BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

MAESTRÍA EN CIENCIAS DE LA ELECTRÓNICA,
OPCIÓN EN AUTOMATIZACIÓN

"VALIDACIÓN EN TIEMPO REAL DE ESTRATEGIAS DE CONTROL DE COBOT EN ESTACIÓN DE ENSAMBLE"

TESIS

Presentada para obtener el título de:
Maestro en Ciencias de la Electrónica, Opción en Automatización

Presenta:
Ing. Rubén Ganiz Mero*

Directores:

Dr. Jorge Dionisio Fierro Rojas (FCE-BUAP)

Dr. Alexandre Zemliak (FCFM-BUAP)

Puebla, México **Julio de 2023**

Becario CONAHCYT*

Agradecimientos

A mi madre, por ser la mayor espectadora de mis triunfos, por siempre anhelar y desear lo mejor para mi vida, por nunca dejar de caminar a mí lado a pesar de todas las adversidades e inconvenientes que la vida nos ha puesto. Mis triunfos también son tuyos.

A mis abuelos[1], por todo ese amor y apoyo incondicional, por ser mis guías y sostén en los momentos más difíciles. Así como, a mi tía Briseida, por siempre estar al pendiente de mí, por siempre estar dispuesta a apoyarme en cualquier momento sin esperar nada a cambio y por su gran corazón.

A mi asesor, Dr. Jorge Dionisio Fierro Rojas, por toda su ayuda y confianza puesta en mí, por siempre estar al pendiente de las dudas y necesidades que iban surgiendo a lo largo de este proceso y sobre todo por su amistad.

A los profesores de la Maestría en Ciencias de la Electrónica, Opción en Automatización quienes me brindaron todos los conocimientos necesarios para poder desarrollar esta tesis y en especial, a aquellos que invirtieron su tiempo en leer y hacer valiosas recomendaciones para mejorar la redacción y presentación de la misma.

A mis compañeros de generación, por todo su conocimiento, apoyo y compañerismo brindado en todo este tiempo.

A CONAHCYT por el apoyo económico brindado.

[1] *Arturo & M.Isabel. Rubén & Isabel.*

Dedicatoria

Dedicado a mis abuelos, sin ustedes
esto no hubiera sido posible.
Y a mi fiel compañero, Hunter.

Resumen

En la presente investigación se aborda el problema de validación en tiempo real de estrategias de control de una estación virtual de ensamble colaborativo entre un operador humano y un cobot en un contexto de Industria 4.0. Siguiendo un proceso de Diseño Basado en Modelos (MBD), el proyecto se realiza en etapas, empezando por el modelado virtual del robot colaborativo, y sustituyendo después gradualmente cada subsistema en software por su correspondiente sistema embebido en una tarjeta Raspberry Pi. El producto final es un sistema informático que emula la estación de ensamble mediante los códigos Python del modelo dinámico del cobot, del generador de trayectorias y del controlador, cada uno embebido en una Raspberry Pi. Además sensores que detectan la presencia de un operador humano en el espacio de trabajo del cobot. Cada tarea de ensamble llevada a cabo en la estación es secuencial e inteligente, y la validez de la estrategia de control para la interacción operador humano - cobot se prueba mediante la precisión con la que se realiza la tarea.

Índice general

Índice de tablas

Índice de figuras

Introducción

En el uso de robots industriales en tareas de ensamble tradicionalmente, se requiere la delimitación del espacio de trabajo con cercas protectoras para evitar el riesgo de accidentes por impactos de la estructura mecánica de los robots con operadores humanos [4]. En contraste, los robots colaborativos (cobots) están diseñados precisamente para tal interacción [9] y, a diferencia de los robots industriales tradicionales, los cobots combinan la destreza, flexibilidad y habilidad para solucionar problemas de los humanos, y la fuerza, resistencia y precisión de los robots [10]. Indudablemente, los cobots son parte de la tecnología de Industria 4.0, pues figuran como uno de los agentes principales en la realización de tareas de la llamada manufactura inteligente [11]. Dentro de este nuevo enfoque de manufactura, uno de los aspectos más importantes en la interacción de un operador humano con un cobot es la garantía de salvaguardar la integridad física del operador a tal punto que se han definido normas industriales para su regulación [12].

Recientemente, algunas de las principales compañías fabricantes de robots han desarrollado y puesto a la venta cobots para diversas aplicaciones industriales [13]. Como es de suponerse, además de los costos elevados, la arquitectura cerrada de los cobots comerciales impide la validación de algoritmos de control. Resulta, pues, conveniente contar con una plataforma tecnológica diseñada ex profeso para experimentación e investigación científica. En este trabajo de investigación se propone el diseño de una plataforma tecnológica para validación de estrategias de control conformada por sistemas embebidos interconectados que emulan la interacción de un cobot con un operador humano. Por los componentes propios de la estación de ensamble - cobot, como lo son sensores de detección de presencia de personas en el área de trabajo, es claro que interactúan subsistemas con diferentes dinámicas entre sí actuando simultáneamente, lo que sugiere una consideración de las velocidades individuales en la transmisión de datos; es decir, se visualiza un problema de selección de un protocolo de comunicación apropiado para la implementación física.

La metodología de investigación que se sigue en el presente trabajo está basada en el proceso de Diseño Basado en Modelos (MBD[2], por sus siglas en inglés) ampliamente extendido en la industria automotriz y en algunos desarrollos de sistemas electrónicos de potencia [6]. Algunas de las ventajas del proceso MBD al combinar subsistemas modelados en software con otros implementados en tarjetas electrónicas de procesador, es la flexibilidad y confiabilidad de los sistemas estudiados a un costo económico pequeño en comparación con el que supone la

[2]MBD; acrónimo de la lengua inglesa *Model Based Design*.

construcción de prototipos físicos. Como un complemento en el proceso MBD, el prototipado virtual (VP[3], por sus siglas en inglés) es una herramienta digital que se caracteriza por incluir un modelo, estructura o aparato; es usado para probar y evaluar la forma, el diseño, el desempeño y la capacidad de fabricación. Además se usa para el estudio y entrenamiento de personal. Su alcance sobrepasa al de las simulaciones convencionales porque el VP intenta abordar todos los aspectos relacionados con el producto con el fin de sustituir los prototipos físicos [14]. El método de VP se ha usado con la intención de estudiar la manipulación y el control de brazos robóticos sin la necesidad de crear las complejas ecuaciones dinámicas, como se reporta en [15].

Existen herramientas de software que facilitan el VP. En el presente trabajo, se considera ADAMS® [16], de la compañía MSC Software, que se posiciona como el software de ingeniería asistida por computadora (CAE[4], de las siglas en inglés) más usado para el estudio de la dinámica de partes móviles, y determina cómo las cargas y las fuerzas son distribuidas en los sistemas mecánicos [16]. Por su parte MATLAB®, desarrollado por Mathworks, es una plataforma para la computación numérica mediante la manipulación de matrices, gráfica de funciones y datos, implementación de algoritmos, creación de interfaces gráficas de usuario y la comunicación con programas escritos en otros lenguajes [8].

La co-simulación de ADAMS®, donde residirá el modelo 3D del robot, y MATLAB®, donde se programará el algoritmo de control y el generador inteligente de trayectorias constituye una potente herramienta para realizar el prototipado virtual [17]. Con dichos elementos y la integración del HIL[5] de los sensores de presencia en una Raspberry Pi, se conforma el sistema de control del cobot en lazo cerrado para validar los resultados de la estrategia de control implementada.

El presente trabajo esta organizado por capítulos, divididos por sus diferencias en cuanto a finalidad y contenido. Inmediatamente después de la introducción, se dan a conocer los objetivos, la justificación y el estado del arte. En el capítulo I se describen los conceptos teóricos básicos para comprender el lenguaje técnico y las generalidades de la tesis, en el capítulo II se abordan los fundamentos teóricos de un robot manipulador como lo son la cinemática directa e inversa, Jacobiano del manipulador, dinámica y planificación de trayectorias los cuales son piezas clave para la elaboración de este proyecto de investigación. Dentro del capitulo III se presenta la metodología seguida en este trabajo que va desde la selección y diseño en CAD[6] del robot hasta la validación de un algoritmo inteligente generador de trayectorias por medio de la co-simulación ADAMS®-MATLAB®. Por último, en el capitulo IV, se aborda la etapa de HIL, donde se sustituye el modelo dinámico del cobot por su correspondiente sistema embebido en Raspberry Pi y se lleva a cabo nuevamente la validación de la estrategia de control implementada mediante la red de comunicación entre la tarjeta del HIL del cobot y la tarjeta del HIL del sensor, en la cual reside también el algoritmo inteligente generador de trayectorias. Finalmente, se realiza el análisis y comparación de los resultados obtenidos en el desarrollo de las distintas etapas de Diseño Basado en Modelos presentadas en el trabajo de investigación.

[3]VP; acrónimo de la lengua inglesa *Virtual Prototyping.*
[4]CAE; acrónimo de la lengua inglesa *Computer Aided Engineering.*
[5]HIL; acrónimo de la lengua inglesa *Hardware in the Loop.*
[6]CAD; acrónimo de la lengua inglesa *Computer Aided Design.*

Objetivos

Los objetivos general y específicos de este trabajo de tesis se enuncian a continuación.

Objetivo general

Validar en tiempo real estrategias de control de una estación virtual de ensamble colaborativo entre un operador humano y un cobot en un contexto de Industria 4.0, siguiendo un proceso de Diseño Basado en Modelos.

Objetivos específicos

Objetivo 1: Modelar virtualmente el cobot y la estación de ensamble.

1. Obtener un modelo CAD del cobot.
2. Importar el modelo CAD del cobot en ADAMS® y ajustar los parámetros dinámicos.
3. Integrar en ADAMS® un objeto que emule la presencia de un operador humano en el espacio de trabajo del cobot.
4. Integrar en ADAMS® una mesa virtual que emule la estación de ensamble.

Objetivo 2: Implementar un algoritmo de control y co-simulación.

5. Implementar un algoritmo de control de posición en MATLAB® para el robot modelado.
6. Realizar la co-simulación ADAMS®–MATLAB®.

Objetivo 3: Desarrollar la cinemática del cobot.

7. Desarrollar la cinemática directa e inversa del cobot.

Objetivo 4: Desarrollar un algoritmo inteligente generador de trayectorias.

8. Desarrollar un algoritmo inteligente generador de trayectorias en MATLAB®.
9. Validar mediante co-simulación ADAMS®-MATLAB® el algoritmo generador inteligente de trayectorias.

Objetivo 5: Integrar el HIL de los sensores.

10. Reemplazar los objetos que emulan en ADAMS® la presencia de un operador humano en el espacio de trabajo del cobot por los correspondientes sensores de presencia conectados al robot mediante una tarjeta Raspberry Pi (HIL).

Objetivo 6: Integrar el MIL[7] en MATLAB® y PIL[8] del algoritmo inteligente.

11. Agregar a los modelos en MATLAB® el modelo dinámico del cobot (MIL).

12. Reemplazar el bloque del algoritmo inteligente generador de trayectorias por el correspondiente sistema embebido en una tarjeta Raspberry Pi (PIL).

Objetivo 7: Integrar el PIL del controlador y red de comunicación.

13. Reemplazar el bloque del algoritmo de control por el correspondiente sistema embebido en una tarjeta Raspberry Pi (PIL).

14. Desarrollar una red de comunicación de tarjetas Raspberry Pi.

Objetivo 8: Integrar tarjetas Raspberry Pi y validar.

15. Reemplazar el bloque de la planta (cobot) por el correspondiente sistema embebido en una tarjeta Raspberry Pi (HIL).

16. Integrar a la red de comunicación de tarjetas Raspberry Pi la tarjeta con la planta embebida.

17. Validar las estrategias de control mediante comparación de los últimos resultados de HIL con los resultados de MIL.

[7]MIL; acrónimo de la lengua inglesa *Model in the Loop*.
[8]PIL; acrónimo de la lengua inglesa *Processor in the Loop*.

Justificación

La ventaja de VP sobre el prototipado convencional es que se puede contar con un prototipo virtual que refleje fielmente el desempeño de un cobot en lazo cerrado con el ahorro de recursos materiales que conlleva evitar la fabricación de un prototipo tangible. Por otra parte, la interacción segura humano-robot en los robots colaborativos puede lograrse por varios mecanismos, entre los que se incluye disminuir la velocidad de movimiento del cobot o de plano detenerlo, si se corre riesgo de colisión con el operador humano [18]. Algunas ventajas de usar ADAMS® para el modelo virtual del cobot son:

- Acelerar la obtención de un modelo dinámico (virtual) del cobot para el análisis y las simulaciones en comparación con el tiempo que requiere la obtención del modelo analítico del robot.
- Permite la co-simulación con otras aplicaciones de software, entre ellas, MATLAB®.

Por su parte, MATLAB® representa una plataforma confiable para el análisis dinámico del cobot sujeto a un algoritmo de control. Ofrece también la facilidad de verificar el comportamiento dinámico de un modelo matemático en forma iterativa con sólo unos cuantos pasos. Resulta evidente, entonces, que aprovechando los beneficios de ADAMS® y de MATLAB® la co-simulación es el camino más llano para realizar un prototipo virtual confiable y en un tiempo razonable. Otra razón de peso para la consideración del proyecto de investigación propuesto es que su realización derivará en algunos beneficios para la comunidad académica, entre los que destacan:

- Contar con un entorno virtual para la evaluación de diferentes esquemas de control automático.
- La documentación de los resultados constituirá una base de conocimiento para estudiantes e investigadores que aborden el estudio de cobots en aplicaciones de manufactura.
- Contar con un método de prototipado virtual replicable y escalable a diferentes configuraciones geométricas de robots y en otras aplicaciones de manufactura que justifique el empleo de cobots.
- Conformar un entorno virtual que incorpora los efectos de sensores de presencia en el desempeño del cobot, entorno que puede ser escalado a otro tipo de sensores.
- Conformar sistemas SIL[9], PIL y HIL con recursos confiables para operación en tiempo real y a precios accesibles.

[9]SIL; acrónimo de la lengua inglesa *Software in the Loop*.

Estado del arte

Parte de las funciones de un robot colaborativo es poder realizar una tarea de manera secuencial evitando obstáculos, los cuales son detectados por medio de una red de sensores que dotan al cobot con la capacidad de detectar si algún objeto o humano se encuentra dentro de su área de trabajo, tal y como se trata en [19] en donde se propone un esquema de control de evasión de obstáculos en tiempo real para un manipulador de 6 grados de libertad que utiliza una cámara Kinect V2 RGB-D para rastrear obstáculos, incluidos humanos y objetos en el entorno de trabajo donde los resultados obtenidos son probados en el cobot TM5 700. De igual manera, el autor de [20] implementa un control de velocidad en lazo cerrado en un cobot UR10 midiendo la distancia en tiempo real por medio de una cámara Kinect 3D, logrando con esto una interacción segura entre el operador humano y el cobot, haciendo que este último logre reducir la velocidad cuando se detecte la presencia del operador dentro del área de trabajo del cobot.

Por su parte [21] brinda múltiples aspectos y consideraciones de la seguridad que debe tenerse en cuenta durante el diseño y despliegue de células de trabajo colaborativo humano-robot, puesto que como menciona [22] la participación de los cobots en aplicaciones industriales todavía está fuertemente influenciada por las estrategias para células robóticas no colaborativas; por ello, propone dentro de su trabajo un algoritmo basado en tareas que permite una interacción fluida a través de bioseñales de una Brain Compute Interface (BCI) como estrategia habilitadora de las estaciones de trabajo colaborativa, con lo cual proporciona un método de implementación de los dispositivos BCI para aplicaciones industriales con el fin de hacer que los cobots sean fáciles de usar e interactuar.

Una estrategia para programar y controlar de manera más amigable y sencilla a los robots en tareas de alta complejidad y que no impliquen un alto nivel de experticia del programador ni conocimiento de la plataforma robótica y las posibles condiciones de operación, y por tanto que no demanden un excesivo período de puesta en funcionamiento es el aprendizaje por demostración (LFD[10], por sus siglas en inglés). El autor W.D. Li menciona en su trabajo [23] que el modelo de mezcla Gaussiana y la regresión de mezcla Gaussiana (GMM y GMR; Gaussian Mixture Model y Gaussian Mixture Regression) son herramientas útiles para implementar este enfoque LFD; por esto, dentro de su trabajo diseña un enfoque mejorado basado en el aprendizaje por refuerzo (RL[11], de las siglas en inglés) para GMM / GMR para llevar a cabo una variedad de tareas complejas.

[10]LFD; acrónimo de la lengua inglesa *Learning From Demonstrations*.
[11]RL; acrónimo de la lengua inglesa *Reinforcement Learning*.

En palabras de Palleshi et al., en su trabajo [24] se menciona que el principal desafío actual de la robótica colaborativa es integrar las restricciones de seguridad humana sin comprometer los objetivos de rendimiento robótico, que requieren la minimización del tiempo de ejecución de la tarea además de garantizar su cumplimiento. Para abordar este desafío, propone un novedoso algoritmo de planificación de trayectorias robóticas para producir planes de movimiento seguros y de tiempo mínimo a lo largo de rutas específicas en espacios de trabajo compartidos con humanos.

Una manera de poder realizar las pruebas mecánicas necesarias que implican la validación de estrategias de control, la planificación de trayectorias, pruebas mecánicas y los diversos estudios que se realizan en un robot colaborativo es la implementación del prototipado virtual y el diseño basado en modelos; lo cual, como menciona [5], los ingenieros pueden continuamente probar los diseños, chequear los requisitos y encontrar errores oportunamente en el proceso de desarrollo, lo que implica mayor facilidad y menor coste para su corrección. Dentro de su trabajo aborda por medio del modelo basado en diseño la validación de una unidad electrónica de control (ECU, por sus siglas en inglés) para poder analizar las respuestas que daría el motor en función de las señales que esta genere.

El software ADAMS® es uno de los más utilizados para el modelado, análisis y simulación del movimiento de sistemas mecánicos complejos. Esto debido a que ofrece diversos beneficios tales como rendimiento, seguridad y comodidad, buena visibilidad y buenos resultados en tiempo real, en comparación con los resultados del modelo físico tal y como se menciona en [25] en donde se presentan una serie de pasos para realizar el modelado de un robot ABB IRB 1600-6/1.2 en ADAMS®, con la finalidad de implementar un análisis cinemático del robot y obtener parámetros tales como posición, velocidad y aceleración angular.

La co-simulación es el proceso dentro del cual se establece la conexión entre diferentes software, de tal manera que el modelo de un sistema de control que es construido en un software pueda tomar los resultados de sus cálculos como señales de entrada del sistema al modelo de la planta construido en otro software. Tal es el caso del presente trabajo de investigación en donde se hace uso de dos software, el primero, MATLAB/Simulink que es utilizado para desarrollar el sistema de control en lazo cerrado con el cual, durante el proceso de cálculo, se intercambian datos entre el prototipo virtual y el programa de control, donde ADAMS®, el segundo software, resuelve la ecuación del sistema mecánico mientras que MATLAB® resuelve la ecuación del sistema de control.

En la literatura existen diversos trabajos en donde se utiliza la co-simulación ADAMS®-MATLAB® para la implementación de estrategias de control de posición basada en PID en sistemas de 1 grado de libertad (gdl) [26], 2 gdl planar [27] y en sistemas de 3 gdl, como se trata en [28] para un manipulador paralelo tipo Delta. Otra aplicación relevante, en conjunto con el control de posición dentro del estudio de la co-simulación, es el seguimiento de trayectoria tal y como se realiza en [29] para un robot Gryphon de 5 gdl y en [30] para el prototipado virtual del robot Staubli TX40 de 6 gdl.

Capítulo 1

Marco teórico

1.1. Introducción a la robótica industrial

Los primeros trabajos que condujeron a los robots industriales de hoy día se remontan al período que siguió inmediatamente a la Segunda Guerra Mundial [31]. En 1948, George Devol consideró el diseño de una máquina flexible, adaptable al entorno y de fácil manejo, y patentó un manipulador programable que fue el principio de lo que se considera un robot industrial [32]. Años más tarde, en 1954, Goertz hizo uso de la tecnología electrónica y del servocontrol sustituyendo la transmisión mecánica por otra eléctrica y desarrollando así el primer sistema de telemanipulación con servocontrol bilateral, el cual se muestra en la figura 1.1; es decir, diseñó un manipulador cuya operación podía ser programada y que podía seguir una secuencia de pasos de movimientos determinados por las instrucciones en el programa [1], [31].

Figura 1.1: Sistema de telemanipulación con servocontrol bilateral [1].

En Marzo de 1954 el inventor británico C. W. Kenward solicitó la primera patente de un dispositivo robótico, la cual fue emitida en el Reino Unido en 1957 con el número 781.465. Sin embargo, fue George C. Devol, ingeniero norteamericano, inventor y autor de varias patentes, el que estableció las bases del robot industrial moderno [1]. Durante el año de 1956, Joseph F. Engelberger y Devol comenzaron a trabajar en la utilización de sus máquinas a nivel industrial, por lo cual crearon la *Universal Automation* [32]. En 1961 instalaron su primera máquina Unimate (Máquina de Transferencia Universal) en la fábrica de General Motors de Trenton, Nueva Jersey, en una aplicación de fundición por inyección como lo muestra la figura 1.2.

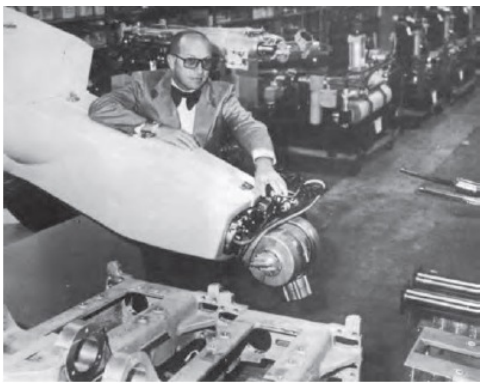


Figura 1.2: Joseph Engelberger con uno de los primeros Unimate [1].

Aunque los robots programados ofrecían una herramienta de fabricación nueva y potente, se hizo patente en los años sesenta que la flexibilidad de estas máquinas se podían mejorar significativamente mediante el uso de una realimentación sensorial [31]. En 1962, H. A. Ernest publicó el desarrollo de una mano mecánica controlada por computador con sensores táctiles. Este novedoso dispositivo, nombrado MH-1, podía sentir bloques y usar esta información para controlar la mano de manera que apilaba los bloques sin la ayuda de un operario [31].

A finales de los años de la década de los sesenta y principios de la década de los setenta se ponen las bases de la investigación en robótica en las universidades. Se crean departamentos de investigación en inteligencia artificial, y se diseñan los primeros robots móviles con un cierto grado de autonomía, como Shakey por el Stanford Research Institute o el Stanford Cart de la Universidad de Stanford [1]. También se ponen las bases de los diseños actuales de brazos manipuladores, principalmente por Victor Scheinman, con el diseño del brazo de Stanford [1]. Este robot fue el primer manipulador controlado por computadora y con accionamiento eléctrico, y que llevaría más tarde al diseño, fabricación y comercialización del robot PUMA, uno de los robots industriales más famosos y utilizados tanto en ambientes industriales como de investigación [32], el cual se muestra en la figura 1.3.

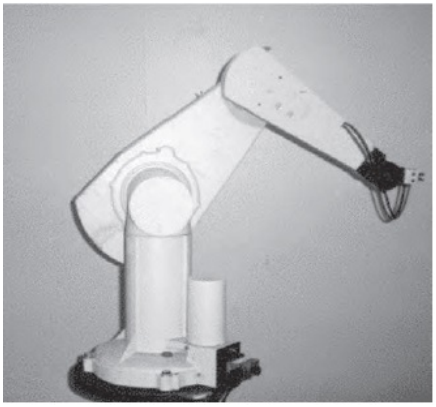


Figura 1.3: Robot PUMA [1].

El crecimiento de la robótica en Japón aventajó en un cierto tiempo a Estados Unidos debido a la formación de la primera asociación de robótica del mundo denominada Asociación de Robótica Industrial de Japón (JIRA) en 1972 [32]. Dos años más tarde, se formó el Instituto de Robótica de América (RIA), que en 1984 cambió su nombre por el de Asociación de Industrias Robóticas [33]. Por su parte, Europa tuvo un despertar más tardío, aunque no menos relevante. En 1973 la firma sueca ASEA construyó el primer robot con accionamiento totalmente eléctrico, el robot IRb6 que se muestra en la figura 1.4, seguido un año más tarde del IRb60. Posteriormente, ya como ABB, se convertiría en una de las empresas más importantes del mundo en la fabricación de robots industriales, y Suecia uno de los países con más robots per capita. En este sentido, en 1980 se fundó la Federación Internacional de Robótica con sede en Estocolmo, Suecia [1]. La empresa alemana KUKA también fue pionera en la fabricación e implantación de robots industriales, siendo una de las primeras en instalar una línea de soldadura equipada únicamente con robots [1].

Figura 1.4: Robot IRb6 de la empresa ASEA [1].

Además de los entornos de fabricación y de investigación, en la década de los setenta y principios de los ochenta los robots comenzaron a utilizarse de forma práctica en otros entornos, como el espacial o el submarino, haciendo uso principalmente de tecnologías de teleoperación [1]. A lo largo de la historia se ha podido constatar que son muchos los avances y aplicaciones que han permitido que los robots tengan gran aceptación en la industria, al grado de que se han convertido en elemento clave del proceso de automatización industrial debido a los beneficios que han traído consigo, tales como reducción de costos, incremento de productividad, mejoramiento de la calidad del producto y reducción de problemas en ambientes peligrosos al ser humano como, por ejemplo, ambientes radioactivos [34]. Una de las características principales de la robótica es que como área tecnológica tiene flexibilidad para automatizar y adaptarse a ambientes laborales y su tendencia siempre estará vigente [34]. Debido a esto, es importante su estudio, abordando aspectos como: características cinemáticas, dinámicas y aspectos de control, ya que de esta manera puede lograrse que los robots realicen nuevas tareas con un mejor desempeño [32].

1.1.1. Elementos que componen a un robot industrial

Los cobots se encuentran dentro de la clasificación de robots industriales para interacción con operadores humanos, dicha clasificación se tratará más adelante. Estos robots, en su mayoría, constan de los siguientes elementos: estructura mecánica, sistema sensorial, sistema de potencia y control, y elementos terminales [1], [32], [34]. Dentro de esta sección se aborda el enfoque de cadena cinemática abierta haciendo referencia a la estructura mecánica del robot.

Estructura mecánica

Mecánicamente, un robot está formado por una serie de elementos o eslabones unidos mediante articulaciones que permiten un movimiento relativo entre cada dos eslabones consecutivos. La constitución física de la mayor parte de los robots manipuladores industriales guarda cierta similitud con la anatomía del brazo humano, por lo que en ocasiones para hacer referencia a los distintos elementos que componen el robot se usan términos como cuerpo, brazo, codo y muñeca [1]. Cuando el dispositivo robótico cuenta con un conjunto determinado de articulaciones y eslabones y se llega a un punto en el cual no se encuentra conectado a ningún otro elemento se dice que dicho robot es de cadena cinemática abierta. De lo contrario, el robot es de cadena cinemática cerrada [32], la figura 1.5 muestra un ejemplo de cadena cinemática abierta y una cerrada.

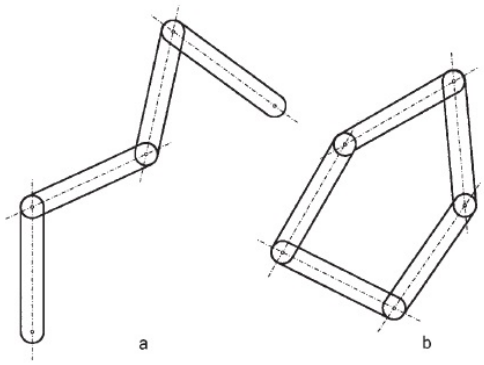


Figura 1.5: Cadena cinemática abierta, cadena cinemática cerrada [1].

Los eslabones de un robot manipulador presentan un movimiento relativo entre sí por medio de las articulaciones que los interconectan. Dependiendo el tipo de movimiento que produzcan las articulaciones del robot pueden ser de tipo rotacional o lineal (prismáticas) [34]. Las unidades de medición que se asocian a una articulación de tipo rotacional están dadas en radianes o grados, mientras que para una articulación de tipo lineal generalmente se encuentran en metros [34]. En 1876, Franz Reuleaux identificó los posibles movimientos relativos entre dos elementos en contacto. Reuleaux denominó pares inferiores (*lower pairs*) a aquellos en los que el contacto se realiza entre superficies, mientras que si el contacto es puntual o lineal, los denominó pares superiores [1]. La figura 1.6 muestra los seis posibles pares inferiores establecidas por Reuleaux [1].

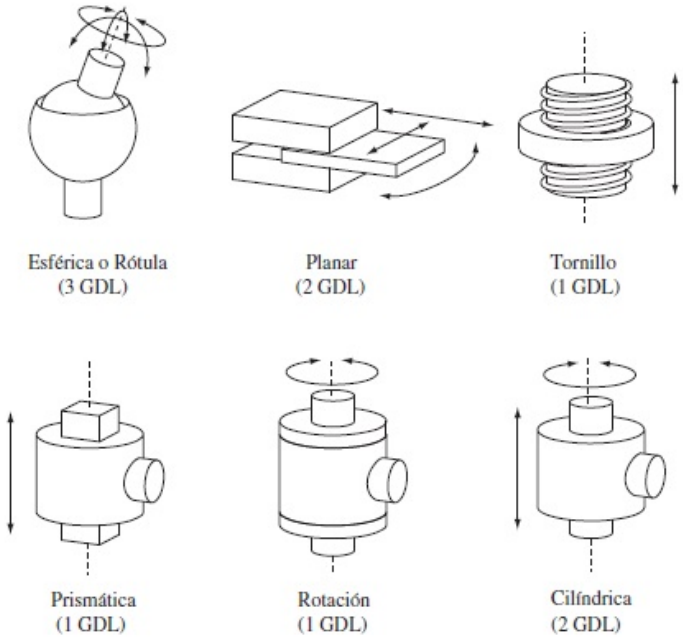


Figura 1.6: Los seis pares inferiores de Reuleaux. [1].

Un concepto importante para el robot es el de grados de libertad, que se refieren al conjunto de variables independientes que permiten determinar la posición y orientación del elemento final del robot. Los eslabones de un robot se mueven por medio de actuadores los cuales pueden ser hidráulicos, neumáticos o eléctricos [32], [34]. Los actuadores se encuentran acoplados con las articulaciones del robot; si las articulaciones son independientes con una única dirección de movimiento o rotación respecto a un eje determinado, el número de articulaciones con que cuenta un manipulador coincide con el número de grados de libertad [32].

Sistema sensorial

El robot como dispositivo mecánico puede posicionarse y orientarse en el espacio para realizar tareas determinadas. Obviamente, para poder asegurar que esas tareas se realizan convenientemente es necesario realizar un control de los movimientos a través de las posiciones y orientaciones que toma el robot en cada momento [33]. Con tal propósito es necesario disponer de una serie de sensores que faciliten la información necesaria del robot: posición, orientación, velocidad, fuerza, etc [33]. Generalmente, este tipo de sensores están ubicados en la propia estructura mecánica del robot, por lo que se les conoce como sensores internos, y siempre están presentes al menos algunos elementales [33]. Cuando además al robot se le quiere dotar de cierta flexibilidad y autonomía de actuación, son precisos los sensores externos para reconocer e interpretar el entorno que le rodea y poder realizar de forma correcta la tarea encomendada [33].

Sistema de potencia y control

Esta parte de un robot se compone de un sistema electrónico con la etapa de potencia encargada de suministrar energía al robot para su movimiento. Incluye un dispositivo llamado

teach-pendant, el cual brinda la interfaz necesaria para que el usuario se comunique con el robot a través de instrucciones de programación [34]. La consola de control también incluye los algoritmos de control programados en el sistema operativo del robot para guiar al robot. La capacidad del robot para llevar a cabo la tarea asignada con alto desempeño esta dada por el algoritmo de control, el cual puede decidir la ejecución de la acción con respecto a las restricciones impuestas por el sistema mecánico y el medio ambiente [34].

Elementos terminales

Los elementos terminales, también llamados *efectores finales* son los encargados de interaccionar directamente con el entorno del robot. Pueden ser tanto elementos de aprehensión como herramientas [1]. Para cada aplicación en particular será necesario un útil especifico a ser acoplado al extremo de la muñeca del robot. Existe un sin fin de efectores finales, por lo cual, se realiza la siguiente clasificación de los mismos [33]:

- Garras o elementos que permiten sujetar y manipular objetos.
- Herramientas que permiten realizar operaciones sobre objetos (pistola de pintura, atornillador, fresadora, lija, etc.).

1.1.2. Clasificación de los robots manipuladores

Un robot puede ser clasificado atendiendo a diferentes criterios o características. Algunas de éstas son dependientes de su propia esencia; otras, de la aplicación o tarea a que se destinan.

Por su número de ejes

Esta característica es aplicable a los robots o telerobot con cadena cinemática (es decir, sería aplicable a los robots manipuladores, pero no lo sería, por ejemplo, a los robots móviles). Se entiende por eje cada uno de los movimientos independientes con que está dotado el robot [1], [33].

- Robots con 3 ejes.
- Robots con 4 ejes.
- Robots con 5 o más ejes.

Por su tipo de control

Atendiendo al tipo de control, la norma ISO 8373 y, en consonancia la Federación Internacional de Robótica (IFR, por sus siglas en inglés), distingue entre los siguientes [1], [33]:

- **Secuencia controlada**. Robot con un sistema de control en el que un conjunto de movimientos se efectúa eje a eje en un orden dado, de tal forma que la finalización de un movimiento inicia el siguiente.

- **Trayectoria operada continua.** Robot que ejecuta un procedimiento controlado por el cual los movimientos de tres o más ejes controlados se desarrollan según instrucciones que especifican en el tiempo la trayectoria requerida para alcanzar la siguiente posición (obtenida normalmente por interpolación).
- **Adaptativo.** Robot que tiene un control sensorial adaptativo o funciones para control mediante aprendizaje. Se dice que el control es adaptativo si los parámetros del sistema de control son ajustados a partir de las condiciones detectadas durante el proceso.
- **Teleoperados.** Robot que puede ser controlado remotamente por un operador humano, extendiendo las capacidades sensoriales y motoras de éste a localizaciones remotas.

Por su área de aplicación en la industria

Desde el punto de vista del uso que se da a los robots es posible clasificarlos bien con base al sector económico en el que se encuentran trabajando o bien con base al tipo de aplicación o tarea que desarrollan, independientemente de en qué sector económico trabajen. Las actividades económicas pueden ser clasificadas de acuerdo al Estándar Internacional de Clasificación de Sectores de las Actividades Económicas (ISIC, por sus siglas en inglés) [1].

- Manipulación en fundición.
- Manipulación en moldeo de plásticos.
- Manipulación en tratamientos térmicos.
- Manipulación en la forja y estampación.
- Soldadura.
- Aplicación de materiales.
- Mecanización.
- Montaje.
- Paletización y empaquetado.
- Medición, inspección, control de calidad.
- Manipulación de materiales.
- Formación, enseñanza e investigación.
- Otros procesos.

Por su tipo de actuador

Dependiendo de cuál sea el tipo de energía utilizada por los ejes principales del robot, éste puede ser clasificado como robot neumático, hidráulico o eléctrico.

Por su configuración

Considerando el aspecto de geometría, las primeras tres articulaciones del robot que le brindan su posicionamiento, permiten clasificar al mismo como [32]:

- **Manipuladores articulados o RRR.** Algunas veces se les denomina rotacionales o antropomórficos. Las primeras tres articulaciones de estos manipuladores son rotacionales. Una de sus ventajas es que permiten una amplia variedad de movimientos en espacios reducidos.

- **Manipuladores RRP.** Dentro de esta clasificación se encuentran los manipuladores esféricos. Su nombre se debe a que la posición del efector final se puede describir por medio de coordenadas esféricas. En estos manipuladores las primeras dos articulaciones son rotacionales mientras que la tercera es prismática.

- **Manipuladores cilíndricos o RPP.** Para estos robots, las variables de las articulaciones pueden expresarse a través de coordenadas cilíndricas del efector final referido a la base. La primera articulación de los mismos es rotacional, mientras que las dos siguientes son prismáticas.

