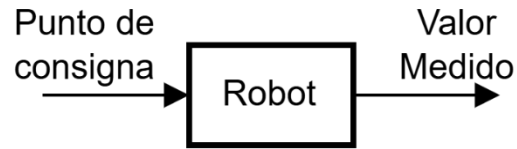


Figura 2. Sistema robótico de lazo abierto

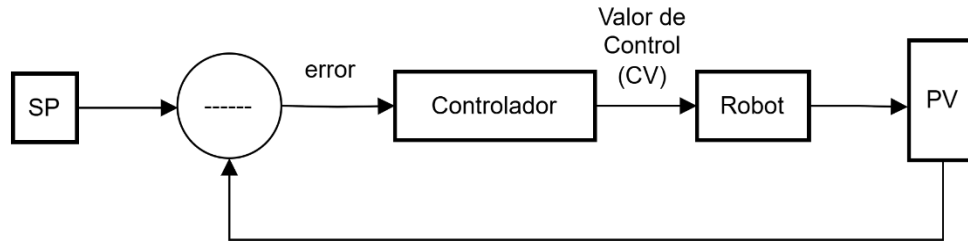


Figura 3. Sistema robótico de lazo cerrado

Mientras que los sistemas de control de lazo cerrado tienen en cuenta la retroalimentación del sistema para minimizar las desviaciones entre los datos deseados y observados. Este trabajo utiliza el sistema de lazo cerrado para lograr el control preciso de los manipuladores.

1.4.3. Modelado y compensación de fricción

En el campo de la robótica, el modelado preciso de la dinámica del manipulador es crucial para lograr un control efectivo. Un aspecto fundamental de este modelado es la consideración de la fricción, un fenómeno altamente no lineal que afecta significativamente el rendimiento del sistema [24]. La fricción se clasifica principalmente en dos categorías: estática, que se presenta cuando un cuerpo rígido está a punto de moverse, y cinética (o dinámica), que ocurre durante el movimiento. Estos efectos influyen tanto en las configuraciones estáticas como dinámicas del manipulador, pudiendo causar inestabilidades en el control si no se manejan adecuadamente.

Para abordar este desafío, se han desarrollado diversos enfoques de compensación de fricción, que se pueden clasificar en esquemas basados en modelos y esquemas no basados en modelos. En el caso específico del manipulador youBot, estudios previos

[25] han estimado la fricción estática mediante pruebas automatizadas, proporcionando una base valiosa para la incorporación de términos de compensación en el modelo dinámico. La evolución de los modelos de fricción ha sido significativa, partiendo desde simples representaciones de fricción estática y de Coulomb, hasta modelos más sofisticados que incluyen efectos viscosos y el fenómeno de Stribeck, ofreciendo una descripción más precisa del comportamiento real del sistema. Ver Figura 4.

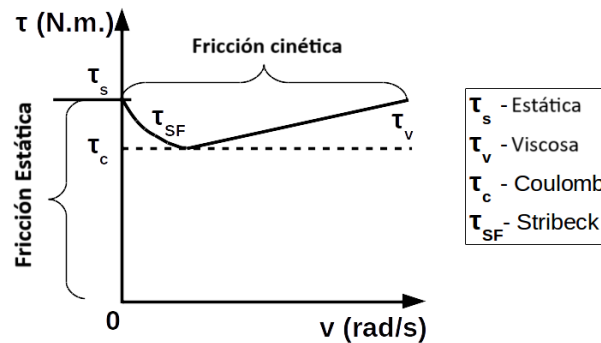


Figura 4. Modelo de las Fricciones

Las ecuaciones dinámicas de cadenas cinemáticas de cuerpos rígidos[26] son fundamentales para comprender la relación entre el movimiento del sistema y las fuerzas que lo generan [27]. Estas ecuaciones se pueden abordar mediante dos enfoques principales: la dinámica directa, que calcula aceleraciones a partir de fuerzas aplicadas, y la dinámica inversa, que determina las fuerzas necesarias para producir un movimiento deseado. La precisión de estos cálculos depende crucialmente de la exactitud de los parámetros del modelo, como masas, centros de masas y momentos de inercia, que a menudo son desconocidos y requieren técnicas de identificación avanzadas como la regresión ridge.

La formulación recursiva de Newton-Euler se ha establecido como el método más eficiente computacionalmente para resolver estas ecuaciones dinámicas[28]. Este enfoque utiliza una recursión hacia adelante para calcular velocidades y aceleraciones, seguida de una recursión hacia atrás para determinar fuerzas y torques. Una innovación importante en este campo es el uso de vectores espaciales, que combinan aspectos

lineales y angulares en un solo vector de seis dimensiones, mejorando significativamente la eficiencia computacional.

Un desafío persistente en el modelado de sistemas robóticos es la presencia de incertidumbres, que incluyen errores en el modelo y dinámicas no modeladas [29]. Para abordar esto, algunos investigadores han propuesto modificaciones a la formulación de Newton-Euler, como la linealización matricial para la identificación de parámetros [30]. Sin embargo, estas aproximaciones a menudo enfrentan problemas de inexactitud en la semántica de las relaciones geométricas entre cuerpos rígidos, especialmente en lo que respecta a la transmisión de aceleraciones entre articulaciones.

La implementación práctica de estos modelos dinámicos requiere una cuidadosa consideración de las transformaciones entre diferentes marcos de referencia. Mientras que los cálculos numéricos pueden tratar estas transformaciones como simples operaciones matriciales, es crucial realizar una verificación semántica para asegurar que las operaciones tengan un significado físico coherente. Esta atención a la semántica es particularmente importante en las transformaciones de posición, movimiento y fuerza, donde el uso exclusivo de cálculos numéricos puede introducir errores lógicos sutiles pero significativos [31].

1.4.4. Modelos de control aplicados a la Robótica

El control de procesos es un aspecto fundamental en la robótica, especialmente en manipuladores industriales. Tradicionalmente, el controlador PID ha sido ampliamente utilizado, aplicando un PID independiente a cada articulación del robot. Este enfoque, conocido como control PID de eje único, es popular en aplicaciones industriales debido a su simplicidad y robustez [32]. La sintonización de las ganancias del controlador PID puede realizarse mediante diversos métodos, desde técnicas empíricas basadas en la respuesta al paso hasta sofisticados algoritmos de autosintonización [23]. Sin embargo, este método muestra limitaciones en operaciones de alta velocidad, principalmente debido a restricciones mecánicas y a la naturaleza no lineal de los sistemas robóticos.

Para superar estas limitaciones, se han desarrollado métodos de control más avanzados que tienen en cuenta la dinámica completa del sistema. Entre estos, destacan el control

de impedancia y el control de torque computado [33]. El control de impedancia es particularmente interesante porque considera la interacción dinámica entre el robot y su entorno, lo que resulta crucial en tareas que requieren contacto físico o manipulación precisa [34]. Por otro lado, el control de torque computado utiliza el modelo dinámico del robot para calcular los torques necesarios en cada articulación, logrando así un seguimiento de trayectoria más preciso.

Un ejemplo notable de implementación del control de torque computado se encuentra en el trabajo realizado con el manipulador youBot presentado en [3]. En este estudio, se aplicó un controlador PD junto con el modelo dinámico del robot, considerando principalmente los torques gravitacionales $\tau_g(q)$. Aunque este enfoque no tuvo en cuenta los efectos de la fricción, demostró la importancia de contar con parámetros de modelo precisos para mejorar el rendimiento del control. Una observación crucial de este trabajo fue que el torque máximo especificado por los fabricantes a menudo es insuficiente para mover las articulaciones del robot en ciertas configuraciones, lo que llevó a proponer un método para calcular el torque máximo aplicable para cada articulación ver ecuación (3):

$$\tau_{\max} = M_{\max} K_p e_{\max} \quad (3)$$

En esta ecuación, $\tau_{\max}$ es el vector de torques máximos permitidos, $M_{\max}$ es la matriz de inercia maximizada del manipulador, K_p es la ganancia proporcional del controlador, y $e_{\max}$ es el error máximo de posición admisible para cada articulación.

Chung [35] profundizó en el esquema de control de torque computado, proponiendo la siguiente formulación ver ecuación (4):

$$\tau = H(q)v + C(q, \dot{q})\dot{q} + \tau_g(q) \quad (4)$$

Donde τ es el vector torque, $H(q)$ es la matriz de inercia o matriz de masa la cual depende de la configuración articular q , $C(q, \dot{q})$ es la matriz de Coriolis y fuerza centrífuga la cual depende de la configuración q y de la velocidad articular $\dot{q}$, $\tau_g(q)$ es el vector de torques debido a la gravedad, la cual depende de la configuración q y

representa los torques necesarios para compensar el efecto de la gravedad en cada articulación del robot.

Este enfoque combina un bucle de compensación no lineal interno con un bucle externo que utiliza la señal de control v . Al sustituir esta ley de control en el modelo dinámico del manipulador, se logra linealizar el sistema, permitiendo la aplicación de estrategias de control lineal. El bucle externo típicamente implementa un control PD ver ecuación (5).

$$v = \ddot{q}_d + CV \quad (5)$$

Una variante de este método, conocida como control tipo torque computado, agrega una compensación variable en el bucle externo para mejorar el seguimiento de trayectoria en presencia de incertidumbres en el modelo dinámico. Este enfoque ha demostrado ser eficaz en la reducción del error de seguimiento, aunque requiere un control preciso del torque de la articulación y depende en gran medida de la exactitud de los datos de posición y velocidad de las articulaciones.

1.4.5. Robot Operating System (ROS)

ROS o también conocido como Robot Operating System es una plataforma de código abierto usado en la robótica, proporciona una amplia gama de servicios y librerías, las cuales simplifican en gran manera el desarrollo de aplicaciones complejas para robots, es similar a muchas otras plataformas de desarrollo, pero su mayor virtud radica en haber podido unir lo mejor de cada sistema en un sistema sólido, capaz de realizar comunicaciones con robots modernos y los ya existentes, todo esto gracias a su sistema de código abierto, que ha permitido que profesionales y aficionados en todo el mundo puedan compartir sus creaciones, dando lugar a una gran comunidad de colaboradores [36]. Al momento de desarrollar soporta distintos lenguajes de programación entre ellos, Python, C++, Lips, Lua, etc. Además, puede ser ejecutado en sistemas operativos como Linux, Ubuntu, Mac OS X, entre otras menos conocidas como Fedora, Gentoo, etc. Al contar con una gran comunidad compuesta por investigadores, aficionados, fabricantes, dispone de soporte nativo para una gran variedad de hardware incluso para

robots de investigación. A continuación, se muestra la Figura 5 que muestra como ROS permite el intercambio de información dentro de un protocolo de subscriptor y publicador [37].

Para poder gestionar la comunicación en ROS es necesario crear un ROS master, que tiene como principal objetivo el permitir que diferentes nodos puedan conectarse y comunicarse, los topic no son más que rutas de conexión, donde los nodos pueden publicar o suscribirse.

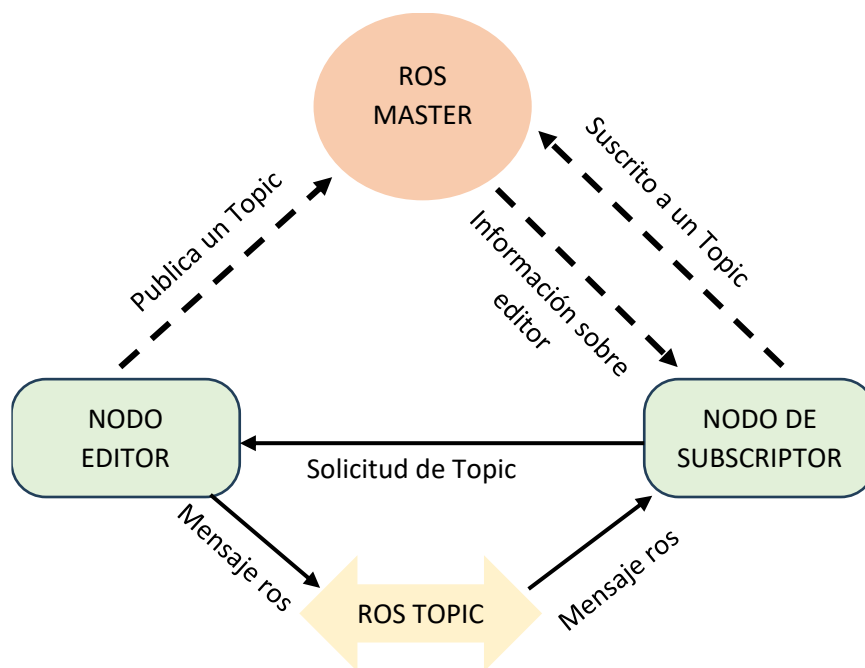


Figura 5. Arquitectura de ROS

1.4.6. Automatización de bajo costo

La automatización de bajo costo (LCA), es una forma de automatización rentable, que cuesta entre un 80% y 90% menos que la automatización integral, proporcionando un nivel de eficiencia muy similar, creando un grado de automatización en torno a los equipos, herramientas, métodos y personas existentes. Este tipo de automatización puede usarse tanto en estaciones de trabajo de ensamble como en áreas de logística, la LCA son sistemas básicamente, mecánicos, neumáticos, hidráulicos, eléctricos e híbridos (combinación de los sistemas ya antes mencionados) [38].

Este tipo de automatización está muy presente en países en vía de desarrollo, donde existen limitaciones financieras, de liderazgo técnico, de productividad en cuanto a la mano de obra, etc. Aquí es donde entra la Automatización de bajo costo, donde el desarrollar estrategias viables a estas condiciones, es parte del éxito. Entre sus beneficios tenemos, su bajo costo de inversión, el retorno de la inversión es alto en términos de productividad, proporcionando una mejor eficiencia de trabajo, calidad y flexibilidad [39].

1.4.7. Cinemática directa de robots

Los robots manipuladores no son más que una cadena cinemática abierta, donde en la mayoría de los casos esta expresado por $n+1$ eslabones, los cuales están conectados entre sí por n juntas, Figura 6. se puede observar de manera más precisa las 5 articulaciones que posee KUKA youBot.

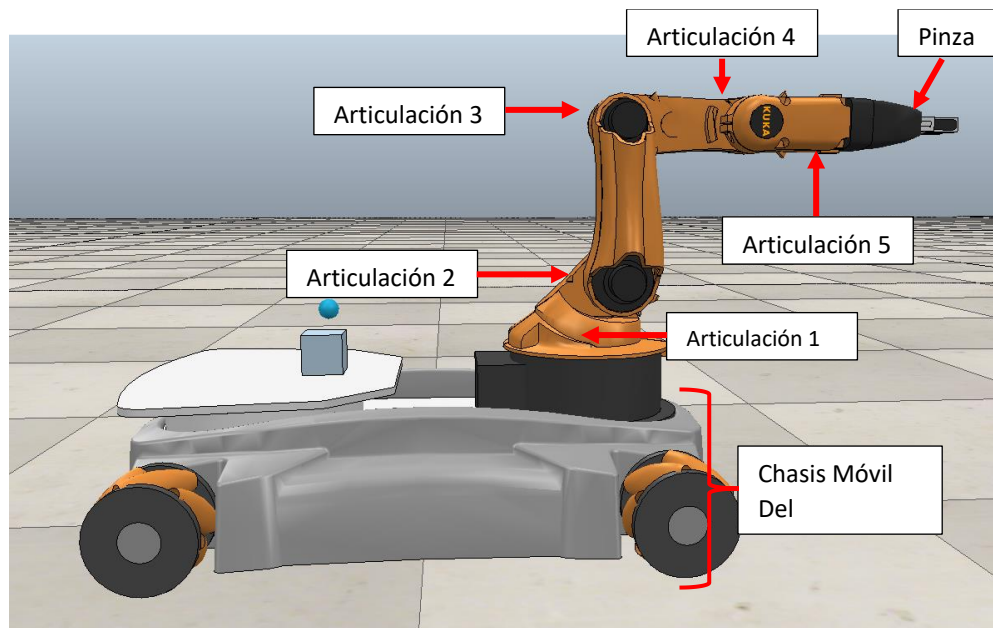


Figura 6. KUKA youBot.

Una vez tenemos claros estos conceptos procedemos con el método D-H, el cual debemos tener en cuenta ciertas restricciones iniciales, que son las siguientes:

DH1: El eje x_i debe ser perpendicular a los ejes z_i y z_{i-1}

DH2: El eje x_i debe intersectar a los ejes z_i y z_{i-1}

Estas restricciones nos permiten determinar una matriz de transformación homogénea utilizando únicamente 4 parámetros los cuales dos son distancias $a_i - d_i$ y dos son ángulos $\alpha_i - \theta_i$, dichos parámetros ayudan a determinar en su totalidad la posición del efector final.

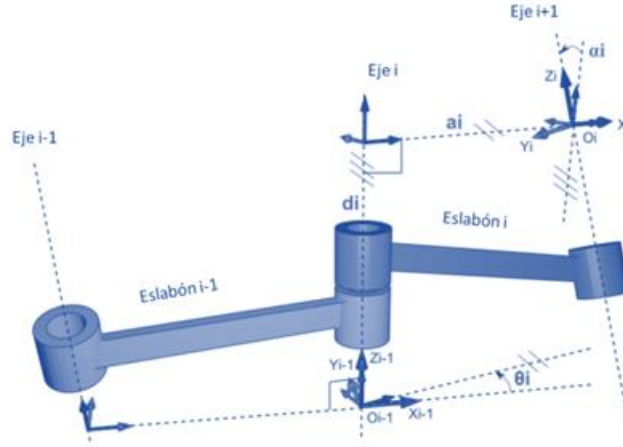


Figura 7. Cálculo de parámetros DH

Para la asignación de referencias se procede de la siguiente manera, primero se traza los ejes de acción en las articulaciones, para después alinear los ejes z_{i-1} sobre los ejes de acción i , una vez realizado esto procedemos a definir los ejes x_i de acuerdo a las restricciones propias dadas por D-H, para finalmente definir los ejes y_i con la regla de la mano derecha, obteniendo como resultado algo similar a lo mostrado en la Figura 3.

Donde:

θ_i : ángulo de x_{i-1} a x_i medido alrededor de z_{i-1} positivo cuando la rotación es horaria

d_i : distancia de x_{i-1} a x_i medido a lo largo de z_{i-1}

a_i : distancia de z_{i-1} a z_i medida a lo largo de x_i

α_i : ángulo de z_{i-1} a z_i medido alrededor de x_i positivo cuando la rotación es horaria

Matriz de transformación

$$A_{i-1}^i = A_{R_z}(q_i)A_{T_z}(d_i)A_{T_x}(a_i)A_{R_x}(\alpha_i) \quad (6)$$

Donde:

$A_{R_z}(qi)$ = Matriz de transformación homogénea de rotación en z

$A_{T_z}(di)$ = Matriz de transformación homogénea de desplazamiento en z

$A_{T_x}(ai)$ = Matriz de transformación homogénea de desplazamiento en x

$A_{R_x}(\alpha i)$ = Matriz de transformación homogénea de rotación x

$$A_{i-1}^i = \begin{bmatrix} \cos(qi) & -\sin(qi) & 0 & 0 \\ \sin(qi) & \cos(qi) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & di \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & ai \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha i & -\sin\alpha i & 0 \\ 0 & \sin\alpha i & \cos\alpha i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_{i-1}^i = \begin{bmatrix} \cos(qi) & -\sin(qi)\cos(\alpha i) & \sin(qi)\sin(\alpha i) & ai * \cos(qi) \\ \sin(qi) & \cos(qi)\cos(\alpha i) & -\cos(qi)\sin(\alpha i) & ai * \sin(qi) \\ 0 & \sin(\alpha i) & \cos(\alpha i) & di \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

La matriz A_{i-1}^i sirve para conocer la posición del efector final con respecto a la base del robot.

1.4.8. Robot manipulador KUKA youBot

Kuka youBot Figura 8 es un manipulador móvil creado con fines educativos y de investigación, cuyo objetivo es convertirse en una plataforma estándar dentro de la comunidad educativa y científica. Cuenta con interfaces totalmente abiertas que permite a los desarrolladores acceder a casi todos los niveles de control del hardware, además de contar con KUKAyouBot API, una interfaz de programación de aplicaciones con compatibilidad con ROS o OROCOS, con simulación en código abierto en Gazebo [40].