- **Manipuladores cartesianos o PPP.** En estos robots manipuladores, las variables de las articulaciones pueden manejarse mediante coordenadas cartesianas para referir el efector final con respecto a su base. Las primeras tres articulaciones de los mismos son prismáticas.

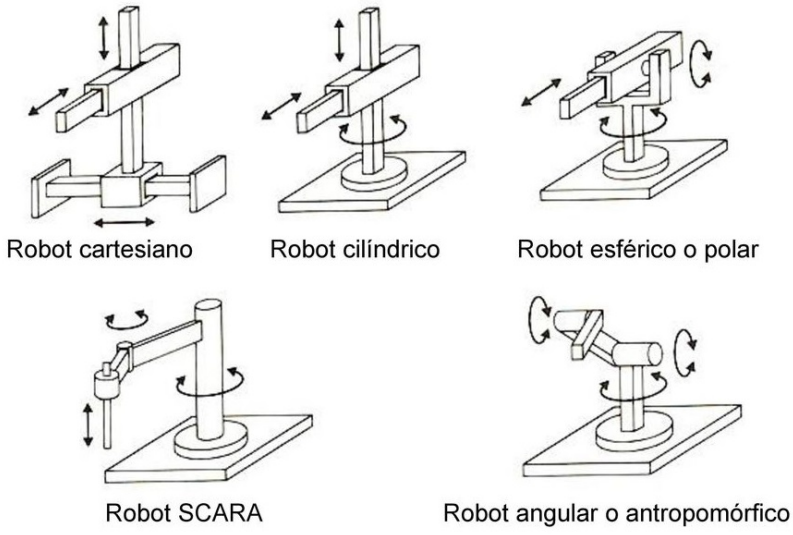


Figura 1.7: Configuración de robots considerando su geometría [1].

1.2. Industria 4.0

A lo largo de la historia se han presenciado varios procesos de transformación radical donde el avance tecnológico ha impactado sustancialmente en las condiciones materiales y sociales de la producción [35]. Tal es el caso de los procesos industriales, los cuales han atravesado tres grandes transformaciones tecnológicas a lo largo de la historia, dentro de las primeras tres revoluciones industriales se tuvieron avances tecnológicos representativos como fueron la creación de la máquina de vapor, la electricidad y la automatización que causaron un profundo impacto en la sociedad luego de su implementación.

El final del siglo XX trajo consigo una nueva transformación, el despliegue de la electrónica y la informática en los procesos industriales que permitió automatizar las líneas de producción y que las máquinas reemplazaran a las personas en tareas repetitivas [2]. Esto logró abrirle el paso a lo que hoy en día conocemos como la Cuarta Revolución Industrial, la cual se asocia con la digitalización de la producción, y con la generación, integración y análisis de una gran cantidad de datos a lo largo del proceso productivo y del ciclo de vida de los productos, facilitados fundamentalmente por internet [35]. Estos cambios en la industria se muestran de manera resumida en la figura 1.8.

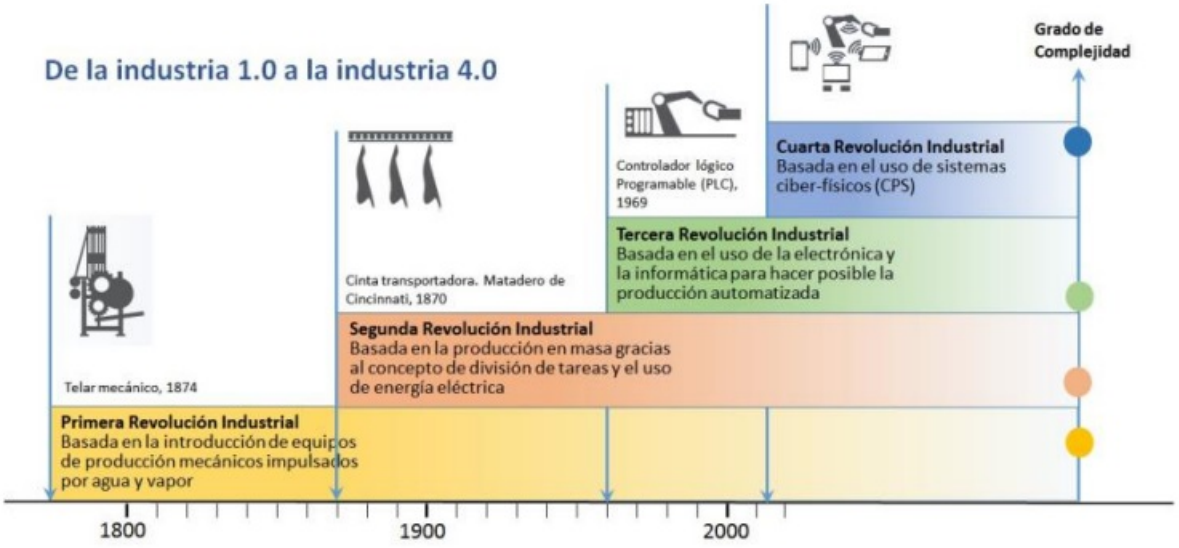


Figura 1.8: Evolución de las Revoluciones Industriales [2].

El término Industria 4.0 se utiliza de manera generalizada en Europa, si bien se acuñó en Alemania. También es habitual referirse a este concepto como "Fábrica Inteligente". Esta nueva ola de tecnologías que avanza en diversas regiones del mundo a distintas velocidades, implica una transformación a partir de nuevas tecnologías industriales con potencial de crear fábricas con procesos productivos totalmente integrados y automatizados, permitiendo que diferentes sistemas interactúen analizando información en tiempo real para optimizar la producción, predecir fallas e integrar las cadenas de suministros para volverlas más eficientes [36].

Dentro de la Industria 4.0 existen una variedad de pilares tecnológicos dentro de los cuales se destacan los siguientes [35], [37]:

- **Internet de las cosas.** Permite una comunicación de forma multidireccional entre máquinas, personas y productos, facilitando la toma de decisiones en base a la información que la tecnología recoge de su entorno. Utiliza nuevos sensores y actuadores que, en combinación con el análisis de big data y de computación en la nube, permite máquinas autónomas y sistemas inteligentes.
- **Big data y análisis de grandes datos.** Se refiere a datos caracterizados por su volumen, velocidad y variedad de datos estructurados.
- **Computación en la nube.** Ofrece almacenamiento, acceso y uso de servicios informáticos en línea.
- **Simulación de entornos virtuales.** Permite ajustar y representar virtualmente el funcionamiento conjunto de máquinas, procesos y personas en tiempo real antes de ser puestos en marcha, lo que ayuda a prevenir averías, ahorrar tiempo y evaluar el resultado final en un entorno controlado.
- **Inteligencia artificial.** Se basa en el desarrollo de algoritmos que permiten a las computadoras procesar datos a una velocidad inusual, logrando además el aprendizaje automático.
- **Ciberseguridad.** Es fundamental para que todas las demás tecnologías logren una adecuada penetración en esta fase de digitalización.
- **Máquinas y sistemas autónomos.** En el en mundo de la industria la tendencia es avanzar sobre la automatización de los procesos productivos, la navegación y el control, la integración de sensores y actuadores, la comunicación de las interfaces. Se busca incrementar la robótica colaborativa para ir hacia fábricas inteligentes donde todas las áreas de la empresa puedan trabajar en forma conectada y con un alto nivel de automatización en las tareas.

Las fábricas inteligentes o "smart factories" merecen mención especial al erigirse como uno de los principales elementos distintivos que mejor representa esta Industria 4.0. Éstas constituyen un "hogar", un ecosistema, donde conviven la digitalización de extremo a extremo, los procesos y productos inteligentes, y las tecnologías habilitadoras. Gracias a la convergencia de estos factores, veremos fábricas más autónomas, ágiles, dinámicas, flexibles, y optimizadas, pero también más complejas debido a los crecientes niveles de integración, funcionalidad y conectividad [2].

Tradicionalmente, la totalidad de componentes que integraban las plantas productivas se comportaban como bloques aislados o células individuales; sin embargo, en las fábricas del futuro éstos se convierten en nodos de red inteligentes. Estos nodos agregarán información constantemente, tanto del estado de los productos como de las máquinas y de los procesos, funcionando así como una comunidad cooperativa. Se busca conseguir todo esto de una manera sencilla, de ahí que se esté empezando a utilizar el concepto de *plug and play* como idea base que guíe la construcción, el funcionamiento y la gestión de las instalaciones [2], [38].

1.3. Cobots

Partiendo de la premisa anterior, la robótica colaborativa aparece de la necesidad del ser humano de interaccionar con robots en el sector industrial, ya que la sociedad y sus exigencias aumentan cada vez más, siendo necesario optimizar los recursos disponibles, disminuyendo tiempos y costo de producción [39].

Sin embargo, esta necesidad no era posible suplirla ya que los brazos robóticos disponibles en la industria no pueden interaccionar con un ser humano cuando están en funcionamiento porque supone un gran peligro para la persona interponerse en la zona de trabajo de dichos robots, es por ello que se utilizan cercas protectoras con la finalidad de disminuir los accidentes entre los operadores humanos y este tipo de robots. Además, dichos robots son costosos, de gran tamaño y muy pesados, y disminuir todos estos factores era una de las principales motivaciones que perseguía el concepto de robótica colaborativa. Pero el factor más importante es romper con el peligro que supone la interacción de un ser humano con los robots industriales existentes hasta el momento [39].

Los robots colaborativos o cobots son unos robots especialmente diseñados para trabajar compartiendo espacio con los humanos e incluso realizando tareas que requieren de la interacción con ellos. Se crearon en 1996 por profesores de la Northwestern University en Illinois, EEUU. La empresa KUKA en una colaboración con el Centro Aerospacial Alemán desarrolló el primer robot colaborativo de uso industrial, el LBR 3 en el año 2004 [37]. Sin embargo, no fue hasta el año 2010 que su diseño, producción y su uso empezó a externderse. Tras el desarrollo del LBR 3 por parte de KUKA, otras grandes empresas del sector también desarrollaron y lanzaron sus modelos de cobots, como pueden ser Universal Robots o FANUC [37].

Los cobots ofrecen una gran variedad de ventajas respecto a los robots industriales tradicionales de las cuales destacan [37], [39], [40]:

- Los robots colaborativos complementan las capacidades del hombre y permiten la automatización de los pasos de producción que anteriormente se realizaban de forma manual.

- Las tareas de los robots colaborativos se pueden adaptar de forma flexible. Además, los robots colaborativos pueden utilizarse con flexibilidad de posicionamiento y con unos requisitos de espacio reducidos.

- El cobot puede operar a velocidades normales cuando ningún ser humano entre en la zona de trabajo. Si una persona entra en el área de trabajo, entonces detecta su presencia, en ese momento el cobot ralentiza su marcha y restringe el movimiento para evitar cualquier peligro de posible contacto.

- Sencillez a la hora de instalarlos y programarlos. Muchos de ellos ofrecen métodos por los que los usuarios de estos robots no requieren de tener estudios superiores o de saber métodos convencionales de programación.

1.3.1. Interacción segura operador humano-cobot

Los robots colaborativos, como se mencionó anteriormente son robots industriales diseñados para interaccionar directamente con un humano dentro de un espacio de trabajo cooperativo sin que sea necesaria la existencia de un espacio de seguridad aislado, como en el caso de un robot industrial convencional. En robots colaborativos pueden existir diferentes grados de interacción humano-robot, según los parámetros de tiempo y espacio. Las situaciones mostradas a continuación no son excluyentes entre ellas. Se estima que las aplicaciones 100 % colaborativas serán minoritarias y lo habitual serán operaciones con cierto grado de colaboración, alternando con otras no colaborativas [3].

- **Cooperación (colaboración indirecta).** Mismo espacio de trabajo, tiempos diferentes (por ejemplo, alternancia de trabajos sobre misma pieza).

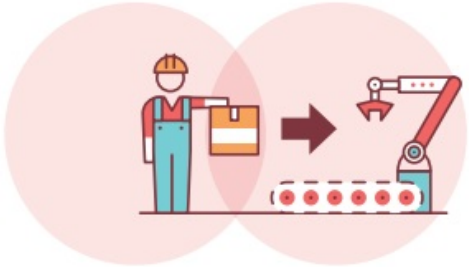


Figura 1.9: Colaboración indirecta [3].

- **Coexistencia.** Mismo tiempo de trabajo, espacios diferentes (por ejemplo, el trabajo en diferentes piezas y zonas).

Figura 1.10: Coexistencia [3].

- **Colaboración directa.** Mismo espacio de trabajo al mismo tiempo (por ejemplo, el trabajo simultáneo sobre misma pieza).

Figura 1.11: Colaboración directa [3].

Para garantizar la seguridad de los operadores humanos, en los últimos años se han introducido estrategias que apuntan a diferentes tipos de seguridad, que incluyen [21]:

- Seguridad de colisión para garantizar que solo las colisiones seguras/controladas solo puedan tener lugar entre robots, humanos y obstáculos. La limitación de la fuerza ejercida sobre los humanos es el principal objetivo.

- Seguridad activa para detectar oportunamente colisiones inminentes entre humanos y equipos y detener la operación de manera controlada. Para tal efecto, podrán utilizarse sensores de proximidad, sistemas de visión y sensores de fuerza/contacto.

A raíz de lo anterior, se han incluido normas, directivas y leyes nacionales e internacionales para permitir que los diseñadores de sistemas integren fácilmente la seguridad en sus sistemas. Considerando el hecho de que un espacio de trabajo colaborativo no solo involucra al ser humano y al robot, sino también a otros dispositivos auxiliares (por ejemplo, destornilladores eléctricos, dispositivos de sujeción eléctricos, etc.), cada celda de trabajo presenta riesgos únicos que deben manejarse con seguridad. Por lo tanto, también se deben respetar las normas y leyes para cada tipo de equipo y operación [21].

A diferencia de los robots no colaborativos, los cobots tienen requisitos críticos extra de seguridad que cumplir ya que una colaboración humano-robot generalmente ocurre en un entorno dinámico y no muy bien estructurado, además la coexistencia de humanos y cobots en un espacio compartido forma una peligrosa situación que requiere mecanismos apropiados para garantizar la seguridad de ambos como se mencionó anteriormente. Es por esto que el diseño de los mecanismos de seguridad debe cumplir con los estándares industriales correspondientes [4].

En primer lugar, un cobot debe cumplir con los requisitos de seguridad como maquinaria general, una serie de leyes y directivas, como la Directiva Europea de Maquinaria (2006/42/EC). En segundo lugar, los componentes utilizados en un cobot deben cumplir con los estándares tipo A de módulos funcionales, por ejemplo, seguridad funcional según IEC 62,508:2010, evaluación de riesgos según ISO 12,100: 2010 y dispositivos de seguridad y controles activos según ISO 12,100. En tercer lugar, la aplicación específica de un sistema cobot debe cumplir con los estándares de tipo B; por ejemplo, las aplicaciones de fabricación por ISO 11,161:2006. Finalmente, los cobots como productos deben cumplir estándares tipo C; por ejemplo, celdas

de trabajo robóticas según ISO 10,218-2:2011 y robots según ISO 10,218-1:2011 [4]. En general, un cobot debe cumplir leyes y normas de un robot industrial común y además cumplir con estándares meramente creados para robots colaborativos. La siguiente tabla 1.1 contiene las normas, directivas y leyes nacionales e internacionales en lo relativo al uso de robótica colaborativa de acuerdo con [4], [21], [38].

Título	**Descripción**
EN ISO 10218-1	Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales - Parte 1: Robots.
EN ISO 10218-2	Robots y dispositivos robóticos. Requisitos de seguridad para robots industriales - Parte 2: Sistemas de robot e integración.
ISO/TS 15066: 2016	Robots y dispositivos robóticos – Robots colaborativos.
EN ISO 13849-1/2	Seguridad de las máquinas. Partes de los sistemas de control relacionadas con la seguridad. Parte 1: Principios generales para el diseño. Parte 2: Validación.
EN ISO 12100: 2010	Seguridad de las máquinas. Principios generales para el diseño. Evaluación y reducción de riesgos

Tabla 1.1: Normas, directivas y leyes nacionales e internacionales en lo relativo al uso de robótica colaborativa.

Actualmente, hay casi 30 directivas activas y alrededor de 600 estándares diferentes relacionados con la seguridad. Específicamente para las celdas robóticas, los tres primeros estándares de la tabla 1.1 proveen una abundante cantidad de estrategias diferentes para lograr la seguridad. Algunas de las más importantes implican [4], [21], [38]:

- **Funciones de parada de robot**: se requiere que cada robot tenga una función de parada de protección, así como una función de parada de emergencia independiente. También deben proporcionar conexión a equipos de seguridad externos.
- **Control de velocidad**: la velocidad del efector final del robot y del punto central de la herramienta (TCP, por sus siglas en inglés) deben ser controlables. Especialmente para lugares de trabajo colaborativos, la velocidad del TCP no debe exceder los 250 mm/s.
- **Establecimiento de la distancia mínima de separación**: dependiendo de la aplicación, se utiliza una evaluación de riesgos para determinar la distancia mínima de separación entre el robot y el operador. La evaluación considera a) Los peligros asociados con el efector final y las partes que puede contener, b) el diseño del espacio de trabajo, c) las tareas de los operadores y d) la usabilidad del sistema.
- **Detección de colisión**: la función de seguridad debe determinar si las posiciones y velocidades actuales tanto del robot como del ser humano pueden hacer que la distancia de separación disminuya por debajo del valor mínimo (colisión).
- **Requisitos de operación colaborativa**: los robots diseñados para la operación colaborativa deben proporcionar una indicación visual cuando el robot está en alguna operación colaborativa como se muestra en la figura 1.12. Se aplica para los siguientes casos:

- Parada supervisada con clasificación de seguridad. El robot debe detenerse cuando un humano está en el espacio de trabajo colaborativo y puede reanudar la operación automática cuando el humano sale de él.

- Guiado manual. El equipo de guía manual debe incluir una parada de emergencia y un dispositivo de habilitación. Durante esta operación, la velocidad del robot debe ser monitoreada de forma segura y certificada. Se pueden usar varias tecnologías, como el control de la impedancia o la rigidez, para permitir el guiado manual por parte del ser humano.

- Monitoreo de velocidad y posición. El robot debe mantener una distancia de separación del operador, monitoreada por características integrales o una combinación de entradas externas.

- Limitación de potencia y fuerza por diseño inherente. Las funciones de limitación de potencia/fuerza deben respetar los límites establecidos por las normas y, si se superan, se debe emitir una parada.

- Limitación de potencia y fuerza por sistema de control. Esto prevé que se utilice una función de control para garantizar que no se excedan los valores máximos de potencia y fuerza.

Figura 1.12: Mecanismos de aseguramiento de la seguridad [4].

1.4. Tiempo real en la industria

El tiempo real se refiere a la capacidad de un sistema o aplicación para responder a estímulos externos dentro de límites de tiempo definidos [41], [42]. Esta característica es esencial en entornos donde la sincronización y la toma de decisiones oportunas son críticas. Existen dos categorías principales de tiempo real: blando y duro. En los sistemas de tiempo real blando, se permite cierta tolerancia a retrasos, mientras que en los sistemas de tiempo real duro, no se admite ninguna violación de tiempo debido a sus consecuencias graves o catastróficas. Los sistemas de tiempo real blando son adecuados para aplicaciones donde una tardanza ocasional no tiene un impacto significativo, mientras que los sistemas de tiempo real duro son necesarios en situaciones donde incluso un pequeño retraso puede tener consecuencias desastrosas [41], [42].

En la industria, el tiempo real es esencial para el control preciso de maquinarias y procesos de producción ya que permite la monitorización y el ajuste inmediato de variables clave, asegurando la calidad, la eficiencia y la seguridad en las operaciones [43]. Dentro de los entornos industriales donde la interacción entre humanos y robots es común, el tiempo real garantiza la detección temprana de situaciones de riesgo y la respuesta ante ellas contribuyendo a un entorno de trabajo más seguro y eficiente. Es por esto que dentro del campo de la robótica resulta importante el concepto de tiempo real ya que es crucial para el control preciso de los movimientos de los robots y su interacción con el entorno y los humanos. Los sistemas robóticos que trabajan en tiempo real deben de tener una respuesta precisa a las entradas sensoriales, lo que es fundamental para tareas como la navegación autónoma y la manipulación de objetos [44]. Por lo tanto la comunicación en tiempo real entre el robot y el control son esenciales para garantizar una interacción segura y eficiente en entornos de trabajo compartidos [45].

Es importante destacar que, dependiendo de la tarea que esté realizando el robot dentro del área de trabajo, puede ser factible utilizar tanto un tiempo real duro como un tiempo real blando. En ciertos escenarios, donde la tarea del robot requiere una respuesta sin tolerancia a retrasos, se debe optar por un tiempo real duro [46]. Por ejemplo, en aplicaciones de seguridad donde el robot debe evitar colisiones con objetos en movimiento, se necesita una respuesta sin retrasos para garantizar la seguridad del entorno como ocurre con los sistemas de control aéreo, sistemas médicos críticos y sistemas de control de procesos industriales [47]. Sin embargo, en otras situaciones donde la tarea del robot no es tan crítica en términos de tiempo de respuesta, se puede utilizar un tiempo real blando que permita cierta tolerancia a retrasos [48], tal es el caso de las aplicaciones de control de calidad donde el robot realiza inspecciones visuales, un pequeño retraso en la respuesta del robot podría ser aceptable sin comprometer significativamente la calidad del resultado. No obstante, es importante tener en cuenta que, incluso en un tiempo real blando, existen límites en cuanto al retraso máximo permitido para mantener la calidad del resultado ya que un retraso en la respuesta del robot que exceda el máximo de tiempo permitido podría afectar la detección de defectos en una línea de producción.

Por lo tanto, aunque se permita cierta tolerancia a retrasos, es necesario establecer límites para garantizar la efectividad de la tarea realizada por el robot teniendo en cuenta que los rangos de tolerancia específicos pueden variar según el contexto, los requisitos de la aplicación y las especificaciones del sistema en particular.

1.5. Diseño Basado en Modelos

El Diseño Basado en Modelos proporciona un entorno de diseño único que permite a los desarrolladores utilizar un único modelo de todo su sistema para análisis de datos, visualización, pruebas y validación y la posibilidad de la generación de código automático [5]. Habilita a los desarrolladores a evaluar múltiples opciones, predecir la actuación del sistema, probar la funcionalidad imponiendo entradas o salidas y probar diferentes diseños a través de la simulación, permitiendo su uso incluso en aplicaciones de tiempo real [5].

Una definición que se encuentra en la literatura [6] sobre el MBD, habla de que es un método matemático y visual que facilita los diseños complejos de sistemas embebidos. Ya que en lugar de usar códigos de programación extensos y complicados, los diseñadores pueden usar MBD para definir modelos con características funcionales avanzadas usando simulaciones de tiempo continuo y tiempo discreto. Los componentes principales de este método son el diseño de una simulación a nivel de sistema y componente, la generación automática de código y las pruebas y verificaciones continuas [6].

El MBD puede considerarse una metodología de desarrollo de software basada en el método V presentado en la figura 1.13 [6]. Este método es una representación gráfica del ciclo de vida de desarrollo de un sistema. Define una metodología de gestión de proyectos creada por la Administración Federal Alemana y de Defensa. Su parte izquierda descompone las necesidades mientras que su parte derecha representa la integración y su verificación. Este método minimiza los riesgos, así como los costes totales del proyecto, mejorando y garantizando la calidad [5].

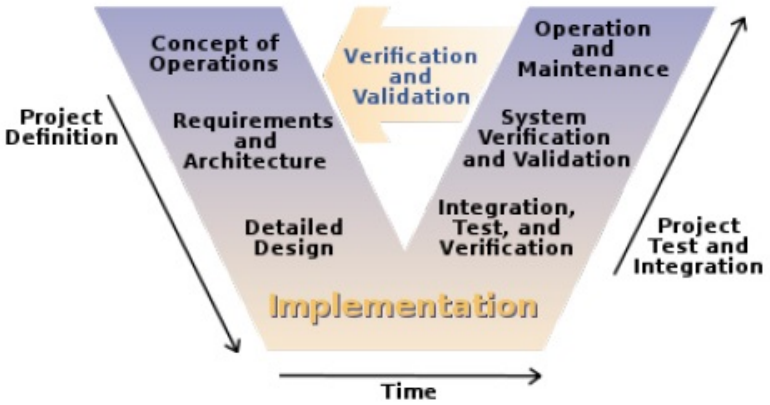


Figura 1.13: Método V [5].

La metodología que implica el Diseño Basado en Modelos y que se basa en el método V tratado anteriormente se muestra en la figura 1.14.

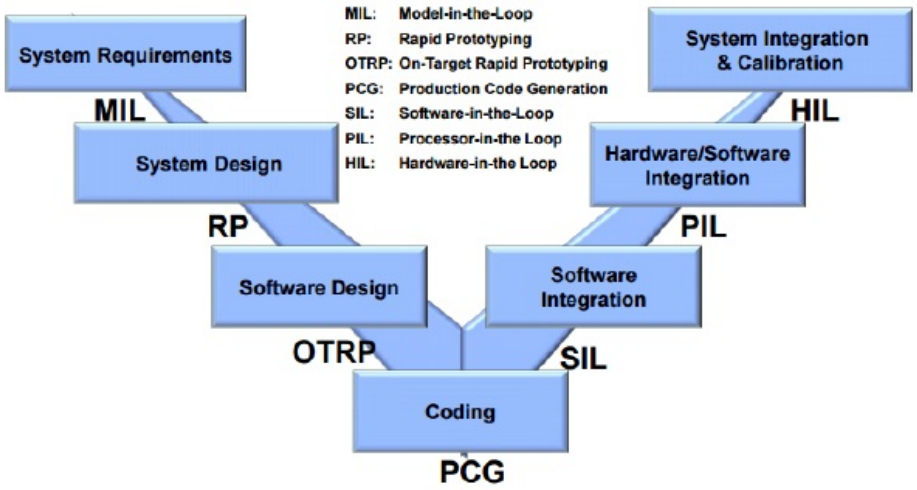


Figura 1.14: Método V aplicado al diseño basado en modelos [5].

Se hace referencia a que la metodología del MBD está basada en el método V debido a que como se puede observar en el diagrama, se parte del lado izquierdo con los requerimientos y diseño del sistema a evaluar, mientras que el lado derecho corresponde a las etapas de verificación e integración del sistema, por lo que el diseño de software comienza antes de disponer de prototipos y sistemas físicos que estén disponibles, anticipando la corrección de errores tardía [6]. Es decir, en la primera etapa (MIL), se busca desarrollar el modelo de la planta a través de datos experimentales, modelos matemáticos o una combinación de ambas, así como el diseño del controlador mediante una máquina de estado y/o diagrama de bloques, y durante la etapa final (HIL) se integran todos los sistemas desarrollados y son verificados para validar que el modelo cumpla con las especificaciones y requisitos para posteriormente poder ser desarrollado, o bien, ser aplicado en algún prototipo físico [5]. A continuación, se realiza un descripción más detallada sobre cada una de las etapas del MBD que son de particular interés para el presente trabajo de tesis.

1.5.1. Etapas del Diseño Basado en Modelos

Model in the loop (MIL)

El escenario MIL es una técnica que se utiliza para abstraer el comportamiento de un sistema o subsistema de modo que este modelo se pueda utilizar para probar, simular y verificar ese modelo [6]. Por ejemplo, el sistema de control de una máquina eléctrica basado en un regulador PID con etapa de potencia, sería posible ajustar y probar su correcto funcionamiento en una planta modelada formada por la etapa de potencia y la máquina eléctrica. Al utilizar una cadena de herramientas estándar de la industria como Simulink para definir el modelo, es posible probar y refinar ese modelo dentro de una computadora, lo que permite administrar un sistema complejo de manera eficiente [6].

Software in the loop (SIL)

A diferencia del MIL, el escenario SIL permite probar el código generado a partir del modelo también en una computadora donde es posible realizar la simulación, pero en base al código del modelo. Es decir, en el mismo ejemplo del sistema de control anterior, el controlador PID basado en un modelo, se genera su código y por lo tanto se prueba en el mismo entorno. Esta fase requiere solo el modelo de simulación y es independiente del hardware, centrándose en las interfaces de software y los resultados numéricos [6].

Processor in the loop (PIL)

PIL es un proceso de prueba en el que el código compilado que describe un controlador se puede ejecutar en un procesador digital de señales (DSP[1], por sus siglas en inglés) externo, FPGA[2], procesador común o microprocesador y el modelo de la planta se ejecuta en un simulador offline, ambos subsistemas, controlador y planta, están vinculados por un protocolo de comunicación o intercambio por un puerto que permita el intercambio de datos. Aunque PIL se usa típicamente para evaluar un controlador; no obstante, puede resultar útil probar partes de un sistema complejo, donde dichas partes pueden integrarse en la plataforma digital y el resto del sistema puede ejecutarse en el simulador offline. Lo importante de PIL es que permite depurar el controlador o el subsistema bajo prueba para corregir posibles errores sobre su desempeño [49].

Hardware in the loop (HIL)

Las pruebas de HIL son simulaciones en tiempo real que permiten comenzar a probar el código embebido sin necesidad de hardware de sistema. Esto permite probar condiciones anormales y de fallo que pueden dañar el hardware si el código en desarrollo no opera dentro de las especificaciones [50]. El rendimiento de la simulación de HIL depende tanto de la complejidad de la dinámica del sistema que se modela como del hardware informático en tiempo real que se utiliza. En la simulación HIL, se utiliza una computadora en tiempo real como una representación virtual de su modelo de planta y una versión real de su controlador [50].

A manera de resumen, considérese la figura 1.15, la cual ilustra las etapas de dicho proceso.

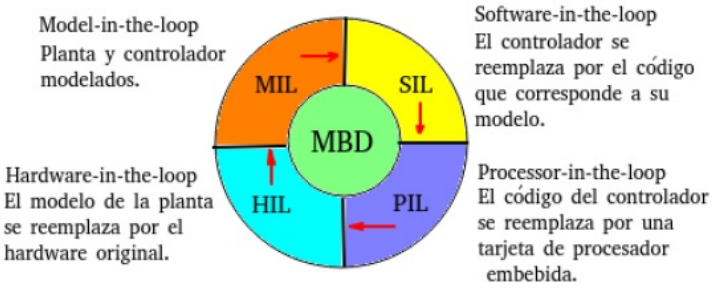


Figura 1.15: Proceso de Diseño Basado en Modelos [6].

[1]DSP; acrónimo de la lengua inglesa *Digital Signal Processor*.
[2]FPGA; acrónimo de la lengua inglesa *Field Programmable Gate Arrays*.

1.6. Prototipado virtual

El prototipado virtual es el diseño y generación de un producto con suficientes características claves que permiten su evaluación frente a los requerimientos propios de su aplicación. La denominación "virtual" indica que el producto aún no es desarrollado físicamente; sin embargo, existe una representación visual que permite realizar observaciones sobre aspectos constructivos, análisis de movimientos relativos y absolutos [28]. Se busca que el prototipo virtual posea el mismo comportamiento que el modelo físico; no obstante, éste se encuentra en un ambiente virtual generado por computadora. Dentro de la literatura se encuentran diversas definiciones de prototipado virtual como la anterior, dependiendo del marco de trabajo de los investigadores, algunas de ellas son:

- El prototipo virtual se trata de la presentación, prueba y análisis de modelos CAD tridimensionales antes de crear cualquier prototipo físico [51].
- Por prototipado virtual, nos referimos al proceso de simular el usuario, el producto y su interacción dentro del software a través de diferentes etapas del diseño de un producto, además del análisis de desempeño cuantitativo del mismo [52].
- En la definición de prototipado virtual de ingeniería mecánica, la idea es reemplazar modelos (maquetas) físicos por prototipos de software. Esto incluye también toda clase de simulaciones funcionales y geométricas, involucrando o no personas [53].

Con las definiciones anteriores se puede decir entonces que un prototipo virtual:

- Es un modelo de una estructura o aparato (producto).
- Usado para probar forma, satisfacción de diseño, desempeño, y facilidad de manufactura.
- Usado para estudio y aprendizaje.
- Un prototipo virtual requiere que preste las mismas funciones o aún más que su homólogo físico.
- Permite la evaluación sensorial de un producto, como color, características estéticas, tacto, adaptabilidad, entre otros aspectos como la ergonomía.

1.6.1. Métodos de prototipado virtual

La creación de prototipos virtuales consta y requiere de muchas capacidades, de las cuales la más conocida es la creación y visualización de modelos sólidos tridimensionales con varios colores y texturas superficiales [51]. Una cantidad relativamente grande de herramientas de simulación de software están integradas directamente en los paquetes CAD tridimensionales existentes o las ofrecen proveedores externos como módulos adicionales. En un nivel superior, el prototipado virtual puede incluso incluir la simulación de usuarios humanos y sus interacciones con un producto y su entorno [51]. La utilización de este tipo de herramientas computacionales que apoyan el diseño y fabricación de un producto ha sido una constante desde hace ya varias décadas, ejemplo de estas herramientas son [54], [55]:

- **Diseño asistido por computadora (CAD).**

  Comprende los sistemas informáticos (software) para el diseño asistido por computadora cuya codificación permite generar modelos en ambiente 2D y 3D con todas las características de un determinado producto, a fin de obtener una representación de gran precisión del objeto deseado. El CAD puede combinarse a otras tecnologías como lo es la Ingeniería Asistida por Computadora (CAE, por sus siglas en inglés) para permitir el desarrollo integral de un proyecto desde su fase de diseño hasta su producción en línea, con lo que se consigue un importante ahorro de tiempo, mínima intervención humana y mayor precisión de diseño. Además este tipo de software permite ensamblar diferentes componentes para visualizar la interacción entre ellos. Ejemplo de este tipo de software es SolidWorks.

- **Ingeniería asistida por computadora (CAE).**

  Corresponde al uso de software computacional para simular el desempeño y así poder realizar mejoras a los diseños de producto o bien apoyar a la resolución de problemas de ingeniería para una amplia gama de industrias. Esto incluye el diseño, simulación, validación y optimización de productos, procesos y herramientas de manufactura. Las aplicaciones CAE soportan una gran variedad de disciplinas y fenómenos de la ingeniería, incluyendo el análisis de cinemática y de dinámica de mecanismos, análisis de control de sistemas y el análisis de sistemas de control.

  Ejemplo de este tipo de software es ADAMS®, el cual permite crear y probar fácilmente prototipos virtuales de sistemas mecánicos en una fracción del tiempo y el costo requeridos para la construcción y prueba física. A diferencia de la mayoría de las herramientas integradas de CAD, este software incorpora física real al resolver simultáneamente ecuaciones para cinemática, estática, cuasiestática y dinámica [56]. Otra de las grandes ventajas de utilizar este software es que dentro de los módulos adicionales se incluye ADAMS/Control, el cual ayuda a integrar fácilmente los mundos de la simulación de movimiento y el diseño de sistemas de control de una manera verdaderamente multidisciplinaria. Con este módulo, se pueden incorporar los modelos de ADAMS® dentro de diagramas de bloques en software de diseño de sistema de control. Como alternativa, también se puede importar actuadores y/o controladores directamente desde el software de diseño de controles al entorno de simulación de ADAMS®.

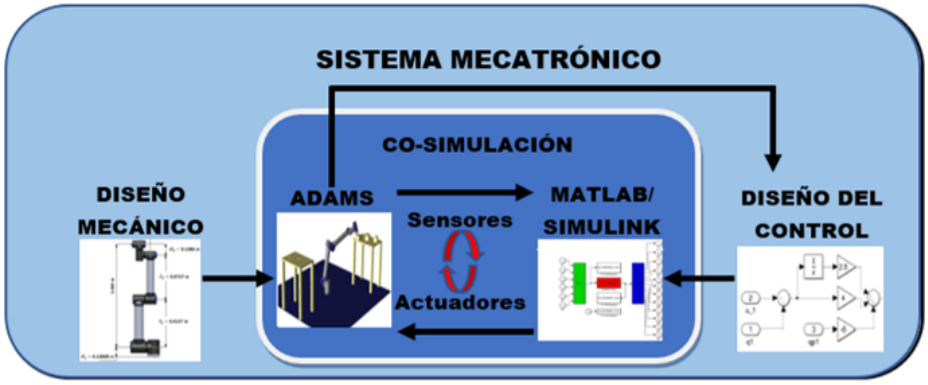


Figura 1.16: Integración de ADAMS®-MATLAB®.