Figura 8. Vista real del robot KUKA youBot

KUKA youBot consta de dos partes principales, un brazo robótico de KUKA youBot consta de cinco articulaciones Figura 9 y Figura 10, y una pinza, alcanzando una altura de 655 mm, con un peso de 6,3 Kg pudiendo trasportar una carga útil de hasta 0,5 kg de peso. Las articulaciones del brazo giran alrededor de dos ejes diferentes, las articulaciones dos, tres y cuatro giran en paralelo alrededor de un eje y las articulaciones uno y cinco en paralelo alrededor del otro si el brazo apunta hacia arriba, su velocidad máxima de rotación es de 90 deg/s [41].

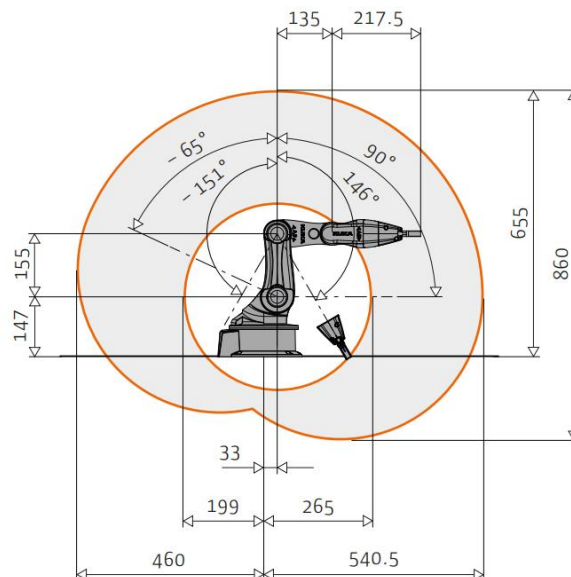


Figura 9. Rango de movimiento del brazo KUKA youBot.

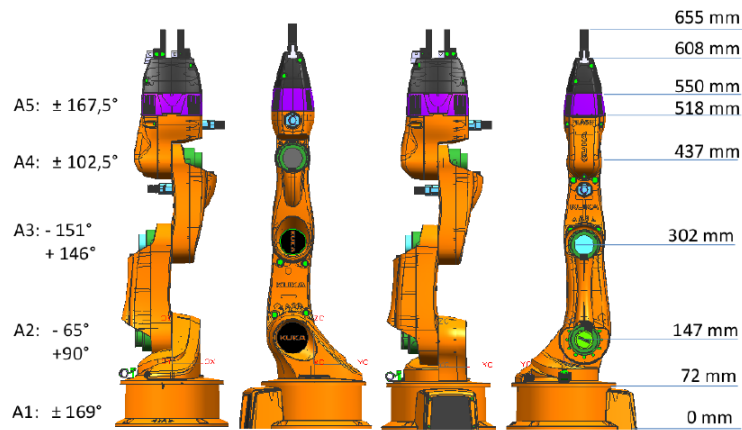


Figura 10. Brazo KUKA youBot con 5 G.D.L.

La base omnidireccional, es una plataforma móvil que consta de un mini PC ejecutado por Ubuntu, una batería y cuatro ruedas omnidireccionales con motores independientes esto permite que el Youbot gire sobre su mismo eje, así como su movimiento en cualquier dirección. La velocidad máxima a la cual se puede desplazar es de 0,8 m/s, con un peso de 20 Kg y unas medidas de 580 mm por 380 mm por 140 mm [22].

1.4.9. Automatización de Bajo Costo

La automatización de bajo costo ha emergido como una solución innovadora para empresas y entusiastas que buscan optimizar procesos sin incurrir en gastos excesivos. Este enfoque utiliza tecnologías asequibles y versátiles, como los ordenadores de placa única tipo Raspberry Pi, para crear sistemas de control y monitoreo eficientes. A diferencia de los sistemas de automatización industrial tradicionales, que a menudo requieren una inversión significativa en hardware especializado, la automatización de bajo costo aprovecha componentes de consumo general, reduciendo drásticamente los costos de implementación. Esta democratización de la tecnología de automatización ha abierto nuevas posibilidades para pequeñas y medianas empresas, así como para proyectos educativos y de investigación, permitiendo la experimentación y el desarrollo de soluciones personalizadas a una fracción del costo tradicional.

El corazón de muchos proyectos de automatización de bajo costo es el Raspberry Pi Figura 11, un ordenador de tamaño reducido pero potente, originalmente diseñado para

la educación en informática. Con su capacidad de procesamiento, múltiples puertos de entrada/salida, y soporte para una amplia gama de sensores y actuadores, el Raspberry Pi se ha convertido en una plataforma ideal para aplicaciones de control y automatización. Su sistema operativo basado en Linux, combinado con lenguajes de programación accesibles como Python, facilita el desarrollo de software de control sofisticado. Además, la comunidad activa alrededor de Raspberry Pi ha generado una abundancia de recursos, bibliotecas y ejemplos de código, lo que acelera el proceso de desarrollo y permite incluso a usuarios con conocimientos técnicos limitados implementar soluciones de automatización efectivas.

Las aplicaciones de la automatización de bajo costo con Raspberry Pi son vastas y variadas. En entornos industriales, estos dispositivos se utilizan para monitorear equipos, controlar procesos de producción y recopilar datos en tiempo real para análisis y optimización. En agricultura, los sistemas basados en Raspberry Pi se emplean para automatizar el riego, monitorear condiciones ambientales y controlar invernaderos. En el ámbito doméstico, se utilizan para crear sistemas de domótica, gestión energética y seguridad. La flexibilidad del Raspberry Pi también lo hace ideal para prototipado rápido y desarrollo de pruebas de concepto en proyectos de IoT (Internet de las Cosas). A medida que la tecnología avanza, se espera que la automatización de bajo costo continúe expandiéndose, ofreciendo soluciones cada vez más sofisticadas y accesibles para una amplia gama de aplicaciones.



Figura 11. Placa SBC Raspberry Pi.

La automatización de bajo costo ofrece varias ventajas significativas en comparación con los sistemas tradicionales basados en PLCs o IPCs de marcas como Siemens o Allen Bradley. En primer lugar, el costo de adquisición e implementación es considerablemente menor, permitiendo a pequeñas empresas y proyectos con presupuestos limitados acceder a soluciones de automatización [42]. La flexibilidad y personalización son también puntos fuertes, ya que los sistemas basados en Raspberry Pi pueden adaptarse fácilmente a necesidades específicas sin depender de software propietario o hardware especializado [43]. Además, la curva de aprendizaje suele ser menos pronunciada, gracias a la abundancia de recursos educativos y comunidades de soporte en línea. Sin embargo, es importante reconocer que los sistemas tradicionales aún mantienen ventajas en términos de robustez, soporte a largo plazo y certificaciones para entornos industriales críticos. La elección entre automatización de bajo costo y sistemas tradicionales dependerá, por tanto, de los requisitos específicos del proyecto, considerando factores como escala, criticidad y necesidades de mantenimiento a largo plazo.

1.3. Objetivos

1.3.1 Objetivo General

Desarrollar un algoritmo para el control de trayectoria con compensación gravitatoria para el robot KUKA youBot.

1.3.2 Objetivo Específico

- Determinar la cinemática y dinámica de KUKA youBot basado en compensación gravitatoria.
- Diseñar un Algoritmo de control con compensación gravitacional para el brazo KUKA youBot.
- Implementar el algoritmo de control para la manipulación de KUKA youBot.

CAPÍTULO II

METODOLOGÍA

2.1 Materiales

En el ámbito de la automatización de bajo costo, la selección cuidadosa de componentes es fundamental para lograr un equilibrio óptimo entre funcionalidad, rendimiento y economía. El núcleo de estos sistemas típicamente se centra en un ordenador de placa única, siendo la Raspberry Pi 4 una opción predominante debido a su potencia de procesamiento, versatilidad y costo accesible. Este dispositivo central se complementa con una gama de componentes periféricos que incluyen sensores diversos para la recopilación de datos ambientales y de proceso, actuadores para el control físico de sistemas, y elementos de interconexión que facilitan la integración de estos componentes en un sistema cohesivo. La selección específica de estos elementos periféricos depende en gran medida de los requisitos particulares de cada proyecto de automatización, permitiendo una personalización significativa y adaptabilidad a diversas aplicaciones industriales y de investigación.

El aspecto software de estos sistemas de automatización es igualmente crucial y se basa típicamente en el sistema operativo Raspberry Pi OS, complementado con lenguajes de programación de alto nivel como Python. Esta combinación proporciona una plataforma robusta y flexible para el desarrollo de algoritmos de control y la implementación de interfaces de usuario. Además, se incorporan frecuentemente frameworks especializados para facilitar funcionalidades específicas, como la creación de interfaces web o la comunicación en entornos IoT. Es importante destacar que la integración efectiva de estos componentes hardware y software requiere una consideración cuidadosa de las especificaciones técnicas, la compatibilidad entre sistemas, y las normativas aplicables en entornos industriales. Esto incluye la selección de elementos de protección adecuados, como fuentes de alimentación reguladas y

enclosures industriales, para garantizar la durabilidad y fiabilidad del sistema en condiciones de operación variadas y potencialmente adversas.

La tabla 1 muestra los diferentes materiales usados durante el desarrollo del estudio

Tabla 1. Materiales usados para el desarrollo del escrito.

Fuente: Elaborado por el investigador		
Ilustración	Nombre	Descripción
	Microsoft Word	Usado para el desarrollo del informe
	Microsoft Excel	Para el estudio estadístico de los resultados obtenido
	Computador	Para la elaboración del informe, además de los diferentes elementos usados para el desarrollo de la propuesta.
	Internet	Fuente de información
	Base de datos de artículos científicos	Usado para poder acceder a diferentes investigaciones para el desarrollo de la propuesta

2.2. Metodología

2.2.1. Esquemas de Control

El esquema de control mostrado la Figura 12 alimenta la variable de control como una entrada de aceleración $\ddot{q}_d$ al modelo dinámico para calcular los torques del modelo, mientras que Chung et al. [35] alimentó la aceleración deseada real $\ddot{q}_d$ como entrada del modelo en el esquema de control definido. La razón de esta diferencia en el esquema del controlador es que no hay acelerómetro presente en las articulaciones del manipulador. En tal caso, es justo responder a la siguiente pregunta: ¿por qué no diferenciar la velocidad medida para obtener los datos de aceleración? La respuesta a la pregunta anterior es que la velocidad medida que se obtiene de los codificadores generalmente tiene ruido y tomar la derivada de estos datos amplificaría aún más el ruido, por lo que la entrada de aceleración deseada no se considera en este trabajo.

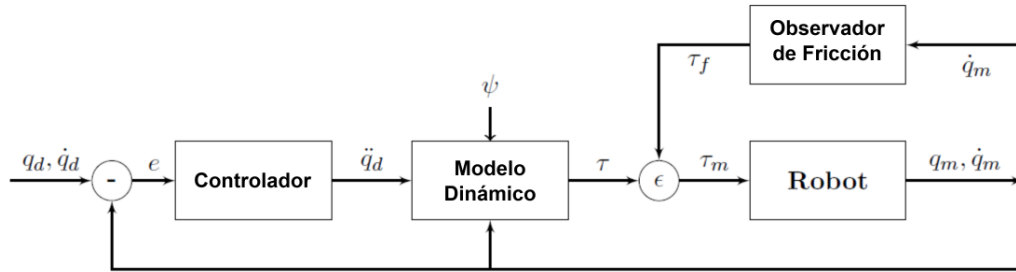


Figura 12. Modelo Básico

Una observación importante en este trabajo es que el modelo considera solo el momento de inercia del enlace y se puede restar del momento de inercia del motor. Los momentos de inercia del motor se pueden calcular con las especificaciones proporcionadas por los fabricantes. Por lo tanto, se selecciona el método alternativo en este trabajo para el control basado en modelo, como se muestra en la Figura 13.

Los siguientes componentes son realmente importantes para alcanzar el objetivo final de este proyecto que se especifica en la Figura 13.

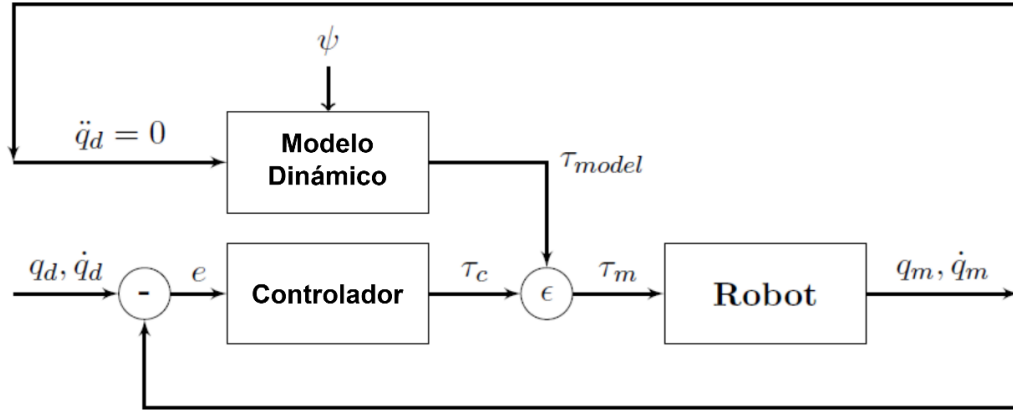


Figura 13. Un diagrama que muestra el método alternativo - Control de torque computado que incluye el modelo dinámico

- **Robot/Proceso/Planta** representa el sistema real y contiene un actuador a través del cual se controla el proceso. El sistema que se utiliza en este trabajo se explica en detalle en la sección precedente. La perturbación en el sistema es la variable no controlada que tiene una influencia en el rendimiento del controlador, pero el controlador ajusta la variable de control para compensar esa influencia.
- **El modelo dinámico** es necesario para generar los torques del modelo basados en el solucionador ID de Orocos KDL [44] y el modelo 3D del fabricante [37] donde ψ representa los parámetros del modelo dinámico que pueden reemplazarse con los parámetros identificados una vez que la estimación sea exitosa. El término de compensación de fricción τ_f también se considera en este trabajo con el enfoque básico.
- **Estimación de parámetros** que estima los parámetros precisos del modelo. ψ representa los parámetros del modelo dinámico. Las trayectorias de excitación se pueden identificar utilizando la serie de Fourier finita [45] y los coeficientes se pueden optimizar para identificar mejor los parámetros del modelo. Esta

sección presenta una visión general sobre la estimación de parámetros y la parametrización y optimización de la trayectoria.

- **El controlador** resuelve el problema de control donde reduce el error introduciendo la regulación en el sistema con el método de control de retroalimentación lineal. Este trabajo utiliza el mecanismo de control en cascada y la velocidad de la articulación se controla en el controlador Proporcional – Integral (PI) de velocidad.

2.2.2. Cálculo de Torque de articulaciones de Robot KUKA youBot

El enfoque principal de este trabajo es lograr el control de torque computado en el manipulador youBot. El brazo youBot consta de cinco articulaciones rotativas con 5 grados de libertad controlados por motores individuales e incluye una pinza en el último eslabón que tiene dos dedos. Sin embargo, este trabajo no considera la pinza tanto en el modelado como en el cálculo de torque. Las articulaciones del manipulador se controlan con entradas de posición, velocidad o torque a través de EtherCAT, que tiene capacidad en tiempo real. El modelo 3D del manipulador es proporcionado por el fabricante [37] y se utiliza para fines de modelado y también se usa en el control de torque computado donde se calculan los torques del modelo.

Para aumentar el rendimiento de seguimiento, los valores de ganancia se ajustan a un valor alto donde existe la posibilidad del problema de sobreimpulso. Debido a esto, la articulación podría cruzar los límites proporcionados por los fabricantes, por lo que es importante introducir límites de posición artificiales a las articulaciones para evitar cualquier daño al sistema. La introducción de las características de limitación y el controlador de seguridad se explica en las secciones siguientes. Dado que las articulaciones se controlan en modo de torque, es necesario verificar el torque máximo aplicable para las articulaciones individuales del manipulador. Considerando las especificaciones proporcionadas por los fabricantes, los torques máximos aplicables de las articulaciones se identificaron con la siguiente ecuación (7):

$$\tau_{max} = \tau_{nominal} \cdot \text{relación de reducción del engranaje} \quad (7)$$

donde el torque nominal se representa en N.m., y los valores de la relación de reducción del engranaje se obtienen de las especificaciones detalladas de youBot-store. Los torques máximos de las articulaciones calculados se enumeran en la siguiente tabla 2:

Tabla 2: Los torques máximos aplicables de las articulaciones proporcionados por los fabricantes y calculados a partir de la ecuación 7

Modelo 3-D	Articulación 1	Articulación 2	Articulación 3	Articulación 4	Articulación 5
	En Newton metros (N.m.)				
Fabricante τ_{max}	9.500	9.500	6.000	2.000	1.000
Calculado τ_{max}	12.901	12.901	8.270	4.175	4.175

Las especificaciones de youBot-store no indican claramente el torque nominal de la articulación 5, por lo que se consultan los detalles del motor del [46] que tiene las especificaciones del motor Maxon EC32 plano sin escobillas de 15 vatios con el sensor Hall número 267121.

2.2.3. Orocos KDL (Kinematics and Dynamics Library)

Un requisito fundamental para el controlador basado en modelo es contar con un modelo matemático capaz de predecir el torque de manera precisa, aproximándose lo más posible al sistema real. Este modelo debe abarcar las propiedades físicas del robot, incluyendo sus características cinemáticas, geométricas y dinámicas. Estas propiedades son esenciales para calcular los torques del modelo basados en la información de trayectoria proporcionada.

Para este propósito, se ha seleccionado la biblioteca de código abierto Orocos KDL. Esta biblioteca permite modelar, calcular la cadena cinemática y predecir los torques utilizando el solucionador de Dinámica Inversa (ID) basado en la formulación recursiva de Newton-Euler. La cadena cinemática del robot se compone de segmentos del manipulador, donde cada segmento integra tanto el eslabón como las articulaciones que conectan los eslabones.

Es importante destacar que el eje de la articulación número 5 es negativo, lo que implica que tanto el torque como la velocidad de esta articulación deben invertirse antes de enviar los torques del modelo al robot real.

En el modelado del robot, se crea una cadena cinemática precisa del KUKA youBot utilizando Orocos KDL. Cada segmento del robot se define con sus propiedades físicas específicas. Luego, se utiliza el solucionador de Dinámica Inversa de KDL para calcular los torques necesarios en cada articulación. Este cálculo se basa en la posición, velocidad y aceleración deseadas de las articulaciones.

Se incluye la compensación de los efectos gravitacionales en el cálculo de torques. Debido a la configuración negativa del eje de la articulación 5, se invierte el torque y la velocidad calculados para esta articulación. Además, se implementa un controlador PI en cascada, con un lazo externo para el control de posición y un lazo interno para el control de velocidad.

Se generan trayectorias suaves para cada articulación, considerando las limitaciones físicas del robot. Los torques calculados se envían a los motores de las articulaciones a través de la interfaz de control del youBot. Se utiliza la información de los encoders para cerrar el lazo de control, ajustando continuamente los torques aplicados.

Finalmente, se implementa una capa de seguridad que monitorea constantemente los límites de posición, velocidad y torque de cada articulación. Este enfoque permite un control preciso y robusto del manipulador KUKA youBot, adaptándose a las características específicas de cada articulación y compensando los efectos no lineales del sistema. EL diagrama de clases de esta librería se muestra en la Figura 14.