Capítulo 2

Cinemática y dinámica de un robot manipulador

Dentro de este capítulo se abordan los fundamentos teóricos de un robot manipulador como lo son la cinemática directa e inversa, Jacobiano del manipulador, dinámica y planificación de trayectorias los cuales son piezas clave para la elaboración de este proyecto de investigación.

2.1. Cinemática

La cinemática se encarga de estudiar el movimiento de los objetos sin considerar las fuerzas que intervienen. Así pues, en el campo de la robótica la cinemática se interesa por la descripción analítica del movimiento espacial del robot como una función del tiempo, y en particular por las relaciones entre la posición y la orientación del extremo final del robot con los valores que toman sus coordenadas articulares [1]. Existen dos problemas fundamentales a resolver en la cinemática del robot; el primero de ellos se conoce como el problema cinemático directo, y consiste en determinar cuál es la posición y orientación del extremo final del robot con respecto a un sistema de coordenadas que se toma como referencia; el segundo, denominado problema cinemático inverso, resuelve la configuración que debe adoptar el robot para una posición y orientación del efector final conocidas [1].

2.2. Cinemática directa

El estudio de la cinemática directa de robots manipuladores industriales proporciona elementos para analizar y diseñar el desplazamiento de trayectorias del robot, así como la orientación del efector final del mismo [34]. La obtención del modelo cinemático directo puede ser abordado mediante dos enfoques diferentes denominados métodos geométricos y métodos basados en cambios de sistemas de referencia [1]. Los primeros son adecuados para casos simples, pero al no ser sistemáticos, su aplicación queda limitada a robots con pocos grados de libertad. Los métodos basados en cambio de sistemas de referencia, permiten de una manera sistemática abordar la obtención de la cinemática directa del robot, en particular, para robots de n grados de libertad, siendo estos, por tanto, los más utilizados frecuentemente [1].

2.2.1. Métodos geométricos

Este método es relativamente sencillo, no obstante, al no ser sistemático, apenas sirve para manipuladores de pocos grados de libertad o configuraciones muy concretas como lo son los robots planares [1]. Debido a que no existe un procedimiento específico para resolver la cinemática directa mediante este método, resulta de ello, por lo general, inoperativo para robots de mayor número de grados de libertad.

2.2.2. Métodos de cambio de sistema de referencia

Dado que un robot se puede considerar como una cadena cinemática formada por objetos rígidos o eslabones unidos entre sí mediante articulaciones, se puede establecer un sistema de referencia fijo en la base del robot y describir la localización de cada uno de los eslabones con respecto a dicho sistema de referencia [1]. De esta forma el problema cinemático se reduce a encontrar una matriz de transformación homogénea que relacione la posición y orientación del extremo final del robot con respecto al sistema de referencia fijo situado en la base del mismo [1].

Transformaciones homogéneas

La transformación homogénea es una herramienta matemática que involucra operaciones de rotación y traslación dentro de una matriz que estructura el modelo de cinemática directa [34]. La notación más común para representar la matriz de rotación y el vector de traslación en forma compacta se conoce como transformación homogénea la cual está dada por la siguiente expresión [34]:

$$H = \begin{bmatrix} R_{z,\theta} & \mathbf{d}_0^1 \\ \mathbf{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \text{Matriz de rotación} & \vdots & \text{Vector de traslación} \\ \cdots & \vdots & \cdots \\ \mathbf{0}^T & & 1 \end{bmatrix} \tag{2.1}$$

donde $R_{z,\theta} \in SO(3)$ es una matriz ortogonal de rotación y $\mathbf{d}_0^1 \in \mathbb{R}^3$ es un vector de traslación. Para propósitos de acoplamiento de dimensiones, el vector $\mathbf{0}^T$ y el número 1 aparecen en el último renglón [34]. Las matrices de rotación permiten modelar la orientación de la herramienta de trabajo colocada en el extremo final del robot, y junto con las tranformaciones homogéneas dentro de una sola matriz incluye la orientación y posición de la herramienta de trabajo, formando la estructura del modelo cinemático [34].

Convención Denavit-Hartenberg (D-H)

El método Denavit-Hartenberg, que fue propuesto en 1955 por Jaques Denavit y Richard S. Hartenberg, consiste en determinar una tabla de parámetros relacionados con los eslabones del robot por medio de la cual se recurre a un método matricial que establece la localización que debe tomar cada sistema de coordenadas S_i ligado a cada eslabón i de una cadena cinemática articulada [1], [34]. Frente a otros métodos de cálculo para la cinemática, D-H presenta un mecanismo sistemático para la obtención de la matriz de transformación homogénea que representa la relación entre el sistema de coordenadas del efector final respecto a la base del

robot. D-H se basa en que es posible representar cada una de estas transformaciones entre los consecutivos sistemas de referencia de cada eslabón mediante 4 transformaciones básicas que consisten en una sucesión de rotaciones y traslaciones que permiten relacionar el sistema de referencia del elemento $i-1$ con el sistema del elemento i [1].

Las transformaciones en cuestión son las siguientes [1]:

1. Rotación alrededor del eje $\mathbf{z}_{i-1}$ un ángulo θ_i.
2. Traslación a lo largo de $\mathbf{z}_{i-1}$ una distancia d_i.
3. Traslación a lo largo de $\mathbf{x}_i$ una distancia a_i.
4. Rotación alrededor del eje $\mathbf{x}_i$ un ángulo α_i.

Dado que el producto de las matrices no es conmutativo, las transformaciones se han de realizar en el orden indicado. De este modo se tiene [34]:

$$
\begin{aligned}
H_i =& R_{z,\theta_i} T_{z,d_i} T_{x,l_i} R_{x,\alpha_i} \\
=& \begin{bmatrix} \cos(\theta_i) & -\text{sen}(\theta_i) & 0 & 0 \\ \text{sen}(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
& \begin{bmatrix} 1 & 0 & 0 & l_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_i) & -\text{sen}(\alpha_i) & 0 \\ 0 & \text{sen}(\alpha_i) & \cos(\alpha_i) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
=& \begin{bmatrix} \cos(\theta_i) & -\text{sen}(\theta_i)\cos(\alpha_i) & \text{sen}(\theta_i)\text{sen}(\alpha_i) & l_i\cos(\theta_i) \\ \text{sen}(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\text{sen}(\alpha_i) & l_i\text{sen}(\theta_i) \\ 0 & \text{sen}(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\tag{2.2}
$$

Las variables articulares que se presentan en la convención D-H (figura 2.1) son denotadas por θ_i para el tipo rotacional, prismática o lineal por d_i, la longitud del eslabón está representada por l_i y el ángulo de separación entre los ejes z_i y z_{i-1} es denotada α_i.

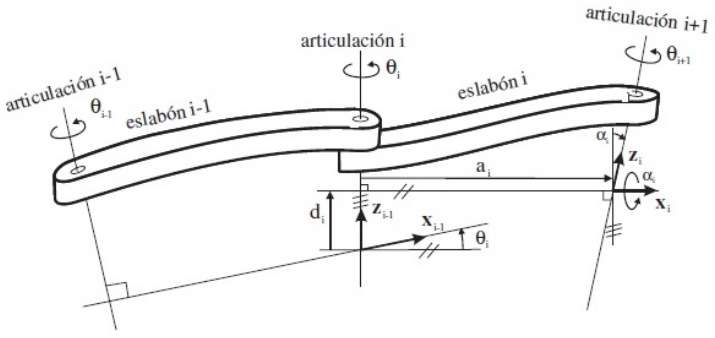


Figura 2.1: Parámetros de la convención D-H [1].

El ángulo θ_i es el ángulo entre los ejes x_{i-1} y x_i medido alrededor del eje z_{i-1}. d_i es la distancia del origen del sistema de referencia $i-1$ a la intersección del eje x_i con el eje z_{i-1}. Su medición se realiza a lo largo del eje z_{i-1}. En el caso de articulaciones rotacionales d_i es un parámetro constante y representa el $offset$ o espesor del servomotor [34]. Adicionalmente a las variables articulares d_i y θ_i, hay dos parámetros constantes que describen características específicas del eslabón i-ésimo. Esos parámetros son: el parámetro l_i o a_i y se define como la distancia a lo largo del eje x_i desde el origen del sistema de referencia coordenado $i-1$ hasta la intersección del eje z_{i-1} con el eje x_i [34]. El ángulo entre los ejes z_i y z_{i-1} se denota por α_i, su medición es respecto a un plano normal a x_i. Una medición de ángulo positivo para α_i se toma en dirección del eje z_{i-1} hacia z_i [34].

La obtención de la cinemática directa del cobot seleccionado en esta investigación se realiza mediante el método basado en cambios de sistemas de referencia para lo cual, en la figura 2.2 se muestra el prototipado virtual del cobot UR10 con las correspondientes medidas reales que aparecen en [57] y, así mismo, en la figura 2.3, se presenta al robot en una posición de casa conveniente con sus correspondientes sistemas de referencia colocados a partir del algoritmo de D-H [1]. Estas medidas, en conjunto con las relaciones de los sistemas de referencia de los elementos S_{i-1} al S_i, permiten obtener los 4 parámetros necesarios para el llenado de la tabla 2.1 de Denavit-Hartenberg correspondiente al robot.

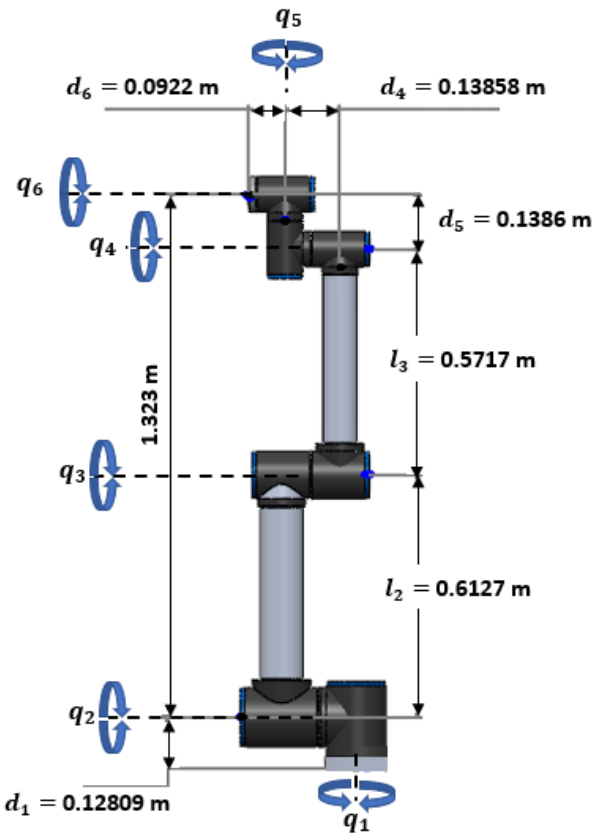


Figura 2.2: Medidas reales de los eslabones del cobot UR10.

Eslabón	l_i	α_i	d_i	θ_i
1	0	$\frac{\pi}{2}$	d_1	q_1
2	l_2	0	0	q_2
3	l_3	0	0	q_3
4	0	$\frac{\pi}{2}$	d_4	q_4
5	0	$-\frac{\pi}{2}$	d_5	q_5
6	0	0	d_6	q_6

Tabla 2.1: Parámetros D-H del cobot UR10.

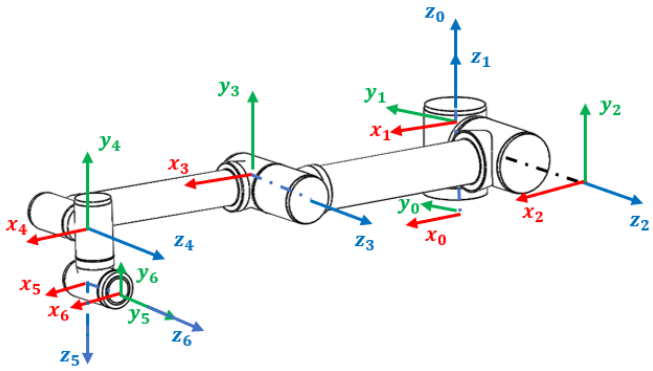


Figura 2.3: Localización de los sistemas de referencia del cobot UR10.

La matriz de transformación homogénea para cada articulación se obtiene a partir de la sustitución de los parámetros de la tabla 2.1 en la matriz (2.2) quedando de la siguiente manera:

$$H_0^1 = R_{z,\theta_1} T_{z,d_1} T_{x,l_1} R_{x,\alpha_1} = \begin{bmatrix} \cos(q_1) & 0 & \text{sen}(q_1) & 0 \\ \text{sen}(q_1) & 0 & -\cos(q_1) & 0 \\ 0 & 1 & 0 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

$$H_1^2 = R_{z,\theta_2} T_{z,d_2} T_{x,l_2} R_{x,\alpha_2} = \begin{bmatrix} \cos(q_2) & -\text{sen}(q_2) & 0 & l_2\cos(q_2) \\ \text{sen}(q_2) & \cos(q_2) & 0 & l_2\text{sen}(q_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

$$H_2^3 = R_{z,\theta_3} T_{z,d_3} T_{x,l_3} R_{x,\alpha_3} = \begin{bmatrix} \cos(q_3) & -\text{sen}(q_3) & 0 & l_3\cos(q_3) \\ \text{sen}(q_3) & \cos(q_3) & 0 & l_3\text{sen}(q_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.5}$$

$$H_3^4 = R_{z,\theta_4} T_{z,d_4} T_{x,l_4} R_{x,\alpha_4} = \begin{bmatrix} \cos(q_4) & 0 & \text{sen}(q_4) & 0 \\ \text{sen}(q_4) & 0 & -\cos(q_4) & 0 \\ 0 & 1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

$$\begin{aligned} H_4^5 =& R_{z,\theta_5} T_{z,d_5} T_{x,l_5} R_{x,\alpha_5} \\ =& \begin{bmatrix} \cos(q_5) & 0 & -\text{sen}(q_5) & 0 \\ \text{sen}(q_5) & 0 & \cos(q_5) & 0 \\ 0 & -1 & 0 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.7}$$

$$\begin{aligned} H_5^6 =& R_{z,\theta_6} T_{z,d_6} T_{x,l_6} R_{x,\alpha_6} \\ =& \begin{bmatrix} \cos(q_6) & -\text{sen}(q_6) & 0 & 0 \\ \text{sen}(q_6) & \cos(q_6) & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{2.8}$$

La matriz de transformación homogénea que representa la relación entre el sistema de coordenadas del efector final con respecto a la base del robot está dada por:

$$H_0^6 = H_0^1 H_1^2 H_2^3 H_3^4 H_4^5 H_5^6 \tag{2.9}$$

$$H_0^6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

Debido a que las expresiones resultantes dentro de cada elemento de la matriz son muy extensas, se hace uso de las siguientes propiedades trigonométricas para reducir las expresiones:

$$\begin{aligned} \text{sen}(a+b) &= \text{sen}(a)\cos(b) + \text{sen}(b)\cos(a) & \qquad \text{sen}(a-b) &= \text{sen}(a)\cos(b) - \text{sen}(b)\cos(a) \\ \cos(a+b) &= \cos(a)\cos(b) - \text{sen}(a)\text{sen}(b) & \qquad \cos(a-b) &= \cos(a)\cos(b) + \text{sen}(a)\text{sen}(b) \end{aligned}$$

A partir de aplicar dichas propiedades se obtienen resultados simplificados para los cuales se utiliza la siguiente nomenclatura:

$$\begin{aligned} \text{s}_{234} &= \text{sen}(q_2 + q_3 + q_4) \\ \text{c}_{234} &= \cos(q_2 + q_3 + q_4) \end{aligned}$$

cada elemento de la matriz H_0^6 está dado por:

$$n_x = c_6 s_1 s_5 + c_1 c_5 c_6 c_{234} - s_6 c_1 s_{234} \tag{2.11}$$
$$n_y = -c_6 c_1 s_5 + s_1 c_5 c_6 c_{234} - s_6 s_1 s_{234} \tag{2.12}$$
$$n_z = s_6 c_{234} + c_5 c_6 s_{234} \tag{2.13}$$
$$o_x = -s_6 s_5 s_1 - s_6 c_5 c_1 c_{234} - c_6 c_1 s_{234} \tag{2.14}$$
$$o_y = s_6 c_1 s_5 - s_6 c_5 s_1 c_{234} - c_6 s_1 s_{234} \tag{2.15}$$
$$o_z = c_6 c_{234} - c_5 s_6 s_{234} \tag{2.16}$$
$$a_x = c_5 s_1 - s_5 c_1 c_{234} \tag{2.17}$$
$$a_y = -c_5 c_1 - s_5 s_1 c_{234} \tag{2.18}$$
$$a_z = -s_5 s_{234} \tag{2.19}$$
$$p_x = d_5 c_1 s_{234} + d_4 s_1 + d_6 c_5 s_1 - d_6 s_5 c_1 c_{234} + l_2 c_1 c_2 + l_3 c_1 c_{23} \tag{2.20}$$
$$p_y = d_5 s_1 s_{234} - d_4 c_1 - d_6 c_5 c_1 - d_6 s_5 s_1 c_{234} + l_2 s_1 c_2 + l_3 s_1 c_{23} \tag{2.21}$$
$$p_z = d_1 + l_2 s_2 - d_5 c_{234} + l_3 s_{23} - d_6 s_5 s_{234} \tag{2.22}$$

2.3. Cinemática inversa

La cinemática inversa consiste en encontrar los valores que deben adoptar las coordenadas articulares del robot $\boldsymbol{q} = [q_1, q_2, ..., q_n]^T$ para que su efector final se posicione y oriente según una determinada localización espacial [1]. A diferencia de la cinemática directa, la cinemática inversa se complica debido a que el proceso de obtener las variables articulares en función de la posición y orientación del efector final involucra un conjunto de ecuaciones no lineales que pueden tener múltiples soluciones o incluso la solución puede no existir [32]. Si el problema de la cinemática inversa tiene múltiples soluciones, significa que el manipulador puede adoptar más de un conjunto de valores de variables articulares que le permitan obtener el valor deseado de posición y orientación. Si la solución no existe, se relaciona éste hecho con posiciones y/u orientaciones que el manipulador no puede obtener, por ejemplo si se encuentra fuera de su espacio de trabajo [32].

Así como es posible abordar la cinemática directa de manera sistemática a partir de la utilización de matrices de transformación homogénea independientemente de la configuración del robot, no ocurre lo mismo con la cinemática inversa, siendo el procedimiento de obtención de las ecuaciones fuertemente dependiente de la configuración del robot [1]. Las técnicas para resolver el problema de la cinemática inversa pueden ser muy diversas y de modo general pueden clasificarse en [32]:

- **Soluciones en forma cerrada o analíticas**: se trata de soluciones o expresiones no iterativas que proporcionan de modo directo el valor deseado.
- **Soluciones numéricas**: se trata de procedimientos iterativos que permiten la determinación del valor de una variable determinada.

El problema de este último tipo de soluciones es que la velocidad de convergencia e incluso su convergencia en si no siempre está garantizada [1].

En las soluciones en forma cerrada puede emplearse el método geométrico, o el algebraico. Los métodos geométricos permiten, normalmente, obtener los valores de las primeras variables articulares, que son las que consiguen posicionar el robot (prescindiendo de la orientación del efector final). Para ello, se utilizan relaciones trigonométricas y geométricas sobre los elementos del robot. Se suele recurrir a la resolución de triángulos formados por los elementos y articulaciones del robot [1]. El método algebraico consiste en el uso de expresiones algebraicas basadas en relaciones con matrices de transformación homogénea para obtener una solución cerrada [32].

El primer paso a dar para resolver la cinemática inversa mediante este método es obtener la matriz de transformación homogénea H_0^i que relaciona el sistema de referencia asociado a la base con el sistema de referencia asociado al extremo final del robot [1], [32]. Del análisis cinemático directo se sabe que la matriz H_{i-1}^i es una función de la variable q_i por lo tanto el problema de cinemática inversa puede resolverse mediante la solución de las siguientes ecuaciones para las variables de las articulaciones ejemplificándolo con un robot manipulador de 6 grados de libertad se obtienen las siguientes ecuaciones [32]:

$$H_1^6 = \left[H_0^1\right]^{-1} H_0^6 \tag{2.23}$$
$$H_2^6 = \left[H_1^2\right]^{-1} \left[H_0^1\right]^{-1} H_0^6 \tag{2.24}$$
$$H_3^6 = \left[H_2^3\right]^{-1} \left[H_1^2\right]^{-1} \left[H_0^1\right]^{-1} H_0^6 \tag{2.25}$$
$$H_4^6 = \left[H_3^4\right]^{-1} \left[H_2^3\right]^{-1} \left[H_1^2\right]^{-1} \left[H_0^1\right]^{-1} H_0^6 \tag{2.26}$$
$$H_5^6 = \left[H_4^5\right]^{-1} \left[H_3^4\right]^{-1} \left[H_2^3\right]^{-1} \left[H_1^2\right]^{-1} \left[H_0^1\right]^{-1} H_0^6 \tag{2.27}$$
$$I = \left[H_5^6\right]^{-1} \left[H_4^5\right]^{-1} \left[H_3^4\right]^{-1} \left[H_2^3\right]^{-1} \left[H_1^2\right]^{-1} \left[H_0^1\right]^{-1} H_0^6 \tag{2.28}$$

La razón por la cual las ecuaciones anteriores resuelven el problema de la cinemática inversa es que, por ejemplo, empleando la primera ecuación (2.22) se observa que:

- La matriz $\left[H_0^1\right]^{-1}$ es una función de q_1, mientras que los elementos H_1^6 son cero, constantes o funciones de $q_2, ..., q_6$.
- Los elementos constantes proporcionan ecuaciones algebraicas que se resuelven en función de q_1 empleando funciones trigonométricas.

Las mismas conclusiones anteriores pueden ser establecidas para las demás ecuaciones y así obtener ecuaciones para las variables $q_2, ..., q_6$. Esta técnica es algunas veces denominada **Técnica de Pieper** [32]. Para aplicar las fórmulas anteriores es necesario obtener la matriz inversa de las correspondientes matrices de transformación homogénea la cual viene dada por [1]:

$$\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} n_x & o_x & a_x & -\boldsymbol{n^T p} \\ n_y & o_y & a_y & -\boldsymbol{o^T p} \\ n_z & o_z & a_z & -\boldsymbol{a^T p} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

En la presente investigación, se obtiene la cinemática inversa mediante el método geométrico en conjunto con el método algebraico. Este procedimiento en sí, se basa en encontrar el suficiente número de relaciones geométricas en las que intervendrán las coordenadas del extremo final del robot, sus coordenadas articulares y las dimensiones físicas de sus elementos [1]. A continuación se muestra el desarrollo y la obtención de cada variable articular para el robot de 6 grados de libertad.

Comenzando con en el método geométrico, se realiza un análisis para determinar la posición del sistema de referencia 5 con respecto al sistema de referencia 0 que se encuentra situado en la base del robot. Esto puede llevarse a cabo trasladando el sistema de referencia 6 correspondiente al efector final solamente realizando la resta de la distancia d_6 que existe entre el sistema de referencia 5 y el sistema de referencia 6.

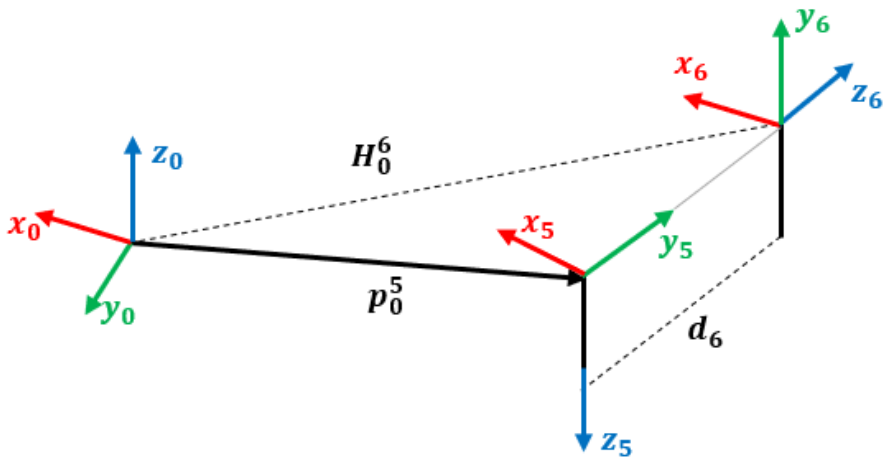


Figura 2.4: Distancia entre los sistemas de referencia 5 y 6.

La traslación descrita anteriormente y que se muestra en la figura 2.4 se interpreta de la siguiente manera:

$$p_0^5 = p_0^6 - d_6 \cdot a_0^6 \tag{2.29}$$

$$p_0^5 = H_0^6 \begin{bmatrix} 0 \\ 0 \\ -d_6 \\ 1 \end{bmatrix} \tag{2.30}$$

A continuación, en la figura 2.5, se puede ver que la rotación de θ_1 se puede analizar con respecto a la posición del sistema de referencia 5, donde la distancia que hay entre el origen y la posición del sistema de referencia 5 está dada por el vector de posición p_{0xy}^5 con respecto al sistema de referencia 0.

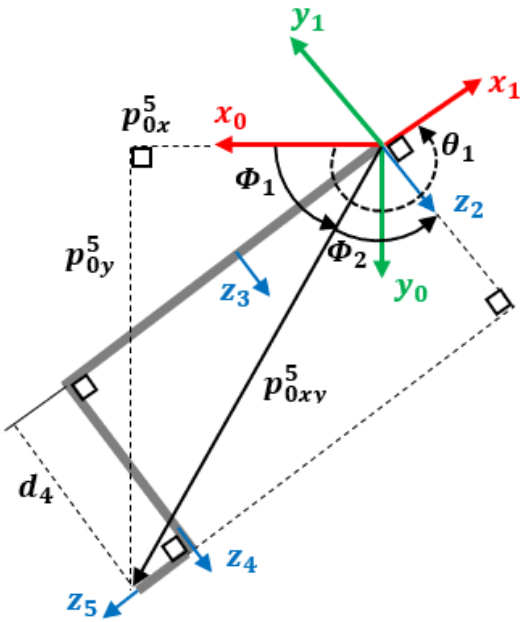


Figura 2.5: Distancia entre los sistemas de referencia 5 y 1.

El ángulo θ_1 puede ser obtenido de la siguiente manera:

$$\phi_1 = \text{atan2}(p^5_{0y}, p^5_{0x}) = \text{atan2}\left(\frac{p^5_{0y}}{p^5_{0x}}\right) \tag{2.31}$$

Y por su parte, el ángulo ϕ_2 puede ser calculado como:

$$\cos(\phi_2) = \frac{d_4}{p^5_{0xy}} \tag{2.32}$$

por lo tanto

$$\phi_2 = \pm\text{acos}\left(\frac{d_4}{p^5_{0xy}}\right) = \pm\text{acos}\left(\frac{d_4}{\sqrt{\left(p^5_{0x}\right)^2 + \left(p^5_{0y}\right)^2}}\right) \tag{2.33}$$

La ecuaciones anteriores permiten encontrar el ángulo correspondiente a θ_1 que en este caso corresponde a la variable articular q_1 quedando de la siguiente manera:

$$q_1 = \text{atan2}(p^5_{0y}, p^5_{0x}) \pm \text{acos}\left(\frac{d_4}{\sqrt{\left(p^5_{0x}\right)^2 + \left(p^5_{0y}\right)^2}}\right) + \frac{\pi}{2} \tag{2.34}$$

El ángulo de la variable articular q_1 se obtuvo mediante el sistema de referencia 5 con respecto al origen, lo cual servirá para el cálculo de la variable articular q_5.

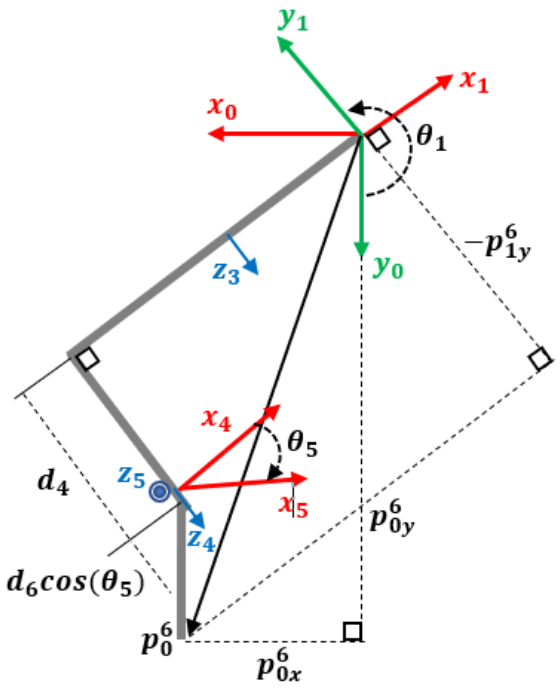


Figura 2.6: Distancia entre los sistemas de referencia 6 y 1.

En la figura 2.6 se observa que la componente $-p^6_{1y}$ solamente depende del ángulo θ_5, por lo tanto, la distancia que hay entre los sistemas de referencia 1 y 6 está dada por la siguiente ecuación:

$$-p^6_{1y} = d_4 + d_6\cos(\theta_5) \tag{2.35}$$

Con la expresión anterior y con base a las componentes del vector p^6_0 se obtiene la siguiente expresión:

$$p^6_{1y} = p^6_{0x}(\text{-sen}(\theta_1)) + p^6_{0y}\cos(\theta_1) \tag{2.36}$$

sustituyendo $-p^6_{1y}$ tenemos

$$-d_4 - d_6\cos(\theta_5) = p^6_{0x}(\text{-sen}(\theta_1)) + p^6_{0y}\cos(\theta_1) \tag{2.37}$$

Por lo tanto, el valor correspondiente de θ_5 que representa el valor de la variable articular q_5 se obtiene de la siguiente expresión:

$$q_5 = \pm\text{acos}\left(\frac{p^6_{0x}\text{sen}(q_1) - p^6_{0y}\cos(q_1) - d_4}{d_6}\right) \tag{2.38}$$

Para determinar la expresión del ángulo θ_6 se examina el eje y_1 visto desde el sistema de referencia 6 y, como se observa en la figura 2.7, el ángulo θ_6 puede ser asumido en coordenadas esféricas. Por otro lado, también es posible utilizar el método algebraico para encontrar una expresión que relacione estos dos ángulos, debido a que la matriz H_0^6 y el ángulo θ_1 son conocidos se puede utilizar la siguiente igualdad basado en la Técnica de Pieper:

$$\left[H_0^1\right]^{-1} H_0^6 = H_1^2 H_2^3 H_3^4 H_4^5 H_5^6 \tag{2.39}$$

Realizando el desarrollo de las operaciones matriciales se obtiene en ambos lados de la igualdad lo siguiente:

$$\begin{bmatrix} n_x c_1 + n_y s_1 & o_x c_1 + o_y s_1 & a_x c_1 + a_y s_1 & p_x c_1 + p_y s_1 \\ n_z & o_z & a_z & p_z - d_1 \\ n_x s_1 - n_y c_1 & o_x s_1 - o_y c_1 & a_x s_1 - a_y c_1 & p_x s_1 - p_y c_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.40}$$

$$= \begin{bmatrix} c_{234}c_5c_6 - s_{234}s_6 & -s_{234}c_6 - c_{234}c_5s_6 & -c_{234}s_5 & d_5 s_{234} + l_3 c_{23} + l_2 c_2 + d_6 s_5 c_{234} \\ c_{234}s_6 + s_{234}c_5c_6 & c_{234}c_6 - s_{234}c_5s_6 & -s_{234}s_5 & d_5 c_{234} + l_3 s_{23} + l_2 s_2 - d_6 s_5 s_{234} \\ c_6 s_5 & -s_5 s_6 & c_5 & d_4 + d_6 c_5 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Como se puede observar en los resultados de ambas matrices, se pueden relacionar los elementos de ambos lados de la igualdad que se encuentran en la columna 1 y 2 de la fila 3 para poder encontrar la siguiente expresión correspondiente a la variable articular q_6:

$$q_6 = \text{atan2}\left(\frac{o_x \text{sen}(q_1) - o_y \cos(q_1)}{\text{sen}(q_5)}, \frac{n_x \text{sen}(q_1) - n_y \cos(q_1)}{-\text{sen}(q_5)}\right) \tag{2.41}$$

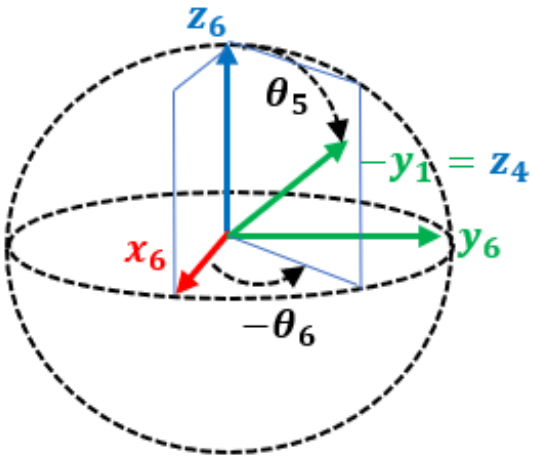


Figura 2.7: Sistema de coordenadas esférico.

Para la obtención de θ_3 se realiza el análisis de la figura 2.8, en donde se puede observar que los eslabones se encuentran en paralelo uno con otro lo cual se aprecia de mejor manera en la figura 2.3, esto genera que el robot pueda considerarse como un robot planar de 3 grados de libertad.

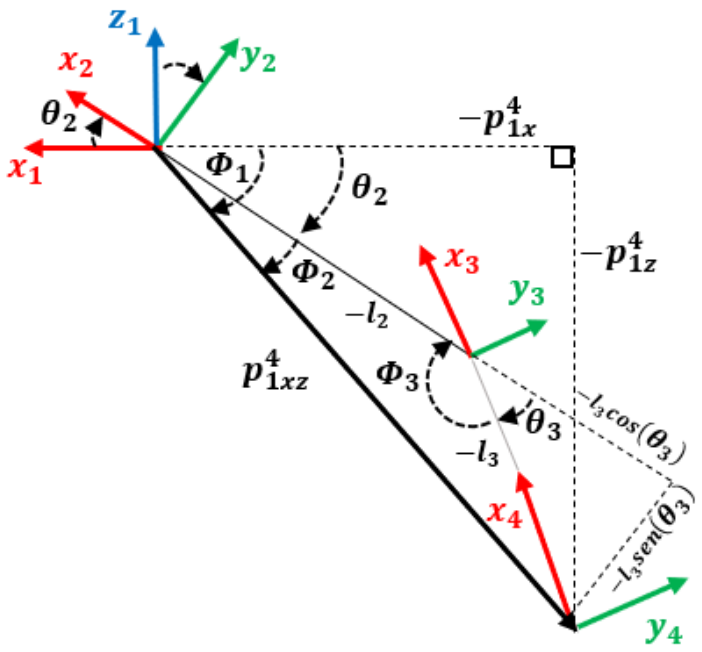


Figura 2.8: Análisis geométrico para θ_3.

En este punto, nuevamente es conveniente implementar el método algebraico debido a que se requiere conocer H_1^4 para obtener el vector que relaciona al sistema de referencia 1 con el 4 dado que la distancia entre 4 y 1 solamente depende de θ_3, por lo tanto, para encontrar la matriz correspondiente es necesario aplicar la siguiente relación de matrices:

$$H_4^6 = H_4^5 H_5^6 \tag{2.42}$$

$$H_6^4 = [H_4^6]^{-1} \tag{2.43}$$

$$H_1^6 = [H_0^1]^{-1} H_0^6 \tag{2.44}$$

$$H_1^4 = H_1^6 H_6^4 \tag{2.45}$$

$$H_1^4 = [H_0^1]^{-1} H_0^6 [H_4^5 H_5^6]^{-1} \tag{2.46}$$

De la relación de matrices anterior se obtienen las siguientes expresiones:

$$-p_{1x}^4 = l_3 c_{23} + l_2 c_2 \tag{2.47}$$

$$-p_{1z}^4 = l_3 s_{23} + l_2 s_2 \tag{2.48}$$

Por lo cual, es posible obtener la variable articular q_3 de la siguiente manera:

$$q_3 = \text{acos}\left(\frac{|p_{1xz}^4|^2 - (l_2^2) - (l_3^2)}{2 l_2 l_3}\right) \tag{2.49}$$

El valor del ángulo θ_2 puede ser determinado mediante ϕ_1-ϕ_2 donde cada ángulo se calcula de la siguiente manera con base a la figura 2.8:

$$\phi_1 = \text{atan2}\left(-p_{1z}^4, -p_{1x}^4\right) \tag{2.50}$$

$$\phi_2 = \text{asen}\left(\frac{-l_3 \text{sen}(\phi_3)}{|p_{1xz}^4|}\right) \tag{2.51}$$

En la figura se puede visualizar que ϕ_3 puede ser reemplazado por θ_3 debido a que $\text{sen}(\phi_3) = \text{sen}(180° - \theta_3) = \text{sen}(\theta_3)$. Por lo tanto, con la combinación de ambos ángulos encontrados se obtiene la variable articular q_2 expresada de la siguiente manera:

$$q_2 = \phi_1 - \phi_2 = \text{atan2}\left(-p_{1z}^4, -p_{1x}^4\right) - \text{asen}\left(\frac{-l_3 \text{sen}(\phi_3)}{|p_{1xz}^4|}\right) \tag{2.52}$$

Por último, el ángulo θ_4 puede ser obtenido mediante la matriz de transformación homogénea H_4^3 ya que se puede interpretar como el ángulo visto entre los ejes x_3 y x_4 por ser paralelos. Al igual que el caso anterior, esta matriz puede ser determinada mediante el método algebraico de la siguiente forma:

$$H_1^3 = H_1^2 H_2^3 \tag{2.53}$$

$$H_3^1 = [H_1^3]^{-1} \tag{2.54}$$

$$H_4^3 = [H_1^3]^{-1} H_1^4 \tag{2.55}$$

Para este resultado de la matriz de transición de 3 con respecto a 4, unicamente es necesario utilizar los componentes con respecto al eje x y con lo cual se obtiene la variable articular como:

$$q_4 = \text{atan2}\left(n_{4y}^3, n_{4x}^3\right) \tag{2.56}$$

2.4. Jacobiano del robot

En la sección anterior se obtuvo la cinemática directa e inversa las cuales relacionan las posiciones de las articulaciones del robot con las posiciones y orientaciones del efector final. Por otra parte, es claro ver que otro punto importante dentro del control de movimiento de los robots manipuladores es el análisis de la velocidad, ya que en la mayoría de tareas que realizan los robots no solo se requiere alcanzar una posición final sino que se requiere llegar a esa posición con cierta velocidad en el efector final, por lo cual los aspectos relacionados con las velocidades que se desarrollan en los manipuladores son de mucho interés. Es por ello que se cuenta con la cinemática diferencial la cual tiene como objetivo obtener la relación entre las velocidades lineales y angulares del efector final con las velocidades de las articulaciones [58], [59]. Esta relación está dada por la siguiente expresión [34]:

$$\begin{bmatrix} \boldsymbol{v} \\ \boldsymbol{w} \end{bmatrix} = J(\boldsymbol{q})\dot{\boldsymbol{q}} = \begin{bmatrix} J\boldsymbol{v}(\boldsymbol{q})\dot{\boldsymbol{q}} \\ J\boldsymbol{w}(\boldsymbol{q})\dot{\boldsymbol{q}} \end{bmatrix} \tag{2.57}$$

donde $J\boldsymbol{v}(\boldsymbol{q}) \in \mathbb{R}^{3xn}$ relaciona la velocidad articular $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ con la velocidad lineal $\boldsymbol{v} \in \mathbb{R}^3$, mientras que $J\boldsymbol{w}(\boldsymbol{q}) \in \mathbb{R}^{3xn}$ relaciona la velocidad angular $\boldsymbol{w} \in \mathbb{R}^3$ con la velocidad articular $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ [34]. La matriz $J(\boldsymbol{q}) \in \mathbb{R}^{6xn}$ formada por $J\boldsymbol{v}(\boldsymbol{q})$ y $J\boldsymbol{w}(\boldsymbol{q})$ es conocida como Jacobiano o Jacobiano del robot.

El Jacobiano es una de las funciones matriciales más importantes dentro del análisis y control de movimiento de un robot manipulador debido a que es utilizado en diferentes tareas y aspectos tales como planificación y ejecución de trayectorias, determinación de singularidades, obtención de las ecuaciones dinámicas del robot manipulador empleando Euler-Lagrange, describir la relación entre la fuerza aplicada y los pares o torques resultantes del extremo final del robot, entre muchas otras aplicaciones [34], [58], [59].

Existen diferentes maneras de elegir la orientación de la herramienta del robot manipulador: si dicha orientación se representa mediante los ángulos de Euler (θ, ϕ, ψ) donde se asocia un sistema de referencia al extremo final del robot o a la herramienta de trabajo, la velocidad angular estará dada por $\boldsymbol{w} = [\dot{\theta}, \dot{\phi}, \dot{\psi}] \in \mathbb{R}^3$ a lo cual se le conoce como Jacobiano analítico [34]. En cambio, si se modela la orientación de la herramienta del robot mediante un sistema de referencia específico como el que se encuentra localizado en la base del robot, entonces al Jacobiano se le conoce como Jacobiano geométrico y este puede ser calculado usando las matrices de transformación homogénea derivadas de la cinemática directa [34], [59].

La manera en que se calcula el Jacobiano para la presente investigación es mediante el Jacobiano geométrico debido a que se cuenta con las matrices de transformación homogénea que se obtuvieron en la sección anterior. A continuación se lleva a cabo el desarrollo de las ecuaciones correspondientes para la obtención del Jacobiano para el cobot UR10.

2.4.1. Jacobiano del cobot UR10

Recordando que la matriz de transformación homogénea que relaciona el sistema de referencia de la base con el sistema de referencia del efector final está dada por:

$$H_0^n = \begin{bmatrix} R_0^n & \mathbf{d}_0^n \\ \mathbf{0}^T & 1 \end{bmatrix}$$

La parte angular del Jacobiano $J_{g,\boldsymbol{w}}$ se puede obtener como [58], [59]:

$$J_{g,\boldsymbol{w}} = \begin{bmatrix} z_0^0, ..., z_0^{n-1} \end{bmatrix} \tag{2.58}$$

donde $z_0^{n-1} = R_0^{n-1}\boldsymbol{k}$ y $\boldsymbol{k} = [0,0,1]^T$ es el vector unitario del sistema de referencia $i-1$ expresado en la orientación del sistema de referencia inercial, es decir, el eje de acción del sistema de referencia S_0 en este caso el de la base. Por lo tanto, el elemento $z_0^0 = \boldsymbol{k} = [0,0,1]^T$ [58].

Considerando lo anterior, la parte angular del Jacobiano del cobot UR10 se obtiene mediante la siguiente expresión:

$$\begin{aligned} J_{g,\boldsymbol{w}} &= \begin{bmatrix} z_0^0 & z_0^1 & z_0^2 & z_0^3 & z_0^4 & z_0^5 \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{k} & R_0^1\boldsymbol{k} & R_0^2\boldsymbol{k} & R_0^3\boldsymbol{k} & R_0^4\boldsymbol{k} & R_0^5\boldsymbol{k} \end{bmatrix} \end{aligned} \tag{2.59}$$

A manera de demostración, se muestra solamente el desarrollo de la obtención del elemento z_0^1 de la matriz debido a que las expresiones que se obtienen son demasiado extensas.

$$z_0^1 = \begin{bmatrix} \cos(q_1) & 0 & \text{sen}(q_1) \\ \text{sen}(q_1) & 0 & -\cos(q_1) \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \text{sen}(q_1) \\ -\cos(q_1) \\ 0 \end{bmatrix} \tag{2.60}$$

La parte lineal $J_{g,\boldsymbol{v}}$ del Jacobiano está dada por [58], [59]:

$$J_{g,\boldsymbol{v_i}} = \frac{\partial \boldsymbol{d_0^n}}{\partial \boldsymbol{q_i}} \tag{2.61}$$

Por lo tanto, la parte lineal del Jacobiano del cobot UR10 se obtiene de la siguiente manera:

$$J_{g,\boldsymbol{v}} = \begin{bmatrix} \frac{\partial p_{0x}^6}{\partial q_1} & \frac{\partial p_{0x}^6}{\partial q_2} & \frac{\partial p_{0x}^6}{\partial q_3} & \frac{\partial p_{0x}^6}{\partial q_4} & \frac{\partial p_{0x}^6}{\partial q_5} & \frac{\partial p_{0x}^6}{\partial q_6} \\ \frac{\partial p_{0y}^6}{\partial q_1} & \frac{\partial p_{0y}^6}{\partial q_2} & \frac{\partial p_{0y}^6}{\partial q_3} & \frac{\partial p_{0y}^6}{\partial q_4} & \frac{\partial p_{0y}^6}{\partial q_5} & \frac{\partial p_{0y}^6}{\partial q_6} \\ \frac{\partial p_{0z}^6}{\partial q_1} & \frac{\partial p_{0z}^6}{\partial q_2} & \frac{\partial p_{0z}^6}{\partial q_3} & \frac{\partial p_{0z}^6}{\partial q_4} & \frac{\partial p_{0z}^6}{\partial q_5} & \frac{\partial p_{0z}^6}{\partial q_6} \end{bmatrix}$$

Para el cálculo de la parte lineal y angular del Jacobiano se realizó un código en MATLAB® que permite obtener los valores de los elementos de las matrices, los cuales no son expresados en su totalidad aquí debido a que son muy extensos.

2.4.2. Jacobiano para un punto arbitrario en el eslabón

Para la obtención del modelo dinámico del robot manipulador es necesario no solo obtener el Jacobiano en el efector final, sino que se requiere el Jacobiano para puntos arbitrarios en un eslabón, específicamente en los centros de masa de cada eslabón. Dado un vector $\boldsymbol{r}_0^{Pj}$ que apunta a un punto P en el eslabón l_j expresado en el sistema de referencia S_0, los elementos correspondientes a la parte lineal del Jacobiano $J_{Pj}(\boldsymbol{q})$ son [58], [59]:

$$J_{Pj,\boldsymbol{v}} = \frac{\partial \boldsymbol{r}_0^{Pj}}{\partial \boldsymbol{q_i}} = z_0^{i-1} \times (\boldsymbol{r}_0^{Pj} - \boldsymbol{d}_0^{i-1}), \qquad \forall \;\; i \leq j \tag{2.62}$$

$$J_{Pj,\boldsymbol{v}} = 0 \qquad \forall \;\; i > j \tag{2.63}$$

para $i \in 1, 2, ..., n$. Las columnas de $J_{Pj,\boldsymbol{v}}$ para $i > j$ son cero debido a que la velocidad del j-ésimo eslabón no se ve afectada por el movimiento de los eslabones que vienen después. Esta característica se da también para el Jacobiano angular [59]:

$$J_{Pj,\boldsymbol{w}} = z_o^{i-1}, \qquad \forall \;\; i \leq j \tag{2.64}$$

$$J_{Pj,\boldsymbol{w}} = 0 \qquad \forall \;\; i > j \tag{2.65}$$

Entonces, la expresión para el Jacobiano en un punto arbitrario está dada por:

$$J_{Pj}(\boldsymbol{q}) = \begin{bmatrix} z_0^0 \times (\boldsymbol{r}_0^{Pj} - \boldsymbol{d}_0^0) & \dots & z_0^{j-1} \times (\boldsymbol{r}_0^{Pj} - \boldsymbol{d}_0^{j-1}) & \mathbf{0} & \dots & \mathbf{0} \\ \\ z_0^0 & \dots & z_0^{j-1} & \mathbf{0} & \dots & \mathbf{0} \end{bmatrix} \tag{2.66}$$

Considerando que el fabricante [57] proporciona el vector de posición de los centros de masa de cada eslabón expresados en un sistema de referencia adjunto a su respectivo eslabón, es posible obtener el Jacobiano para un punto arbitrario en cada eslabón primeramente transformando los vectores de los centros de masa al sistema de referencia inercial del robot S_0. Esto puede hacerse de la siguiente manera [59]:

$$\begin{aligned} \boldsymbol{r}_0^{M1} &= \boldsymbol{d}_0^1 + R_0^1 \boldsymbol{r}_{c1}^1 \\ \boldsymbol{r}_0^{M2} &= \boldsymbol{d}_0^2 + R_0^2 \boldsymbol{r}_{c2}^2 \\ \boldsymbol{r}_0^{M3} &= \boldsymbol{d}_0^3 + R_0^3 \boldsymbol{r}_{c3}^3 \\ \boldsymbol{r}_0^{M4} &= \boldsymbol{d}_0^4 + R_0^4 \boldsymbol{r}_{c4}^4 \\ \boldsymbol{r}_0^{M5} &= \boldsymbol{d}_0^5 + R_0^5 \boldsymbol{r}_{c5}^5 \\ \boldsymbol{r}_0^{M6} &= \boldsymbol{d}_0^6 + R_0^6 \boldsymbol{r}_{c6}^6 \end{aligned} \tag{2.67}$$

donde $\boldsymbol{d}_0^i$ y R_0^i pueden ser obtenidos de las matrices de transformación homogénea H_0^i correspondiente y $\boldsymbol{r}_{ci}^i$ es el vector de posición del centro de masa de cada eslabón.

Finalmente, para obtener el Jacobiano de cada punto de los centros de masa, se sustituyen las ecuaciones (2.59) y (2.67) dentro de la ecuación (2.66), donde ahora $\boldsymbol{r}_0^{Mi} = \boldsymbol{r}_0^{Pj}$. Por lo tanto, la matriz del Jacobiano $J_{Mj}(\boldsymbol{q}) \in \mathbb{R}^{6x6}$ del cobot UR10 formada por su parte lineal $J_{Mj,\boldsymbol{v}} \in \mathbb{R}^{3x6}$ y su parte angular $J_{Mj,\boldsymbol{w}} \in \mathbb{R}^{3x6}$ está dada por:

$$J_{Mj}(\boldsymbol{q}) = \begin{bmatrix} J_{Mj,\boldsymbol{v}} \\ \hline J_{Mj,\boldsymbol{w}} \end{bmatrix} \tag{2.68}$$

$$= \begin{bmatrix} \frac{\partial p_{0x}^{M1}}{\partial q_1} & \frac{\partial p_{0x}^{M2}}{\partial q_2} & \frac{\partial p_{0x}^{M3}}{\partial q_3} & \frac{\partial p_{0x}^{M4}}{\partial q_4} & \frac{\partial p_{0x}^{M5}}{\partial q_5} & \frac{\partial p_{0x}^{M6}}{\partial q_6} \\ \frac{\partial p_{0y}^{M1}}{\partial q_1} & \frac{\partial p_{0y}^{M2}}{\partial q_2} & \frac{\partial p_{0y}^{M3}}{\partial q_3} & \frac{\partial p_{0y}^{M4}}{\partial q_4} & \frac{\partial p_{0y}^{M5}}{\partial q_5} & \frac{\partial p_{0y}^{M6}}{\partial q_6} \\ \frac{\partial p_{0z}^{M1}}{\partial q_1} & \frac{\partial p_{0z}^{M2}}{\partial q_2} & \frac{\partial p_{0z}^{M3}}{\partial q_3} & \frac{\partial p_{0z}^{M4}}{\partial q_4} & \frac{\partial p_{0z}^{M5}}{\partial q_5} & \frac{\partial p_{0z}^{M6}}{\partial q_6} \\ \hline \boldsymbol{k} & R_0^1\boldsymbol{k} & R_0^2\boldsymbol{k} & R_0^3\boldsymbol{k} & R_0^4\boldsymbol{k} & R_0^5\boldsymbol{k} \end{bmatrix}$$

Como se puede observar, solamente es necesario calcular la parte lineal del Jacobiano debido a que la parte angular resulta ser igual a la calculada en la sección anterior para el Jacobiano del efector final.

2.5. Tensores de inercia

Para obtener las ecuaciones dinámicas de un robot manipulador es necesario conocer el tensor de inercia de cada eslabón. A menudo no los proporciona el fabricante y deben calcularse. Sin embargo, una de las ventajas al realizar el prototipado virtual del cobot UR10 en ADAMS® mediante los parámetros dinámicos que brinda el fabricante en [57] es posible obtener los valores de los tensores de inercia de cada uno de los eslabones, los cuales se muestran a continuación y sus unidades están en kgm^2.

$$\mathcal{I}_1 = \begin{bmatrix} 0.03408 & 0.00002 & -0.00425 \\ 0.00002 & 0.03529 & 0.00008 \\ -0.00425 & 0.00008 & 0.02156 \end{bmatrix} \quad \mathcal{I}_2 = \begin{bmatrix} 0.02814 & 0.00005 & -0.01561 \\ 0.00005 & 0.77068 & 0.00002 \\ -0.01561 & 0.00002 & 0.76943 \end{bmatrix}$$

$$\mathcal{I}_3 = \begin{bmatrix} 0.01014 & 0.00008 & 0.00916 \\ 0.00008 & 0.30928 & 0.00000 \\ 0.00916 & 0.00000 & 0.30646 \end{bmatrix} \quad \mathcal{I}_4 = \begin{bmatrix} 0.00296 & -0.00001 & 0.00000 \\ -0.00001 & 0.00222 & -0.00024 \\ 0.00000 & -0.00024 & 0.00258 \end{bmatrix}$$

$$\mathcal{I}_5 = \begin{bmatrix} 0.00296 & -0.00001 & 0.00000 \\ -0.00001 & 0.00222 & -0.00024 \\ 0.00000 & -0.00024 & 0.00258 \end{bmatrix} \quad \mathcal{I}_6 = \begin{bmatrix} 0.0004 & 0.00000 & 0.00000 \\ 0.00000 & 0.00041 & 0.00000 \\ 0.00000 & 0.00000 & 0.00034 \end{bmatrix}$$

2.6. Dinámica

Un robot manipulador es un sistema dinámico muy complejo cuya descripción analítica requiere de ecuaciones diferenciales. La naturaleza no lineal, multivariable y acoplada de su comportamiento dinámico ofrece un amplio espectro en la formulación de problemas de control teóricos y prácticos [34]. El modelo dinámico de un robot manipulador permite explicar todos los fenómenos físicos que se encuentran en su estructura mecánica, tales como efectos inerciales, fuerzas centrípetas y de Coriolis, par gravitacional y fricción los cuales son fenómenos físicos intrínsecos o propios de la naturaleza dinámica del robot [34].

La mecánica analítica representa la herramienta sólida de las ciencias exactas para formular modelos matemáticos de sistemas mecánicos, en este contexto la dinámica es la parte de la física que estudia la relación que existe entre las fuerzas que actúan sobre un cuerpo y el movimiento que en él se origina. A diferencia de otros métodos de modelado de la física como el de Newton-Euler donde el desarrollo del modelo dinámico se obtiene mediante métodos recursivos aplicando métodos numéricos, las ecuaciones de movimiento de Euler-Lagrange representan la mejor alternativa de modelado para robots manipuladores debido a las propiedades matemáticas que se deducen de manera natural usando esa metodología [34].

2.6.1. Ecuaciones de Euler-Lagrange

Un método para obtener el modelo dinámico de un robot está basado en las ecuaciones de movimiento de Euler-Lagrange. El lagrangiano está representado como la diferencia de la energía cinética menos la potencial.

$$\partial\mathcal{L}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \mathcal{K}(\boldsymbol{q},\dot{\boldsymbol{q}}) - \mathcal{U}(\boldsymbol{q}) \tag{2.69}$$

La energía cinética del manipulador muestra una estructura matemática cuadrática bien definida que se encuentra en función de la velocidad articular [34]:

$$\mathcal{K}(\boldsymbol{q},\dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{2.70}$$

donde $M(\boldsymbol{q}) \in \mathbb{R}^{nxn}$ es la matriz de inercia del manipulador, la cual es simétrica y definida positiva. Por otro lado, la energía potencial $\mathcal{U}(\boldsymbol{q})$ no presenta una forma especifica, pero guarda una dependencia unicamente del vector de posición $\boldsymbol{q}$ debido a que este tipo de energía se da como consecuencia de la interacción de campos conservativos como la fuerza de gravedad [34]. Las ecuaciones de movimiento de Euler-Lagrange de un robot manipulador de n grados de libertad están dadas por [34]:

$$\frac{d}{dt}\left[\frac{\partial\mathcal{L}(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial\dot{\boldsymbol{q}}}\right] - \frac{\partial\mathcal{L}(\boldsymbol{q},\dot{\boldsymbol{q}})}{\partial\boldsymbol{q}} = \boldsymbol{\tau} - \boldsymbol{v}(\dot{\boldsymbol{q}},\boldsymbol{f}_e) \tag{2.71}$$

donde $\boldsymbol{q} = [q_1, q_2, \ldots, q_n]^T \in \mathbb{R}^n$ representa el vector de posiciones articulares o coordenadas generalizadas, $\dot{\boldsymbol{q}} = [\dot{q}_1, \dot{q}_2, \ldots, \dot{q}_n]^T \in \mathbb{R}^n$ es el vector de velocidades articulares, $\boldsymbol{\tau} = [\tau_1, \tau_2, \ldots, \tau_n]^T \in \mathbb{R}^n$ es el vector de pares aplicados, y $\boldsymbol{v}(\dot{\boldsymbol{q}},\boldsymbol{f}_e) \in \mathbb{R}^n$ es el vector de fuerzas o pares de fricción estática $\boldsymbol{f}_e$ que se encuentra en cada una de las articulaciones; $n \in \mathrm{N}$ es el número de grados de libertad.

2.6.2. Modelo dinámico de un robot manipulador

El modelo dinámico representa la base matemática para llevar a cabo el análisis y estudio de los fenómenos físicos presentes en la estructura mecánica de un robot manipulador de n grados de libertad en cadena cinemática abierta, con eslabones rígidos, en su rango de operación nominal o ancho de banda [34]. La ecuación de un robot manipulador de n grados de libertad en su forma compacta y con la notación más ampliamente utilizada se encuentra dada por [34]:

$$\boldsymbol{\tau} = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{f}_f(\dot{\boldsymbol{q}}, \boldsymbol{f}_e) \tag{2.72}$$

donde

- $\boldsymbol{q} \in \mathbb{R}^n$ es el vector de coordenadas generalizadas o posiciones articulares.
- $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ es el vector de velocidades articulares.
- $\ddot{\boldsymbol{q}} \in \mathbb{R}^n$ es el vector de aceleraciones articulares.
- $M(\boldsymbol{q}) \in \mathbb{R}^{nxn}$ es la matriz de inercia, la cual es simétrica y definida positiva.
- $C(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^{nxn}$ es la matriz de fuerzas centrípetas y de Coriolis.
- $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ es el vector de fuerzas o pares gravitacionales.
- $\boldsymbol{f}_f(\dot{\boldsymbol{q}}, \boldsymbol{f}_e) \in \mathbb{R}^n$ es el vector de pares de fricción que incluye la fricción viscosa, de Coulomb y estática ($\boldsymbol{f}_e$) de cada articulación del robot.

2.6.3. Modelo dinámico del UR10

Para el desarrollo del modelo dinámico del robot estudiado en la presente investigación se debe encontrar la energía cinética y potencial del mismo. Las ecuaciones que permiten su obtención se presentan a continuación y están basadas en [58], [59].

Energía cinética

Partiendo de la idea general de que la energía cinética total de un cuerpo rígido que rota es la suma de la energía cinética de rotación y la energía cinética de traslación del centro de masa [59]. La energía cinética total se puede obtener mediante la siguiente expresión:

$$\mathcal{K} = \frac{1}{2} m \boldsymbol{v}^T \boldsymbol{v} + \frac{1}{2} \boldsymbol{w}^T \mathcal{I} \boldsymbol{w} \tag{2.73}$$

donde $\boldsymbol{v} \in \mathbb{R}^3$, $\boldsymbol{w} \in \mathbb{R}^3$ son las velocidades lineales y angulares del cuerpo rígido, $m \in \mathbb{R}$ es la masa total, $\mathcal{I} \in \mathbb{R}^{3x3}$ es el tensor de inercia. Basado en lo anterior y suponiendo que cada eslabón del robot es un cuerpo rígido y usando la ecuación (2.68), la energía cinética total de un robot manipulador puede ser obtenida de la siguiente manera:

$$\mathcal{K}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2} \dot{\boldsymbol{q}}^T \left[\sum_{i=1}^{n} m_i J_{\boldsymbol{v}_{mi}}(\boldsymbol{q})^T J_{\boldsymbol{v}_{mi}}(\boldsymbol{q}) + J_{\boldsymbol{w}_{mi}}(\boldsymbol{q})^T \mathcal{I}_i J_{\boldsymbol{w}_{mi}}(\boldsymbol{q}) \right] \dot{\boldsymbol{q}} \tag{2.74}$$

donde $m_i \in \mathbb{R}$ es la masa de cada eslabón, $J_{\boldsymbol{v}_{mi}} \in \mathbb{R}^{3xn}$ y $J_{\boldsymbol{w}_{mi}} \in \mathbb{R}^{3xn}$ son la parte lineal y angular de la matriz Jacobiano $J_{Mi}(\boldsymbol{q})$ del centro de masa de cada eslabón, $\mathcal{I}_i$ es el tensor de inercia correspondiente a cada eslabón.

Energía potencial

La energía potencial de cada eslabón se puede calcular suponiendo que la masa de todo el objeto está concentrada en su centro de masa y viene dada por [58], [59]:

$$\mathcal{U}(\boldsymbol{q}) = \sum_{i=1}^{n} m_i \boldsymbol{g}_0^T \boldsymbol{r}_{ci}^i \tag{2.75}$$

donde $m_i \in \mathbb{R}$ es la masa de cada eslabón, $\boldsymbol{g}_0$ es el vector de gravedad y $\boldsymbol{r}_{ci}^i$ es el vector de posición del centro de masa de cada eslabón.

Matriz de inercia

Con base a la ecuación (2.70), es claro observar que mediante la expresión (2.74) de la energía cinética se puede obtener la matriz de inercia $M(\boldsymbol{q}) \in \mathbb{R}^{6x6}$ de la siguiente forma:

$$M(\boldsymbol{q}) = \left[\sum_{i=1}^{n} m_i J_{\boldsymbol{v}_{mi}}(\boldsymbol{q})^T J_{\boldsymbol{v}_{mi}}(\boldsymbol{q}) + J_{\boldsymbol{w}_{mi}}(\boldsymbol{q})^T \mathcal{I}_i J_{\boldsymbol{w}_{mi}}(\boldsymbol{q}) \right] \tag{2.76}$$

Matriz de Coriolis

El cálculo de los elementos de la matriz de Coriolis $C(\dot{\boldsymbol{q}}, \boldsymbol{q})$ se realiza mediante los símbolos de Christoffel que se encuentran definidos de la siguiente manera [58]:

$$c_{ij} = \sum_{k=1}^{n} \frac{1}{2} \left(\frac{\partial m_{ij}}{\partial q_k} + \frac{\partial m_{ik}}{\partial q_j} - \frac{\partial m_{kj}}{\partial q_i} \right) \dot{q}_i \tag{2.77}$$

donde m_{ij} representa el ij-ésimo elemento de la matriz de inercia.

Par gravitacional

Finalmente, para obtener el par gravitacional se utiliza la siguiente expresión:

$$\boldsymbol{g}(\boldsymbol{q}) = \frac{\partial \mathcal{U}(\boldsymbol{q})}{\partial \boldsymbol{q}} \tag{2.78}$$

Debido a la complejidad que conlleva obtener de manera analítica los cálculos del par gravitacional y las matrices de inercia y de Coriolis, se realizó un programa en MATLAB® basado en las ecuaciones anteriores que permite obtener mediante variables simbólicas los resultados de cada uno de estos elementos, los cuales no pueden ser expresados en su totalidad aquí puesto que son muy extensos. La simulación correspondiente a la implementación del modelo dinámico se aborda en los capítulos siguientes.

2.7. Estrategia de control

Dentro de las diversas clasificaciones comerciales de los algoritmos de control de robots manipuladores existe la de ***arquitectura cerrada*** y la ***arquitectura abierta***. La primera, hace referencia a todo el conjunto de elementos que integran un sistema, en este caso el robot manipulador junto con su unidad de control de origen sin posibilidad de modificación alguna, mientras que la arquitectura abierta se refiere a la posibilidad de modificación del sistema para lograr operarlo de diferente manera integrando nuevos elementos que permitan que el funcionamiento del robot tenga un desempeño específico [60].

Una de las ventajas más importantes que presenta la arquitectura abierta frente a la arquitectura cerrada es que permite el desarrollo y evaluación de mejoras al sistema mediante la implementación de nuevas estrategias de control que logran aumentar el desempeño del robot; en cambio, un robot que mantiene su arquitectura cerrada tiene desempeño limitado por el algoritmo de control programado y solo le permite realizar tareas específicas [61]. Para poder evaluar diversas estrategias de control en un robot manipulador de arquitectura abierta diseñado para realizar diversas tareas, es necesario contar con una plataforma experimental que emule la dinámica del robot y su interacción con el entorno, y que permita la programación de diversos algoritmos de control [60].

De acuerdo con [62] una estrategia de control es un conjunto de técnicas, herramientas matemáticas y tecnologías implementadas para resolver el problema de control de sistemas robóticos y que se logren alcanzar los objetivos deseados. Tal es el caso de [63] que implementa una nueva estrategia de control *master-helper* en conjunto con un algoritmo de autoorganización y la técnica de distribución de carga para lograr que un sistema multi-robot se emplee solo cuando dos robots originalmente dispuestos no pueden realizar la tarea encomendada y es necesario adicionar un nuevo robot. Por su parte, dentro de [62] se abordan también diversas estrategias de control para controlar el movimiento y las fuerzas de contacto de manera simultánea entre un robot manipulador y su entorno.

Con base a la premisa anterior, dentro de la presente investigación se propone como estrategia de control la implementación de sensores de proximidad ultrasónicos, variables de estado $(\boldsymbol{q}, \dot{\boldsymbol{q}})$, tarjetas únicas de procesamiento (Raspberry Pi), planificador inteligente de trayectorias y algoritmos de control bien conocidos en la industria como lo es la familia de los controladores PID, para que el robot logre los objetivos deseados.

El algoritmo de control PID está dado por la siguiente estructura matemática [64]:

$$\boldsymbol{u}(t) = K_p \boldsymbol{e}(t) + K_i \int_0^t \boldsymbol{e}(\tau) d\tau + K_d \frac{d\boldsymbol{e}(t)}{dt} \tag{2.79}$$

donde $K_p, K_i, K_d \in \mathbb{R}^{nxn}$ son las ganancias proporcional, integral y derivativa respectivamente y, $\boldsymbol{e}(t) = \boldsymbol{q}_d - \boldsymbol{q}(t)$ es el error, que se define como la diferencia entre el valor deseado y el valor actual.

2.8. Planificación de trayectorias

El control de movimiento se encarga de mover al robot libremente en su espacio de trabajo siguiendo una trayectoria deseada en posición y velocidad sin interactuar con su medio ambiente [34]. La planificación de trayectorias consiste en la generación de un conjunto de parámetros de control para que el robot realice una determinada tarea. El problema de control de movimiento se puede dividir en dos subproblemas: la planificación de la trayectoria y el control y seguimiento de esta trayectoria [65].

- **Planificación de la trayectoria.** Consiste en describir el movimiento deseado del manipulador como una secuencia de puntos en el espacio articular a través de los cuales debe pasar. La curva espacial que el efector final del robot manipulador recorre desde la posición inicial hasta la posición final con restricción de tiempo se le llama trayectoria. La planificación de trayectoria interpola y/o aproxima la trayectoria deseada por una clase de funciones polinomiales y genera una secuencia de puntos en función del tiempo para el control del manipulador desde la posición inicial hasta el destino [65].

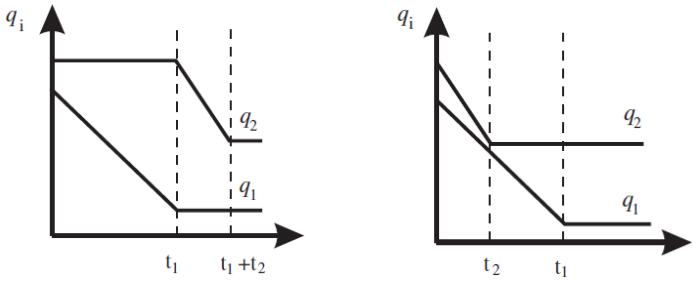
- **Control de movimiento.** Dada la trayectoria que debe seguir el robot manipulador, el control de movimiento puede realizarse mediante dos métodos como se menciona en [1].

  - **Control cinemático.** Establece cuáles son las trayectorias que debe seguir cada articulación del robot a lo largo del tiempo para lograr los objetivos fijados por el usuario (punto de partida, la trayectoria cartesiana del efector final, velocidad, etc.). Dentro de la práctica, el movimiento especificado al robot para que siga la trayectoria definida no es del todo posible, debido a que mediante la implementación de este tipo de control no se consideran las características dinámicas del robot (inercias, fricción, etc.) generando que la trayectoria deseada y la real no coincidan completamente.

  - **Control dinámico.** Tiene por objetivo que las trayectorias reales coincidan lo más cercano posible a las trayectorias deseadas ya que en este caso si se considera el modelo dinámico del robot y distintas herramientas de análisis y diseño aportadas por la teoría de control.

2.8.1. Tipos de trayectorias

Trayectorias punto a punto. En este tipo de trayectorias cada articulación evoluciona desde su posición inicial a la final sin realizar consideración alguna sobre el estado o evolución de las demás articulaciones. Normalmente, cada actuador trata de llevar a su articulación al punto de destino en el menor tiempo posible, pudiéndose distinguir dos casos: movimiento eje a eje (figura 2.9a) y movimiento simultáneo de ejes (figura 2.9b), siendo el segundo tipo de especial interés para la presente investigación debido a que a diferencia del primer caso, en donde el movimiento de cada articulación se inicia una vez que la articulación anterior haya alcanzado su punto final, en este caso se pretende que las articulaciones puedan moverse de manera independiente o coordinada [1].



(a) Movimiento eje a eje. (b) Movimiento simultáneo de ejes.

Figura 2.9: Trayectorias punto a punto [1].

Trayectorias continuas. Cuando se pretende que la trayectoria que sigue el extremo del robot sea conocida por el usuario (trayectoria en el espacio cartesiano o de la tarea), es preciso calcular y controlar de manera continua las trayectorias articulares, que, por lo general, presentarán una compleja evolución temporal tal y como se muestra en la figura 2.10 [1].

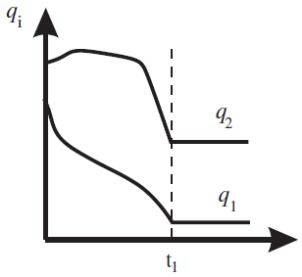


Figura 2.10: Trayectoria continua [1].

2.8.2. Generación de trayectorias cartesianas

Normalmente, el usuario del robot indica el movimiento que éste debe realizar especificando las localizaciones espaciales por las que debe pasar el efector final, junto con otros datos, como instantes de paso, velocidades o tipos de trayectorias. Así, por ejemplo, es frecuente especificar que el robot debe ir de un punto inicial hasta otro final, siguiendo en el espacio de la tarea una línea recta a velocidad constante. Por lo anterior, es preciso entonces, establecer un interpolador entre las localizaciones expresadas en el espacio de la tarea que dará como resultado una expresión analítica de la evolución de cada coordenada [1]. Una vez que se dispone de la secuencia de configuraciones articulares por las que debe pasar el robot, la siguiente función del control cinemático es la de unir está sucesión de puntos articulares garantizando, junto con las condiciones de configuración-tiempo de paso, que se cumplan las restricciones de velocidad y aceleración máximas asociadas a los actuadores del robot, de manera que se consiga una trayectoria realizable y suficientemente suave [1].