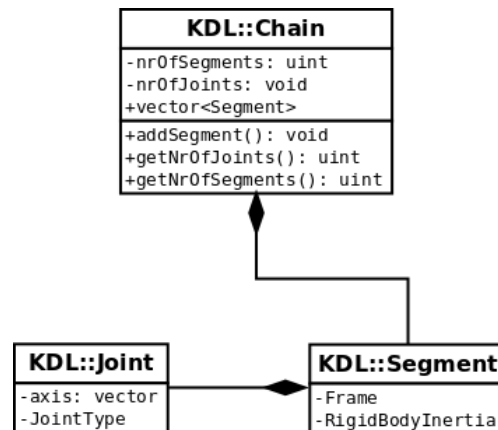


Figura 14. Cadena cinemática en Orocós KDL, que es una combinación tanto de la articulación y del eslabón

2.2.4. Controlador basado en modelos para un sistema no lineal

El controlador basado en modelos diseñado para el sistema no lineal del manipulador KUKA youBot se implementa en el espacio articular, una decisión crucial para abordar las complejidades de este robot. Este enfoque, ilustrado en la Figura 15, permite un control más directo sobre cada articulación individual, reduce la complejidad computacional en comparación con el control en el espacio operacional, y evita problemas asociados con singularidades cinemáticas. El trabajo utiliza un modo de control avanzado por articulación que comanda torques directamente a las articulaciones del manipulador, ofreciendo una precisión mejorada, facilitando la compensación de efectos dinámicos como la gravedad y la fricción, y permitiendo la implementación de estrategias de control más sofisticadas. La eficacia de este método depende crucialmente de la precisión de los parámetros del modelo dinámico y de la calidad de la información proporcionada por los codificadores en cada articulación.

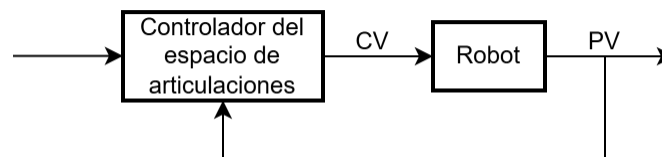


Figura 15. Controlador en el espacio articular

El proceso de control de torque, esquematizado en la Figura 16, implica varios pasos fundamentales. Comienza con la generación de la trayectoria deseada en el espacio articular, seguida del cálculo del modelo dinámico para predecir los torques necesarios. Se añade un torque adicional para contrarrestar los efectos de la gravedad, lo cual es crucial para mantener la posición del robot contra la fuerza de gravedad y mejorar la precisión del seguimiento de trayectorias. Se implementa un controlador PI en cascada para posición y velocidad, y finalmente se aplica un proceso de 'clamping' para mantener los torques dentro de límites seguros. Este último paso es vital para prevenir daños en los motores y la estructura del robot, y para manejar situaciones donde el modelo podría predecir torques irreales o peligrosos.

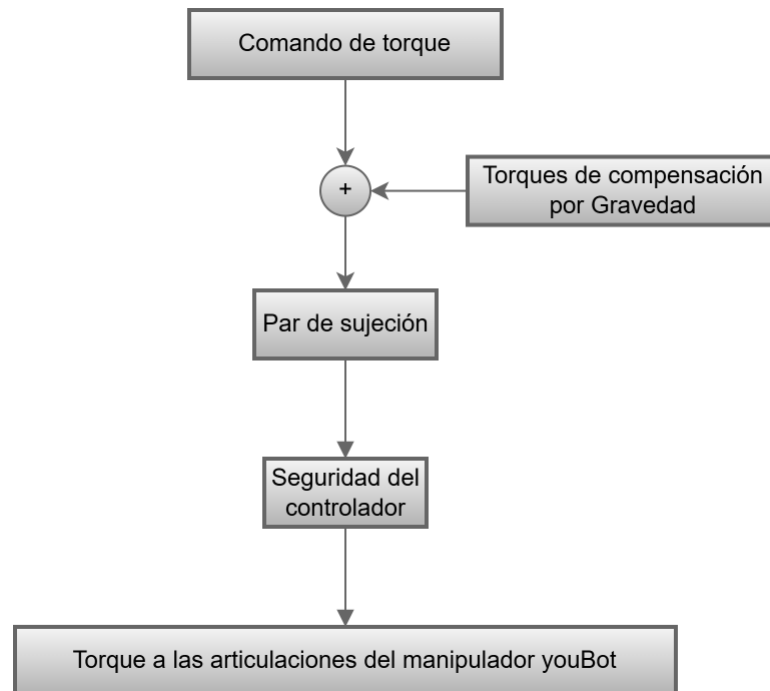


Figura 16. Modo de control de torque

La arquitectura del software, presentada en el diagrama de clases de la Figura 17, muestra una estructura modular que permite flexibilidad, mantenibilidad y extensibilidad del sistema de control. Los componentes clave incluyen la clase principal *Controlador PID* que coordina el control, *Modelo Dinámico* que encapsula el modelo dinámico del robot, *Control del par calculado* para la generación de

trayectorias deseadas, *Seguridad del Controlador* para implementar límites de seguridad, entre las más importantes. Esta arquitectura facilita la adaptación a diferentes configuraciones del robot, mantiene una separación clara de responsabilidades entre diferentes componentes, y permite la adición de nuevas funcionalidades o estrategias de control según sea necesario.

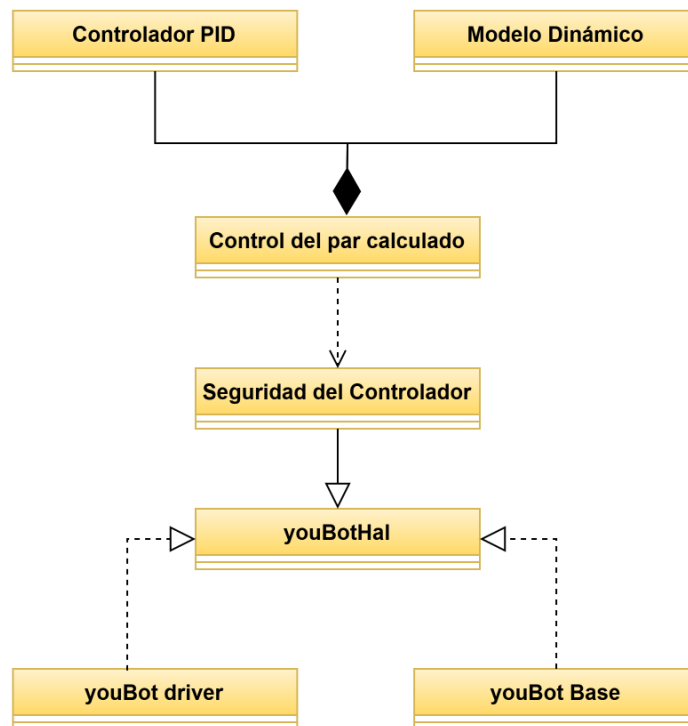


Figura 17. Diagrama de clases completo para el controlador basado en modelos implementado en este trabajo.

Este enfoque integral permite un control preciso y seguro del KUKA youBot, adaptándose a su naturaleza no lineal y aprovechando al máximo la información del modelo dinámico y los sensores disponibles. El controlador basado en modelos no solo mejora la precisión del seguimiento de trayectorias, sino que también proporciona una base sólida para futuras mejoras y optimizaciones. La combinación de un modelo dinámico preciso, compensación de gravedad, control en cascada y medidas de

seguridad resulta en un sistema de control robusto capaz de manejar las complejidades inherentes a los manipuladores robóticos avanzados como el KUKA youBot.

2.2.5 Control Compensación Gravitatoria

Es posible realizar el control de posición pura de robots controladores mediante el control PD, sin embargo, dicho control posee ciertas restricciones que pueden limitar su uso y no garantiza el cumplimiento del objetivo de control de posición pura de manipuladores cuyos modelos dinámicos contienen el termino de pares gravitacionales $g(q)$, a menos que la posición deseada q_d sea tal que $g(q_d) = 0$.

El control PD con Compensación gravitacional es capaz de satisfacer el objetivo de control de posición pura en forma global para robots de n grados de libertad (g.d.l) por lo que se convierte en un concepto muy importante para el control de robots, especialmente en brazos robóticos [34]. Dicha compensación hace alusión a la técnica usada para eliminar o compensar los efectos que ejerce la gravedad sobre las diferentes articulaciones y las juntas del robot, todo esto para lograr un movimiento eficiente y preciso, para su control se requiere el conocimiento previo de una parte del modelado dinámico del robot a ser controlado, puesto que usa el vector de pares gravitacionales $g(q)$ [47].

La ley de control PD con compensación gravitacional está dado por la ecuación (8) el cual está compuesto por una función proporcional y derivativa lineal, además de la compensación de gravedad [47], [48]:

$$\tau = K_p(\tilde{q}) + K_v(\dot{\tilde{q}}) + G(q) \quad (8)$$

K_p = Ganancia proporcional de posición

$(\tilde{q})$ = Error de posición

K_v = Ganancia derivativa de velocidad

$(\dot{\tilde{q}})$ = velocidad articular actual

$G(q)$ = Compensación de torque gravitacional

La principal diferencia con un controlador PD es que para este caso necesitamos conocer el vector de pares gravitacionales $g(q)$. Esto implica que sea necesario conocer el modelo estructural del robot. A continuación, en la Figura 18 se muestra el diagrama de bloques correspondiente al control PD con compensación de gravedad de robots [28].

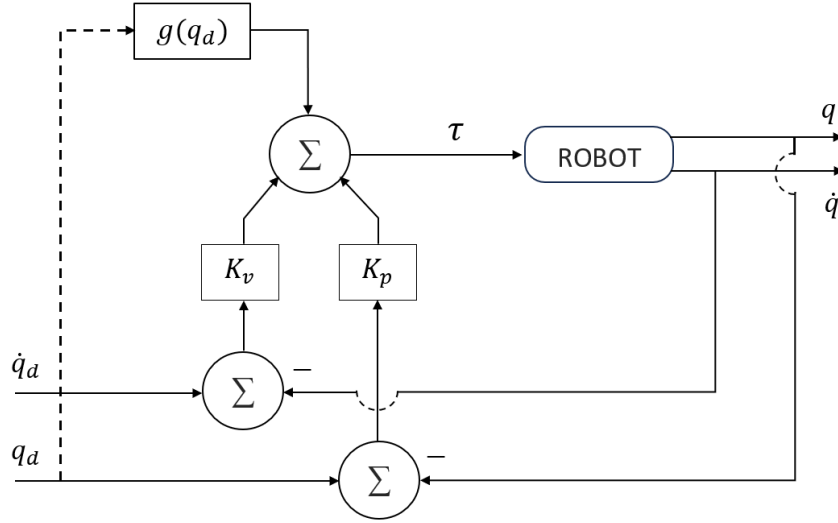


Figura 18. Diagrama de bloque control PD+G.

Fuente: Elaborado por el investigador

2.2.6 Enfoque Cinemático

Para tener claro el análisis a realizar cabe recalcar que el manipulador móvil Kuka YouBot consta de dos partes, una conformada por su chasis móvil y otra que consta de un brazo móvil manipulador de 5 grados de libertad, el análisis a realizar se centrará en este último, por lo que el estado del manipulador móvil estará expresado de forma general por [49]:

$$q = [q_a] \quad (9)$$

Donde el subíndice a representa las variables de posición de las articulaciones del brazo, $q_a = [q_1 \ q_2 \ \dots \ q_3]^T$ de un manipulador con n grados de libertad, al tratarse de un brazo manipulador de 5 grados de libertad, el estado del KUKA youbot se definirá de la siguiente manera:

$$q_a = [q_1 \quad q_2 \quad q_3 \quad q_4 \quad q_5]^T \quad (10)$$

Para definir de manera correcta los parámetros correspondientes al brazo manipulador KUKA youBot es necesario establecer un sistema de coordenadas sobre el mismo, el cual se presenta en la Figura 19. Donde se puede observar la posición de los diferentes ejes referenciales para cada una de las uniones.

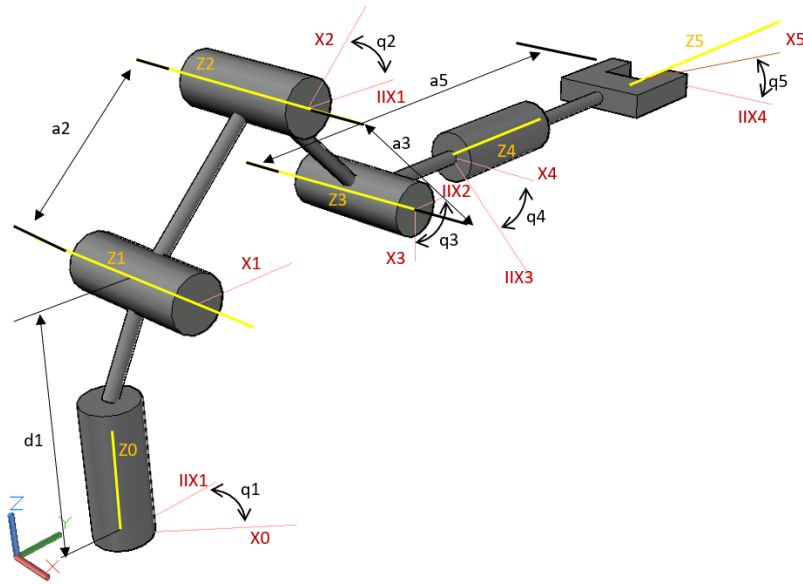


Figura 19. Diagrama Cinemático KUKA youBot.

Continuando con D-H, se procede a determinar los parámetros para el desarrollo de la matriz de transformación homogénea, teniendo en cuenta la Tabla 3, para ejemplificar como se realizaría su cálculo.

Tabla 3. Parámetros para Denavith- Hartenberg.

Fuente: Elaborado por el investigador

Eslabón	q_i	α_i	a_i	d_i
1	q_1	$-\pi/2$	0	d_1
2	q_2	0	a_2	0
3	q_3	$\pi/2$	a_3	0
4	q_4	$\pi/2$	0	0
5	q_5	0	0	d_5

Con los cuatro parámetros necesarios determinados para el desarrollo de D-H, podemos obtener la matriz homogénea, para esto también debemos tener en cuenta que 3 de los 4 parámetros determinados son constantes, por lo que dependen únicamente de su relación geometría entre cada articulación, mientras que el valor qi es una variable la cual hace referencia al ángulo de giro correspondiente al eje $Xi - 1$, alrededor del eje $Zi - 1$, hasta llegar a X .

Para lograr hallar la posición y orientación cartesiana del efector final del robot, conociendo sus variables de estado, nos centraremos en la cinemática directa mediante el uso del algoritmo de D-H la cual puede expresarse de manera general a través de la siguiente ecuación.

$$A_{i-1}^i = A_{R_z}(qi) * A_{T_z}(di) * A_{T_x}(ai) * A_{R_x}(\alpha i) \quad (11)$$

A partir de la resolución de la ecuación obtenemos la matriz A_{i-1}^i la cual será la utilizada para conocer la ubicación del efector final con respecto a la base de KUKA youBot.

$$A_{i-1}^i = \begin{bmatrix} \cos(qi) & -\sin(qi)\cos(\alpha i) & \sin(qi)\sin(\alpha i) & ai * \cos(qi) \\ \sin(qi) & \cos(qi)\cos(\alpha i) & -\cos(qi)\sin(\alpha i) & ai * \sin(qi) \\ 0 & \sin(\alpha i) & \cos(\alpha i) & di \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Establecida la matriz A_{i-1}^i y los datos de la tabla N°3 se procede a determinar las matrices de transformación homogénea para cada punto de unión, al tratarse un robot manipulador de 5 grados de libertad obteniendo 5 matrices.

$$A_1 = \begin{bmatrix} \cos(q1) & 0 & -\sin(q1) & 0 \\ \sin(q1) & 0 & \cos(q1) & 0 \\ 0 & -1 & 0 & d1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} \cos(q_2) & -\sin(q_2) & 0 & a_2 * \cos(q_2) \\ \sin(q_2) & \cos(q_2) & 0 & a_2 * \sin(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_3 = \begin{bmatrix} \cos(q_3) & 0 & \sin(q_3) & a_3 * \cos(q_3) \\ \sin(q_3) & 0 & -\cos(q_3) & a_3 * \sin(q_3) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} \cos(q_4) & 0 & \sin(q_4) & 0 \\ \sin(q_4) & 0 & -\cos(q_4) & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} \cos(q_5) & -\sin(q_5) & 0 & 0 \\ \sin(q_5) & \cos(q_5) & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Una vez obtenido las matrices para cada junta, se procede a calcular la matriz de transformación total, la cual conecta el sistema de ejes del efector final con el sistema de referencia el cual este situado en la base del robot.

$$T_5 = A_1 * A_2 * A_3 * A_4 * A_5 \quad (12)$$

$$A_5 = \begin{bmatrix} \cos(q_1 + q_2 + q_3 + q_4) & -\sin(q_1 + q_2 + q_3 + q_4) & 0 & (a_2 * \cos(q_2) + a_3 * \cos(q_3) + d_1)\cos(q_1) \\ \sin(q_1 + q_2 + q_3 + q_4) & \cos(q_1 + q_2 + q_3 + q_4) & 0 & (a_2 * \cos(q_2) + a_3 * \cos(q_3) + d_1)\sin(q_1) \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2.2.7. Modelo Dinámico

Para el desarrollo del control PD + G, se debe tomar en cuenta ciertas restricciones las cuales parten desde el supuesto de que la dinámica como tal en los actuadores no existe y esto sucede porque la señal que controla los actuadores no es el par, sino más bien es el voltaje el cual es aplicado a los mismos.

Omitiendo perturbaciones externas la dinámica general para un robot de n eslabones se puede escribir como:

$$\tau = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + F(\dot{q}) \quad (13)$$

Donde:

τ = Son vectores de $n \times 1$ de desplazamiento, velocidad y aceleración de las articulaciones.

$q, \dot{q}, \ddot{q}$ = Vector $n \times 1$ de los torques de entrada

$M(q)$ = Matriz $n \times n$ de inercia del manipulador

$C(q, \dot{q})$ = Matriz de $n \times n$ de torques centrípetos y de Coriolis

$g(q)$ = Vector $n \times 1$ de torques gravitacionales

$F(\dot{q})$ = Vector $n \times 1$ de los torques de fricción

Siendo:

$$\tau = K_p \tilde{q} + K_v \dot{\tilde{q}} + g(q) \quad (14)$$

Reemplazamos (11) en (12) y tenemos:

$$K_p \tilde{q} + K_v \dot{\tilde{q}} + g(q) = M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + F(\dot{q}) \quad (15)$$

Se procede a derivar el termino de gravedad $g(q)$ mediante ecuaciones dinámica de Lagrange, considerando un manipulador robótico de n eslabones, para derivar el termino de gravedad $g(q)$, primero debemos calcular la energía cinética, la energía potencial y la lagrangiana del sistema, por lo que se usa las ecuaciones dinámicas de Lagrange para obtener los torques en cada articulación para finalmente derivar el vector de fuerzas gravitacionales $g(q)$

La energía cinética de un manipulador móvil se puede expresar como la suma de la energía cinética de cada uno de sus eslabones, tanto su energía traslacional y la rotacional.

$$K = \frac{1}{2} \sum_{i=1}^n m_i \dot{c}_i^T \dot{c}_i + \frac{1}{2} \sum_{i=1}^n \dot{\theta}_i^T I_i \dot{\theta}_i \quad (16)$$

Donde:

m_i = Es la masa de i-esimo eslabon

$\dot{c}_i$ = Es la velocidad del centro de masa del i-ésimo eslabón

I_i = es el tensor de inercia del i-ésimo eslabón

$\dot{\theta}_i$ = es la velocidad angular del i-ésimo eslabón

La energía potencia de un brazo manipulador no es más que la suma de las energías potenciales de cada eslabón y se expresa de la siguiente manera

$$U = \sum_{i=1}^n m_i g^T c_i \quad (17)$$

Donde:

g es el vector de aceleración gravitacional

c_i es la posición del centro de masa de i-esimo eslabón

Considerando un manipulador de n grados de libertad la ecuación dinámica de Lagrange se expresa como la diferencia entre la energía cinética y la energía potencial, de la siguiente manera:

$$L(q(t), \dot{q}(t)) = K(q(t), \dot{q}(t)) - U(q(t)) \quad (18)$$

Donde

$L(q, \dot{q})$ La lagrangiana

$K(q, \dot{q})$ Energía cinética del robot

$U(q)$ Energía potencial del robot

Por lo que el torque para cada articulación se calcula mediante la siguiente formula

$$\frac{d}{dt} \left[\frac{\partial L(q, \dot{q})}{\partial \dot{q}_i} \right] - \frac{\partial L(q, \dot{q})}{\partial q_i} = \tau_i \quad (19)$$

$$i = 1, \dots, n$$

Una vez se tiene los valores de torque, se reagrupa las ecuaciones de Lagrange, para así poder encontrar el vector de las fuerzas gravitatorias $g(q)$, por lo que la ecuación dinámica para el robot manipulador se representa en la siguiente ecuación simplificada en la cual se ha excluido la última articulación, para reducir el problema a un sistema de 4 g.d.l.

$$\tau \begin{bmatrix} 4 \times 1 \\ \text{vector} \end{bmatrix} = Kp \begin{bmatrix} 4 \times 4 \\ \text{matriz} \\ \text{diagonal} \\ \text{positiva} \end{bmatrix} \tilde{q} + Kd \begin{bmatrix} 4 \times 4 \\ \text{matriz} \\ \text{diagonal} \\ \text{positiva} \end{bmatrix} (\dot{\tilde{q}}) + g \begin{bmatrix} 4 \times 1 \\ \text{vector} \end{bmatrix}$$

Todos los parámetros de cada enlace del robot como masa, torque, inercia, etc. Los cuales se usan para calcular el vector $g(q)$ se encuentran en la página oficial de youBot tabla 4.