Existen diversas funciones interpoladoras que son utilizadas frecuentemente, tal como los interpoladores lineales en los cuales se mantiene la velocidad constante para la trayectoria pla-

nificada. Sin embargo, este tipo de interpoladores origina saltos bruscos en la velocidad $\dot{\boldsymbol{q}}$ de la articulación, y como consecuencia de ello se generan aceleraciones $\ddot{\boldsymbol{q}}$ infinitas, lo cual en la práctica no es viable [1]. También, se precisa de otro tipo de interpoladores utilizando funciones polinómicas de grado n, lo cual significa que entre mayor grado posea el polinomio mayor es el número de parámetros a utilizar (posiciones, velocidades, aceleraciones). Esto por un lado genera una mayor precisión y suavidad en el seguimiento de la trayectoria y, por otro lado presenta inconvenientes tales como mayor tiempo de cálculo, lo que puede generar oscilación entre los puntos de paso y menor precisión numérica para el cálculo [1], [66]. Considerando lo anterior, para la presente investigación, se hará uso de un interpolador cúbico, el cual se describe a continuación.

Interpolador spline cúbico.

Para asegurar que la trayectoria que une los puntos por los que tiene que pasar la articulación considerada presente continuidad en velocidad, puede recurrirse a utilizar un polinomio de grado 3 que una cada pareja de puntos adyacentes [1]. De este modo en cada intervalo, al tener cuatro parámetros disponibles se podrán imponer cuatro condiciones de contorno, dos de posición y dos de velocidad. Con esto se consigue una trayectoria compuesta por una serie de polinomios cúbicos, cada uno válido entre dos puntos consecutivos. Este conjunto de polinomios concatenados, escogidos de modo que exista continuidad en la posición y velocidad se denominan *splines* [1].

La expresión de la trayectoria que une dos puntos adyacentes $(\boldsymbol{q}_i, \boldsymbol{q}_{i+1})$ será [1]:

$$
\begin{aligned}
&\boldsymbol{q}(t) = a + b(t - t_i) + c(t - t_i)^2 + d(t - t_i)^3 \\
&a = \boldsymbol{q}_i \\
&b = \dot{\boldsymbol{q}}_i \\
&c = \frac{3}{(t_{i+1} - t_i)^2}(\dot{\boldsymbol{q}}_{i+1} - \dot{\boldsymbol{q}}_i) - \frac{1}{(t_{i+1} - t_i)}(\dot{\boldsymbol{q}}_{i+1} + 2\dot{\boldsymbol{q}}_i) \\
&d = -\frac{2}{(t_{i+1} - t_i)^3}(\dot{\boldsymbol{q}}_{i+1} - \dot{\boldsymbol{q}}_i) + \frac{1}{(t_{i+1} - t_i)^2}(\dot{\boldsymbol{q}}_{i+1} + \dot{\boldsymbol{q}}_i)
\end{aligned}
\tag{2.80}
$$

En los capítulos anteriores, se presentaron las generalidades y fundamentos tanto teóricos como matemáticos en los cuales está sustentada la presente investigación y que permiten poder llevar a cabo el desarrollo de la misma. A continuación, dentro del siguiente capítulo se presenta la descripción de la metodología implementada para el desarrollo de la plataforma tecnológica en donde se abordan los temas de diseño en CAD del cobot, diseño del entorno virtual en ADAMS®, co-simulación ADAMS®-MATLAB® y validación de la implementación.

Capítulo 3

Metodología de diseño de la plataforma tecnológica

En el presente capítulo se aborda la propuesta de una metodología de diseño para una plataforma tecnológica que consta de 5 pasos mediante los cuales se llega a una validación de resultados, en este caso el control de un cobot por medio de la implementación de una estrategia de control. A continuación, se describe detalladamente el procedimiento de cada parte del proceso.

3.1. Proceso de diseño de la plataforma tecnológica

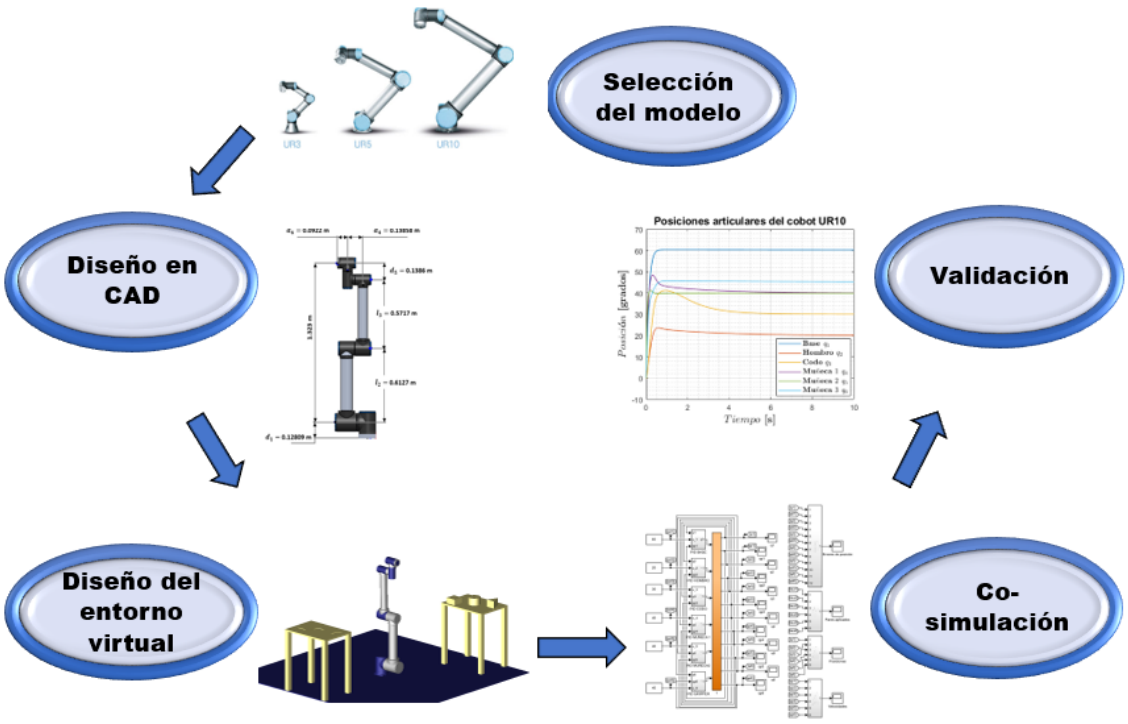


Figura 3.1: Proceso de diseño de la plataforma tecnológica.

3.1.1. Selección del modelo de cobot

El modelo diseñado está basado en el cobot UR10 de la empresa Universal Robots. Se seleccionó este modelo en particular por dos razones principales:

- Es una empresa líder y pionera en la robótica colaborativa [57] ya que según un reporte [67] de la empresa de inteligencia, investigación y asesoramiento del mercado *BIS Research*, Universal Robots ha mantenido su posición de líder del mercado con una cuota global del 60 % del mercado de cobots, alcanzando un volumen de ventas de cobots mayor que todos los competidores juntos.

- El cobot UR10 es el más vendido de todos sus modelos de acuerdo con [7].

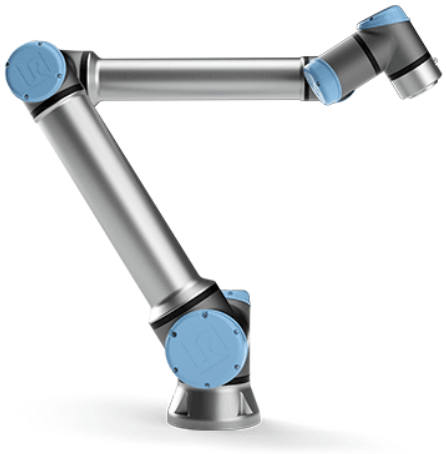


Figura 3.2: Cobot UR10 de Universal Robots [7].

3.1.2. Diseño en SolidWorks®

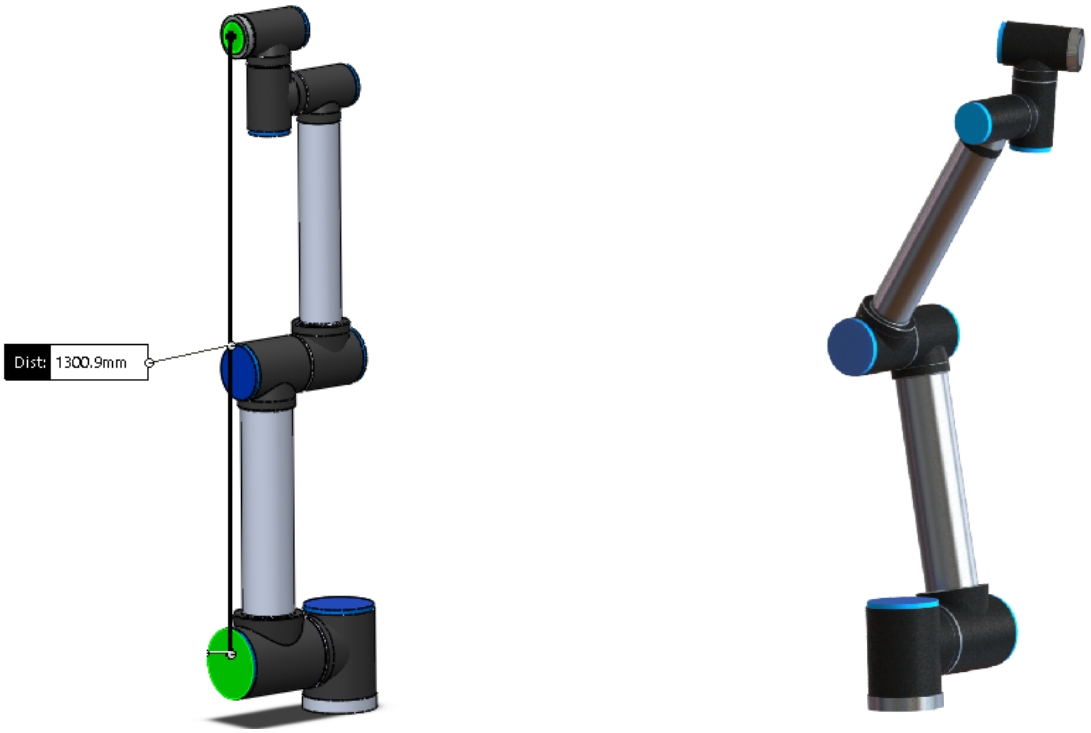
Para la elaboración del cobot se utilizó el software de diseño CAD SolidWorks®, para lo cual se tomaron las especificaciones que brinda el fabricante en [7] donde se mencionan las características del cobot UR10 de las cuales se destacan:

Eslabón	**Masa [kg]**	**Centro de masa [m]**	**Par máximo [Nm]**
Base	7.1	[0.021, 0.0, 0.027]	330
Hombro	12.7	[0.38, 0.0, 0.158]	330
Codo	4.27	[0.24, 0.0, 0.068]	150
Muñeca 1	2.0	[0.0, 0.007, 0.018]	56
Muñeca 2	2.0	[0.0, 0.007, 0.018]	56
Muñeca 3	0.365	[0.0, 0.0, -0.026]	56

Tabla 3.1: Especificaciones del cobot UR10.

- Alcance: 1300 mm (51.2 in).
- Materiales de los eslabones:
  - Aluminio.
  - Plástico PP.
  - Acero.
- Carga útil: 10 kg (22 lb).
- Rango de las articulaciones: +/- 360°.

Con base a las especificaciones anteriores se realizó el diseño del cobot UR10 que se muestra en la figura 3.3b ya renderizado y en la figura 3.3a se muestra que se diseñó con base a las medidas que brinda el fabricante en relación a su alcance.



(a) Alcance del cobot diseñado en SolidWorks®. (b) Cobot diseñado (renderizado).

Figura 3.3: Diseño de cobot UR10 realizado en SolidWorks®.

3.1.3. Diseño del entorno virtual en ADAMS®

Para realizar la exportación del diseño del cobot UR10 realizado en SolidWorks® hacia el software ADAMS®, se siguió el procedimiento como se muestra en la figura 3.4 el cual se describe a continuación:

1. Seleccionar la pestaña de estudio de movimiento dentro de la misma interfaz de SolidWorks® la cual se encuentra indicada en la parte inferior izquierda de la ventana.

2. Activar la opción SolidWorks Motion que se incluye en la barra de herramientas dentro de la pestaña SolidWorks Add-Ins, esto permite habilitar todas las funciones del estudio de movimiento.

3. Dentro del menú desplegable que se señala, aparecerán 3 opciones: Animation, Basic Motion y Motion Analysis. Seleccionar esta última opción.

4. Una vez que se tienen las opciones de estudio de movimiento correspondientes, se añade un actuador a cualquier articulación del diseño, en este caso, se añadió un motor en la articulación del hombro del cobot.

5. Realizar la simulación de movimiento de la articulación seleccionando el botón que se indica.

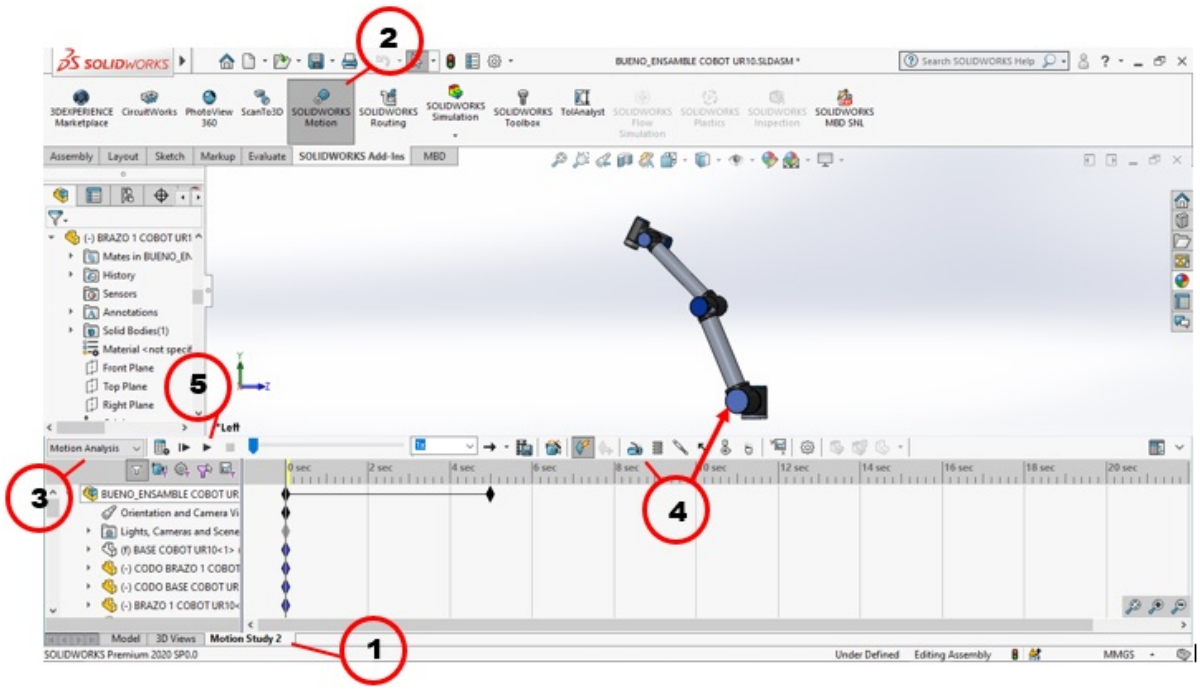


Figura 3.4: Procedimiento para exportar el diseño de SolidWorks® a ADAMS®.

El procedimiento anterior habilita la opción que permite exportar el diseño con todas las relaciones de posición configuradas en el modelo. Una vez que se completó el proceso de simulación de movimiento, se realiza el último paso como se muestra en la figura 3.5, en donde se observa que aparece la opción *export to ADAMS* ya habilitada, la cual, al seleccionarla permite guardar el modelo diseñado dentro de un archivo con una extensión *.adm* que puede ser leída y ejecutada dentro del software ADAMS®.

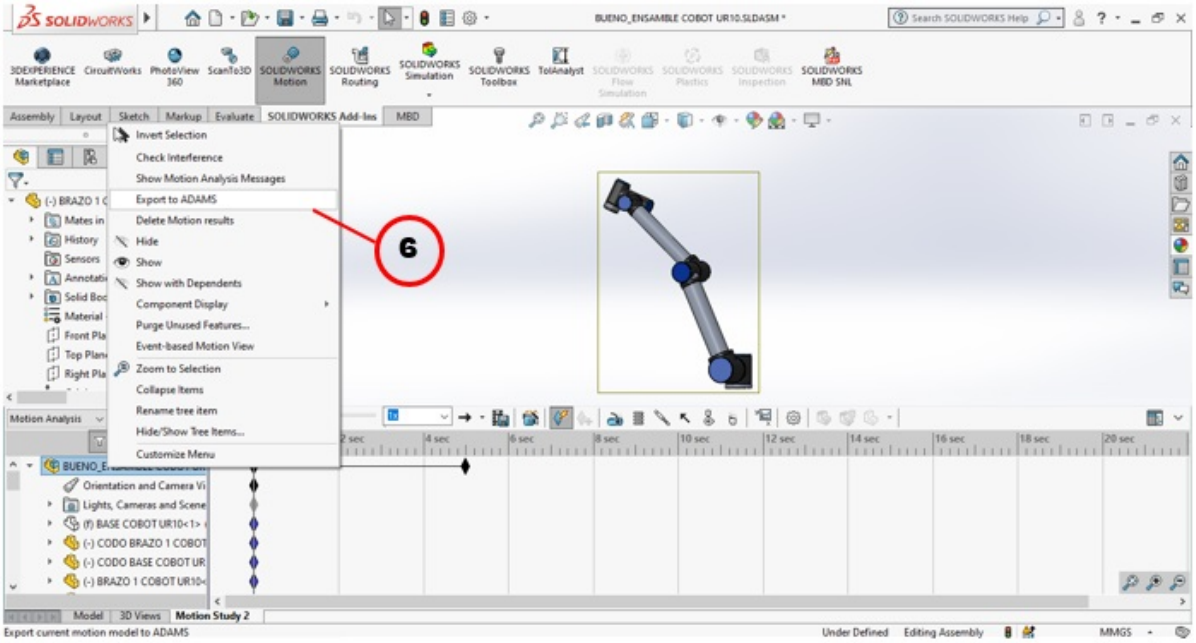


Figura 3.5: Opción habilitada para exportar el diseño a ADAMS®.

Al ejecutar el software ADAMS®, dentro de la interfaz se abrirá automáticamente la ventana que se muestra en la figura 3.6, dentro de las opciones que se presentan se selecciona *Existing Model*, la cual, permitirá realizar la búsqueda del archivo con extensión *.adm* que se exportó desde SolidWorks® con el procedimiento anteriormente descrito.

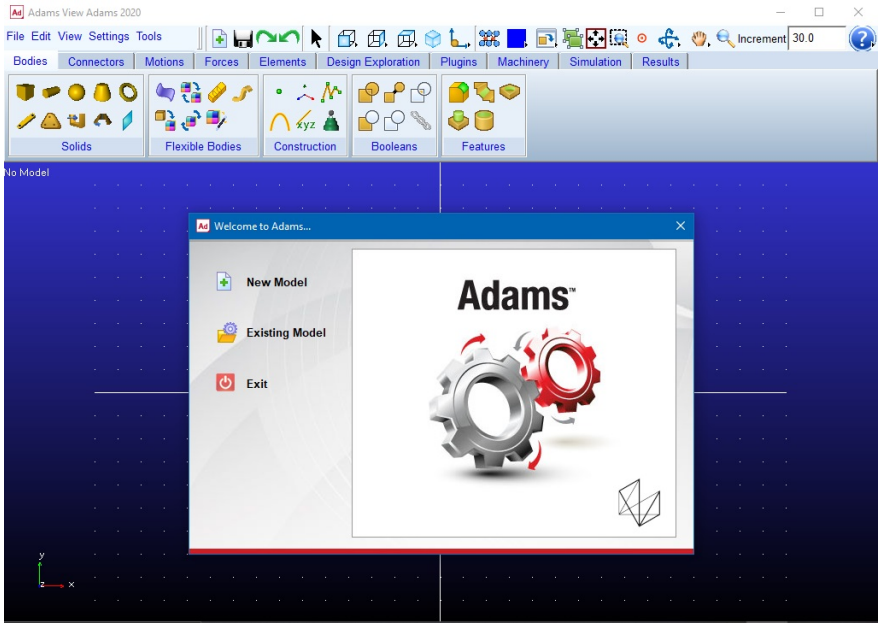


Figura 3.6: Software ADAMS®.

.

La figura 3.7 muestra el diseño del cobot UR10 dentro de la interfaz de ADAMS® al cual, se le añadieron dos mesas con el fin de simular la estación de ensamble y poder darle movimiento al robot desde una hacia otra simulando la trayectoria que seguiría al realizar una tarea dentro del espacio de trabajo. Además, se colocaron restricciones de movimiento a las mesas de ensamble y a la base del robot con la finalidad de que al momento de realizar la simulación estos se mantuvieran en su lugar.

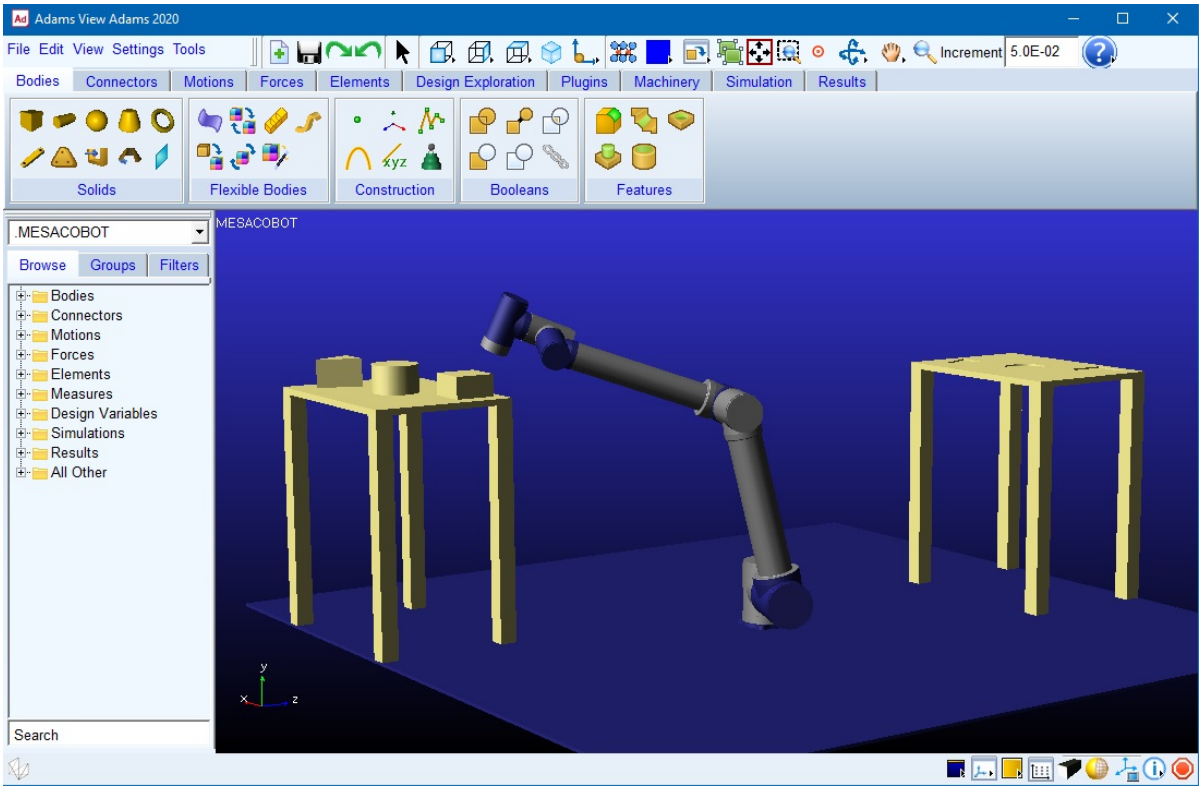


Figura 3.7: Diseño del cobot UR10 en software ADAMS®.

3.1.4. Co-simulación ADAMS®-MATLAB®

Para poder llevar a cabo la co-simulación entre los dos software ADAMS® y MATLAB®, es necesario realizar una serie de pasos que por cuestiones de simplicidad y para un mejor entendimiento de ellos se dividen en 3 secciones:

- Preparación del entorno de ADAMS®.
- Configuración de las entradas y salidas del robot.
- Preparación del entorno en MATLAB®.

A continuación se realiza una descripción detallada de cada una de las secciones anteriores.

Preparación del entorno de ADAMS®

1. Primeramente se requiere habilitar el complemento *ADAMS/Controls*, para ello, dentro de la opción *Tools*, que se encuentra en la barra de herramientas, se selecciona *Plugins Manager* de la lista de opciones que se despliega y, posteriormente, en la ventana que se muestra se selecciona la opción *ADAMS/Controls*. Una vez realizado lo anterior, dentro de la pestaña *Plugins* se puede visualizar que el complemento se activó correctamente. Este proceso se muestra en la figura 3.8 donde se señala cada paso mencionado anteriormente.

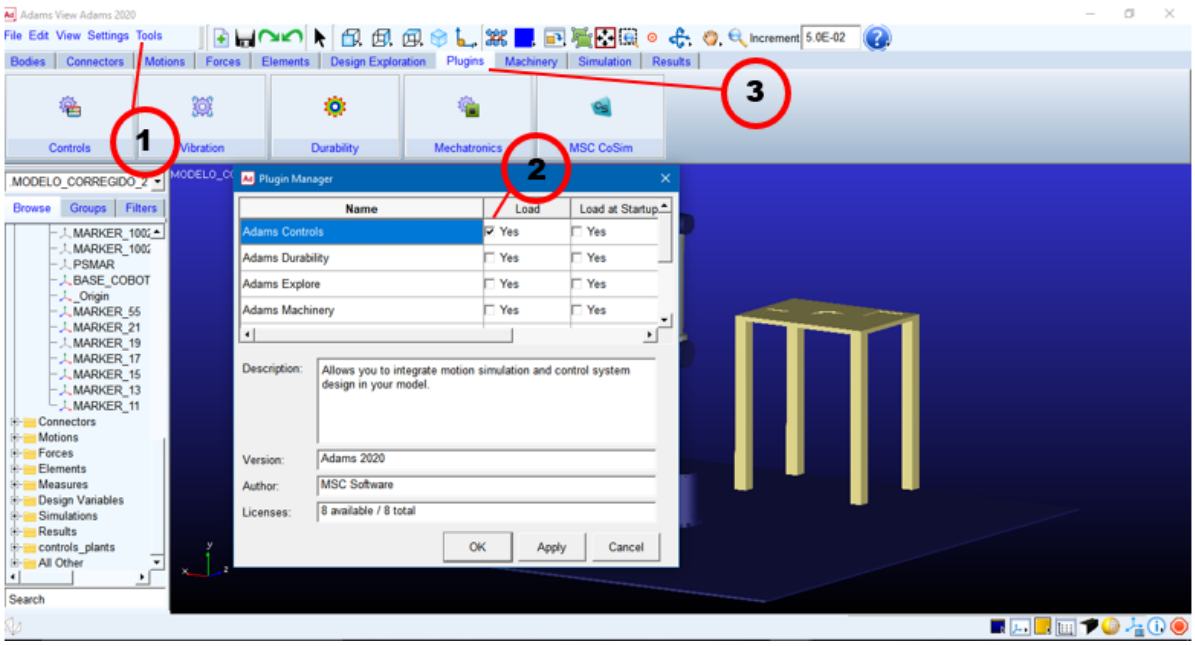


Figura 3.8: Activación del complemento ADAMS/Controls.

2. Para poder controlar al prototipo virtual del cobot es necesario que, como parte del diseño del entorno virtual, se añada a cada una de las articulaciones los distintos elementos que brinda ADAMS® para lograr este objetivo de mover al robot. Uno de los elementos más importantes son las *joints* debido a que permiten el movimiento en las articulaciones del robot y, además, mediante estas mismas se obtiene la posición y velocidad para cada articulación. Otro de los elementos que se requiere insertar en cada articulación son las *Sforce*, básicamente estos elementos representan el par aplicado en las articulaciones, es decir, las entradas del robot.

3. Una vez que se tienen añadidos los elementos anteriores, lo siguiente es crear variables de estado $\boldsymbol{q}$ y $\dot{\boldsymbol{q}}$ para cada articulación, las cuales representan las salidas del robot. En este punto, debe realizarse la asignación de cada estado del sistema a cada una de las variables creadas; es decir, para las variables de estado q_1 y $\dot{q}_1$ se asignan la posición y velocidad del joint 1, etc. Este procedimiento de asignación se realiza con cada una de las 6 articulaciones.

Para el caso de los pares aplicados que fueron añadidos de igual manera en las articulaciones del robot (*Sforce*), es necesario crear variables de entrada $\boldsymbol{\tau}$ para realizar la asignación correspondiente de cada uno de los pares. La figura 3.9 muestra los elementos anteriormente descritos ya insertados en una articulación del robot dentro del espacio de trabajo de ADAMS®.

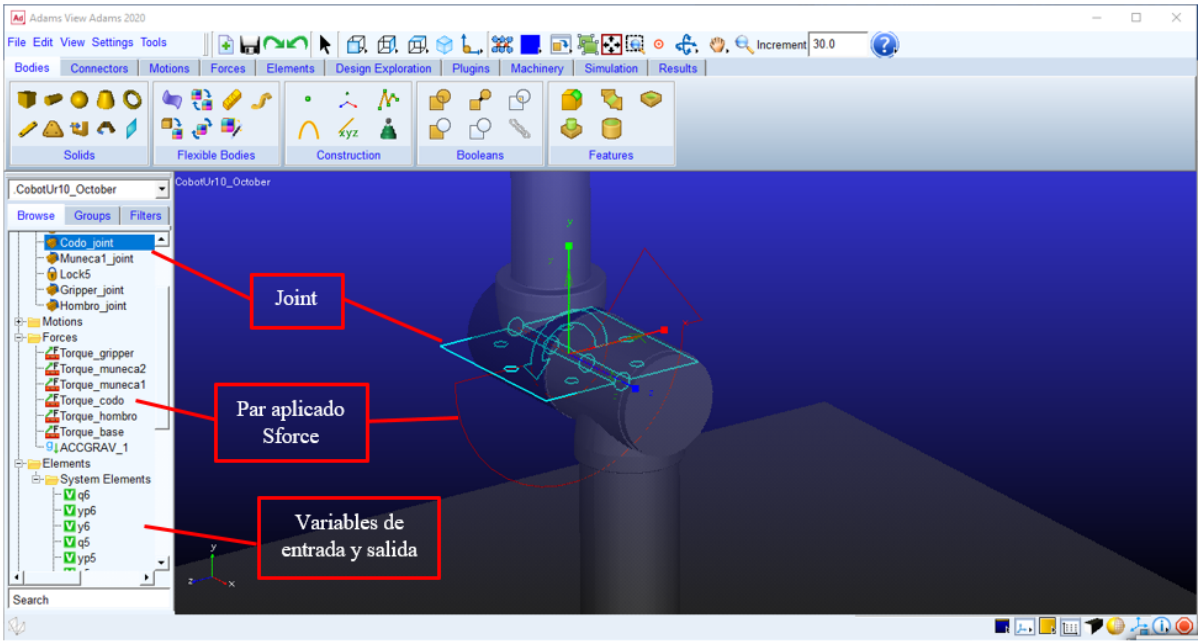


Figura 3.9: Asignación de elementos para las articulaciones del robot.

Configuración de las entradas y salidas del robot

Dentro de esta etapa de diseño se hace uso del complemento *ADAMS/Controls* que fue habilitado en un principio. Para acceder a él se muestran los pasos en la figura 3.10. Dentro de la ventana que se despliega en el paso 3 se pueden observar 3 tipos de datos señalados los cuales corresponden a la configuración que el usuario debe introducir, la primera columna corresponde a las variables de entrada del sistema que son específicamente los pares aplicados que fueron asignados en la etapa anterior, la segunda columna corresponde a las variables de salida que de igual manera se asignaron dentro del proceso anterior y son los estados de cada una de las articulaciones, posición y velocidad. Por último, dentro del recuadro *Target software* se realiza la selección del software con el cual se hará la co-simulación, en este caso para la presente investigación se selecciona MATLAB®.

La finalidad de realizar este proceso es para generar un archivo con extensión *.m* que contiene las configuraciones de entradas y salidas realizadas en el robot para que pueda ser leído y ejecutado por MATLAB® y, posteriormente, obtener el modelo del robot para poder trabajar dentro de MATLAB/Simulink con él en forma de bloque con sus correspondientes entradas y salidas. Este proceso corresponde a la última sección que se explica a continuación.

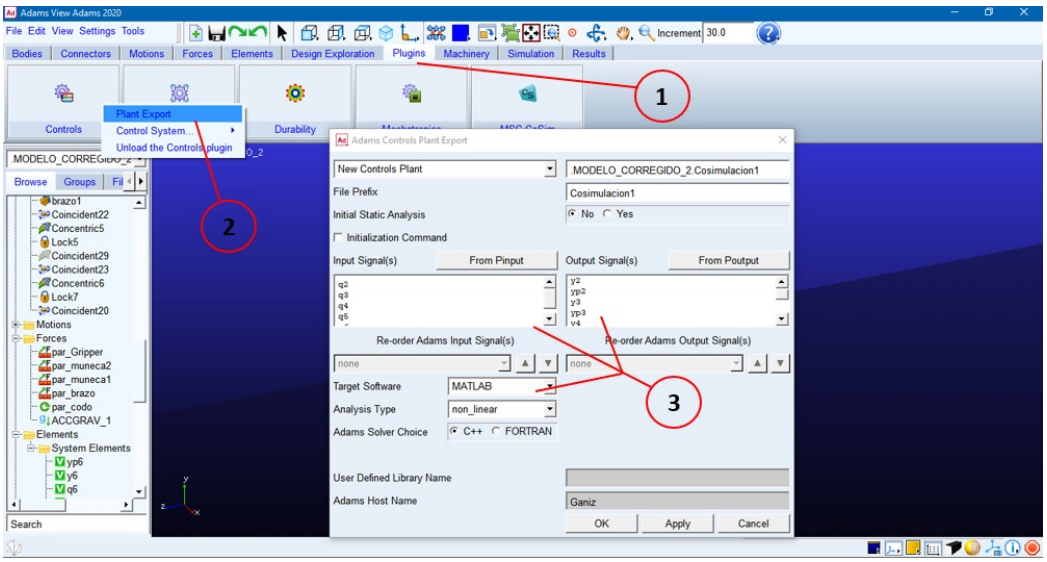


Figura 3.10: Configuración de entradas y salidas en ADAMS/Controls.

Preparación del entorno en MATLAB®

Para esta última etapa del diseño se requiere el archivo con extensión *.m* generado en el proceso anterior, ya que como se mencionó, en él se encuentran todos los datos del prototipo virtual del cobot y son necesarios para llevar a cabo la co-simulación. Una vez localizado el archivo, es necesario que se compile para poder habilitar la comunicación con ADAMS® y cargar todos los datos. Posteriormente, dentro de la ventana de comandos de MATLAB® se requiere ejecutar el comando *adams_sys* el cual genera el bloque que contiene dichos datos del prototipo virtual del cobot con el que se trabajará dentro de Simulink. Lo descrito anteriormente se puede visualizar en la figura 3.11.