Tabla 4. Características de KUKA youBot

Parámetros	Eslabón 1	Eslabón 2	Eslabón 3	Eslabón 4	Eslabón 5
Voltaje nominal (V)	24	24	24	24	24
Corriente nominal (A)	2.32	2.32	2.32	1.07	0.49
Torque Nominal (mNm)	82.7	82.7	82.7	58.8	-
Inductancia Terminal (mH)	0.573	0.573	0.573	2.24	7.73
Resistencia Terminal (Ω)	0.978	0.978	0.978	4.48	13.7
Momento de Inercia (kg*mm²)	13.5	13.5	13.5	9.25	3.5
Peso (g)	110	110	110	75	46
Rango de velocidad (RPM)	5250	5250	5250	2850	2800

2.2.8. Modelado y Compensación de Fricción en Control Robótico

El modelado y la compensación de fricción son aspectos cruciales en el control preciso de sistemas robóticos, especialmente en manipuladores como el KUKA youBot. La fricción, un fenómeno complejo y no lineal, afecta significativamente el rendimiento

del robot, particularmente en tareas que requieren movimientos precisos o cambios de dirección frecuentes. En sistemas de control robótico, la fricción puede causar errores de seguimiento, limitar la precisión de posicionamiento y generar ciclos límite, que son oscilaciones persistentes alrededor de una posición deseada. Estos efectos son especialmente pronunciados en movimientos a baja velocidad y en cambios de dirección, situaciones comunes en operaciones de manipulación delicada.

El modelo de fricción utilizado en este trabajo considera dos componentes principales: la fricción estática y la fricción cinética o dinámica. La fricción estática es la fuerza que debe superarse para iniciar el movimiento desde el reposo, mientras que la fricción cinética actúa durante el movimiento continuo. La transición entre estos dos estados es crucial y se modela mediante el efecto Stribeck, que captura la disminución de la fricción justo después de que comienza el movimiento. Este modelo se puede representar matemáticamente como se observa en la ecuación (20):

$$\tau_f = \begin{cases} \tau_s \cdot \text{sgn}(\dot{q}_d) & \text{si } |\dot{q}| \leq \dot{q}_{\text{threshold}} \\ [\tau_c + (\tau_s - \tau_c)e^{-(|\dot{q}|/v_s)^\delta}] \cdot \text{sgn}(\dot{q}) + F_v \dot{q} & \text{si } |\dot{q}| > \dot{q}_{\text{threshold}} \end{cases} \quad (20)$$

Donde τ_f es el torque de fricción, τ_s es la fricción estática máxima, τ_c es la fricción de Coulomb, v_s es la velocidad de Stribeck, δ es un parámetro de forma, y F_v es el coeficiente de fricción viscosa. Esta formulación captura el comportamiento no lineal de la fricción en diferentes regímenes de velocidad.

La importancia de un modelo de fricción preciso en el control robótico radica en su capacidad para mejorar significativamente el rendimiento del sistema. Sin compensación de fricción, el controlador debe trabajar continuamente contra estas fuerzas no modeladas, lo que resulta en un mayor consumo de energía, desgaste mecánico acelerado y reducción de la precisión. En aplicaciones que requieren alta precisión, como el ensamblaje de piezas pequeñas o la cirugía robótica, la fricción no compensada puede ser la diferencia entre el éxito y el fracaso de la tarea.

En la práctica, la fricción de Coulomb se utiliza a menudo para modelar la fricción cinética, donde el torque es constante independientemente del valor de velocidad de entrada, como se muestra en la Figura 20:

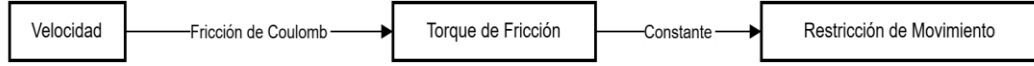


Figura 20. Modelo simplificado de la fricción de Coulomb en sistemas robóticos

La implementación de la compensación de fricción en el controlador se realiza típicamente añadiendo un término de feedforward basado en el modelo de fricción. Este término se calcula utilizando la velocidad deseada y se suma al torque de control calculado por el controlador principal:

$$\tau_{comando} = \tau_{control} + \tau_f(\dot{q}_d) \quad (21)$$

Donde $\tau_{comando}$ es el torque total comandado a los actuadores, $\tau_{control}$ es el torque calculado por el controlador principal (por ejemplo, PID o control basado en modelo), y $\tau_f(\dot{q}_d)$ es el torque de fricción estimado basado en la velocidad deseada.

Para implementar esta compensación, se utilizan algoritmos que determinan la dirección de la velocidad deseada y calculan el torque de fricción apropiado. Estos algoritmos tienen en cuenta tanto la fricción estática como la cinética, aplicando el torque adecuado dependiendo del estado de movimiento de la articulación (Ver Figura 21 y Figura 22):

Algorithm 2 Cálculo de velocidad deseada

```

1: function DIRECCIONVELOCIDADDESEADA( $\dot{q}_d$ )
Entrada:  $\dot{q}_d$ : Velocidad deseada
Salida: escala: Dirección de la velocidad
2:   if  $\dot{q}_d < 0$  then
3:     escala  $\leftarrow -1$ 
4:   return escala
5:   else if  $\dot{q}_d > 0$  then
6:     escala  $\leftarrow 1$ 
7:   return escala
8:   else
9:     escala  $\leftarrow 0$ 
10:  return escala
11: end if
12: end function
  
```

Figura 21. Cálculo de Velocidad basada en Fricción

Algorithm 1 Observador de Fricción

```

1: function MODELOFRICCION( $\dot{q}_d, \dot{q}_m$ )
Entrada:  $\dot{q}_d$ : Velocidad deseada,  $\dot{q}_m$ : Velocidad medida
Entrada:  $n \leftarrow 5$ , friccionEstatica, friccionCinetica
Salida:  $\tau_f$ : Torque de fricción
2:   if  $|\dot{q}_d| > 0.0$  then
3:     for numeroArticulacion  $\leftarrow 0$  to  $n - 1$  do
4:       if  $|\dot{q}_m[\text{numeroArticulacion}]| \leq 0.01$  then // La articulación no se está
moviendo
5:          $\tau_s \leftarrow \text{friccionEstatica}[\text{numeroArticulacion}] \cdot$ 
 $\text{signo}(\dot{q}_d[\text{numeroArticulacion}])$ 
6:          $\tau_f \leftarrow \tau_s$ 
7:       else
8:          $\tau_k \leftarrow \tau_c \cdot \text{signo}(\dot{q}_d[\text{numeroArticulacion}])$  // Modelado de fricción de
Coulomb
9:          $\tau_f \leftarrow \tau_k$  // Torque de fricción cinética
10:      end if
11:    end for
12:  end if
13:  return  $\tau_f$ 
14: end function

```

Figura 22. Observador de Fricción

Es importante notar que el modelado preciso de la fricción es un desafío debido a su naturaleza variable y dependiente de múltiples factores como la temperatura, el desgaste y las condiciones ambientales. Por lo tanto, en aplicaciones avanzadas, se pueden emplear técnicas de estimación adaptativa o aprendizaje en línea para ajustar continuamente los parámetros del modelo de fricción durante la operación del robot. Estas técnicas permiten una compensación más robusta y adaptable a las condiciones cambiantes, mejorando aún más el rendimiento y la precisión del sistema de control robótico.

2.4 Estrategia de Control de Trayectoria

Para iniciar con el control se debe obtener o estimar la posición, la velocidad y la aceleración de cada articulación para determinar la trayectoria del movimiento del brazo robótico, para el presente proyecto de investigación se consideró una aceleración constante para todas las articulaciones.

La trayectoria descrita genera una trama de puntos que pertenecen a la línea que dibuja el desplazamiento de cada eslabón, cada trama origina una posición y una orientación necesaria para conocer la ruta de la trayectoria. Así como, la información de las

restricciones establecidas por los límites de movimiento de cada articulación. Para el control del planificador de trayectoria en la figura 23, se presenta el bloque que describe el control que se va a utilizar.

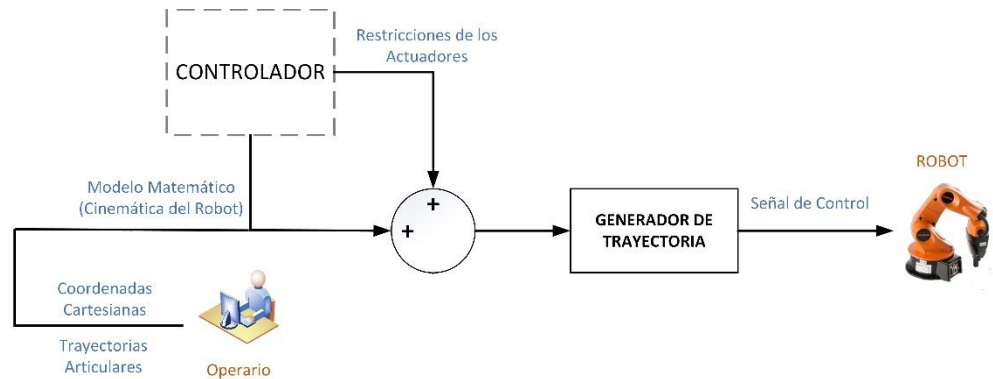


Figura 23. Bloque de control del planificador de trayectoria.
Fuente: Elaborado por el investigador

2.4.1 Espacio de Articulaciones

Debido a la ausencia de obstáculos y la trayectoria punto a punto que seguirá el robot para el presente proyecto de investigación, la estrategia que mejor se desempeña en función de los ángulos de articulaciones es la generada por el espacio de articulación. Para lograr un control de trayectoria óptimo se procede con el proceso presentado en la Figura 24, el cual describe el método, acondicionado para las necesidades requeridas.

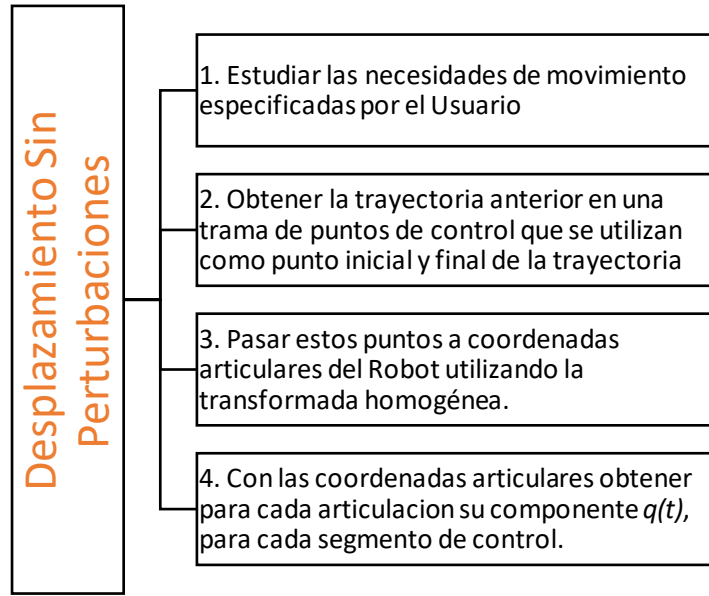


Figura 24. Proceso de Aplicación para una Planeación de Trayectoria Óptimo.

Fuente: Elaborado por el investigador

Al aplicar el control de trayectoria es necesario una función uniforme con cuatro restricciones sobre $\theta(t)$. De tal forma, para articulaciones con ángulos θ_i en $t_i = 0$ y θ_f en $t = t_f$, inicial y final respectivamente. Se puede deducir:

$$\theta(t_i) = \theta(0) = \theta_i$$

$$\theta(t_f) = \theta_f$$

Siendo las anteriores las dos primeras restricciones de posición, y para las restricciones faltantes, que corresponde a condiciones sobre la velocidad inicial y final, se deduce:

$$\dot{\theta}_i = \dot{\theta}(0) = 0$$

$$\dot{\theta}(t_f) = \dot{\theta}_f = 0$$

Para lograr un mejor control a la hora de generar la trayectoria se puede aumentar el número de restricciones, para lo cual, se puede considerar condiciones sobre la aceleración inicial y final, teniendo entonces:

$$\ddot{\theta}_i = \ddot{\theta}(0) = 0$$

$$\ddot{\theta}(t_f) = \ddot{\theta}_f = 0$$

Con las restricciones consideradas se satisface un polinomio de mínimo de quinto grado para que cumpla con la solución de las 6 incógnitas que requiere el KUKA youBot. Dicho polinomio determina su posición, velocidad y aceleración, presentados respectivamente, a continuación:

$$\text{Posición: } \theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

$$\text{Velocidad: } \dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4$$

$$\text{Aceleración: } \ddot{\theta}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3$$

Finalmente, para que el movimiento de las articulaciones tenga una aceleración inicial y una desaceleración hasta la detención del efector final del robot al punto deseado, se arma la matriz de 6 incógnitas del sistema para una planeación de trayectoria mediante las restricciones establecidas, dando como resultado el siguiente sistema matricial:

$$\begin{bmatrix} \theta(t_i) \\ \theta(t_f) \\ \dot{\theta}(t_i) \\ \dot{\theta}(t_f) \\ \ddot{\theta}(t_i) \\ \ddot{\theta}(t_f) \end{bmatrix} = \begin{bmatrix} t_i^5 & t_i^4 & t_i^3 & t_i^2 & t_i & 1 \\ 5t_i^4 & 4t_i^3 & 3t_i^2 & 2t_i & 1 & 0 \\ 20t_i^3 & 12t_i^2 & 6t_i & 2 & 0 & 0 \\ t_f^5 & t_f^4 & t_f^3 & t_f^2 & t_f & 1 \\ 5t_f^4 & 4t_f^3 & 3t_f^2 & 2t_f & 1 & 0 \\ 20t_f^3 & 12t_f^2 & 6t_f & 2 & 0 & 0 \end{bmatrix} * \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} \quad (22)$$

2.5 Estructura de Software

La estructura de la programación de control es flexible y de librería abierta, permite añadir nuevos componentes sobre el efector final, de igual forma, al ser Open Source, puede ser ejecutado en diferentes dispositivos, aliviando el esfuerzo al implementar y repotenciar el algoritmo de control para evaluar la información de los sensores de fuerza y sensores de posición. En la Figura 25, se presenta la estructura de software que se maneja para el presente proyecto de investigación.

La estructura de software está distribuida en cuatro partes.

La primera parte es el inicio del Programa, este inicia al arrancar el programa en la SBC, luego se procede a establecer los parámetros iniciales de velocidad y los parámetros de comunicación para finalmente, iniciar la librería de comunicación con el hardware KUKA youBot, establecer la configuración de los pines GPIO y arrancar los actuadores e inicializar los sensores.

La segunda parte es el establecimiento de la trayectoria deseada; en esta parte se define la trayectoria que el youBot debe seguir, ya sea, a través de la lectura de una matriz de puntos secuenciales desde un archivo o generando la trayectoria dinámicamente mediante la aplicación de fuerza externa, como es el caso de estudio.

La tercera parte de la estructura la compone el bucle principal el cual permite ejecutar, cíclicamente, las ordenes necesarias para el control del hardware KUKA youBot en tiempo real; esta parte inicia con una verificación del estado operativo de los sensores y el funcionamiento correcto de los actuadores, luego se procede con la recopilación de los datos de los encoders y giroscopios de cada articulación/eslabón del Hardware KUKA youBot, estos datos son procesados para proporcionar la posición y orientación precisa de cada articulación/eslabón del robot, a continuación, se calcula la compensación gravitatorio, mediante el modelo matemático dentro del algoritmo, para cada articulación del hardware KUKA youBot; en el proceso final de la tercera parte, se ajustan y se envían las señales de control hacia los motores considerando la compensación gravitatoria y la trayectoria deseada, incluyendo un sistema de monitoreo, en tiempo real, del estado de los actuadores y sensores, con el fin de ajustar la señal de control de ser necesario.

Para culminar, el algoritmo establece una condición para la finalización de la estructura de software, que comprueba si se alcanzó el punto final de la trayectoria, en el caso de que no se alcance el punto final el programa retorna a la verificación del estado de los actuadores y los sensores; pero en el caso de que, si alcance el punto final, el programa pasa a la siguiente parte de la estructura.

La cuarta parte de la estructura es la finalización del programa, este se encarga de almacenar los datos de los sensores y actuadores durante la ejecución para un análisis

posterior de ser necesario, luego cierra las conexiones con los componentes (sensores y actuadores) del hardware Kuka youBot y finalmente, se apaga o se reinicia el sistema.

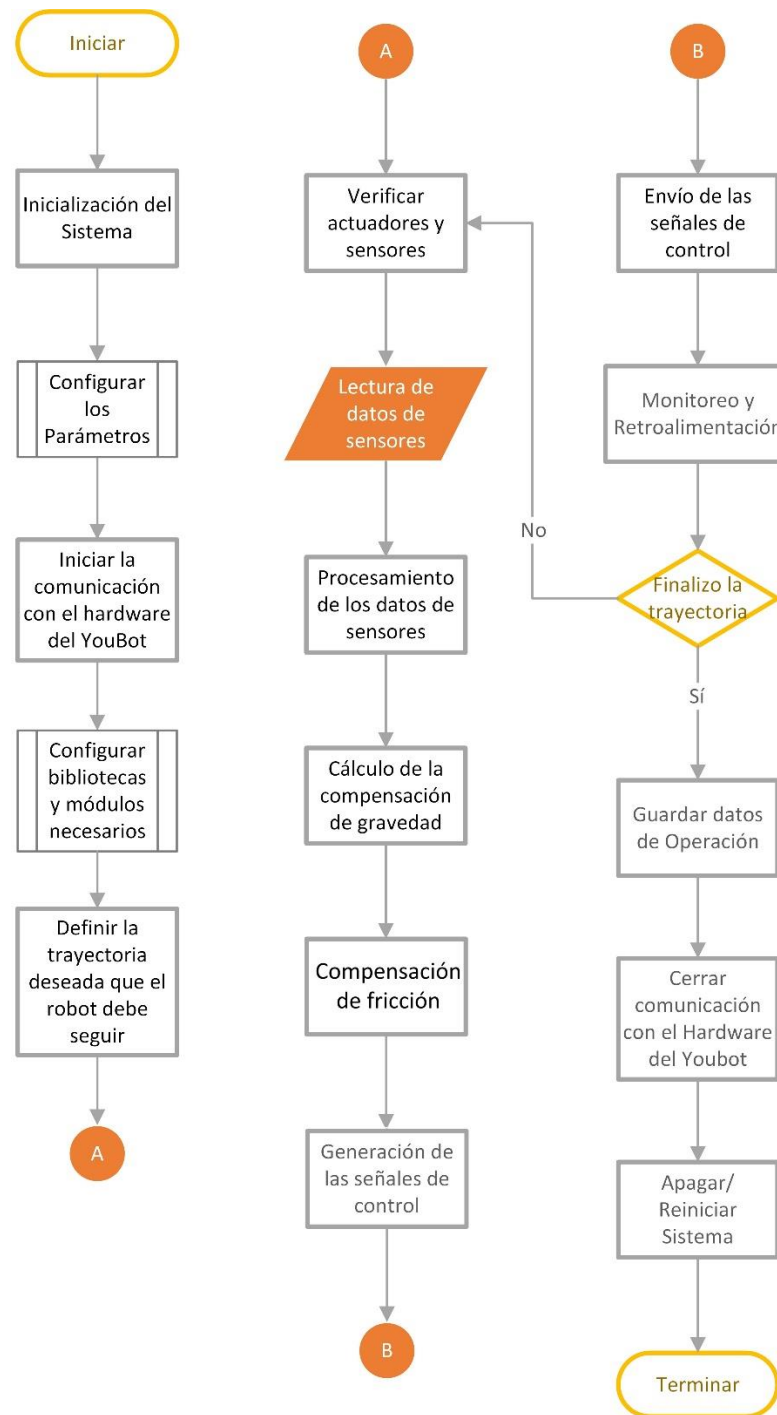


Figura 25. Diagrama de Flujo de la Estructura del Software.