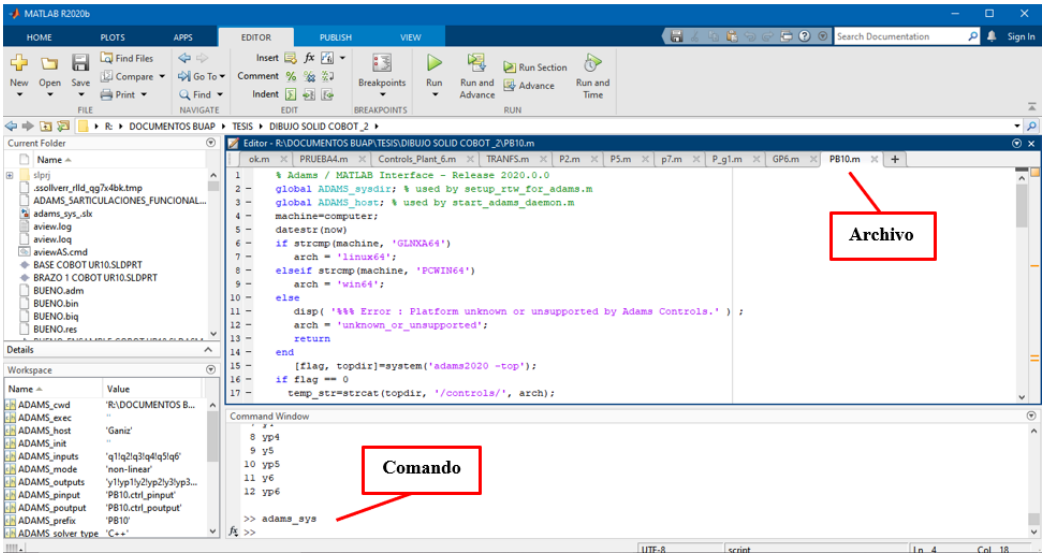
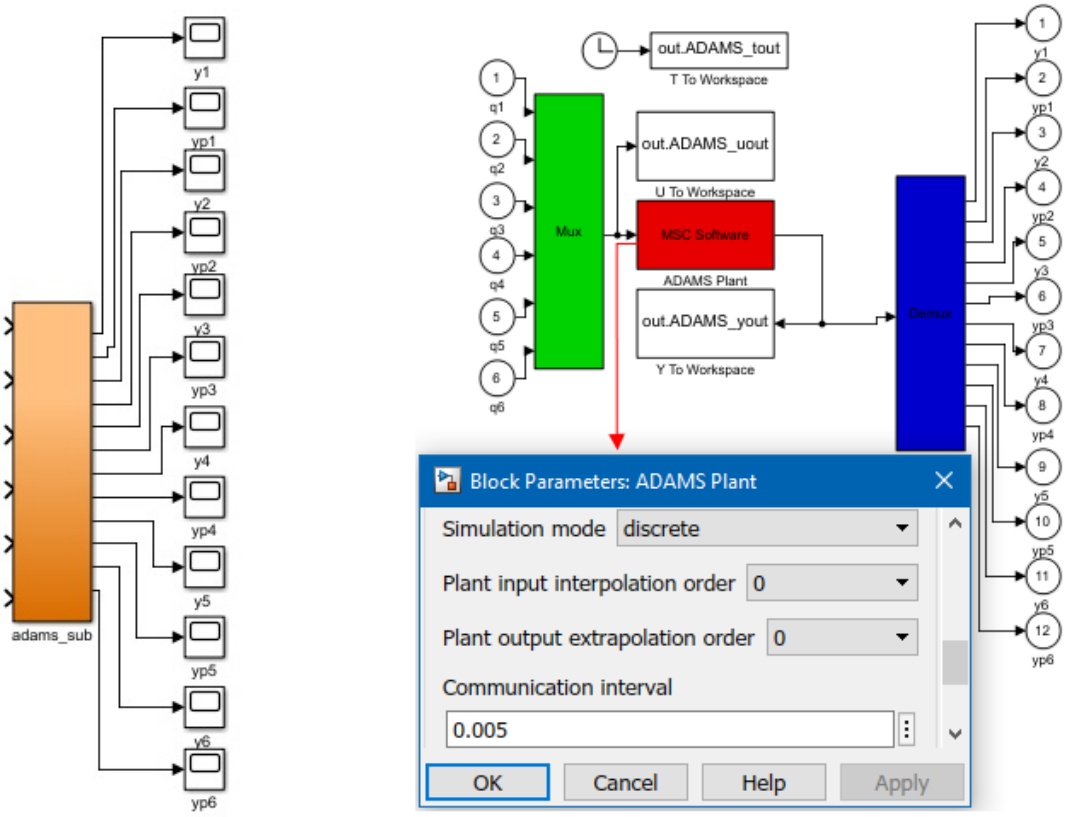


Figura 3.11: Archivo del robot en MATLAB®.

En la figura 3.12a se puede visualizar el bloque de color anaranjado que es generado por MATLAB® mediante el comando *adams_sys*, el cual, contiene las entradas y salidas correspondientes del cobot que fueron insertadas desde el entorno de ADAMS®. Por otra parte, en la figura 3.12b se muestra la estructura interna del bloque, donde se logra visualizar los diferentes parámetros y bloques del cobot que integra ADAMS® dentro del mismo tales cómo modo de simulación, intervalo de comunicación, salidas al espacio de trabajo de MATLAB®, multiplexor para entradas y demultiplexor para salidas.



(a) Bloque que contiene el modelo del cobot.

(b) Estructura interna del bloque.

Figura 3.12: Bloque que contiene el modelo del cobot en Simulink.

Finalmente, teniendo el bloque anterior, dentro de Simulink se realiza el diseño del diagrama de bloques correspondiente al sistema en lazo cerrado, insertando en la entrada de cada articulación un control PID que estará siendo realimentado por la posición y velocidad de las mismas tal y como se observa en la figura 3.13. La estructura del control PID implementada en cada articulación se muestra en la figura 3.14. Las ganancias para cada controlador fueron sintonizadas de manera manual [68] y se muestran en la tabla 3.2.

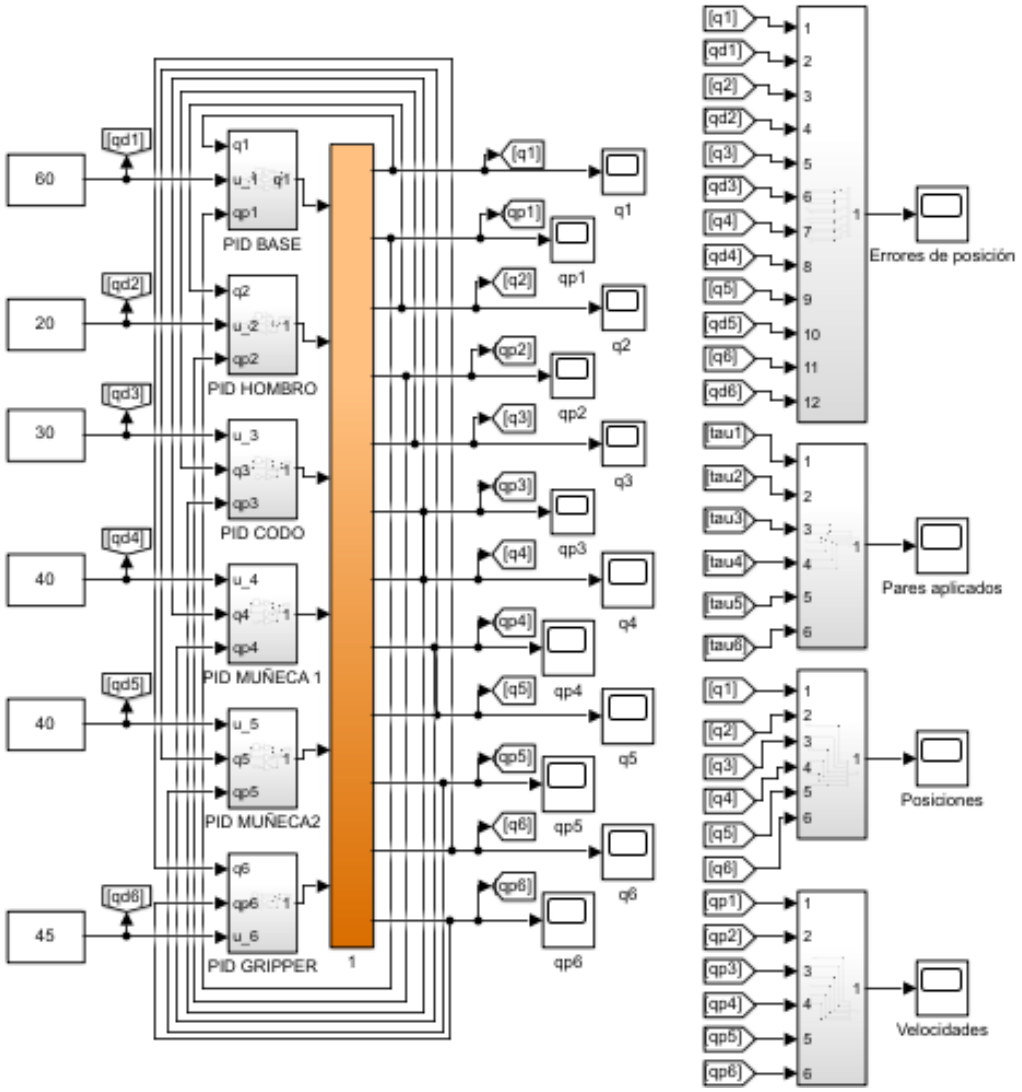


Figura 3.13: Diseño del sistema en lazo cerrado en Simulink.

Articulación	***P***	***I***	***D***
Base	3	0.1	0.7
Hombro	15	4	5
Codo	9	1	2
Muñeca 1	2	0.1	0.08
Muñeca 2	0.8	0.01	0.02
Muñeca 3	0.0025	0.003	0.0035

Tabla 3.2: Ganancias del control PID.

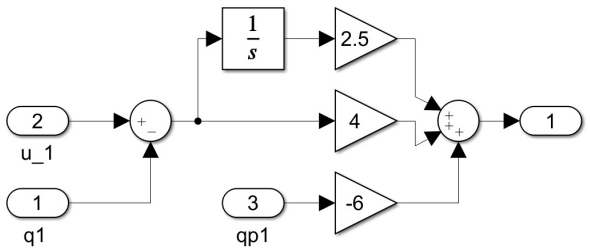


Figura 3.14: Estructura del control PID implementada en cada articulación.

3.1.5. Validación de resultados

A continuación, se realiza el análisis y validación de los resultados obtenidos mediante la implementación de una estrategia de control basada en PID para el control de posición del cobot modelado.

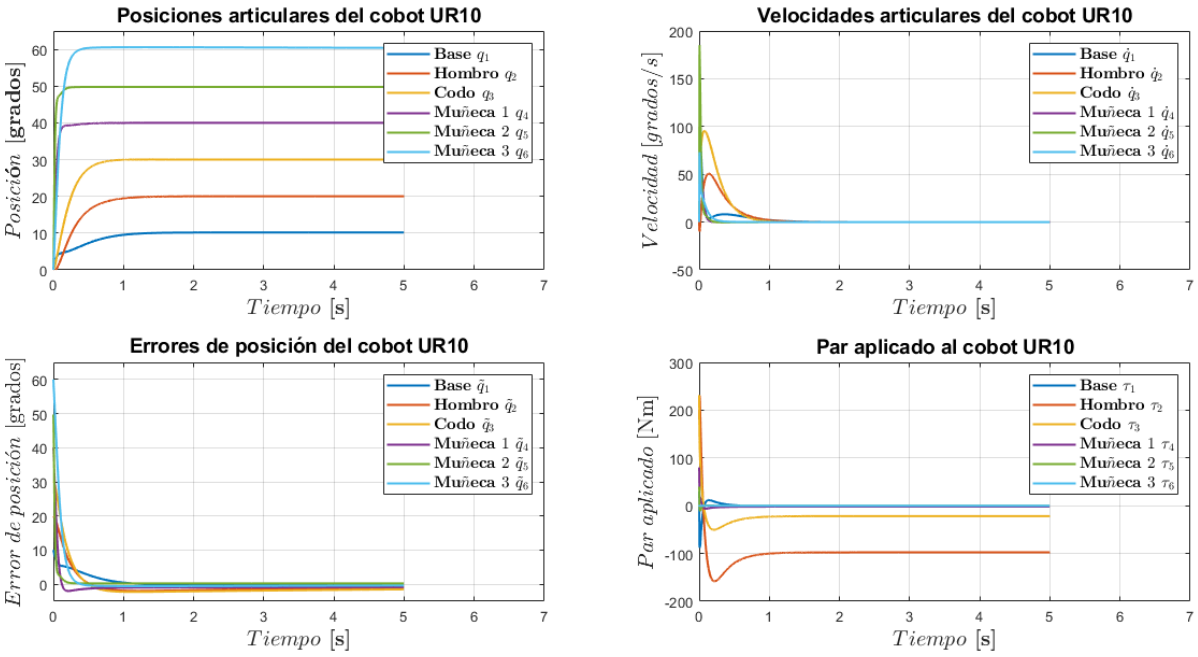


Figura 3.15: Resultados obtenidos de la co-simulación ADAMS® - MATLAB® mediante la implementación de estrategia de control basada en PID para control de posición.

En la figura 3.15 se muestran los resultados obtenidos de la co-simulación correspondiente a las 6 articulaciones del cobot. Partiendo del análisis de las gráficas de posición, se pueden observar parámetros de desempeño en el dominio del tiempo tales cómo sobreimpulsos, convergencia desde la posición inicial $\boldsymbol{q_0} = [0\ 0\ 0\ 0\ 0\ 0]^T$ a la posición deseada $\boldsymbol{q_d} = [10\ 20\ 30\ 40\ 50\ 60]^T$ la cual fue elegida de manera arbitraria. Por otro lado, en la gráfica de velocidad se puede ver como todos los valores de $\dot{\boldsymbol{q}}$ tienden asintóticamente a cero y de igual manera se observan los parámetros de desempeño en tiempo continuo anteriormente mencionados.

De igual manera, se tienen las gráficas correspondientes a los errores de posiciones y pares aplicados que de una manera muy general, en el caso del error de posición se puede apreciar que en la mayoría de ellas, converge a cero en periodos de tiempo de aproximadamente 1 segundo; y de igual manera, en las gráficas de par aplicado se ve cómo solamente 2 articulaciones toman un valor constante para poder mantener la posición deseada, mientras que el valor de par aplicado para las otras 4 converge asintóticamente a cero, destacando también que en ninguno de los casos se supera el par máximo que se muestra en la tabla 3.1. Por último, en la figura 3.16 se muestra el cambio en la configuración de las articulaciones que toma el cobot al llegar a la posición deseada $\boldsymbol{q_d}$ mediante la implementación de la estrategia de control basada en PID.

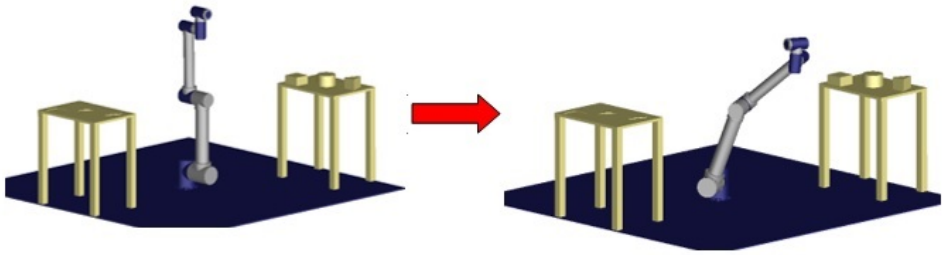


Figura 3.16: Cambio de pose con el control de posición del cobot.

3.2. Algoritmo inteligente generador de trayectorias

Parte importante de las funciones de un robot colaborativo es poder realizar una tarea de manera secuencial dentro de su espacio de trabajo, donde en ocasiones tiene que evitar obstáculos los cuales son detectados por medio de una red de sensores que dotan al cobot con la capacidad de detectar si algún objeto o humano se encuentra dentro de su área de trabajo. Por lo cual, el siguiente paso después de validar la estrategia de control de posición basada en PID del cobot modelado, es el desarrollo de un algoritmo inteligente generador de trayectorias dentro de MATLAB/Simulink, el cual, consiste en un algoritmo que permite identificar una señal de entrada proveniente de un sensor ultrasónico HC-SR04 que emula la presencia de un operador humano dentro del espacio de trabajo del cobot. El principal objetivo de este algoritmo es dotar de inteligencia al cobot para brindarle la capacidad de decidir cuando esté realizando una tarea de *pick-and-place* y se detecte la señal del sensor, éste pueda detenerse dependiendo la distancia a la cual se encuentre el operador humano del cobot dentro del área de trabajo. Si la señal de presencia del operador humano en el área de trabajo es nula, el algoritmo recalcula cuál fue la última posición en la trayectoria y la establece como la nueva posición inicial para continuar con la tarea. Las secciones que se tratan a continuación, describen el proceso que se siguió para el desarrollo del HIL del sensor y la validación del algoritmo inteligente generador de trayectorias mediante la co-simulación ADAMS®-MATLAB®.

3.2.1. HIL del sensor

El proceso para embeber el HIL del sensor requiere una serie de pasos para los cuales se contempló la implementación de una tarjeta Raspberry Pi 4 modelo B de 8 Gb de RAM como la que se muestra en la figura 3.17. Esta serie de pasos se detallan a continuación.

Figura 3.17: Raspberry Pi 4 utilizada para HIL del sensor.

Configuración de la tarjeta.

Para hacer uso de la tarjeta se requiere la instalación del sistema operativo Raspberry Pi OS dentro de una tarjeta micro SD, en este caso se instaló la versión Debian Buster por medio de la herramienta Raspberry Pi Imager (figura 3.18) que brinda el fabricante de manera gratuita en su página oficial. Dentro de las ventajas que brinda esta herramienta se destaca que es posible seleccionar la versión del sistema operativo a instalar, asignación de la IP de la tarjeta, establecer un usuario y contraseña para acceso al sistema, habilitar el Secure Shell (SSH, por sus siglas en inglés), entre otros parámetros que puede definir el usuario.

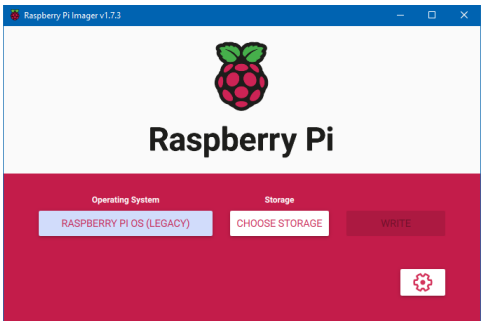


Figura 3.18: Raspberry Pi Imager.

Existen diversas maneras de poder acceder a Raspberry Pi OS ya sea de la manera más tradicional con teclado, monitor y mouse conectados directamente en la tarjeta Raspberry Pi, o bien, haciendo uso de la conexión remota mediante el software Virtual Network Computing (VNC, por sus siglas en inglés). En esta investigación, se utilizó el segundo método por lo que es necesario habilitar la opción de VNC de la Raspberry Pi. Esto se logra a través de la terminal PuTTY que se muestra en la figura 3.19. Esta terminal permite establecer la comunicación con la Raspberry Pi mediante el protocolo de comunicación SSH solamente ingresando la IP establecida en la tarjeta. Dentro de la terminal, se ingresa el usuario y contraseña que se asigno dentro de la configuración inicial para acceder al sistema operativo. Una vez que se ingresa, se hace uso del comando *sudo raspi-config* para poder accesar a la configuración de la Raspberry Pi dentro de la cual es posible habilitar la opción de VNC como se observa en la figura 3.20.

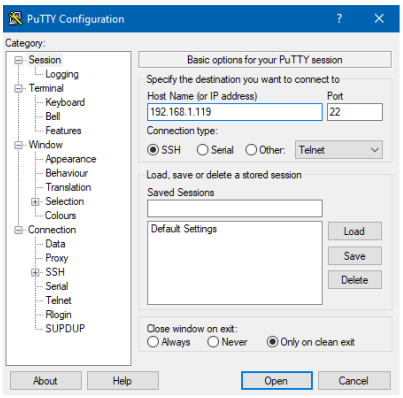


Figura 3.19: Terminal de PuTTY.

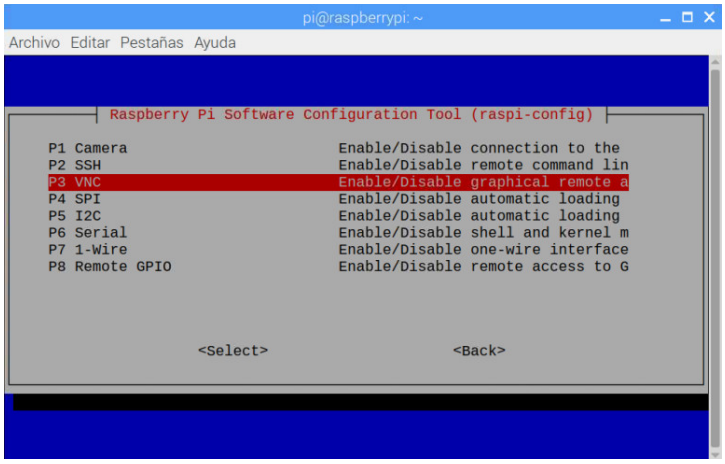


Figura 3.20: Opción VNC dentro de la configuración de Raspberry Pi OS.

Teniendo el VNC habilitado dentro de la configuración de la Raspberry Pi, es posible acceder a su sistema operativo de manera remota a través del software VNC ingresando la IP asignada seguidamente del usuario y contraseña tal y como se muestra en la figura 3.21. El motivo principal por el cual se busca acceder al sistema operativo de la Raspberry Pi es para hacer uso del IDE[1] de Python (figura 3.22), ya que a través de la programación en él es posible realizar la lectura de las señales del sensor ultrasónico y la activación de los pines de la Raspberry Pi.

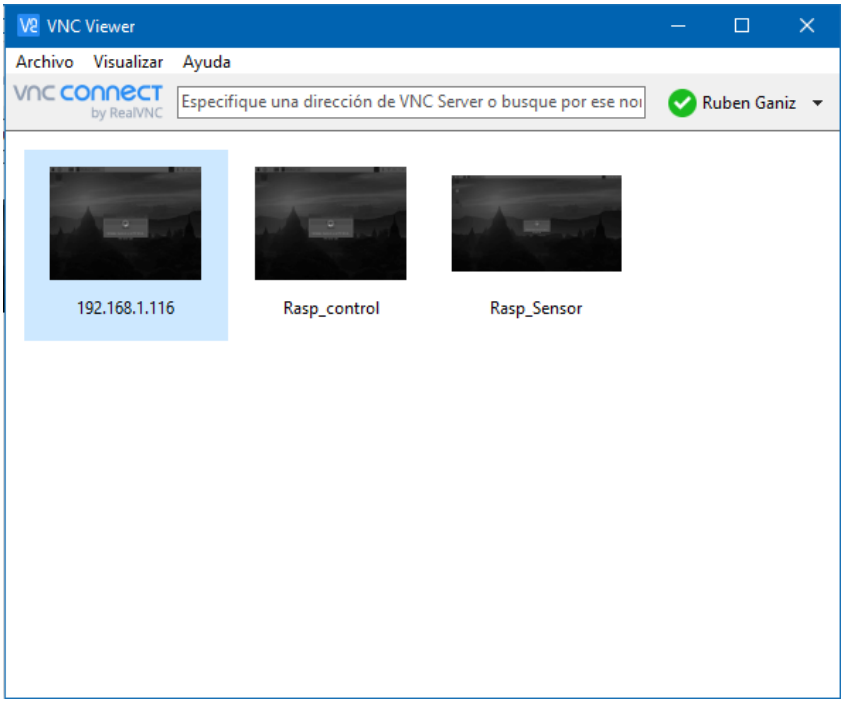


Figura 3.21: Acceso remoto a la Raspberry Pi a través de VNC.

[1]IDE; acrónimo de la lengua inglesa *Integrated Development Environment*.

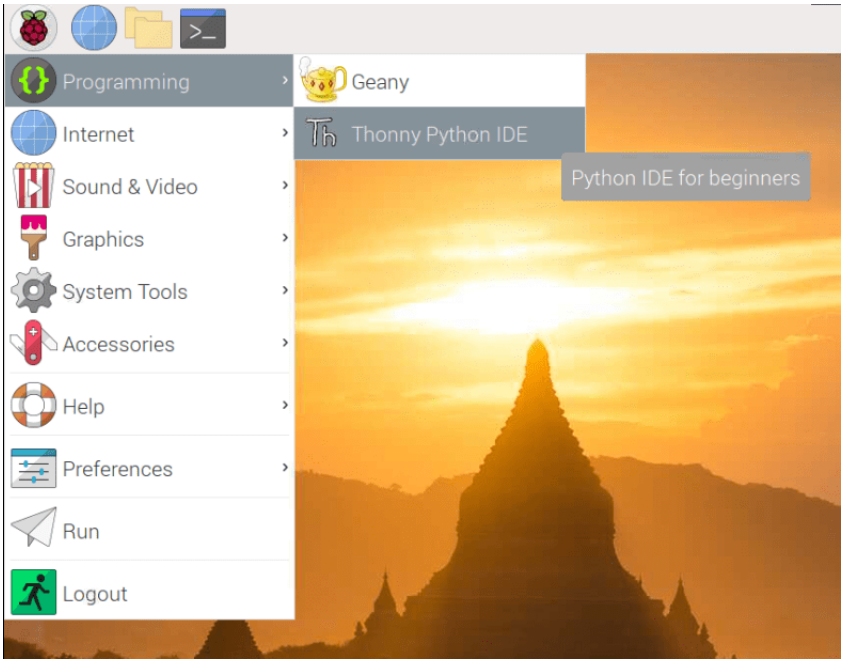


Figura 3.22: IDE de Python en Raspberry Pi OS.

Instalación de paqueterías de Simulink.

Para poder llevar a cabo la comunicación entre la Raspberry Pi y MATLAB/Simulink es necesaria la instalación de dos paqueterías que hacen esto posible:

- **Paquete de soporte de MATLAB® para hardware Raspberry Pi**.
  Esta paquetería permite comunicarse con una Raspberry Pi de forma remota desde una computadora que ejecuta MATLAB® o a través de un navegador web con MATLAB® Online. También, puede adquirir datos de sensores y dispositivos de imágenes conectados a Raspberry Pi y procesarlos en MATLAB®. Asimismo, permite la comunicación con otro hardware a través de los pines GPIO, serie, I2C y SPI [8].

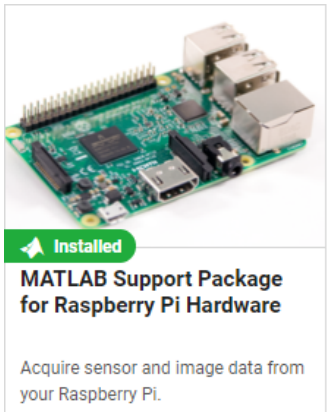


Figura 3.23: Paquetería de MATLAB® para Raspberry Pi [8].

- **Paquete de soporte de Simulink para hardware Raspberry Pi.**
  La paquetería permite crear y ejecutar modelos de Simulink en hardware Raspberry Pi. El paquete de soporte incluye una biblioteca de bloques Simulink para configurar y acceder a periféricos de entrada/salida e interfaces de comunicación. También, permite monitorear y ajustar de forma interactiva los algoritmos desarrollados en Simulink mientras se ejecutan en Raspberry Pi [8].

Figura 3.24: Paquetería de Simulink para Raspberry Pi [8].

Al instalar las paqueterías, dentro de la interfaz de Simulink se habilitan los complementos que permiten seleccionar la tarjeta Raspberry Pi para realizar la comunicación tal y como se muestra en la figura 3.25 en donde además, se requiere nuevamente ingresar la IP de la tarjeta, usuario y contraseña para poder establecer dicha comunicación entre la tarjeta y Simulink. Este último paso permite hacer uso de la biblioteca de bloques para configurar y acceder a periféricos de entrada/salida e interfaces de comunicación de la tarjeta que se presentan señalados en la figura 3.26 dentro el recuadro rojo.

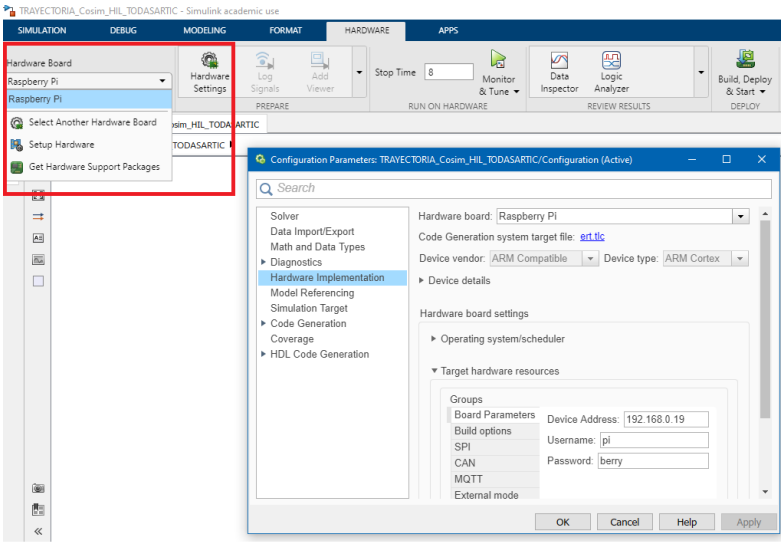


Figura 3.25: Complementos habilitados en Simulink.

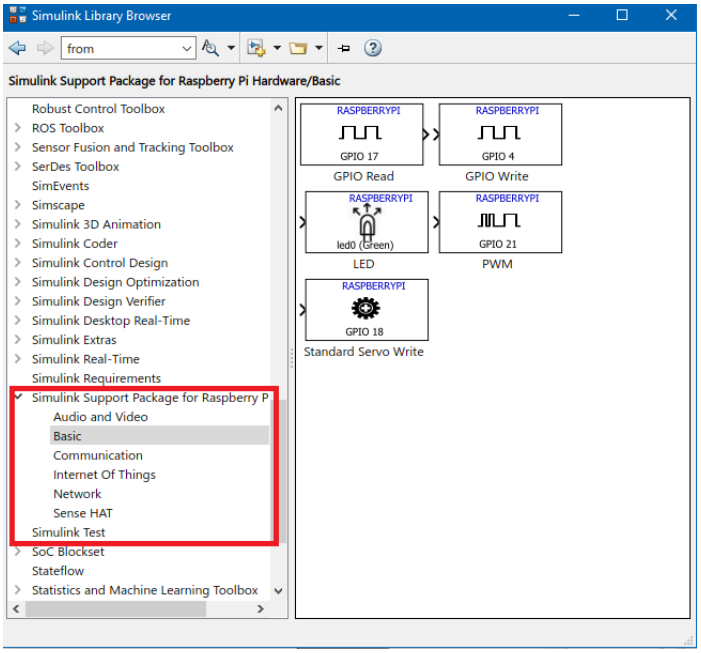


Figura 3.26: Biblioteca de bloques de Simulink para Raspberry Pi.

Diseño e implementación del circuito del sensor ultrasónico HC-SR04.
Para culminar con el proceso del HIL del sensor, se diseñó el circuito que se muestra en la figura 3.27 con base al código desarrollado en el IDE de Python de la Raspberry Pi, ya que dentro del mismo código se establecen los pines de entrada y salida utilizados para la lectura de las señales del sensor.

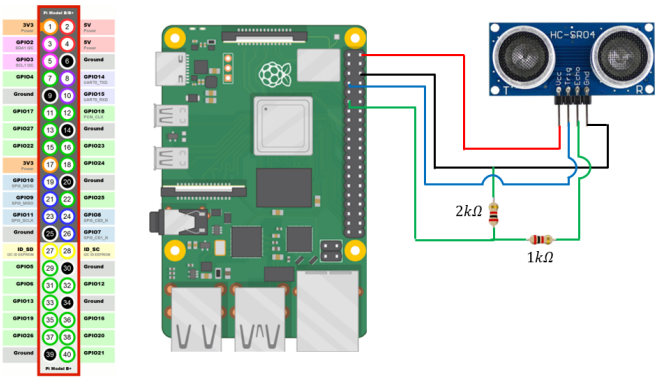


Figura 3.27: Diseño del circuito para el HIL del sensor ultrasónico.

En las conexiones del circuito se puede observar que de por medio se encuentra un divisor de voltaje, el motivo por el cual se colocó es porque el voltaje de alimentación al sensor es de 5 V y el voltaje máximo de entrada de los pines de la tarjeta es de 3.3 V. Al colocar estos valores de resistencia es posible obtener dicho voltaje para no generar daños por sobrepasar los límites que se establecen para el correcto funcionamiento de la Raspberry Pi. La figura 3.28 presenta el circuito implementado de manera física en donde se puede apreciar que se añadió un LED.

Este sirve para emular la presencia del operador humano dentro del área de trabajo del robot, es decir, si la luz del LED es verde, el área de trabajo se encuentra libre y el robot puede continuar realizando la tarea, en caso contrario, si la luz es roja indica que el sensor detectó la presencia de una persona dentro del espacio de trabajo y el robot debe detenerse.

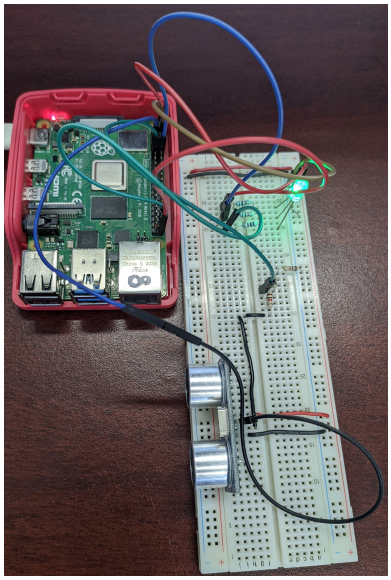


Figura 3.28: Implementación del circuito.

3.2.2. Validación del algoritmo inteligente generador de trayectorias

Las distintas tareas que un robot es capaz de realizar contemplan una serie de requerimientos tales como tolerancias geométricas y de posición, velocidad de procesamiento, tipos de movimiento, etc. Sin embargo, un sistema generador de trayectorias difícilmente podrá usar esta información para generar la ruta dentro del espacio de trabajo del robot. Por lo cual, como menciona [65] es necesario una caracterización de tareas que establezca criterios de selección para interpretar y traducir los requerimientos de la tarea en los parámetros necesarios en la planeación y el control del robot. Como ejemplo, para diversas tareas se pueden tomar en cuenta algunos criterios tales como:

- Suavidad de la curva espacial que representa la ruta a seguir por el efector final.
- Límite de tiempo para el traslado de una pieza de un punto a otro.
- Ausencia de discontinuidades en la trayectoria.
- Mantener la posición fija del efector final.

Esta serie de criterios y muchos más pueden llevarse a cabo con un cierto número de puntos de paso dentro de la trayectoria del robot. Pudiera parecer en principio que conviene tomar un elevado número de puntos, puesto que cuanto más grande es el número de instantes en los que se muestrea la trayectoria, se obtendrán mejores resultados, por cuanto que la trayectoria final pasaría por estos puntos ajustándose a lo especificado con mayor precisión. Sin embargo,

debe considerarse que cada uno de estos puntos debe ser transformado a sus correspondientes coordenadas articulares y ser después utilizado en alguno de los interpoladores que se trataron en el capítulo anterior. Por tanto, es claro que el costo computacional, limitado por la necesaria respuesta en tiempo real del sistema de control del robot, desaconseja que el número de puntos cartesianos a tomar crezca de manera indiscriminada como lo menciona [1].

Existen también criterios adicionales en algunas tareas donde debe ser regulada la velocidad o aceleración entre los distintos puntos de paso, a lo cual se le llama movimiento protegido y es utilizado generalmente en tareas de *pick-and-place* [58] como se muestra en la figura 3.29.

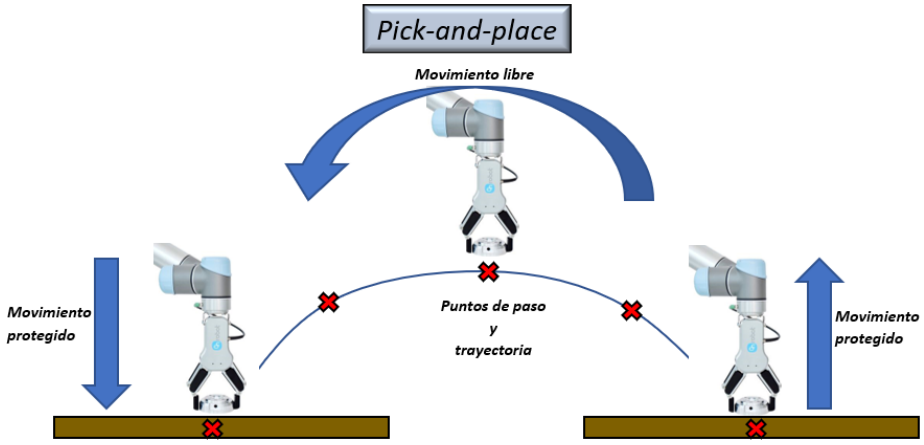


Figura 3.29: Tarea de pick-and-place con movimiento protegido.