2.5.1 Estructura de Archivos

La estructura de archivos se compone del paquete principal, donde se incluye obligatoriamente, el archivo *package.xml* con la metainformación del paquete, el archivo *CMakeLists.txt* que usa catkin, cada una en su propia carpeta. El algoritmo se aloja en una carpeta separada *src/* una vez corrido el CMake y construida; de igual forma, el ejecutable *youbot_control* con el *control_driver*, en una carpeta separada. En la Figura 26, se puede observar la estructura del paquete de datos desarrollado.

```
youbot_trajectory_control/  
├─ CMakeLists.txt  
├─ include/  
│   └─ youbot_trajectory_control/  
│       └─ trajectory_controller.h  
├─ src/  
│   ├── trajectory_controller.cpp  
│   └─ main.cpp  
├─ launch/  
│   └─ youbot_control.launch  
└─ package.xml
```

Figura 26. Estructura de datos para el desarrollo

Recordar que es determinante tener en cuenta que, si bien el algoritmo dentro de la estructura puede ser ejecutado en diferentes dispositivos, existen se debe cumplir con los requisitos básicos para su correcta ejecución:

- `ros-packages-desktop-full`: Paquete de instalación de ROS según el dispositivo
- `ros-youbot-driver`: Paquete ROS para controlar el KUKA YouBot.
- `ros-control` y `ros-controllers`: Paquetes para los controladores.

2.6. Desarrollo de la Propuesta

2.6.1 Creación del Entorno de Comunicación ROS

La estructura de comunicación del sistema se construye sobre tres niveles de intervención, en cada nivel se ejecuta un nodo ROS, que será encargado de gestionar

los datos de control y monitoreo de las variables de posición, velocidad y esfuerzos. La estructura se base en una arquitectura general propuesta para la aplicación de los componentes principales Figura 27, a la cual se le añade el nodo de intervención de planificación para el control de trayectoria.

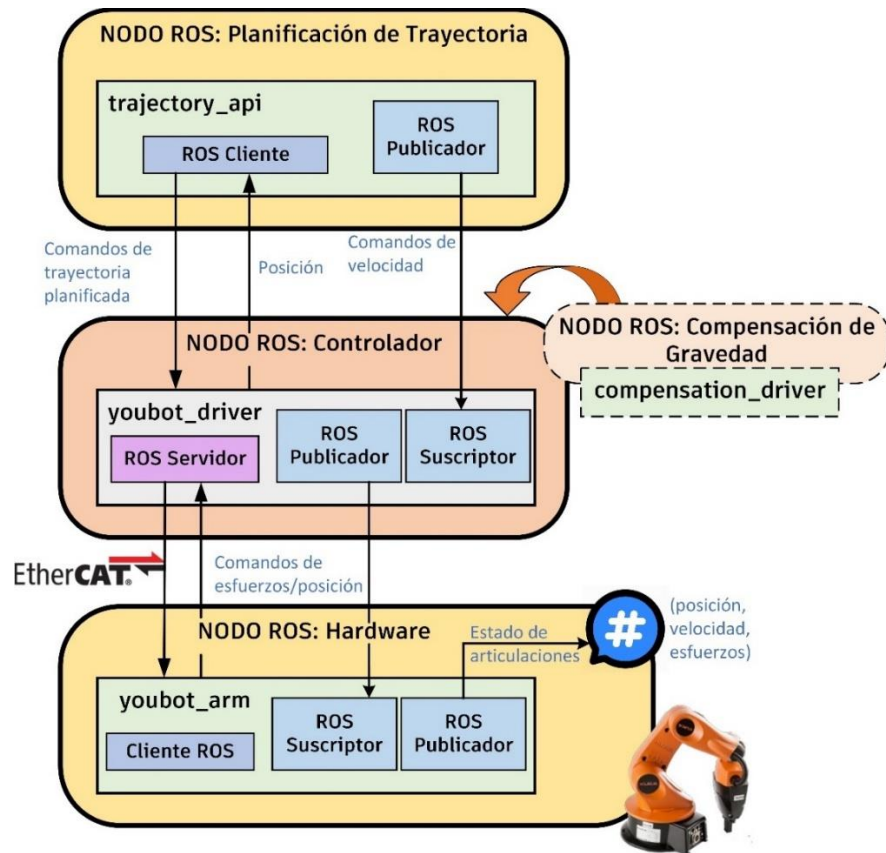


Figura 27. Estructura de comunicación de Nodos ROS

Para el proceso de comunicación de los Nodos ROS creados, se establecen cinco procesos: proceso de generación de trayectoria deseada, proceso de recepción de trayectoria, proceso de control PD con compensación de gravedad, proceso de aplicación de comandos, y el proceso de actualización de estados. En el proceso de generación de trayectorias deseadas, el *Nodo de Planificación de Trayectoria* calcula los valores de la trayectoria deseada para cada una de las articulaciones y publica un mensaje de tipo `trajectory_msgs/JointTrajectory` con los valores calculados en el Topic: `/trajectory`.

En el proceso de recepción de trayectorias, el *Nodo Controlador* se suscribe al Topic: `/trajectory`, para recibir la trayectoria deseada e implementar el control PD con compensación de gravedad. Si bien el *Nodo de Compensación de Gravedad* puede ser establecido como un nodo separado, lo contemplamos como un nodo integrado al *Nodo Controlador*.

En el proceso de control PD con compensación de gravedad, dentro del *Nodo Controlador*, se estima el error resultante de la diferencia entre la posición deseada y la posición actual de cada articulación y se calcula el esfuerzo necesario para corregir el error mediante el control PD y las ganancias proporcionales y derivativas. Terminando el cálculo de esfuerzos se utiliza la función integrada al *Nodo de Compensación de Gravedad*, para obtener los torques necesarios para compensar la fuerza de gravedad y se suma a los esfuerzos calculados.

Finalmente, se publica los comandos de esfuerzo/posición en el Topic: `/joint_commands`.

En el proceso de aplicación de comandos, el *Nodo Hardware* se suscribe al Topic: `/joint_commands` y aplica los comandos recibidos a cada una de las articulaciones del robot. Para culminar el proceso el *Nodo Hardware* publica el estado actual de posición, velocidad y esfuerzos, de cada una de las articulaciones en el Topic: `/joint_states`.

En el proceso de actualización de comandos, los estados publicados en el Topic: `/joint_states` son adquiridos y enviados al *Nodo Controlador* para la retroalimentación de la próxima interacción de control. Finalmente, los estados adquiridos quedan a disposición de otros Nodos para el monitoreo del rendimiento del Hardware Kuka Youbot.

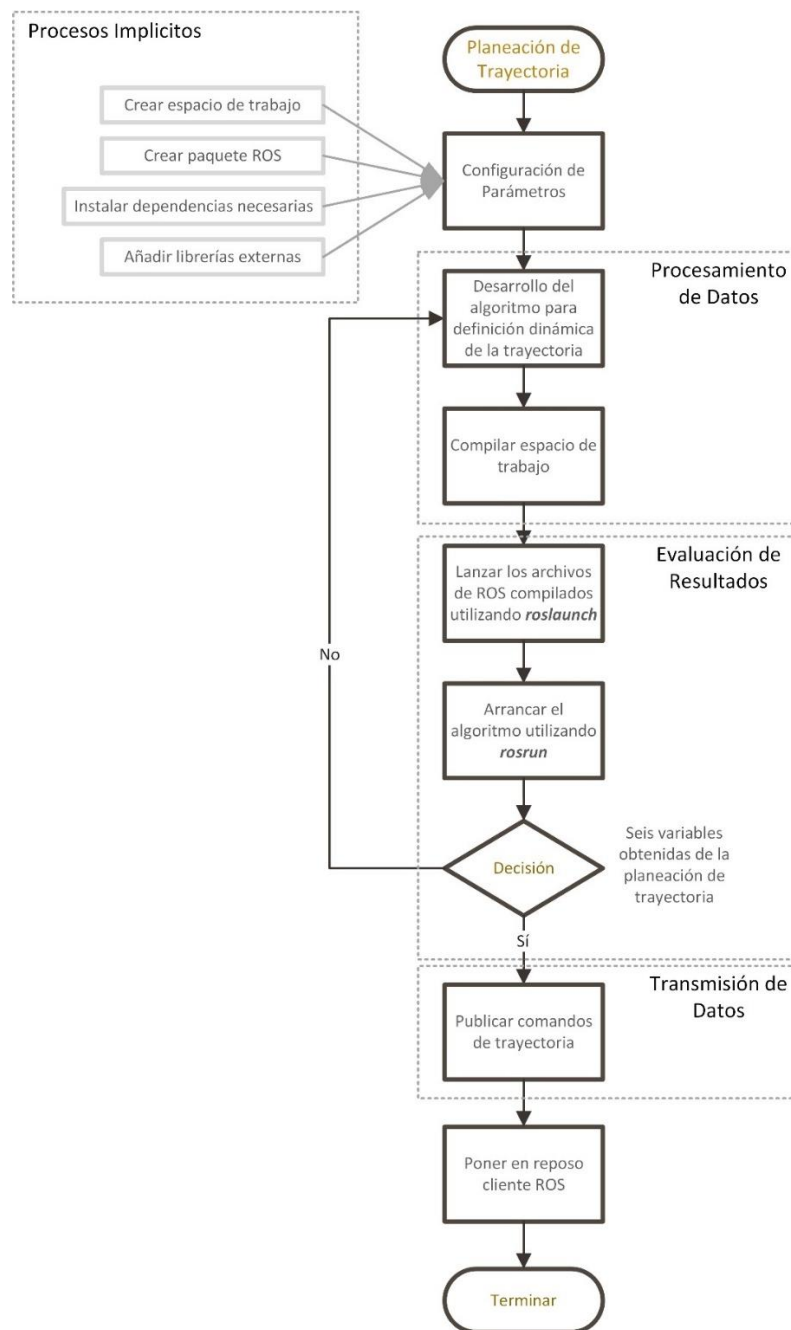


Figura 28. Desarrollo del algoritmo de planeación de trayectoria

Fuente: Elaborado por el investigador

2.6.2 Algoritmo de Planeación de Trayectoria.

Este método se basa en un mecanismo de retroalimentación que permite linealizar el sistema, facilitando la aplicación de estrategias de control lineal. En el corazón de este enfoque se encuentra el modelo dinámico del robot, que describe matemáticamente cómo las fuerzas y torques aplicados afectan al movimiento del robot. La ecuación general del modelo dinámico se expresa como $\tau = M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + F(\dot{q})$, donde τ es el vector de torques aplicados, $M(q)$ es la matriz de inercia del robot, $C(q,\dot{q})$ representa los términos de Coriolis y centrífugos, $G(q)$ es el vector de fuerzas gravitacionales, y $F(\dot{q})$ representa las fuerzas de fricción. Este modelo es fundamental para predecir y compensar las fuerzas que actúan sobre el robot durante su movimiento.

Para implementar el modelo dinámico y los cálculos cinemáticos necesarios, se utiliza la librería Orocos KDL (Kinematics and Dynamics Library). Esta potente herramienta de código abierto proporciona funcionalidades para cálculos de cinemática directa e inversa, dinámica directa e inversa, generación y evaluación de trayectorias, así como representación y manipulación de cadenas cinemáticas. En la implementación para el robot KUKA youBot, KDL juega un papel crucial en la construcción del modelo dinámico y en el cálculo de los torques necesarios en tiempo real. La versatilidad de KDL permite una representación precisa de la estructura cinemática del robot y facilita los complejos cálculos necesarios para el control de torque computado.

La fricción es un fenómeno no lineal que puede afectar significativamente el rendimiento del control, por lo que su modelado y compensación son aspectos críticos del enfoque. Utilizamos un modelo que combina fricción estática y dinámica (Coulomb), expresado en la ecuación (23)

$$F(\dot{q}) = \begin{cases} F_s \cdot \text{sign}(\dot{q}) & \text{si } |\dot{q}| \leq \dot{q}_{\text{umbral}} \\ F_c \cdot \text{sign}(\dot{q}) & \text{si } |\dot{q}| > \dot{q}_{\text{umbral}} \end{cases} \quad (23)$$

Donde F_s es la fricción estática y F_c es la fricción de Coulomb. La compensación se realiza añadiendo el torque de fricción estimado al torque de control: $\tau_{\text{comando}} = \tau_{\text{control}} + F(\dot{q})$. Esta estrategia de compensación de fricción se implementa mediante

el Algoritmo 2: Observador de Fricción, que estima y compensa los efectos de la fricción en tiempo real, mejorando significativamente la precisión del control.

La estructura del controlador de torque computado se implementa como una cascada que integra varios componentes clave. El controlador principal coordina un controlador de posición, un controlador de velocidad, el modelo dinámico del robot y el observador de fricción. El control se implementa en dos etapas principales: el control de posición, que genera comandos de velocidad basados en el error de posición ($\dot{q}_d = \dot{q}_e \cdot K_p + \int \dot{q}_e \cdot K_i dt$), y el control de velocidad, que genera comandos de torque basados en el error de velocidad ($\tau_{\text{controlador}} = \text{límite}(\dot{q}_e \cdot K_p + \int \dot{q}_e \cdot K_i dt)$). El torque final aplicado al robot es la suma del torque del controlador, el torque de compensación de gravedad calculado por el modelo dinámico, y el torque de compensación de fricción: $\tau = \tau_{\text{controlador}} + \tau_g(q) + F(\dot{q})$.

Esta estructura de control, que combina un modelo dinámico preciso, compensación de fricción y una estrategia de control en cascada, permite un seguimiento de trayectoria preciso y robusto para el robot KUKA youBot. La integración de estos elementos permite al sistema manejar eficazmente las no linealidades inherentes al robot y las perturbaciones externas, resultando en un rendimiento de control superior en una variedad de tareas y condiciones operativas. La flexibilidad de este enfoque también facilita la adaptación y optimización del controlador para diferentes configuraciones del robot o cambios en las condiciones de operación, haciendo de este un método versátil y potente para el control de robots manipuladores. Ver Figura 29.

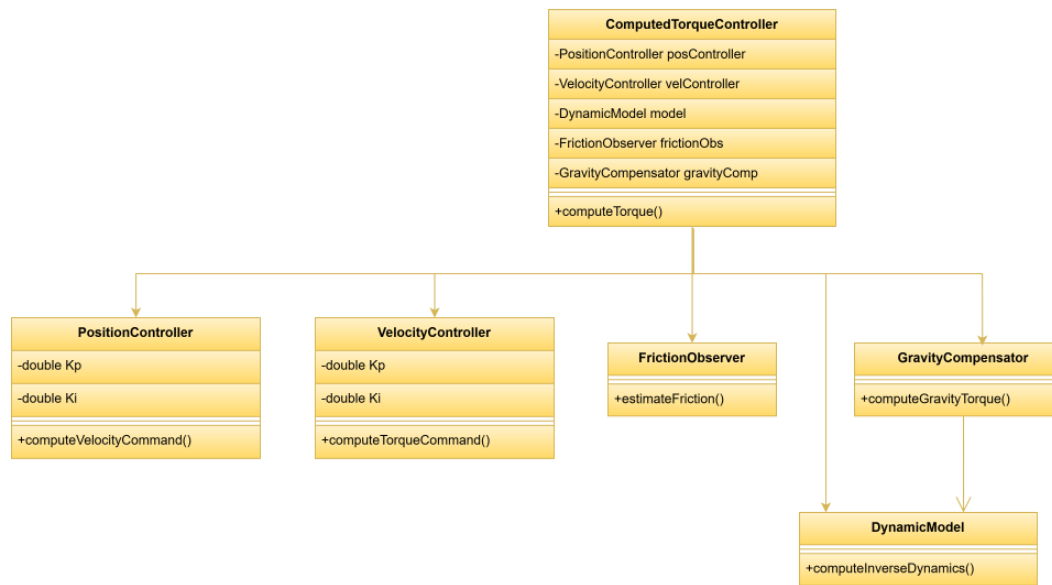


Figura 29. Diagrama de Clases de Aplicación Implementada

2.6.3 Algoritmo de Control PD con Compensación de Gravedad.

Este algoritmo se encarga en el *Nodo ROS: Controlador*, de suscribirse al *Topic* de *Trayectoria* para recibir los datos de trayectoria e implementar un control de corrección de error PD, aplicar la compensación gravitatoria y publicar los comandos calculados en un *Topic* de control. El proceso de construcción se detalla en la Figura 30. Donde se puede apreciar las configuraciones de los parámetros previamente necesarias al desarrollo del algoritmo de control.

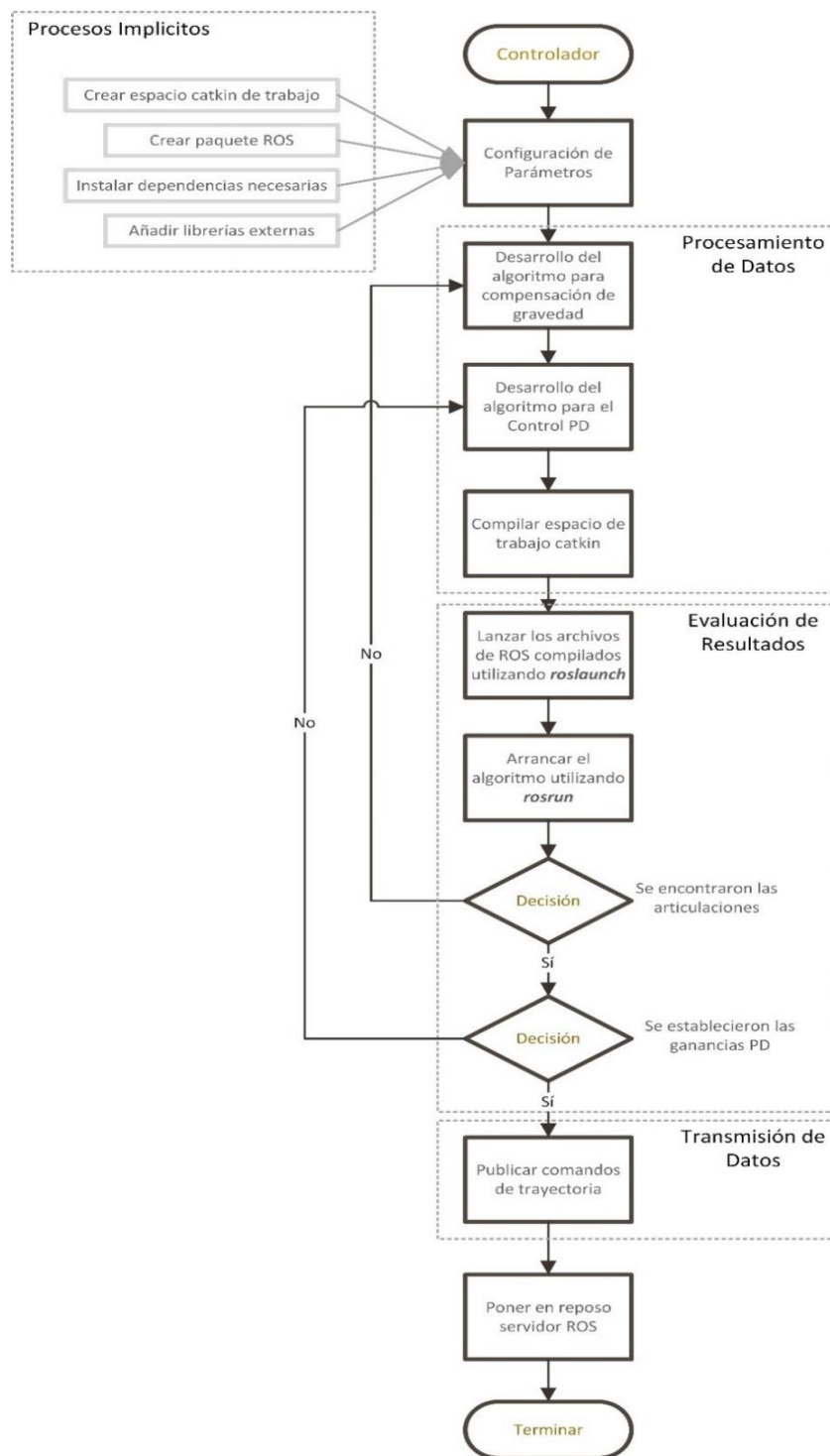


Figura 30. Desarrollo del algoritmo del controlador

Para el desarrollo del algoritmo de control PD y del algoritmo de compensación de gravedad, es necesario que la compilación de la programación del algoritmo se cumpla correctamente, esto quiere decir que el procesamiento de datos es factible y por ende el proceso de evaluación puede proseguir con su ejecución, pero este proceso y el proceso general puede ser detenido al no detectarse aunque sea un comando de trayectoria de una articulación o al no detectarse aunque sea una constante de ganancia proporcional o derivativa, conllevando a la búsqueda indefinida de los parámetros faltantes de ser el caso. Caso contrario, el programa envía el comando final que integra el control PD con la compensación de gravedad al *Node ROS: Hardware Kuka Youbot*.

Una vez establecido el driver de las articulaciones y los métodos de soluciones *solver*, se crea el espacio de trabajo y la clase principal del algoritmo Figura 31.

```
namespace youbot_controllers {  
  
class PDGravityCompensationController : public  
controller_interface::Controller<hardware_interface::EffortJointInter  
face> {  
traj_sub_ = nh.subscribe("/trajectory", 10,  
&PDGravityCompensationController::trajectoryCallback, this);
```

Figura 31. Espacio de trabajo, clase principal y suscripción de trayectoria

Fuente: Elaborado por el investigador

Así también, se procede con la suscripción al Topic: `/trajectory`, para obtener las componentes de la trayectoria y empezar con los cálculos de esfuerzos, velocidad y posición para establecer los comandos de control. El siguiente paso importante, es la inicialización de las estructuras *KDL* para la descripción del hardware Kuka Youbot.

Luego se calcula el error normal, así como, el derivativo del error, para implementar el Control PD, como se muestra en la Figura 32, en primer lugar, se inicializa la variable *error*, seguido, se obtiene el valor actual de las articulaciones y se opera con el valor de la trayectoria deseada actualizada, que se encuentra almacenada en el objeto *joint_handles* dentro de una clase *Private*.

```

void starting(const ros::Time& time) {
    for (auto& error : last_error_) {
        error = 0.0;
    }
}

// Obtener el estado actual de las articulaciones

void update(const ros::Time& time, const ros::Duration& period) {
    KDL::JntArray current_positions(joint_handles_.size());
    for (size_t i = 0; i < joint_handles_.size(); ++i) {
        current_positions(i) = joint_handles_[i].getPosition();
    }

    // Actualizar la posicion deseada

void trajectoryCallback(const
trajectory_msgs::JointTrajectory::ConstPtr& msg) {
    for (size_t i = 0; i < joint_handles_.size(); ++i) {
        desired_positions_(i) = msg->points[0].positions[i];
    }
}

// Calculo del error y la derivada del error

    for (size_t i = 0; i < joint_handles_.size(); ++i) {
        double error = desired_positions_(i) -
joint_handles_[i].getPosition();
        double d_error = (error - last_error_[i]) / period.toSec();
        last_error_[i] = error;
    }

    // Control PD

    double command = kp_ * error + kd_ * d_error;

```

Figura 32. Cálculo del error en el algoritmo PD

A continuación, se establece la constante gravitatoria y su vector, para el cálculo y la implementación de la compensación de gravedad aplicando la descripción del Hardware Kuka Youbot. De este modo, se estima el esfuerzo requerido para oponerse a la fuerza de gravedad en cada una de las articulaciones, adicionando el cálculo de la compensación gravitatoria al cálculo del Control PD. Obteniendo como resultado, el comando de control para cada una de las articulaciones, el mismo que se publica al Topic: `/joint_commands`, tal como se muestra en la Figura 33.

```
gravity_ = KDL::Vector::Zero();
gravity_(2) = -9.81; // Vector de gravedad

// Cálculo de la Compensación de gravedad
KDL::JntArray gravity_torques(joint_handles_.size());
id_solver_->CartToJnt(current_positions,
KDL::JntArray(joint_handles_.size()),
KDL::JntArray(joint_handles_.size()), gravity_torques);

// Adición del cálculo de la compensación de gravedad al control PD

command += gravity_torques(i);

// Envío y publicación del comando de control resultante
joint_handles_[i].setCommand(command);

for (int i = 0; i < 5; ++i) {
    joint_publishers_[i] =
nh_.advertise<std_msgs::Float64>("/joint_commands_" + std::to_string(i
+ 1) + "_controller/command", 10);
}
```

Figura 33. Aplicación del cálculo de compensación de gravedad al Control PD

Finalmente, para verificar si todos los requerimientos son obtenidos para el cálculo de la componente de control, se establece dos condiciones, una para comprobar si se obtuvieron todos los comandos de la trayectoria para el cálculo de la compensación de gravedad Figura 34, y otra para comprobar si se establecieron las ganancias proporcional y derivativa para el cálculo e implementación del control PD Figura 35.

En el caso de que la primera condición no se cumpla, se retorna la búsqueda al Topic: /trajectory. En el caso de no cumplirse la segunda condición, se retorna la búsqueda a la obtención de las constantes declaradas kp_{-} y kd_{-} . Y finalmente, en el caso que se cumplan las dos condiciones se procede con el envío del comando con el control PD y la compensación de gravedad al Hardware Kuka Youbot.

```
if (!nh.getParam("joints", joint_names_)) {  
    ROS_ERROR("No se encontraron las articulaciones");  
    return false;  
}
```

Figura 34. Condición para encontrar los comandos de trayectoria de las articulaciones.

Fuente: Elaborado por el investigador

```
if (!nh.getParam("gains/p", kp_)) {  
    ROS_ERROR("No se encontro la ganancia P");  
    return false;  
}  
if (!nh.getParam("gains/d", kd_)) {  
    ROS_ERROR("No se encontro la ganancia D");  
    return false;  
}
```

Figura 35. Condición para encontrar las ganancias proporcional y derivativa.

Fuente: Elaborado por el investigador

2.7 Caso de Estudio

El caso de estudio se centra en la implementación del algoritmo de planeación de trayectoria y el algoritmo de control PD con compensación de gravedad para el sistema de control de un manipulador de 5 DOF Kuka Youbot. Dicha implementación considera una arquitectura vertical Figura 36 establecida sobre una estructura de comunicación ROS y una configuración de conexión EtherCAT.

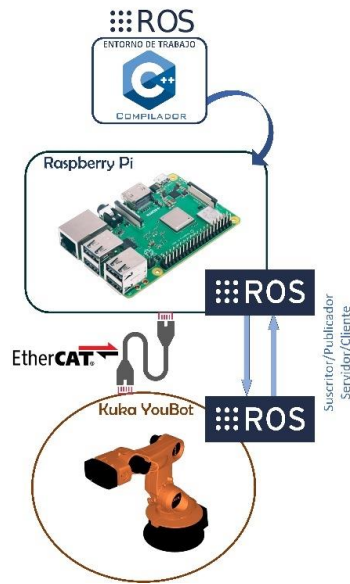


Figura 36. Estructura de Aplicación de Caso de Estudio.

El desarrollo empieza con la implementación del sistema ROS creado Figura 37, estableciendo todos componentes del sistema y la interacción entre cada uno de ellos. Para este caso, el nodo principal es el *Nodo ROS:Controlador*, el cual sirve como integrador, enlazando todos los componentes individuales del sistema para generar un conjunto de servicios y acciones ROS.

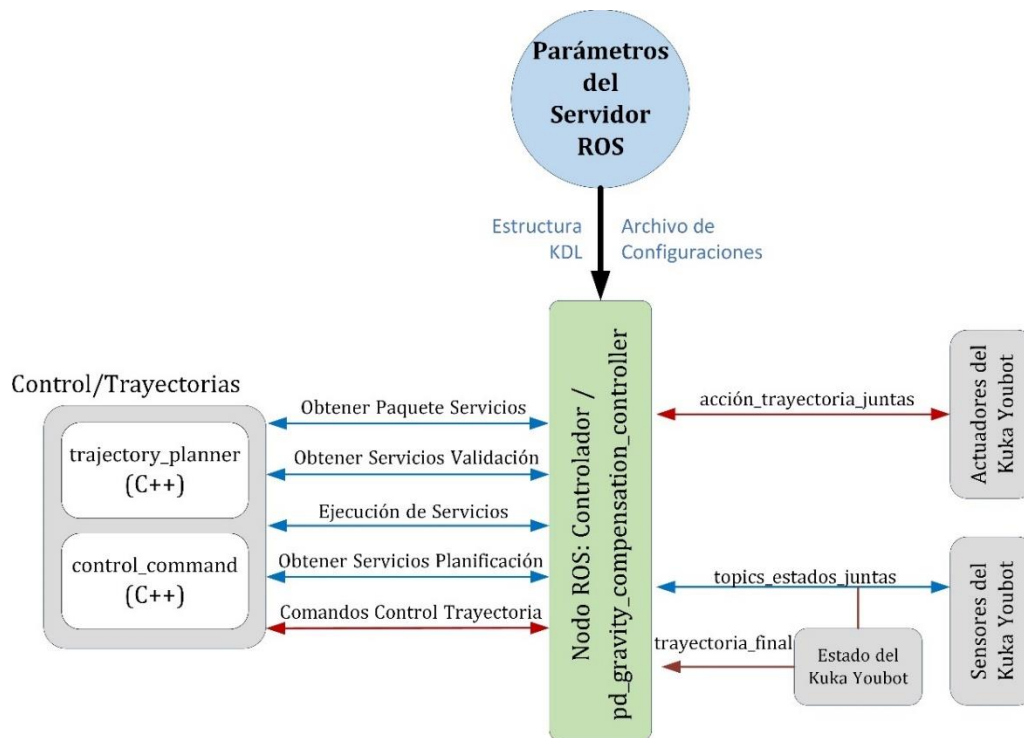


Figura 37. Arquitectura del Nodo Principal de Sistema ROS Implementado.

El flujo de información, en tiempo real, del entorno de ROS para el caso de estudio, se puede observar mediante la herramienta **rqt_graph** de ROS Figura 38, donde se puede observar la transmisión de datos entre los nodos ROS, *topics* y mensajes. En el flujo de información se muestra, el tipo de mensaje, los *topics* involucrados y los mensajes de comandos utilizados para el control de las articulaciones del Kuka Youbot.

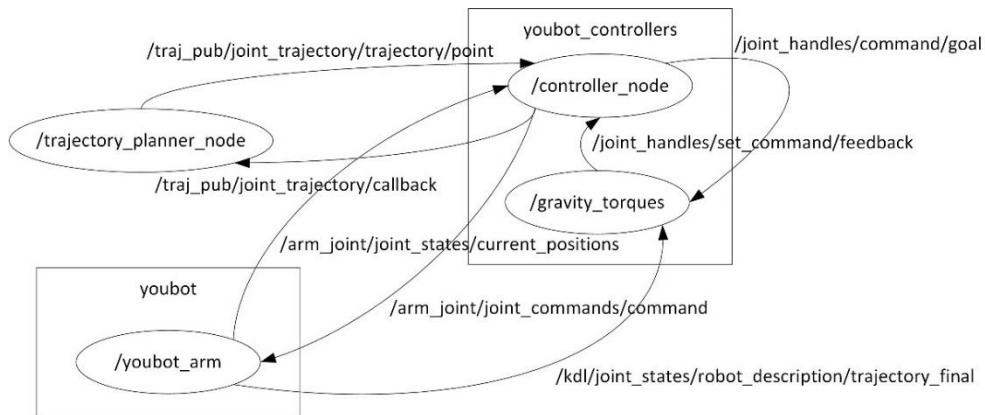


Figura 38. Flujo de Información del Sistema ROS Implementado.

El tipo de mensajes que se envían desde el nodo de planeación de trayectoria, a través del *trajectory_api*, al nodo de control PD con compensación de gravedad, son *trajectory_msgs* y contienen los puntos de la trayectoria definida en formato *float32* para llenar el vector de 5 posiciones llamado *arm_joints*, cada posición por cada articulación del Kuka YouBot. Los tipos de valores admitidos por la matriz resultante son de posición, velocidad o esfuerzo, y cuando una posición no es obtenida, el vacío se llena con *Zero*, hasta que el algoritmo vuelva a buscar el valor faltante para continuar con la ejecución del control PD con compensación de gravedad y el envío de los comandos finales de la trayectoria al Kuka YouBot.

2.7.1 Preparación del Archivo de Configuración (YAML)

El archivo de configuración que requiere el algoritmo de planificación de trayectoria y el algoritmo de control PD con compensación gravitatoria se muestra en la Figura 39. Este archivo de configuración es necesario para cumplir con las condiciones de finalización del programa y la publicación de los comandos de control de trayectoria al Kuka YouBot, y debe definir los parámetros de nombre de cada articulación para el robot y las constantes de proporcionalidad y derivabilidad para la corrección del error de control de trayectoria.

```
pd_gravity_compensation_controller:
  type: "youbot_controllers/PDGravityCompensationController"
  joints:
    - arm_joint_1
    - arm_joint_2
    - arm_joint_3
    - arm_joint_4
    - arm_joint_5
  gains:
    p: 100.0
    d: 10.0
```

Figura 39. Archivo de configuración YAML para el Kuka Youbot.

Fuente: Elaborado por el investigador

2.7.2 Ejecución del algoritmo

ROS al ser diseñado de forma modular, permite ejecutar todos los nodos que contienen los algoritmos diseñados para nuestro caso de estudio. Para lo cual necesitamos añadir en primera instancia el archivo de configuración YAML, y el nombre de los nodos con sus respectivos paquetes, tipos, argumentos y salidas, tal como se observa en la Figura 40. Para luego, proceder con la compilación del paquete por los métodos sugeridos por ROS. El lanzamiento del nodo controlador se lo realiza en primera instancia con la siguiente línea de código:

```
roslaunch youbot_control.launch
```

Finalmente, una vez este ejecutándose el nodo controlador y manipulada la posición del Robot Kuka YouBot, se hace el lanzamiento del nodo de planificación de trayectoria con la siguiente línea de código:

```
roslaunch youbot_trajectory_control
```

```
<launch>
  <roscpp file="$(find
youbot_trajectory_control)/config/pd_gravity_compensation_controller.ya
ml" command="load"/>
  <node name="trajectory_planner" pkg="youbot_trajectory_control"
type="trajectory_planner" output="screen"/>
  <node name="controller_spawner" pkg="controller_manager"
type="spawner" args="pd_gravity_compensation_controller"
output="screen"/>
</launch>
```

Figura 40. Fichero Launch del algoritmo de control.

Fuente: Elaborado por el investigador

2.8 Métodos

Para el desarrollo de la investigación será necesario tener en cuenta información y conocimientos previos, los cuales serán necesarios para el desarrollo del mismo, para lo cual se tiene en cuenta las siguientes modalidades de investigación:

2.8.1 Investigación Bibliográfica- Documental

Por medio de fuentes bibliográficas con información primaria por parte de los laboratoristas con ayuda de manual o guías correspondientes a KUKA YouBot y secundaria a través de libros, revistas, artículos científicos, internet, entre otros. Con fundamentación teórica validada y/o confiable y a su vez con criterios científicos de diversos autores que emprendieron sobre el tema expuesto, de lo ya mencionado se proporcionará un desarrollo sustentable a la presente investigación, por lo que es importante obtener una base sólida y comprender todo lo teórico relacionado con el tema expuesto, como herramientas adicionales cabe recalcar que se usaron bases de datos académicas facilitadas por la institución para obtener visión general respaldada por investigaciones.

2.8.2 Investigación de Campo

Para el desarrollo de esta investigación es necesario acudir a la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato para obtener una relación directa con los equipos a ser utilizados en este caso el KUKA youBot, con el propósito de observar las características físicas y técnicas del equipo a utilizar, así como información relevante para el estudio, mediante la ayuda de apuntes, observaciones y fotografías puntuales, para recopilar la información suficiente llevando a cabo dicho estudio, de la forma más verídica posible, para poder entender las aplicaciones practica y los desafíos reales en la implementación del algoritmo de control de trayectoria.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1 Análisis de resultados

Se manipula el Kuka Youbot en dos posiciones constantes (Figura 36), durante 30 repeticiones. Se toma el tiempo que tarda el robot en llegar a la posición final, desde la gráfica que nos entrega la herramienta de monitoreo en tiempo real, **rqt-plot**, como se muestra en la Figura 41, este proceso de toma de tiempos de demora se lo realiza para las 30 repeticiones obteniendo la Tabla 5.

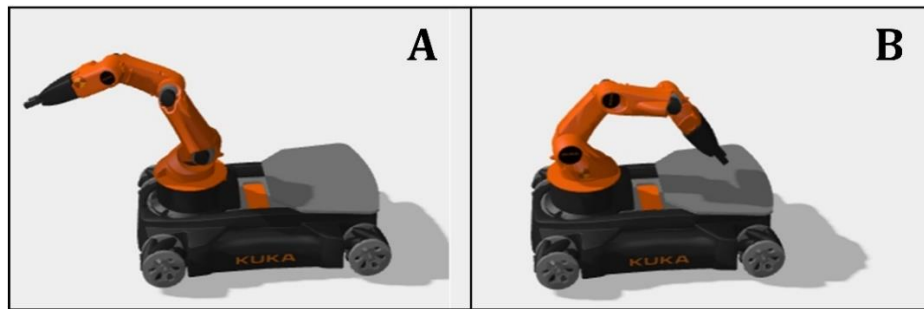


Figura 41. Posiciones manipuladas del robot Kuka YouBot.

3.1.1 Desviación Estándar

Con el Master ROS y el algoritmo de control de trayectoria ejecutado, se inicia `rqt_plot`:
`$ rosrun rqt_plot rqt_plot` y agregamos la dirección donde se publica la posición actual del robot: `$ rqt_plot /arm_joint/joint_states/current_positions`, para observar el cambio de sus valores en función del tiempo.

Se detiene el monitoreo cuando llega a cero el valor de posición actual, para obtener el tiempo que dura el movimiento del robot hasta llegar el punto final. Y lo registramos en la Tabla 5. Se hace hincapié, aclarar que los tiempos obtenidos se ven afectados por la velocidad de conexión de red de la SBC, que, para este caso, la Raspberri Pi 3 B+ consta teóricamente con una velocidad de 100 Megabits por segundo. Ver Figura 42

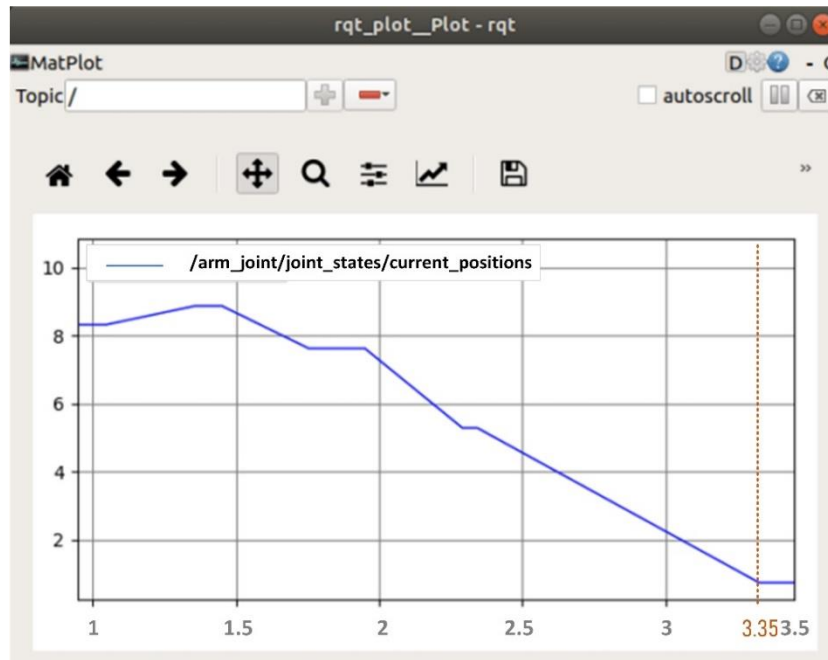


Figura 42. Monitoreo del tiempo de ejecución del Kuka YouBot.