Considerando lo anterior, para generar la trayectoria con la cual se validó el algoritmo inteligente, se eligió una tarea de *pick-and-place* dentro del espacio de trabajo del robot. Para ello, se seleccionaron 5 puntos de paso dentro del prototipo virtual de la estación de ensamble en ADAMS® que debe seguir el efector final y, mediante la cinemática inversa se obtuvieron las posiciones articulares para cada uno de los puntos. En la figura 3.30 se muestra la trayectoria generada por los puntos de paso en ADAMS®.

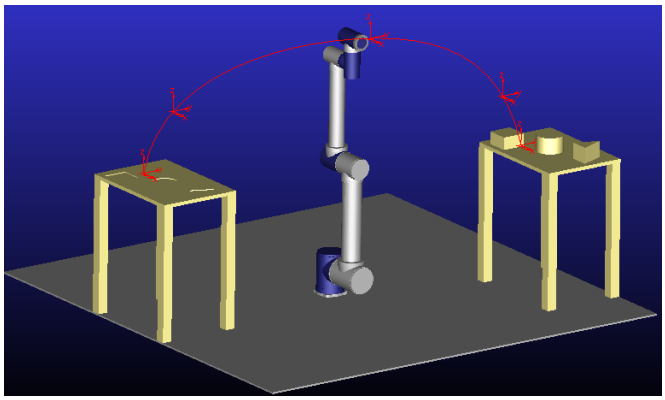


Figura 3.30: Puntos de paso para la trayectoria del efector final.

Una vez que se dispone de la secuencia de configuraciones articulares por las que debe pasar el cobot para realizar la tarea, lo siguiente es unir mediante un spline cúbico la sucesión de puntos articulares implementando condiciones de configuración-tiempo de paso. En este caso por ser un spline cúbico se asignaron cuatro condiciones, dos de posición y dos de velocidad. La figura 3.31 presenta el spline cúbico que se genera mediante las condiciones de configuración asignadas para cada articulación.

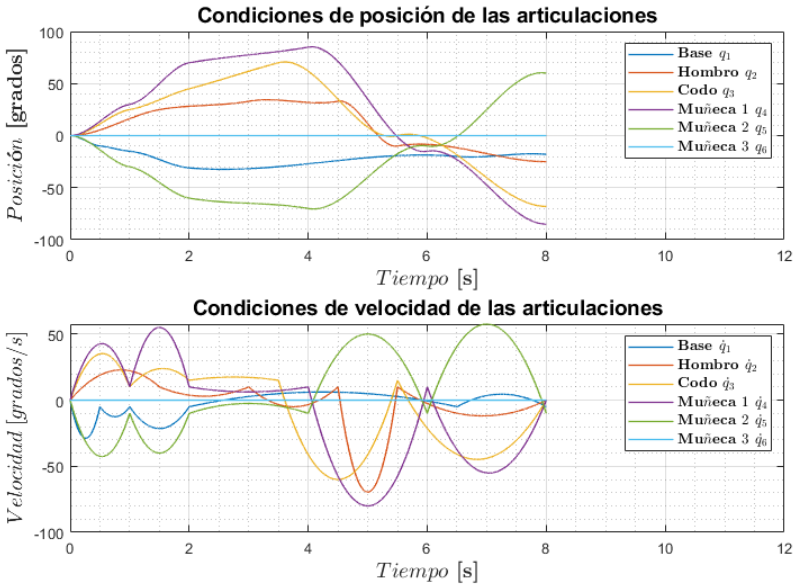


Figura 3.31: Condiciones de configuración-tiempo para la trayectoria.

Finalmente, se diseñó dentro de Simulink el sistema que dota al cobot de inteligencia para realizar una acción una vez que se detecte la señal del sensor. En este caso se asignó la acción para que el robot se coloque en una configuración articular segura, asignando para cada una de las articulaciones una posición deseada las cuales evitan que un operador humano se impacte de manera directa con la estructura mecánica del cobot. La estructura del sistema del algoritmo inteligente generador de trayectorias se presenta en la figura 3.32, el cual fue implementado para cada una de las articulaciones del cobot.

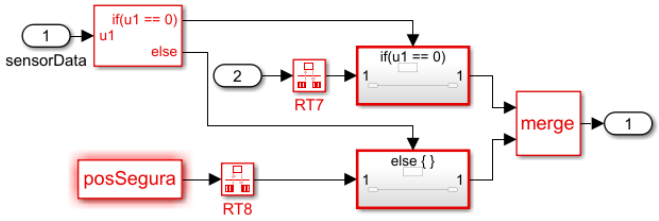


Figura 3.32: Algoritmo inteligente generador de trayectorias.

Teniendo todos los elementos necesarios para el algoritmo inteligente generador de trayectorias, lo siguiente es integrarlos al sistema en lazo cerrado como muestra en la figura 3.33.

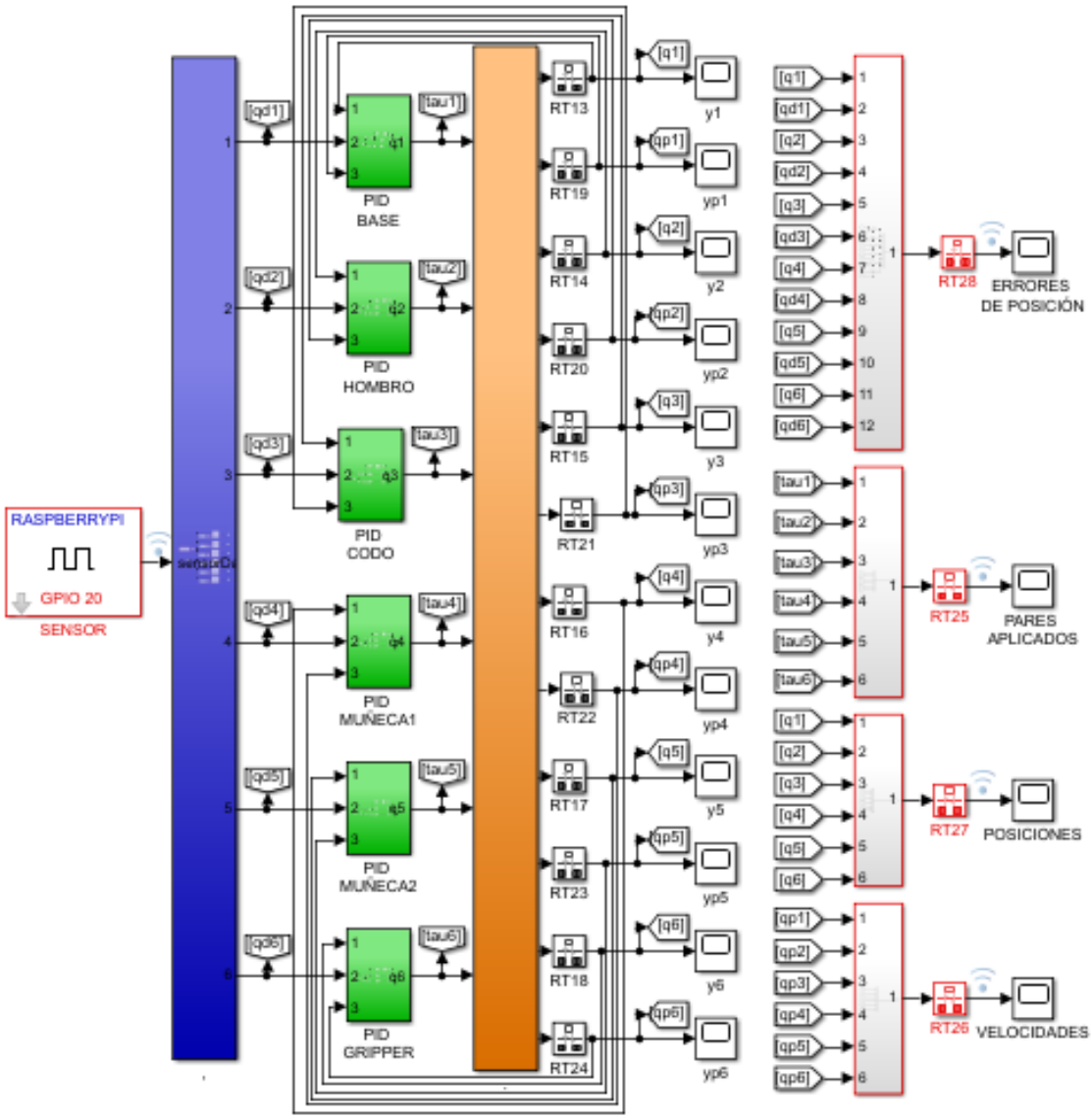


Figura 3.33: Sistema en lazo cerrado con algoritmo inteligente generador de trayectorias.

En la figura del sistema se pueden observar los distintos elementos que lo integran, de los cuales se destacan el bloque que se encarga de realizar la lectura proveniente del HIL del sensor, de color azul el algoritmo inteligente generador de trayectorias, el control PID para cada articulación y el bloque que contiene al cobot. También, es posible visualizar un elemento en cada una de las salidas de los distintos bloques, este elemento lleva el nombre de *Rate transition*. Estos bloques son parte fundamental para que el sistema en lazo cerrado pueda llevar a cabo todo el proceso de co-simulación ya que estos permiten manejar la transferencia de datos entre bloques que operan a diferentes velocidades, es decir, el bloque rate transition puede actuar como una retención de orden cero en una transición rápida a lenta o de caso contrario, como una unidad de retardo en una transición lenta a rápida [8]. Por lo cual, el principal motivo de que se integraran estos elementos en cada una de las salidas de los bloques

fue porque la lectura del HIL del sensor, el generador de trayectorias y el bloque del cobot funcionan a distintas velocidades de transferencia de datos. Por lo tanto, con esta herramienta fue posible homologar las distintas velocidades de transferencia de datos y llevar a cabo la co-simulación. A continuación, se realiza el análisis de los resultados obtenidos en la simulación de la tarea de pick-and-place mediante la implementación del algoritmo inteligente generador de trayectorias en el cobot modelado.

Partiendo con el análisis de la figura 3.34 se cuenta con la gráfica correspondiente a las condiciones de posición de las articulaciones que se establecieron mediante el spline cúbico para que el robot pudiera realizar la trayectoria establecida, también se observa en la gráfica donde no se implementa el HIL del sensor, que el cobot sigue las configuraciones articulares de una manera casi idéntica debido a que se presentan errores de posición en algunos intervalos de tiempo. Además, se tiene la gráfica de los resultados de las posiciones articulares donde se implementó el HIL del sensor y, a diferencia del análisis anterior, en esta se pueden visualizar sobreimpulsos abruptos los cuales son generados por la lectura de la señal del sensor, es decir, cuando el cobot se encuentra realizando la trayectoria y se tiene una señal de activación en el sensor, el algoritmo inteligente generador de trayectorias manda a cada una de las articulaciones a una posición establecida para que el robot se coloque en una pose segura. Posteriormente, si esta señal del sensor desaparece, el algoritmo realiza un análisis para identificar la última posición de las articulaciones antes de la activación del sensor y establece esta posición como la inicial para retomar la trayectoria. Esto se puede comprobar con la gráfica que se muestra de la señal del sensor, en donde cada señal en alto coincide con los sobreimpulsos que se generan en las distintas posiciones articulares.

Por otra parte, las gráficas correspondientes a las velocidades articulares del cobot se presentan en la figura 3.35. En las cuales, nuevamente se tienen las condiciones establecidas en el spline cúbico para la velocidad y, analizando la gráfica donde no se implementa el HIL del sensor se observa que también el cobot intenta seguirlas, sin embargo, en algunos intervalos de tiempo se presentan sobreimpulsos que superan a la velocidad establecida, en mayor parte en las articulaciones de las 3 muñecas. Para el caso en donde se implementó el HIL se tienen los mismos resultados que en las posiciones, sobreimpulsos abruptos debido a la activación de la señal de lectura del sensor. Finalmente, se tiene en la figura 3.36 los errores de posición y pares aplicados en cada una de las articulaciones, en donde los errores de posición sin la implementación del HIL se mantienen en un rango de 5 grados a excepción del codo y el hombro que presentan mayor error en los intervalos de 4-8 segundos. En el caso de los pares aplicados se puede ver la comparación en ambos casos, en donde se observan unos picos muy elevados que corresponden a un incremento abrupto en el par aplicado en las articulaciones cuando se detecta la señal de activación del sensor por parte del algoritmo.

Los picos que se observan en las gráficas de la simulación donde se implementa el HIL del sensor, se deben a que cuando se realiza la activación del sensor, el algoritmo inteligente generador de trayectorias cambia en un instante de tiempo muy corto la posición deseada que originalmente estaban siguiendo las articulaciones del robot, ocasionando un incremento en los distintos parámetros de la simulación en ese instante de tiempo debido a que se incrementa el par aplicado para poder realizar ese cambio de posición deseada de manera rápida.

Con este análisis se llega a la conclusión que la implementación del algoritmo inteligente generador de trayectorias funciona de una manera correcta ya que se observa que se cumple con la acción asignada al mismo, de mandar a las articulaciones a una posición establecida al detectar la señal del sensor. Se resalta también que el control PID implementado funciona de manera eficaz, ya que este consigue el objetivo de mantener a las articulaciones siguiendo la trayectoria deseada a pesar del cambio de posición que se genera con la activación del sensor lo cual ocasiona incrementos abruptos en las velocidades de las articulaciones, sin embargo, una mejora en la sintonización de las ganancias del controlador reduciría los errores de posición que se presentan y se obtendría una mejora en el seguimiento de las trayectorias articulares deseadas.

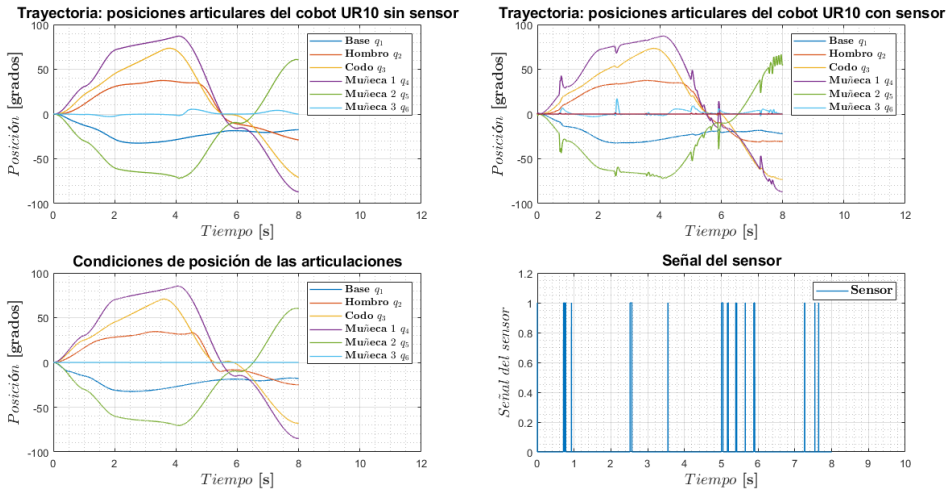


Figura 3.34: Resultados obtenidos para las posiciones articulares mediante la implementación del HIL del sensor.

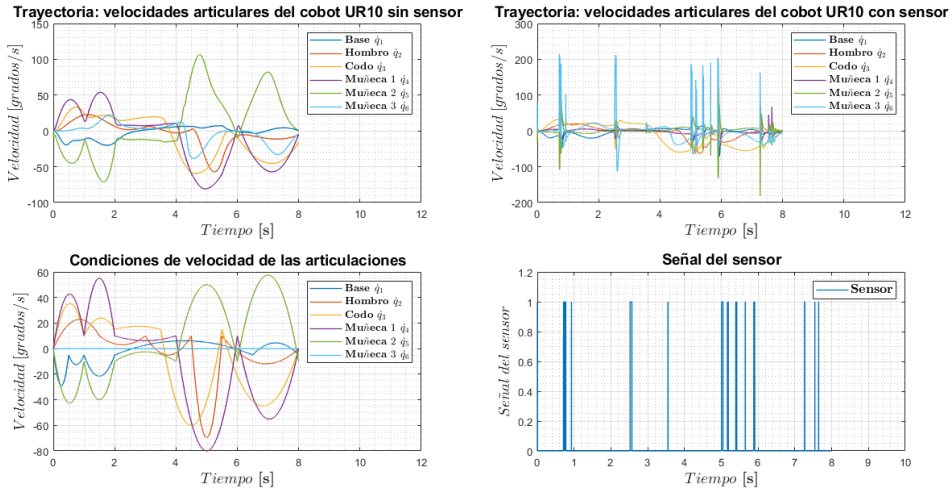


Figura 3.35: Resultados obtenidos para las velocidades articulares mediante la implementación del HIL del sensor.

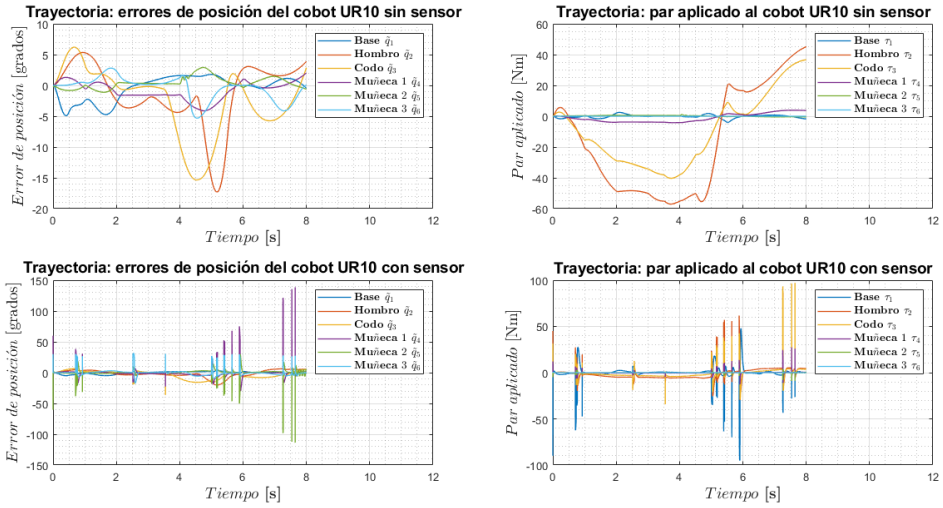


Figura 3.36: Resultados obtenidos para los errores de posición y pares aplicados mediante la implementación del HIL del sensor.

La figura 3.37 muestra una comparación de la trayectoria deseada (rojo) y la trayectoria real (blanco) del efector final que se obtuvo mediante ADAMS® en la simulación del cobot UR10 sin sensor, se puede observar que las trayectorias son similares en algunos puntos, sin embargo, en los puntos de acercamiento a las mesas es donde existe un mayor error de posición. Lo anterior se puede percibir en las gráficas de error de posición que se presentaron en las figuras anteriores.

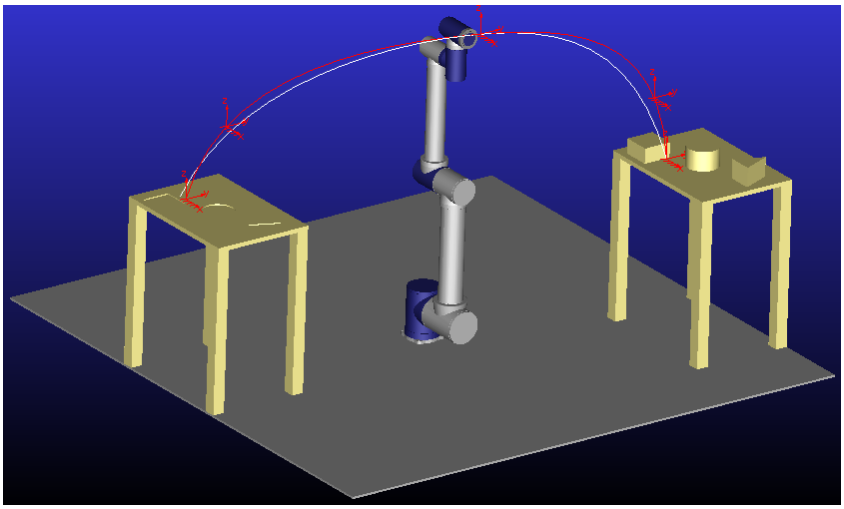


Figura 3.37: Comparación de trayectoria deseada vs trayectoria real obtenida en ADAMS®.

3.3. Validación del modelo dinámico del cobot

El enfoque de Diseño Basado en Modelos ha revolucionado la forma en que se aborda el diseño y desarrollo de robots. Anteriormente, la creación de un modelo dinámico detallado para un robot requería un extenso trabajo analítico, especialmente en el caso de robots con múltiples grados de libertad. Las ecuaciones que describen su comportamiento pueden volverse altamente complejas y difíciles de manejar. Sin embargo, el MBD ha simplificado este proceso al aprovechar el prototipado virtual como una representación confiable del comportamiento del robot. Es por ello que existe la posibilidad de validar el comportamiento del prototipo virtual mediante la comparación con un modelo dinámico desarrollado. Aunque no es necesario construir este modelo dinámico, se puede desarrollar como parte de la etapa de validación.

Con base a lo anterior, se desarrollo el modelo dinámico del cobot UR10 en MATLAB® basado en las ecuaciones planteadas en el capítulo anterior. Además, se implementó un controlador PID para realizar comparaciones con el sistema en lazo cerrado del prototipo virtual y garantizar la consistencia de los resultados. Para llevar a cabo la validación, se realizó una simulación utilizando el mismo escenario para ambos casos con posiciones articulares elegidas de manera arbitraria $\boldsymbol{q_d} = [10\ 20\ 30\ 40\ 50\ 60]^T$, mismas ganancias especificadas en la tabla 3.2 para los controladores PID y se utilizó el mismo tiempo de simulación.

Los resultados de estas pruebas se presentan en las figuras 3.38 y 3.39. Al analizar los resultados de la primera figura, se puede observar claramente que la mayor diferencia en la posición se encuentra en el transitorio, especialmente en la articulación de la base. Sin embargo, esta diferencia no es significativa, ya que se trata de una pequeña desviación angular que dura solo milisegundos. En el estado estacionario, ambas representaciones del comportamiento del robot alcanzan la misma posición. Para las otras articulaciones, los resultados son idénticos tanto en posición como en velocidad, como se puede apreciar en la figura. Por otro lado, los resultados que se observan para los errores de posición de igual manera son consistentes para ambos casos. Se observa una pequeña diferencia en el transitorio, pero en general, los errores de posición son similares. En cuanto a los pares aplicados, los resultados obtenidos del modelo dinámico no superan los límites máximos establecidos y son similares a los del prototipo virtual. Por otra parte para fortalecer aun más la igualdad de los resultados, un punto clave a destacar para esta comparación y que se puede observar en todas las gráficas de los parámetros, es que la convergencia de estos al estado estacionario se da en menos de un segundo.

Estos resultados respaldan la validez y la confiabilidad del MBD como enfoque de diseño. A través de la utilización del prototipado virtual y la comparación con el modelo dinámico, se puede tener confianza en la representación del comportamiento del robot. La implementación del controlador PID y la similitud en los valores de posición, velocidad, errores de posición y par aplicado en las pruebas refuerzan aún más esta confianza, además, contar con un modelo dinámico que funcione de la misma manera que el prototipo virtual es de gran utilidad para el siguiente capítulo correspondiente a la última parte del proceso. En conclusión, el MBD ofrece una solución efectiva y confiable para evitar la necesidad de desarrollar un modelo dinámico, al tiempo que garantiza resultados consistentes y una adecuada representación del comportamiento del robot.

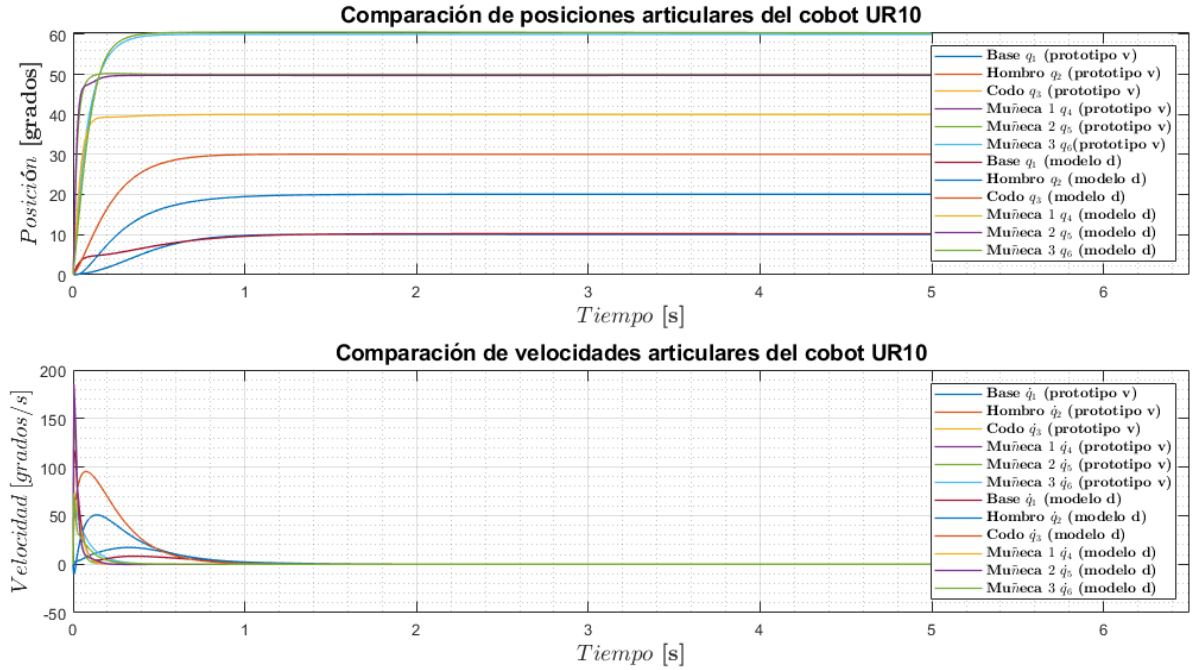


Figura 3.38: Comparación de resultados obtenidos de posiciones y velocidades entre el prototipo virtual y el modelo dinámico del cobot UR10.

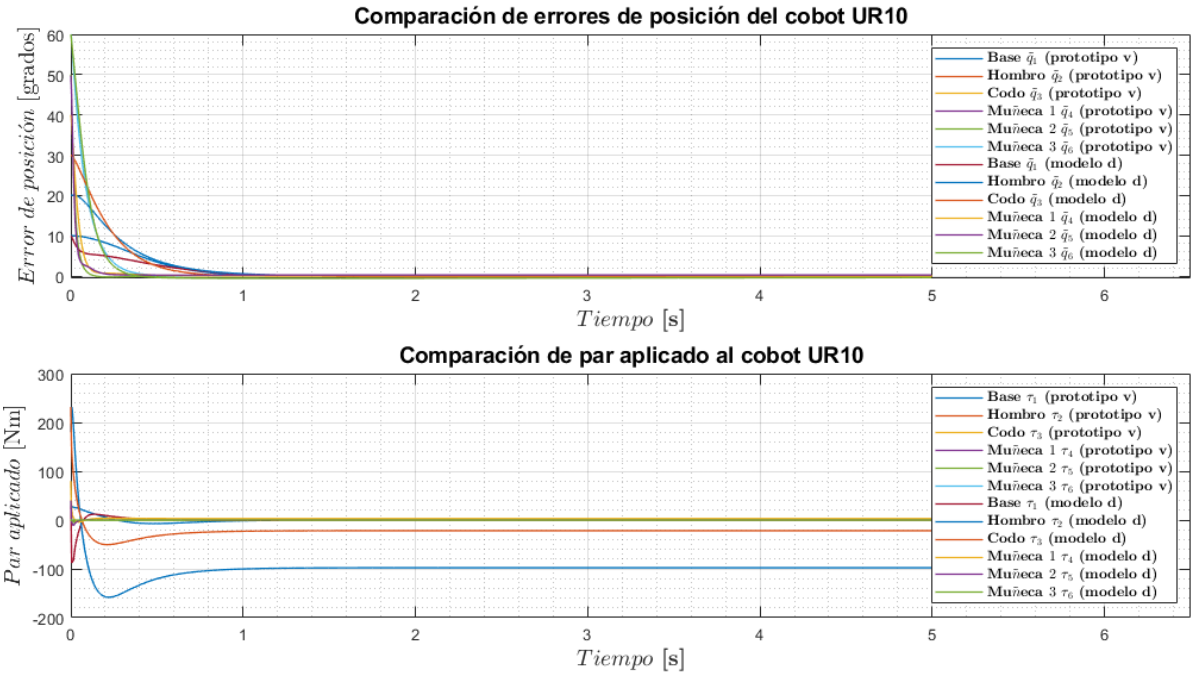


Figura 3.39: Comparación de resultados de errores de posición y par aplicado entre el prototipo virtual y el modelo dinámico del cobot UR10.

Capítulo 4

Validación del proceso MBD

En el capítulo anterior se llevó a cabo la verificación de las diferentes etapas del MBD mediante la implementación de la estrategia de control en el cobot modelado. En este capítulo, nos adentramos en la fase final del procedimiento, donde se presenta toda la información relativa a la integración de los sistemas HIL de cada subsistema desarrollado (figura 3.33), estableciendo una red de comunicación Wi-Fi entre ellos para validar la etapa final del proceso de MBD.

4.1. Comunicación entre los sistemas embebidos (HIL)

Para llevar a cabo la etapa final del MBD la cual implica embeber e integrar los sistemas desarrollados dentro de una tarjeta, en este caso Raspberry Pi, con el objetivo de verificar que el modelo cumpla con los requisitos de funcionamiento y pueda ser posteriormente desarrollado o bien, aplicado en algún prototipo físico. En el contexto de esta investigación, se busca que la estrategia de control desarrollada se implemente en el modelo dinámico del cobot ahora mediante código para poder comparar y validar los resultados de la simulación con los obtenidos en el capítulo anterior. Para ello, se dividió el sistema en lazo cerrado tal y como se muestra en la figura 4.1 en donde la primera parte (S1), consta del sensor en conjunto con el algoritmo inteligente generador de trayectorias y la segunda parte (S2), lo constituye el modelo dinámico del cobot y el control PID. En las secciones posteriores se explica con mayor detalle el desarrollo y funcionamiento de estas dos partes mencionadas anteriormente.

Una vez que ambas partes del sistema están embebidas en sus respectivas tarjetas Raspberry Pi, el siguiente paso consiste en lograr establecer la comunicación entre ellas y así poder realizar el intercambio de datos para llevar a cabo la validación e implementación de la estrategia de control. Para lograr este objetivo se eligió la comunicación a través de Wi-Fi por varias razones, entre las cuales se destacan que es una vía de comunicación entre dispositivos cero invasiva ya que no se requiere de ningún tipo de cableado adicional, como en el caso de Ethernet. Además, se tiene mayor disponibilidad y es fácilmente accesible [69]. Asimismo, el diseño de la plataforma tecnológica tiene como objetivo la escalabilidad a otros tipos de sensores, por lo que Wi-Fi se presenta como una excelente opción para integrar y ampliar la red de comunicación con otras tarjetas sin inconvenientes.

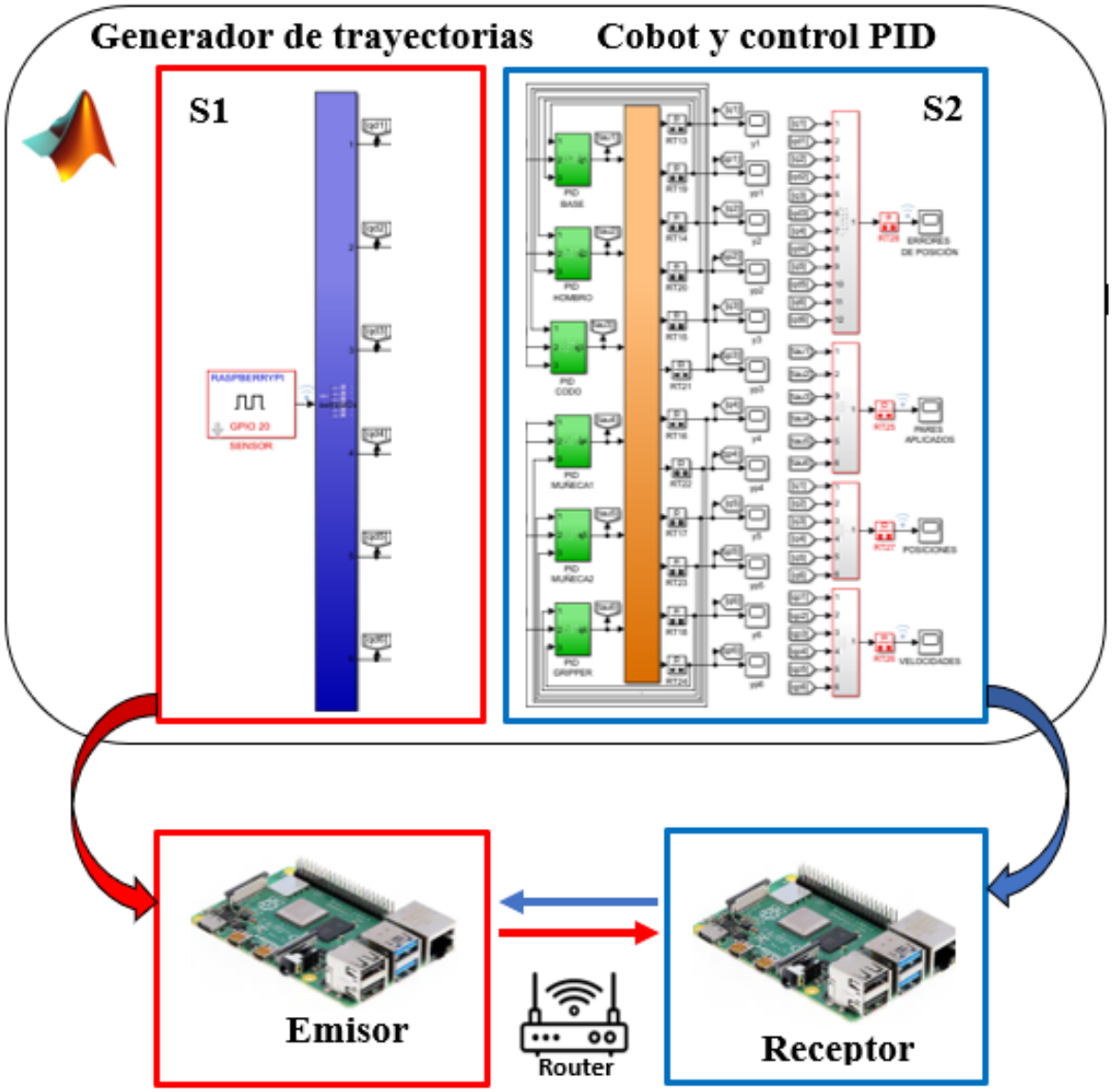


Figura 4.1: Integración de los subsistemas en Raspberry Pi's.

La conexión entre los sistemas HIL está basado en el modelo de comunicación cliente-servidor, donde el servidor es un proceso que se está ejecutando en un nodo de la red, su función es gestionar el acceso a un determinado recurso mientras que un cliente es un proceso que se ejecuta en el mismo nodo o en uno diferente y es el encargado de realizar peticiones al servidor [69]. Ambos utilizan sockets como mecanismo de comunicación, los cuales se definen como puntos de comunicación a través de los cuales un proceso puede enviar y recibir información [70]. Con base a lo anterior, se desarrolló un código en Python para cada sistema embebido, donde el HIL del algoritmo inteligente actúa como emisor (servidor) y el HIL del modelo dinámico del cobot como receptor (cliente). El código para establecer la comunicación entre ambas partes consta, en términos generales, de dos secciones principales, las cuales se representan en el diagrama de flujo mostrado en la figura 4.2 y se describen a continuación.