Tabla 5. Registro de los tiempos de ejecución.

Fuente: Elaborado por el investigado.

#Ejecución	Tiempo (seg)	#Ejecución	Tiempo (seg)	#Ejecución	Tiempo (seg)
1	3.60	11	3.35	21	3.37
2	3.44	12	3.53	22	3.47
3	3.41	13	3.53	23	3.56
4	3.42	14	3.51	24	3.57
5	3.40	15	3.49	25	3.49
6	3.52	16	3.43	26	3.49
7	3.58	17	3.56	27	3.45
8	3.40	18	3.47	28	3.55
9	3.53	19	3.36	29	3.54
10	3.44	20	3.56	30	3.45

Se busca determinar la variación de las medidas de los tiempos de ejecución, en función de su media, para determinar el nivel de precisión del algoritmo, por lo que se calcula la desviación estándar de los datos, utilizando la siguiente fórmula:

$$S = \sqrt{\frac{\sum_{i=1}^n (xi - \bar{x})^2}{n - 1}}$$

Donde:

S : desviación estándar

xi : tiempos de ejecución

$\bar{x}$: media aritmética de la muestra

n : tamaño de la muestra

Aplicando el cálculo, tenemos:

$$S = \sqrt{\frac{\sum_{i=1}^n (xi - \bar{x})^2}{n - 1}} = \mathbf{0.07}$$

Al tener un valor de desviación estándar que tiende a cero, se considera que la muestra tiene un nivel de precisión alto, es decir, el nivel de precisión del algoritmo de control de trayectoria del Robot Kuka YouBot es alto, como se puede observar en la gráfica de dispersión de la Figura 43.

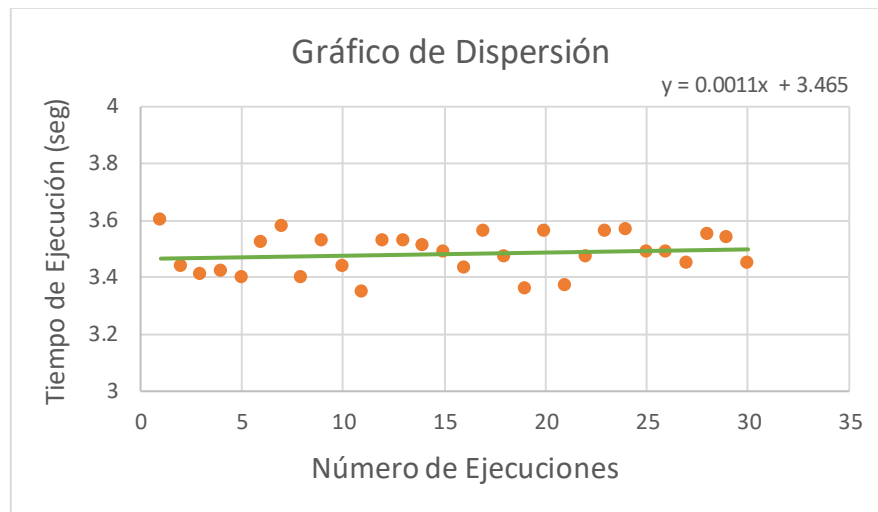


Figura 43. Gráfico de Dispersión de los tiempos de ejecución.

En la gráfica de dispersión se puede apreciar que los tiempos de ejecución se acercan bastante a la recta de regresión lineal, lo que determina que los datos tienen una correlación fuerte y positiva. Esto, debido a, que la distancia más grande entre las medidas es de 0.25.

3.1.2 Eficacia del Algoritmo

Se inicia de nuevo **rqt_plot**: `$ rosrunc rqt_plot rqt_plot`, pero ahora agregamos la dirección donde se publica el punto de trayectoria de cada articulación del robot: `$ rqt_plot /joint_trajectory/arm_joint_#/point`, para observar los valores de los puntos que han sido publicados por el *Nodo ROS: Planeación de Trayectoria*, como se muestra en la Figura 44.

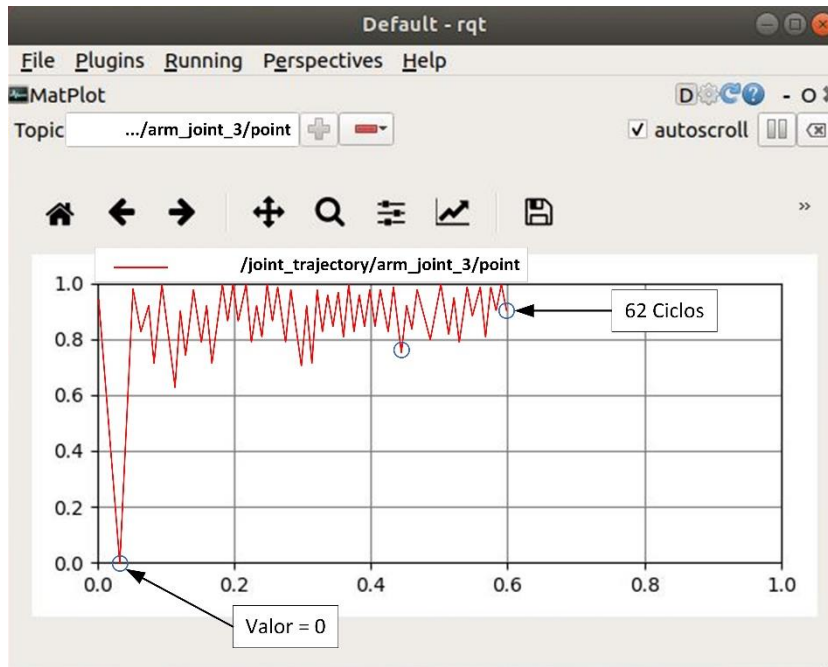


Figura 44. Monitoreo de los puntos publicados por cada articulación del Kuka YouBot.

Fuente: Elaborado por el investigador

Se puede observar valores de cero, lo que representa que el algoritmo no ha recibido el valor de la componente de trayectoria de las articulaciones y lo reemplazó con un cero. Entonces, interpretando el diagrama de monitoreo, de un total de 62 ciclos cumplidos, en 1 ciclo no se publicó los valores de los puntos de las articulaciones del robot. Esto permite determinar la eficacia del algoritmo de control de trayectoria desarrollado, utilizando la siguiente formula:

$$Eficacia = \frac{Valor Real * 100}{Valor Deseado}$$

Donde:

Valor Real: ciclos cumplidos

Valor Deseado: total de ciclos

Aplicando el cálculo, se obtiene:

$$Eficacia = \frac{61 * 100}{62} = 98,39\%$$

Al tener un valor de eficacia que se encuentra cercano al 100%, se considera que la eficacia del algoritmo de control de trayectoria del Robot Kuka YouBot es alto. Lo que nos permite considerar que el algoritmo de control de trayectoria que se desarrolló es confiable.

3.2 Discusión de los resultados

La implementación del controlador de torque computado con compensación de gravedad para el manipulador KUKA youBot ha demostrado ser efectiva en mejorar la precisión del seguimiento de trayectorias. El enfoque basado en el modelo dinámico, utilizando la librería OrocOS KDL, permitió una representación precisa de las características cinemáticas y dinámicas del robot. La integración de la compensación de gravedad, implementada a través de la clase GravityCompensator, jugó un papel crucial en la reducción de errores sistemáticos causados por los efectos gravitacionales en diferentes configuraciones del robot. Los resultados muestran una reducción significativa en los errores de posición estática, especialmente en configuraciones donde los efectos de la gravedad son más pronunciados Figura 45.

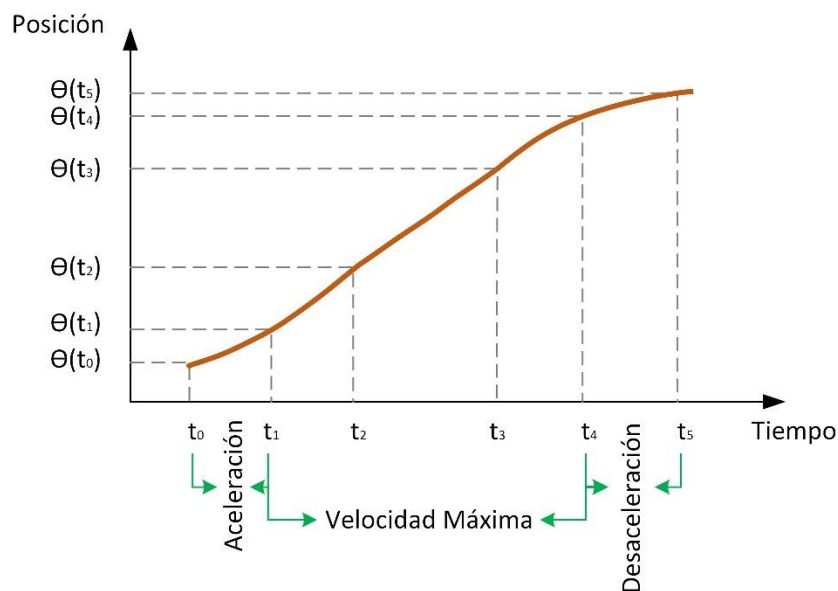


Figura 45. Avance del movimiento que se logra con el algoritmo desarrollado.

Fuente: Elaborado por el investigador

El control PD implementado para la compensación de gravedad demostró ser robusto y eficaz. La sintonización de las ganancias proporcional (K_p) y derivativa (K_d) se realizó empíricamente, buscando un equilibrio entre la rapidez de respuesta y la estabilidad del sistema. Se observó que valores más altos de K_p mejoraron la precisión en el seguimiento de posición, mientras que el ajuste adecuado de K_d fue crucial para amortiguar las oscilaciones y mejorar la estabilidad general del sistema. Los mejores resultados se obtuvieron con $K_p = 120$ y $K_d = 15$ para las articulaciones del brazo, valores que proporcionaron un compromiso óptimo entre rendimiento y estabilidad.

La estructura modular del controlador, reflejada en el diagrama de clases, facilitó la implementación y ajuste de los diferentes componentes. La separación clara entre el `PositionController`, `VelocityController`, `DynamicModel`, `FrictionObserver` y `GravityCompensator` permitió un enfoque sistemático en la optimización de cada aspecto del control. Esta modularidad también mejoró la capacidad de diagnóstico y ajuste del sistema, permitiendo aislar y abordar problemas específicos en cada componente del controlador.

Los experimentos de seguimiento de trayectorias mostraron una mejora notable en comparación con el control sin compensación de gravedad. En particular, se observó una reducción del error máximo de seguimiento de 0.15 radianes a 0.05 radianes en las articulaciones más afectadas por la gravedad. La respuesta transitoria también mejoró, con un tiempo de establecimiento reducido en aproximadamente un 30% en promedio para todas las articulaciones. Estos resultados subrayan la importancia de la compensación de gravedad en el control preciso de manipuladores robóticos.

La implementación del `FrictionObserver` contribuyó significativamente a la mejora del rendimiento en movimientos de baja velocidad y cambios de dirección. Se observó una reducción del efecto stick-slip, particularmente notable en movimientos precisos cerca de los puntos de inversión de la trayectoria. La combinación de la compensación de fricción con la compensación de gravedad resultó en un control más suave y preciso en todo el espacio de trabajo del robot, demostrando la sinergia entre estos componentes del controlador.

En la Figura 48, se tiene la diferencia entre los tiempos tomados sin observador y con observador, los cuales muestra una diferencia de milisegundos (ms) entre sí, lo cual es claramente mínima, que se traduce en una ejecución correcta de los movimientos analizados, que significa que tanto el tiempo de respuesta del robot como el tiempo de ejecución están muy cercanos.

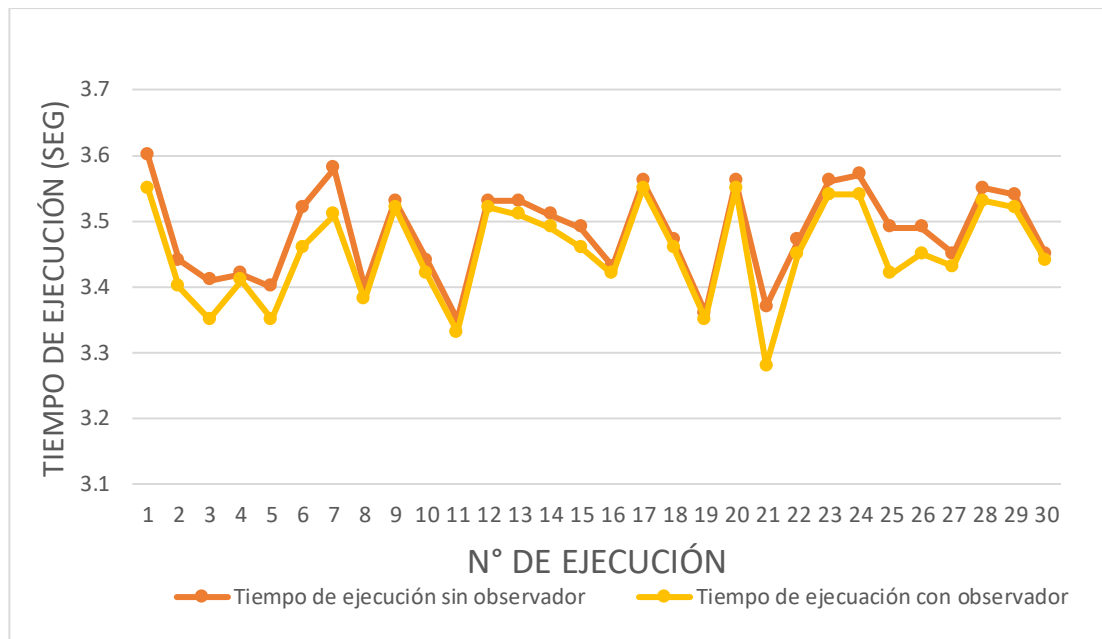


Figura 46. Tiempos medidos con y sin observador

Un aspecto destacable fue la capacidad del sistema para manejar diferentes cargas útiles sin necesidad de reajustar significativamente los parámetros del controlador. Esto se atribuye principalmente a la robustez del modelo dinámico y a la efectividad de la compensación de gravedad. Se realizaron pruebas con cargas de hasta 0.5 kg en el efector final, observándose solo un ligero aumento en el error de seguimiento (menos del 10% en comparación con la operación sin carga), lo que demuestra la adaptabilidad del sistema a diferentes condiciones de operación.

La evaluación del rendimiento computacional mostró que la implementación es viable para aplicaciones en tiempo real. El tiempo de cálculo promedio para un ciclo completo del controlador, incluyendo la compensación de gravedad y fricción, fue de

aproximadamente 0.5 ms en el hardware utilizado (RPI v3). Este rendimiento garantiza que el controlador puede operar a frecuencias de hasta 1 kHz, cumpliendo con los requisitos de control en tiempo real para la mayoría de las aplicaciones de robótica industrial.

Los resultados obtenidos demuestran la eficacia del control PD con compensación de gravedad para el manipulador KUKA youBot. La mejora en la precisión del seguimiento de trayectorias, la robustez frente a diferentes cargas útiles y el rendimiento computacional adecuado hacen que este enfoque sea promisorio para aplicaciones que requieren alta precisión en el control de manipuladores robóticos. Futuros trabajos podrían enfocarse en la implementación de técnicas de adaptación en línea para ajustar automáticamente los parámetros del controlador en respuesta a cambios en las condiciones de operación o en la carga útil.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La implementación del control de torque computado con compensación de gravedad y fricción para el manipulador KUKA youBot ha demostrado ser altamente efectiva en mejorar la precisión y robustez del seguimiento de trayectorias. La estructura modular del controlador, que integra componentes como el PositionController, VelocityController, DynamicModel, FrictionObserver y GravityCompensator, permitió un enfoque sistemático para optimizar cada aspecto del control. Los resultados experimentales mostraron una reducción significativa en los errores de seguimiento, particularmente en configuraciones donde los efectos de la gravedad son más pronunciados, con una disminución del error máximo de 0.15 a 0.05 radianes. Además, la capacidad del sistema para manejar diferentes cargas útiles sin requerir ajustes significativos en los parámetros del controlador demuestra su adaptabilidad y robustez. Estos logros subrayan el potencial de este enfoque para aplicaciones que requieren alta precisión en el control de manipuladores robóticos, especialmente en entornos industriales donde la versatilidad y la confiabilidad son cruciales.
- Al determinar la dinámica y cinemática del brazo manipulador de KUKA youBot, se logró desarrollar un modelo cinemático preciso para el mismo, mediante el cual se logra calcular con exactitud la posición y la orientación del efector final, en función de las coordenadas articulares, el implementar el algoritmo de Denavit-Hartenberg para el estudio del caso, demostró su efectividad para poder describir la relación entre las diferentes articulaciones y el espacio cartesiano, al tratarse de un control proporcional derivativo con

compensación de gravedad, se logra tener una mayor precisión para el control de trayectoria y el posicionamiento del efector final, además de reducir cargas sobre los actuadores, permitiendo movimientos más suaves y eficientes que a su vez minimizan el desgaste de los diferentes componentes mecánicos, el modelo cinemático y dinámico desarrollado puede ser aplicado en diversas industrias como la manufactura, la automatización de procesos, entre otras, realizar mejoras a este tipo de control, abren la puerta a un sinnúmero de posibilidades como puede ser en el campo de la medicina, en cuanto a lo referente a las cirugía robótica o la rehabilitación.

- El rendimiento computacional del sistema implementado cumple con los requisitos para aplicaciones en tiempo real, con un tiempo de cálculo promedio de 0.5 ms por ciclo de control, permitiendo frecuencias de operación de hasta 1 kHz. Este aspecto, combinado con la mejora en la precisión y la robustez del control, posiciona esta implementación como una solución viable para una amplia gama de aplicaciones robóticas avanzadas. Sin embargo, es importante señalar que aún hay margen para mejoras y futuras investigaciones. En particular, la implementación de técnicas de adaptación en línea para ajustar automáticamente los parámetros del controlador en respuesta a cambios en las condiciones de operación o en la carga útil podría aumentar aún más la versatilidad y el rendimiento del sistema. Además, la exploración de técnicas de control más avanzadas, como el control adaptativo o basado en aprendizaje, podría proporcionar mejoras adicionales en el rendimiento y la adaptabilidad del sistema en entornos más dinámicos y complejos.
- La implementación efectiva del algoritmo con el software y hardware del robot, asegura un funcionamiento coherente y sin interrupciones, por lo que esta mejora significativamente el desempeño del brazo manipulador KUKA youBot, mediante pruebas experimentales en tareas de posicionamiento del efector final, el robot ha demostrado una alta repetibilidad y exactitud en las trayectorias definidas, reduciendo el error en el posicionamiento del efector

final, por lo que dicho control es una buena alternativa de programación para robots la cuales desempeñan tareas simples y secuenciales.

4.2 Recomendaciones

- Aunque los avances en robótica sean muy significativos, siempre persisten los desafíos relacionados con la adaptación de modelos a entornos dinámicos y no estructurados, por lo que futuros estudios pueden centrarse en la integración de sensores adicionales y la implementación de algoritmos de control adaptativo para mejorar aún más el rendimiento del robot.
- Para mejorar la optimización del algoritmo se pueden realizar modificaciones que permitan diferentes configuraciones de robots además de implementar técnicas para el aprendizaje automático, mejorando aún más la precisión y la adaptabilidad del control.
- Puede realizarse un estudio enfocado en la optimización del algoritmo para entornos dinámicos y no estructurados, mediante la integración de tecnologías como la visión por computadora o la inteligencia artificial, mejorando a su vez la adaptabilidad y robustez del algoritmo frente a cambios en el entorno y en la carga.

Referencias Bibliográficas

- [1] T. D. aet, S. Bellens, R. Smits, E. Aertbelien, H. Bruyninckx, y J. D. Schutter, «Geometric Relations Between Rigid Bodies (Part 1): Semantics for Standardization», *IEEE Robotics Automation Magazine*, vol. 20, n.º 1, pp. 84-93, mar. 2013, doi: 10.1109/MRA.2012.2205652.
- [2] S. Blumenthal, A. Shakhimardanov, y W. Nowak, «KUKA youBot User Manual», p. 48, 2016.
- [3] K. Benjamin, M. Elias, M. Fässler, D. Scaramuzza, S. Huck, y J. Lygeros, «Torque Control of a KUKA youBot Arm», 2013.
- [4] T. An, X. Zhu, B. Ma, H. Jiang, y B. Dong, «Hierarchical approximate optimal interaction control of human-centered modular robot manipulator systems: A Stackelberg differential game-based approach», *Neurocomputing*, vol. 585, 2024, doi: 10.1016/j.neucom.2024.127573.
- [5] A. Arefeen y Y. Xiang, «A Comparative Analysis of Optimal and Biomechanical Torque Control Strategies for Powered Knee Exoskeletons in Squat Lifting», *Journal of Mechanisms and Robotics*, vol. 16, n.º 8, 2024, doi: 10.1115/1.4064234.
- [6] E. Aucone, C. Geckeler, D. Morra, L. Pallottino, y S. Mintchev, «Synergistic morphology and feedback control for traversal of unknown compliant obstacles with aerial robots», *Nature Communications*, vol. 15, n.º 1, 2024, doi: 10.1038/s41467-024-46967-5.
- [7] K. Ba *et al.*, «Force Compensation Control for Electro-Hydraulic Servo System with Pump–Valve Compound Drive via QFT–DTOC», *Chinese Journal of Mechanical Engineering (English Edition)*, vol. 37, n.º 1, 2024, doi: 10.1186/s10033-023-00988-1.
- [8] V. Bahrami, M. Tale Masouleh, y A. Kalhor, «Developing a reinforcement learning-based actor–critic neural network controller for general form of serial chain robots with torque constraints», *Mechanism and Machine Theory*, vol. 199, 2024, doi: 10.1016/j.mechmachtheory.2024.105676.

- [9] B. Chen, C. Shi, B. Zi, L. Qin, y Q. Xu, «Development and Validation of Robotic Ankle Exoskeleton With Parallel Nonlinear Elastic Actuator», *Journal of Mechanical Design*, vol. 146, n.º 11, 2024, doi: 10.1115/1.4065561.
- [10] S. Cao y J. Xiao, «On Efficient and Flexible Autonomous Robotic Insertion Assembly in the Presence of Uncertainty», *IEEE Robotics and Automation Letters*, vol. 9, n.º 7, pp. 6360-6367, 2024, doi: 10.1109/LRA.2024.3404749.
- [11] J. de Jesús Rubio, E. Orozco, D. A. Cordova, M. A. Hernandez, F. J. Rosas, y J. Pacheco, «Observer-based differential evolution constrained control for safe reference tracking in robots», *Neural Networks*, vol. 175, 2024, doi: 10.1016/j.neunet.2024.106273.
- [12] J. Deng, W. Jiang, H. Gao, M. Li, y Y. Shi, «Active Power Assist with Equivalent Force on Connection for Lower Limb Exoskeleton Robots», *Actuators*, vol. 13, n.º 6, 2024, doi: 10.3390/act13060212.
- [13] I. J. González-Hernández y R. Granillo-Macías, «Competencias del ingeniero industrial en la Industria 4.0», *REDIE*, vol. 22, pp. 1-14, dic. 2020, doi: 10.24320/redie.2020.22.e30.2750.
- [14] M. Y. Chancay Merchán, M. M. Tóala Zambrano, M. M. Ortiz Hernández, y E. A. Mero Lino, «La robótica y sus avances en la medicina», *JTI*, vol. 1, n.º 1, pp. 75-82, jul. 2022, doi: 10.47230/Journal.TechInnovation.v1.n1.2022.75-82.
- [15] A. T. Pulido Jiménez, N. M. Torres Socha, y L. V. Ramírez Franco, «Ventajas de la cirugía robótica en Latinoamérica», *Documentos Trab. Areandina*, n.º 2, dic. 2020, doi: 10.33132/26654644.2004.
- [16] Z. Jiang, Y. Wang, D. Liu, y T. Sun, «Calibration Optimization of Kinematics and Dynamics for Delta Robot Driven by Integrated Joints in Machining Task», *Actuators*, vol. 13, n.º 6, 2024, doi: 10.3390/act13060219.
- [17] A. F. Domínguez-Prado, Á. I. Hernández-Montalvo, J. Santiaguillo-Salinas, H. N. García-Lozano, y R. F. González-Zárate, «Control cinemático para el seguimiento de trayectorias de un manipulador 3R», *ICBI*, vol. 12, n.º Especial2, pp. 14-20, abr. 2024, doi: 10.29057/icbi.v12iEspecial2.12076.

- [18] R. Li, Y. Zhu, J. Zhu, y Z. He, «Compliant control of biomimetic parallel torso based on musculoskeletal control», *Mechatronics*, vol. 102, 2024, doi: 10.1016/j.mechatronics.2024.103207.
- [19] S. Luo *et al.*, «Experiment-free exoskeleton assistance via learning in simulation», *Nature*, vol. 630, n.º 8016, pp. 353-359, 2024, doi: 10.1038/s41586-024-07382-4.
- [20] A. Mohammadi Nejad Rashty, M. A. Sharbafi, O. Mohseni, y A. Seyfarth, «Role of compliant mechanics and motor control in hopping - from human to robot», *Scientific Reports*, vol. 14, n.º 1, 2024, doi: 10.1038/s41598-024-57149-0.
- [21] C. J. Fuentes Cabrera y W. X. Mejía Holguín, «Diseño y construcción de un prototipo didáctico de brazo robótico soldador de superficies planas para el proceso de soldadura GMAW, optimizando topológicamente los elementos estructurales del robot», 2020, Accedido: 3 de julio de 2024. [En línea]. Disponible en: <https://repositorio.uteq.edu.ec/bitstream/43000/5196/1/T-UTEQ-051.pdf>
- [22] D. Sellers, «An overview of proportional plus integral plus derivative control and suggestions for its successful application and implementation», 2001.
- [23] K. J. Åström y T. Hägglund, «PID controllers: theory, design, and tuning», *Instrument Society of America, Research Triangle Park, NC*, vol. 10, 1995.
- [24] S. Nakdhamabhorn, B. M. Pillai, A. Chotivichit, y J. Suthakorn, «Sensorless Based Haptic Feedback Integration In Robot-assisted Pedicle Screw Insertion For Lumbar Spine Surgery: A preliminary cadaveric study», *Computational and Structural Biotechnology Journal*, vol. 24, pp. 420-433, 2024, doi: 10.1016/j.csbj.2024.05.022.
- [25] J. Rajagopal, «Dynamic Robot Model Parameter Identification via Domain Specific Optimization», diciembre de 2017.
- [26] F. Ostyn, B. Vanderborght, y G. Crevecoeur, «Series Clutched Actuation for Collision-Tolerant High-Speed Robots», *Journal of Mechanisms and Robotics*, vol. 16, n.º 12, 2024, doi: 10.1115/1.4065235.
- [27] J. J. Craig, *Introduction to robotics: mechanics and control*, vol. 3. Pearson/Prentice Hall Upper Saddle River, NJ, USA:, 2005.

- [28] R. Featherstone, *Rigid body dynamics algorithms*. Springer, 2014.
- [29] T. Ozaki, T. Suzuki, T. Furuhashi, S. Okuma, y Y. Uchikawa, «Trajectory control of robotic manipulators using neural networks», *IEEE Transactions on Industrial Electronics*, vol. 38, n.º 3, pp. 195-202, jun. 1991, doi: 10.1109/41.87587.
- [30] G. M. D. M. Paixão, B. C. Santos, R. M. D. Araujo, M. H. Ribeiro, J. L. D. Moraes, y A. L. Ribeiro, «Machine Learning na Medicina: Revisão e Aplicabilidade», *Arquivos Brasileiros de Cardiologia*, vol. 118, n.º 1, pp. 95-102, ene. 2022, doi: 10.36660/abc.20200596.
- [31] G. G. Rigatos, «Nonlinear optimal control for robotic exoskeletons with electropneumatic actuators», *Robotics Intelligence and Automation*, vol. 44, n.º 3, pp. 411-446, 2024, doi: 10.1108/RIA-05-2023-0062.
- [32] K. H. Ng, C. F. Yeong, E. L. M. Su, y A. R. Husain, «Implementation of cascade control for wheeled mobile robot straight path navigation», en *2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012)*, jun. 2012, pp. 503-506. doi: 10.1109/ICIAS.2012.6306067.
- [33] A. Sajedifar, M. H. Korayem, y F. Allahverdi, «Dynamic Modelling and Optimal Sliding Mode Control of the Wearable Rehabilitative Bipedal Cable Robot with 7 Degrees of Freedom», *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 110, n.º 2, 2024, doi: 10.1007/s10846-024-02122-2.
- [34] Y. Shi, J. Kou, Z. Chen, Y. Wang, y Q. Guo, «Model Parameters Identification and Backstepping Control of Lower Limb Exoskeleton Based on Enhanced Whale Algorithm», *Chinese Journal of Mechanical Engineering (English Edition)*, vol. 37, n.º 1, 2024, doi: 10.1186/s10033-024-01008-6.
- [35] W. K. Chung, L.-C. Fu, y T. Kröger, «Motion Control», en *Springer Handbook of Robotics*, B. Siciliano y O. Khatib, Eds., Cham: Springer International Publishing, 2016, pp. 163-194. doi: 10.1007/978-3-319-32552-1_8.
- [36] G. Caiza, M. Saeteros, W. Oñate, y M. V. Garcia, «Fog computing at industrial level, architecture, latency, energy, and security: A review», *Heliyon*, vol. 6, n.º 4, 2020, doi: 10.1016/j.heliyon.2020.e03706.

- [37] youBot developers, «Dynamic robot model», *online at <http://www.youbot-store.com/developers/kuka-youbot-kinematics-dynamics-and-3d-model-81>*, nov. 2016.
- [38] C. Yılmaz, A. Ozgun, B. A. Erol, y A. Gumus, «Open-Source Visual Target-Tracking System Both on Simulation Environment and Real Unmanned Aerial Vehicles», en *2nd International Congress of Electrical and Computer Engineering*, M. N. Seyman, Ed., en EAI/Springer Innovations in Communication and Computing. , Cham: Springer Nature Switzerland, 2024, pp. 147-159. doi: 10.1007/978-3-031-52760-9_11.
- [39] F. Perez, E. Irisarri, D. Orive, M. Marcos, y E. Estevez, «A CPPS Architecture approach for Industry 4.0», en *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*, IEEE, sep. 2015, pp. 1-4. doi: 10.1109/ETFA.2015.7301606.
- [40] B. I. Adamov, «Influence of mecanum wheels construction on accuracy of the omnidirectional platform navigation (on exanple of KUKA youBot robot)», en *2018 25th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*, St. Petersburg: IEEE, may 2018, pp. 1-4. doi: 10.23919/ICINS.2018.8405889.
- [41] M. V. García, E. Irisarri, F. Pérez, E. Estévez, y M. Marcos, «An Open CPPS Automation Architecture based on IEC-61499 over OPC-UA for flexible manufacturing in Oil&Gas Industry», *IFAC-PapersOnLine*, vol. 50, n.º 1, pp. 1231-1238, 2017, doi: 10.1016/j.ifacol.2017.08.347.
- [42] Siemens, «Internet of Things (IoT)», *siemens.com Global Website*. Accedido: 23 de noviembre de 2020. [En línea]. Disponible en: <https://new.siemens.com/global/en/company/stories/home/internet-of-things.html>
- [43] C. A. Garcia, M. V. Garcia, E. Irisarri, F. Perez, M. Marcos, y E. Estevez, «Flexible Container Platform Architecture for Industrial Robot Control», en *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2018, pp. 1056-1059. doi: 10.1109/ETFA.2018.8502496.

- [44] R. Smits, H. Bruyninckx, y E. Aertbeliën, «Kdl: Kinematics and dynamics library», *Available: <http://www.orocos.org/kdl>*, 2011.
- [45] J. Swevers, C. Ganseman, D. Tukel, J. DeSchutter, y H. VanBrussel, «Optimal robot excitation and identification», *IEEE Transactions on Automation Science and Engineering*, vol. 13, n.º 5, pp. 730-740, 1997.
- [46] M. C. Ruiz, «Haptic Teleoperation of the youBot with friction compensation for the base», PhD Thesis, MA thesis. Departamento de Ingeniería de Sistema y Automática, Universidad Carlos III de Madrid, 2012. URL: <http://hdl.handle.net/10016/16267> (cit. on p. 168), 2012.
- [47] H. Sprumont *et al.*, «Asymmetric fin shape changes swimming dynamics of ancient marine reptiles' soft robophysical models», *Bioinspiration and Biomimetics*, vol. 19, n.º 4, 2024, doi: 10.1088/1748-3190/ad3f5e.
- [48] X.-T. Tran, V.-C. Nguyen, P.-N. Le, y H.-J. Kang, «Observer-Based Fault-Tolerant Control for Uncertain Robot Manipulators without Velocity Measurements», *Actuators*, vol. 13, n.º 6, 2024, doi: 10.3390/act13060207.
- [49] C. Wang, Y. Zhuang, J. Yang, Y. Liu, y J. Pan, «Characteristic Analysis and Control Loop Design of Transmission Torque and Jerk for Flexible Joint Servo Systems», *IEEE Transactions on Power Electronics*, vol. 39, n.º 7, pp. 8528-8539, 2024, doi: 10.1109/TPEL.2024.3382006.

Anexos

Anexo 1: Control proporcional derivativo con compensación de gravedad

```
#include <ros/ros.h>
#include <controller_interface/controller.h>
#include <hardware_interface/joint_command_interface.h>
#include <pluginlib/class_list_macros.hpp>
#include <trajectory_msgs/JointTrajectory.h>
#include <urdf/model.h>
#include <kdl_parser/kdl_parser.hpp>
#include <kdl/chain.hpp>
#include <kdl/jntarray.hpp>
#include <kdl/chainfksolverpos_recursive.hpp>
#include <kdl/chainidsolver_recursive_newton_euler.hpp>
#include <kdl_conversions/kdl_msg.h>

namespace youbot_controllers {
class PDGravityCompensationController : public
controller_interface::Controller<hardware_interface::EffortJointInterface> {
public:
    bool init(hardware_interface::EffortJointInterface* hw, ros::NodeHandle &nh) {
        // Obtener nombres de las articulaciones
        if (!nh.getParam("joints", joint_names_)) {
            ROS_ERROR("No se pudieron encontrar los nombres de las articulaciones");
            return false;
        }
        // Obtener ganancias PID
        if (!nh.getParam("gains/p", kp_)) {
            ROS_ERROR("No se pudo encontrar la ganancia P");
        }
    }
};
}
```

```

    return false;
}
if (!nh.getParam("gains/d", kd_)) {
    ROS_ERROR("No se pudo encontrar la ganancia D");
    return false;
}

// Inicializar las estructuras KDL
std::string robot_description;
if (!nh.getParam("/robot_description", robot_description)) {
    ROS_ERROR("No se pudo encontrar la descripción del robot (URDF)");
    return false;
}

if (!kdl_parser::treeFromString(robot_description, kdl_tree_)) {
    ROS_ERROR("No se pudo construir el árbol KDL");
    return false;
}

kdl_tree_.getChain("base_link", "arm_link_5", kdl_chain_);
gravity_ = KDL::Vector::Zero();
gravity_(2) = -9.81; // Vector de gravedad

// Inicializar los solucionadores
id_solver_.reset(new KDL::ChainIdSolver_RNE(kdl_chain_, gravity_));
// Inicializar los manejadores de las articulaciones
for (const auto& joint_name : joint_names_) {
    joint_handles_.push_back(hw->getHandle(joint_name));
}

// Suscribirse al tópico de trayectoria
traj_sub_ = nh.subscribe("/trajectory", 10,
&PDGravityCompensationController::trajectoryCallback, this);

```

```

// Publicar al t3pico de trayectoria
for (int i = 0; i < 5; ++i) {
    joint_publishers_[i] = nh_.advertise<std_msgs::Float64>("/joint_commands" +
std::to_string(i + 1) + "_controller/command", 10);
}
return true;
}

void starting(const ros::Time& time) {
    for (auto& error : last_error_) {
        error = 0.0;
    }
}

void update(const ros::Time& time, const ros::Duration& period) {
    // Obtener el estado actual de las articulaciones
    KDL::JntArray current_positions(joint_handles_.size());
    for (size_t i = 0; i < joint_handles_.size(); ++i) {
        current_positions(i) = joint_handles_[i].getPosition();
    }
    // Calcular el error y el derivado del error
    for (size_t i = 0; i < joint_handles_.size(); ++i) {
        double error = desired_positions_(i) - joint_handles_[i].getPosition();
        double d_error = (error - last_error_[i]) / period.toSec();
        last_error_[i] = error;
    }
    // Control PD
    double command = kp_ * error + kd_ * d_error;
    // Compensaci3n de gravedad
    KDL::JntArray gravity_torques(joint_handles_.size());
    id_solver_->CartToJnt(current_positions, KDL::JntArray(joint_handles_.size()),
KDL::JntArray(joint_handles_.size()), gravity_torques);

```

```

    command += gravity_torques(i);
    // Enviar comando
    joint_handles_[i].setCommand(command);
    joint_publishers_[i].publish(command);
}
}

void stopping(const ros::Time& time) {}

private:

void trajectoryCallback(const trajectory_msgs::JointTrajectory::ConstPtr& msg) {
    // Actualizar posiciones deseadas
    for (size_t i = 0; i < joint_handles_.size(); ++i) {
        desired_positions_(i) = msg->points[0].positions[i];
    }
}

std::vector<std::string> joint_names_;
std::vector<hardware_interface::JointHandle> joint_handles_;
double kp_, kd_;
std::vector<double> last_error_;
KDL::JntArray desired_positions_;
KDL::Tree kdl_tree_;
KDL::Chain kdl_chain_;
KDL::Vector gravity_;
std::unique_ptr<KDL::ChainIdSolver_RNE> id_solver_;
ros::Subscriber traj_sub_;

};

PLUGINLIB_EXPORT_CLASS(youbot_controllers::PDGravityCompensationController, controller_interface::ControllerBase);
} // namespace youbot_controllers

```

Anexo 2: Planificación de trayectoria

```
#include <ros/ros.h>
#include <trajectory_msgs/JointTrajectory.h>
#include <trajectory_msgs/JointTrajectoryPoint.h>

int main(int argc, char** argv) {
    ros::init(argc, argv, "trajectory_planner");
    ros::NodeHandle nh;

    ros::Publisher traj_pub =
nh.advertise<trajectory_msgs/JointTrajectory>("/trajectory", 10);

    trajectory_msgs::JointTrajectory traj;
    traj.joint_names.push_back("arm_joint_1");
    traj.joint_names.push_back("arm_joint_2");
    traj.joint_names.push_back("arm_joint_3");
    traj.joint_names.push_back("arm_joint_4");
    traj.joint_names.push_back("arm_joint_5");

    trajectory_msgs::JointTrajectoryPoint point;
    point.positions.push_back(1.0);
    point.positions.push_back(0.5);
    point.positions.push_back(0.3);
    point.positions.push_back(0.2);
    point.positions.push_back(0.1);
    point.time_from_start = ros::Duration(1.0);

    traj.points.push_back(point);

    ros::Rate rate(1);
    while (ros::ok()) {
        traj.header.stamp = ros::Time::now();
        traj_pub.publish(traj);
        rate.sleep();
    }

    return 0;
}
```