4.1.1. Establecimiento de la conexión.

La fase inicial del código tanto para el receptor como para el emisor consiste en la creación de un socket. Esto requiere la especificación de la dirección IP del emisor y la asignación de un puerto de conexión. Estos elementos permiten establecer un canal de comunicación entre los dispositivos. Una vez que se ha implementado el socket, en el código del emisor se emplean las primitivas *bind* y *listen*. Estas primitivas permiten al emisor asignar la dirección IP y el puerto al socket creado, y mantenerlo en espera de una solicitud de conexión. Además, se incluye un ciclo en ambos códigos para mantener abierto el canal de comunicación durante un período de tiempo definido. En caso de que la conexión no se logre establecer, el socket se cierra.

4.1.2. Envío y recepción de los datos.

Partiendo de que la conexión entre ambos dispositivos se estableció de manera correcta, en esta parte del código el receptor comienza con la solicitud de los datos al emisor y, una vez que la recibe, se comienza con el envío de los mismos. Para garantizar la integridad de los datos durante la transferencia, se incluye en el código del emisor un número de secuencia para cada dato enviado. Este número permite a ambas partes realizar un seguimiento de los datos enviados y recibidos. En caso de que el receptor no reciba el número de secuencia correcto, se detiene la transferencia y se envía un mensaje al emisor solicitando la retransmisión del dato faltante. Asimismo, el emisor espera la confirmación del receptor de que el número de secuencia ha sido recibido correctamente antes de continuar con el proceso.

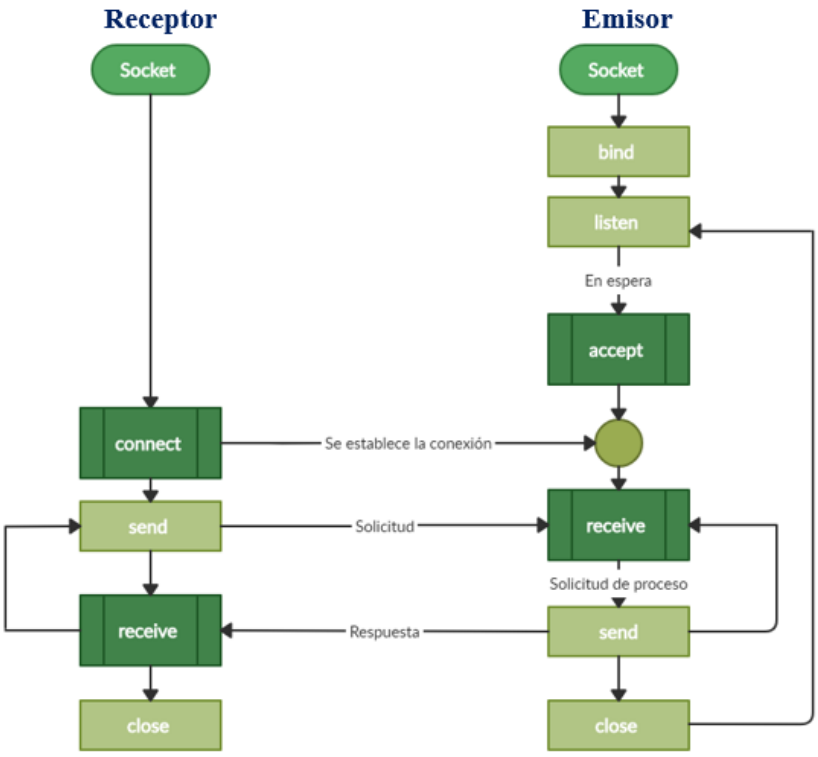


Figura 4.2: Diagrama de flujo de envío y recepción de datos entre HIL.

4.2. HIL del sensor y algoritmo inteligente

El primer sistema embebido que como ya se ha mencionado se encuentra integrado por los códigos Python del sensor y el algoritmo inteligente generador de trayectorias tal y como se observa en la figura 4.3. El código desarrollado consta de 3 bloques principales:

- **Comunicación**. Esta primera parte del código consiste en establecer la comunicación con el receptor que como ya se mencionó en la sección anterior, se lleva a cabo mediante Wi-Fi con la apertura de un socket que se mantiene a la espera de la conexión y solicitud de los datos.
- **Empaquetado de los datos**. Dentro de este bloque se cuenta con la programación de la función del spline que genera las trayectorias articulares que permiten mover al robot para realizar la tarea establecida en la estación de ensamble, también se tiene a las posiciones articulares previamente establecidas en caso de la activación del sensor. Para ambos casos se cuenta con un array que será enviado dependiendo de la señal del sensor.
- **Algoritmo inteligente generador de trayectorias**. El bloque final del código contempla el funcionamiento del algoritmo inteligente generador de trayectorias, que de manera muy general, consta de un ciclo *for* principal que contiene 2 condiciones que pueden ser activadas dependiendo de la señal recibida por parte del sensor. En la primera se realiza el envío del array correspondiente a las posiciones articulares brindadas por la función del spline, si se recibe la señal de activación del sensor se almacena la última posición enviada y se pasa a la segunda condición, en esta se comienza con el envío de las posiciones articulares previamente establecidas para una pose segura del cobot, si la condición vuelve a cambiar, el algoritmo inteligente comienza el envío de las posiciones articulares desde la última posición que dejó almacenada evitando reiniciar la trayectoria.

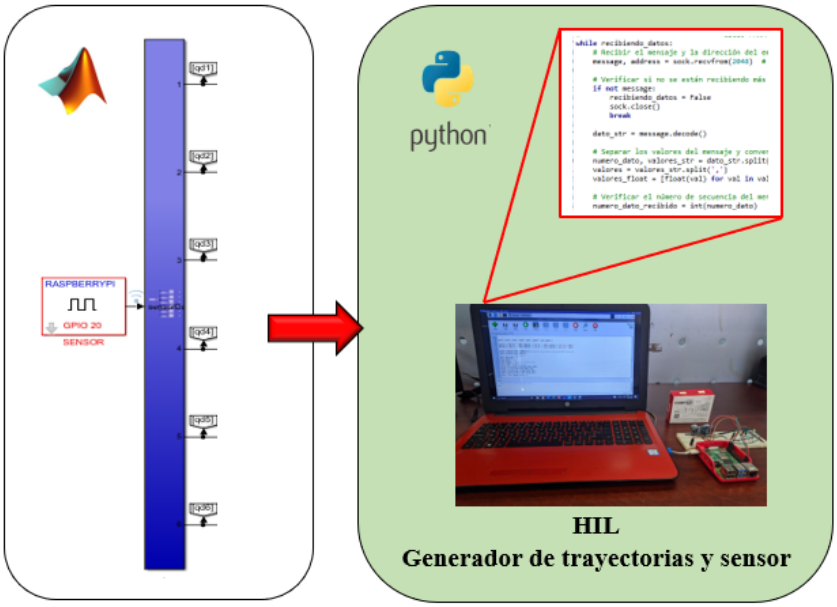


Figura 4.3: HIL del sensor y algoritmo inteligente.

4.3. HIL del modelo dinámico y control PID

Dentro de este segundo sistema (figura 4.4) se implementan el código desarrollado en Python para el modelo dinámico del cobot que se obtuvo dentro del desarrollo del segundo capítulo y el código también en Python correspondiente al control PID, cada uno establecido dentro de una función. Asimismo, al igual que en el sistema anterior se cuenta con un bloque correspondiente a la comunicación, sin embargo la diferencia de la programación en este bloque radica en que como se mencionó desde un principio, el receptor solo se encarga de realizar la solicitud de la conexión y transferencia de datos sin olvidar que también tiene la tarea de notificar al emisor el correcto recibimiento del número de secuencia correspondiente. Una vez establecida la conexión entre ambos sistemas se comienza con la recepción de los datos, conforme estos van llegando se van almacenando en un array y posteriormente son tomados por la función del control PID para establecerlos como la posición deseada y así añadirla a la ecuación del controlador.

Por otro lado, la función del modelo dinámico del cobot se encarga de llamar a la función del control PID para añadirla dentro de la ecuación y así completar los elementos de la misma, después se utiliza el método de integración *Runge-Kutta 4/5* mediante el cual se obtienen las posiciones y velocidades articulares. Para verificar que los datos recibidos durante la transferencia sean los correctos, se implementó la programación de gráficas animadas, estas gráficas van tomando durante cada iteración de la simulación las posiciones articulares que entrega *Runge-Kutta 4/5* y las despliega dentro de una ventana. Este bloque de programación permite visualizar de una mejor manera los resultados en tiempo real y comparar las trayectorias que van tomando las articulaciones para así poder validar la correcta implementación de la estrategia de control.

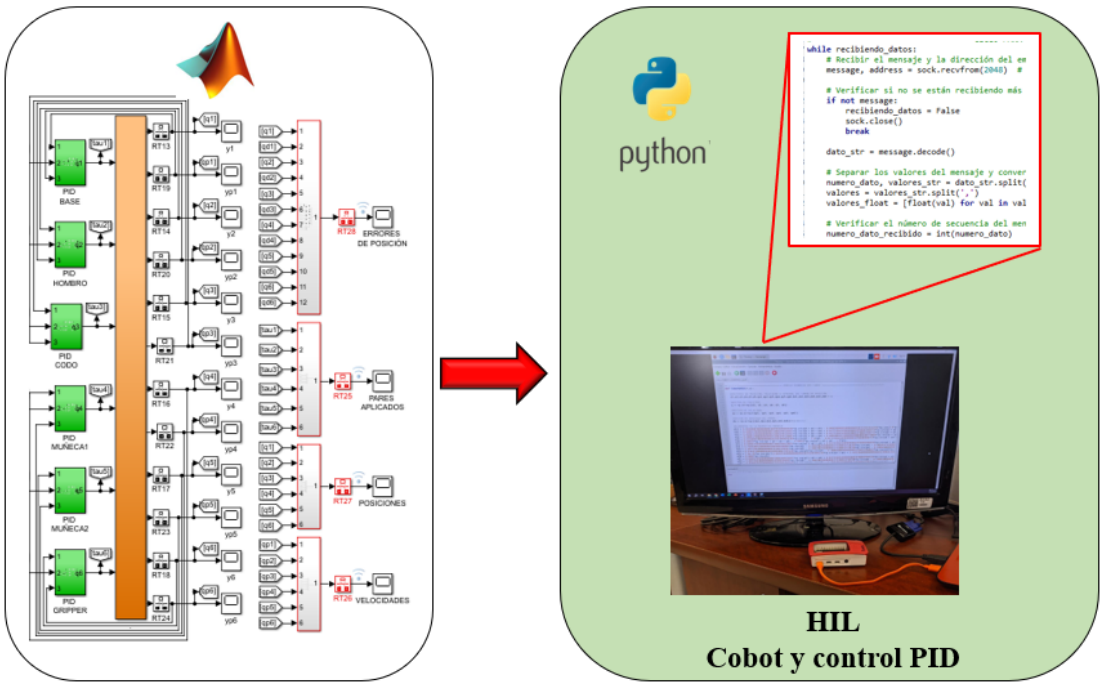


Figura 4.4: HIL del modelo dinámico del cobot y control PID.

4.4. Validación de la red de comunicación

La figura 4.5 muestra la estructura del sistema en lazo cerrado, que es el resultado de la integración de los HIL mencionados anteriormente. En esta representación, se destaca el contenido de cada uno de ellos. La entrada del sistema corresponde a las posiciones articulares proporcionadas por el sistema del algoritmo inteligente generador de trayectorias al sistema del modelo dinámico del cobot, a través de la comunicación vía Wi-Fi.

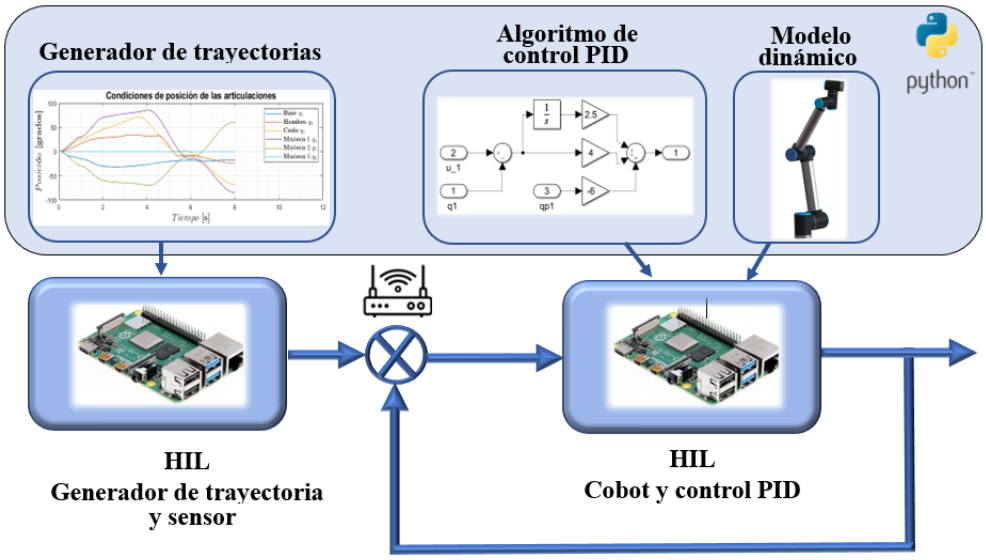


Figura 4.5: Estructura del sistema en lazo cerrado con HIL.

Con el objetivo de visualizar el proceso de transferencia de datos y el funcionamiento de la integración de ambos sistemas, se emplearon dos pantallas, como se ilustra en la figura 4.6. En dicha figura, se puede apreciar que en el lado derecho se encuentra el HIL del sensor y el algoritmo inteligente generador de trayectorias, mientras que en el lado izquierdo se sitúa el HIL del modelo dinámico del cobot. Esta disposición permite una representación clara y separada de ambos sistemas.

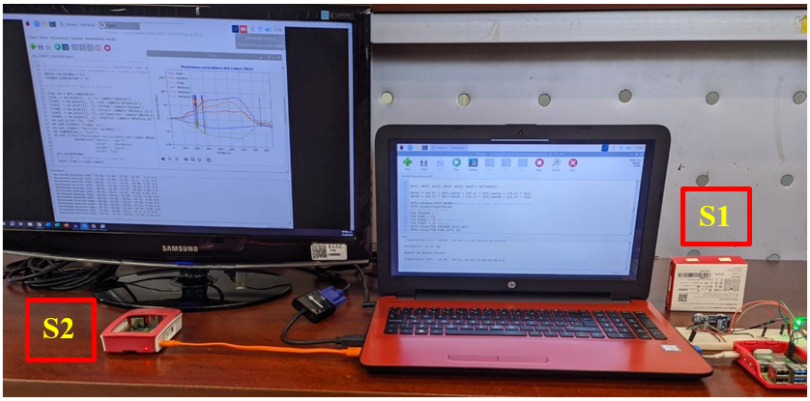


Figura 4.6: Implementación de la red de comunicación entre los HIL.

Los resultados obtenidos mediante la implementación e integración de estos sistemas se presentan en las figuras 4.7 y 4.8. Al analizar la primera figura, donde no se utilizó el sensor, se observa la gráfica correspondiente a las posiciones articulares. Al comparar estos resultados con las implementaciones anteriores, se evidencia una concordancia exacta. Lo mismo ocurre con las gráficas subsiguientes que se muestran en la figura 4.8, ya que los valores de los errores de posición, pares aplicados y velocidades son consistentes para todas las articulaciones.

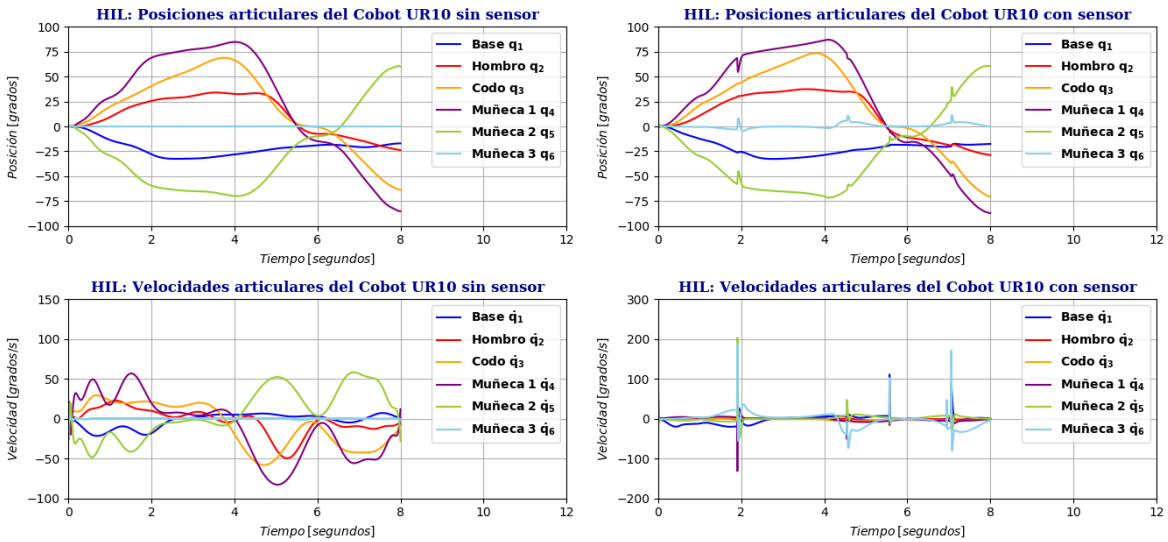


Figura 4.7: Resultados obtenidos mediante la integración de los HIL sin implementar el sensor.

Por otro lado, se presentan las gráficas correspondientes a los resultados obtenidos con la implementación del sensor. En estas gráficas, se observan picos en la simulación en todas las variables, como se mencionó en el capítulo anterior. Estos picos se deben a que, al activar el sensor en el primer sistema, el algoritmo inteligente generador de trayectorias cambia de manera abrupta la posición deseada que originalmente estaban siguiendo las articulaciones del robot. Este cambio ocurre en un instante muy corto de tiempo y provoca un aumento en los distintos parámetros de la simulación, ya que se incrementa el par aplicado para lograr el cambio de posición de manera rápida. Esto es notable en la gráfica de los pares aplicados, se puede observar que también se producen picos elevados durante la implementación del sensor. Estos picos corresponden a que como ya se menciono un aumento repentino en el par aplicado en las articulaciones cuando se detecta la señal de activación del sensor por parte del algoritmo. Sin embargo, es importante destacar que los valores máximos de los pares aplicados no superan los límites establecidos en la tabla 3.1.

Uno de los aspectos destacados de esta implementación, en comparación con la primera en la que se utilizó la co-simulación, es que en estas gráficas se observa una menor cantidad de activaciones inesperadas del sensor. Es decir, el sensor se activa únicamente cuando es necesario, lo que se refleja en gráficas más limpias y consistentes.

Por otra parte, los resultados muestran que modelo dinámico desarrollado ha cumplido con las expectativas al seguir las trayectorias articulares en un tiempo de 8 segundos, sin superar los límites de los pares aplicados. Por lo tanto, se puede afirmar que el desarrollo del modelo ha sido exitoso. Además, el algoritmo inteligente generador de trayectorias ha demostrado un funcionamiento correcto al cumplir con su objetivo de posicionar las articulaciones en una posición establecida al detectar la señal del sensor. Es importante resaltar que el control PID implementado ha mostrado eficacia al mantener las articulaciones siguiendo la trayectoria deseada, a pesar del cambio brusco de posición generado por la activación del sensor. Sin embargo, una mejora en la sintonización de las ganancias del controlador permitiría reducir los errores de posición y lograr un mejor seguimiento de las trayectorias articulares deseadas. Cabe destacar que el código utilizado para la comunicación entre las tarjetas ha funcionado correctamente, permitiendo la recepción completa de los datos y la realización exitosa de las pruebas necesarias. En resumen, la implementación de esta etapa del MBD ha demostrado un funcionamiento coherente y satisfactorio, cumpliendo con los objetivos establecidos y abriendo oportunidades para mejoras futuras en la sintonización del controlador y el seguimiento de las trayectorias articulares deseadas.

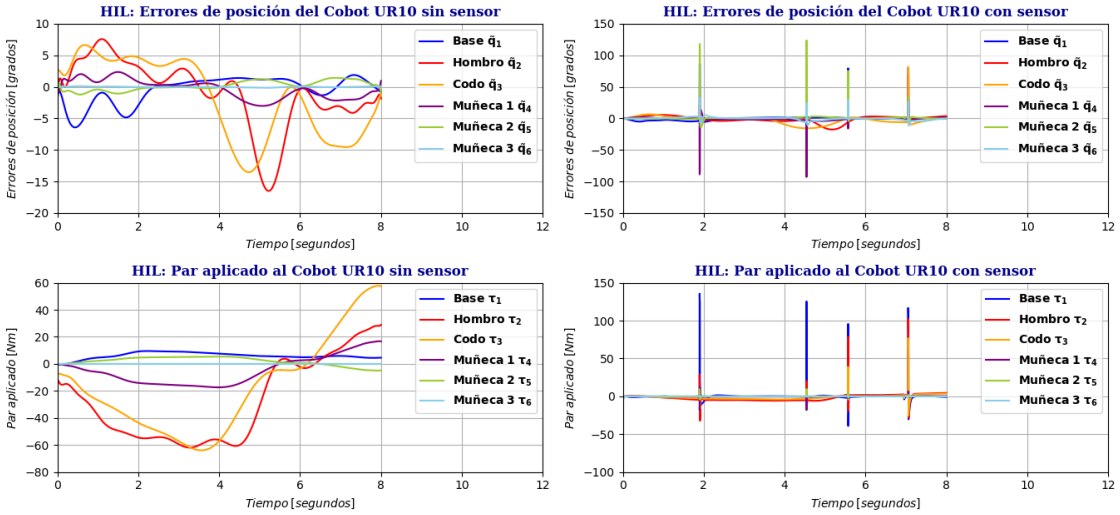


Figura 4.8: Resultados obtenidos mediante la integración de los HIL con implementación del sensor.

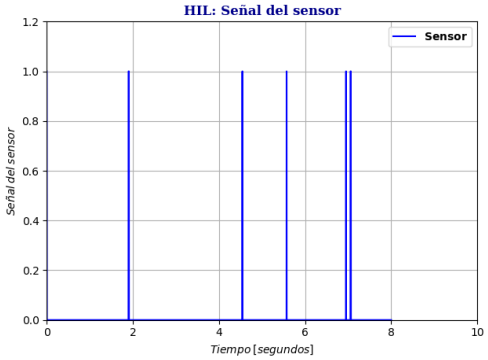


Figura 4.9: Señal de activación del sensor.

Para concluir con la última etapa del MBD, se llevaron a cabo una serie de pruebas con el objetivo de medir el tiempo que tarda el sistema en responder a la activación del sensor y cambiar de posición. Se realizaron un total de 50 pruebas, alternando la activación y desactivación del sensor, como se muestra en la figura 4.10. Para medir el tiempo de respuesta del cobot a la activación del sensor, se tomó en cuenta el tiempo que tarda el HIL del algoritmo inteligente en enviar el cambio de posición cuando detecta la señal de activación del sensor sumado al tiempo que tarda el HIL del cobot en detectar el cambio de posición y ajustar sus articulaciones en consecuencia. El resultado de la suma de estos dos tiempos se muestra en la figura 4.11 para cada una de las pruebas. Es importante destacar que los tiempos de respuesta no superaron los 10 milisegundos, lo cual indica que el robot responde de manera precisa a la activación del sensor.

Además de evaluar el tiempo de respuesta, se realizó un análisis considerando la velocidad lineal máxima de cada uno de los eslabones del robot durante toda la trayectoria. Esto permitió contemplar el peor escenario en el que el robot responde en el tiempo máximo medido durante las pruebas y las articulaciones se mueven a la velocidad máxima. Se determinó la distancia que cada eslabón avanzaría antes de responder a la activación. Los resultados se presentan a continuación en la tabla 4.1 para cada eslabón:

Eslabón	**Velocidad máxima [m/s]**	**Tiempo de respuesta máximo [s]**	**Distancia recorrida [m]**
Base	0.0043122	0.010016	0.000043
Hombro	0.21894	0.010016	0.002192895
Codo	1.0595	0.010016	0.010611914
Muñeca 1	1.4124	0.010016	0.014146548
Muñeca 2	1.3229	0.010016	0.013250119
Muñeca 3	1.3121	0.010016	0.013141947

Tabla 4.1: Resultados obtenidos para caso crítico.

Como se puede observar, las distancias que recorre el robot son menores a dos centímetros. Esto indica que la respuesta del sistema es altamente eficaz, ya que las probabilidades de impactar al operador humano en el tiempo máximo de respuesta son muy bajas. La capacidad de posicionar al robot en tiempos muy cortos contribuye a minimizar los riesgos de accidentes o colisiones. Es importante recordar que los tiempos específicos de tolerancia pueden variar según el contexto, los requisitos de la aplicación y las especificaciones del sistema en particular. En general, si se establece como límite de tolerancia los 10 milisegundos durante estas pruebas, los resultados obtenidos garantizan la respuesta del sistema en tiempo real ya que no se sobrepasan y pudiendo considerarse como tiempo real duro debido al escenario que se plantea.

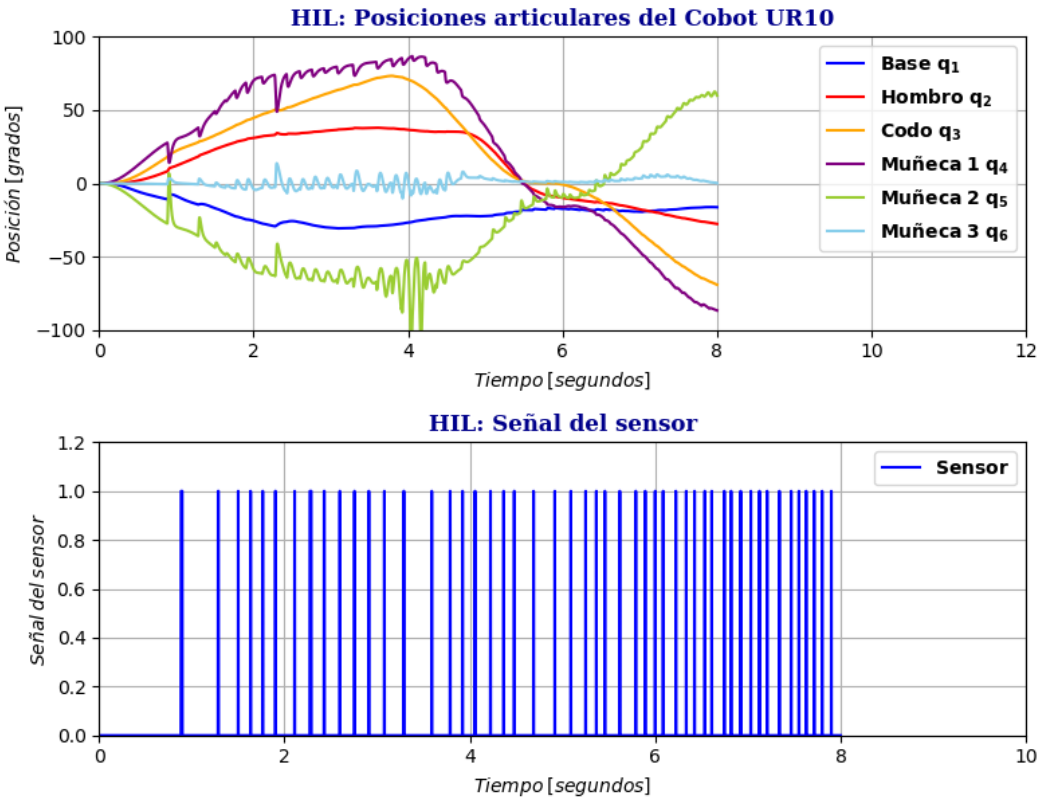


Figura 4.10: Pruebas realizadas para la medición del tiempo de respuesta del robot.

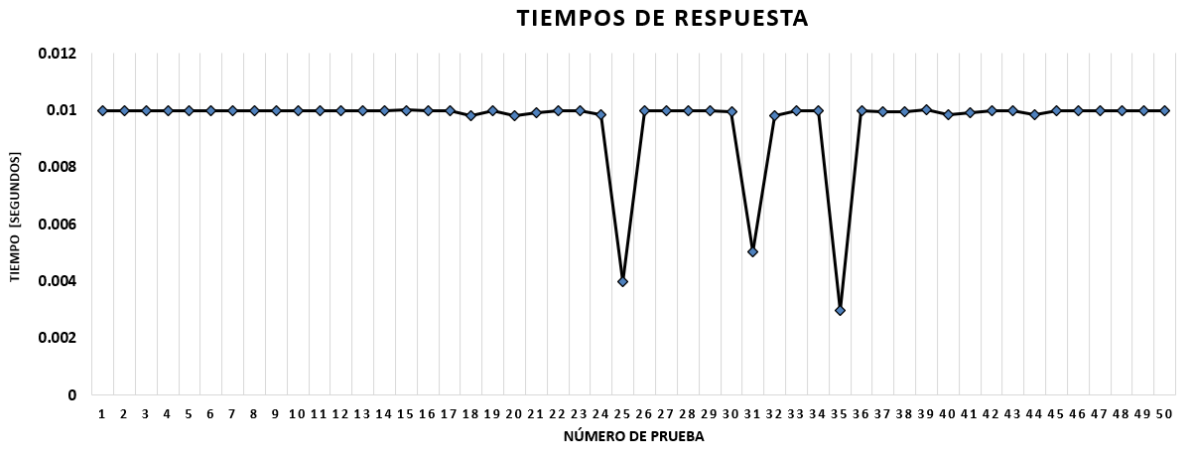


Figura 4.11: Tiempos de respuesta obtenidos en las pruebas realizadas.

Conclusiones

Con base a los resultados obtenidos el proceso de Diseño Basado en Modelos ha demostrado ser una herramienta poderosa y efectiva en el diseño y desarrollo de sistemas de control de robots. Una de las principales ventajas de la implementación del MBD es la capacidad de realizar prototipado virtual. Esto implica el desarrollo y la evaluación de un sistema completo en un entorno virtual, antes de la implementación física por lo cual, la técnica de prototipado virtual ha sido una herramienta valiosa en este proceso, permitiendo realizar pruebas y validaciones de manera rápida y eficiente en el prototipo virtual del cobot UR10 realizado, lo cual resulta en ahorros significativos de costos y tiempo. Estas pruebas habrían sido imposibles de probar de la manera tradicional debido a que el cobot comercialmente tiene una arquitectura cerrada, lo que genera limitaciones para desarrollo y evaluación de mejoras al sistema.

La implementación exitosa de la comunicación entre los sistemas HIL ha permitido una integración fluida y confiable de los componentes clave del sistema que se desarrollaron a lo largo de este trabajo de investigación, lo cual permitió garantizar la reacción en tiempo real del cobot frente a las señales recibidas por parte del algoritmo inteligente ya que la respuesta del cambio de posiciones articulares no superó los límites de retraso que fueron establecidos para esta tarea. Recordando que los límites de tolerancia específicos pueden variar según la tarea a realizar por parte del robot y las especificaciones del sistema en particular, los resultados aquí obtenidos brindan las pautas para adaptar y ajustar el sistema a otras situaciones y realizar pruebas para distintas áreas de la industria donde las tareas sean diferentes.

El sistema en lazo cerrado del robot implementado en este estudio ha demostrado su eficacia al cumplir con los objetivos establecidos. Uno de los logros más destacados es la capacidad de seguir las trayectorias articulares de manera precisa. Esto se debe en gran medida al uso del controlador PID, que ha demostrado un control efectivo sobre los pares aplicados en las articulaciones del robot. Este controlador PID controló eficazmente los pares aplicados, manteniéndolos dentro de los límites especificados en la tabla de especificaciones. Esto es crucial para garantizar la seguridad y el rendimiento óptimo del robot durante su funcionamiento en una aplicación real. Las ganancias utilizadas en el control PID se mantuvieron constantes a lo largo de todo el proceso de MBD, lo que demuestra la robustez y consistencia del controlador en diferentes escenarios y condiciones de operación.

La capacidad del modelo dinámico para seguir las trayectorias articulares de manera precisa respalda su precisión y fiabilidad en la representación del comportamiento del cobot. Estos resultados validan la elección y desarrollo del modelo dinámico utilizado en el MBD como una

representación adecuada del comportamiento del cobot. El modelo dinámico ha sido capaz de capturar de manera precisa las interacciones y dinámicas del robot, lo que ha permitido simular y evaluar su desempeño en diferentes situaciones.

La co-simulación llevada a cabo entre los software ADAMS® y MATLAB® aumentó las capacidades de simulación, análisis y control durante el proceso de MBD. La combinación de la simulación precisa de ADAMS® con la versatilidad y herramientas de control de MATLAB® brindaron un entorno de desarrollo completo y eficiente. Al combinar estas dos plataformas a través de la co-simulación, se obtiene un enfoque integral para el diseño y la implementación de sistemas robóticos logrando una mayor integración y colaboración entre los sistemas de simulación y control.

El producto final de este trabajo de investigación que es la plataforma tecnológica diseñada del sistema en lazo cerrado, servirá como una herramienta de pruebas y evaluación virtual, basada en la metodología del MBD, proporcionando una gran ventaja en el desarrollo de sistemas robóticos ya que se podrán realizar simulaciones exhaustivas y validaciones previas a la implementación física, lo que ahorra tiempo y recursos significativos. Además, al contar con esta herramienta de pruebas, se puede optimizar el rendimiento del robot y evaluar su comportamiento en diversas situaciones y escenarios, lo que garantiza un desempeño óptimo y confiable en el mundo real.

En resumen, la co-simulación de ADAMS® y MATLAB®, junto con las ventajas inherentes del MBD, proporciona una metodología sólida y versátil para el desarrollo de sistemas robóticos y, con base a los resultados obtenidos se sientan las bases para la replicación de esta metodología en diferentes configuraciones de robots y la exploración de diferentes controladores, con el objetivo de establecer una herramienta de pruebas más versátil y confiable para aplicaciones de robótica en la industria.

Participación en congreso

Se realizó una presentación en la 9na Jornada de Ciencia y Tecnología Aplicada.

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

EL TECNOLÓGICO NACIONAL DE MÉXICO
A TRAVÉS DEL CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO

OTORGA EL PRESENTE

RECONOCIMIENTO

A

RUBEN GANIZ MERO
BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

POR LA PRESENTACIÓN DEL ARTICULO:
PROTOTIPADO VIRTUAL Y CO-SIMULACIÓN ADAMS-MATLAB DE UN COBOT UR10 PARA LA IMPLEMENTACIÓN DE UNA ESTRATEGIA DE CONTROL DE POSICIÓN BASADA EN PID
EN EL MARCO DE LA 9ª JORNADA DE CIENCIA Y TECNOLOGÍA APLICADA, CELEBRADA DEL 16 AL 18 DE NOVIEMBRE DE 2022, EN EL TECNM/CENIDET

CUERNAVACA, MORELOS, 16-18 DE NOVIEMBRE DE 2022

DRA. YESICA IMELDA SAAVEDRA BENÍTEZ
DIRECTORA DEL CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO

GR0100922
http://constancias.cenidet.tecnm.mx

Sello Digital:
eyWpQh355j+9Rw+zj9ZuhvKT3Q0aTfhIJytPuztM6IO18rnL4jUwnrC2goRSrWogBNemQEHlAxzai2K/L9autV
E4/oEiXqC58PMx3/qpq3WjXO//n/IM/U3xtRoZCBJoGfz9pKeohq0eqQ6uVZYA9u4fyVQckCg4N16L24buMte0
ALRhYOqQl+NKZdrEr0SYfriz0pA6ITebNdzG4+G6QJOQBEIKRF8NVj0oOhcjgJ+0qKZVmZJXpVAIzhYv9MwQ2f
2I0cEDIzv1Vig7Y+lXmjSqMrdbZdzvKO8P/3e1vjkCkpEKRAfLPB4Zb2HpOJC6rRCQ6+hBjosa3wB/JllMnw==

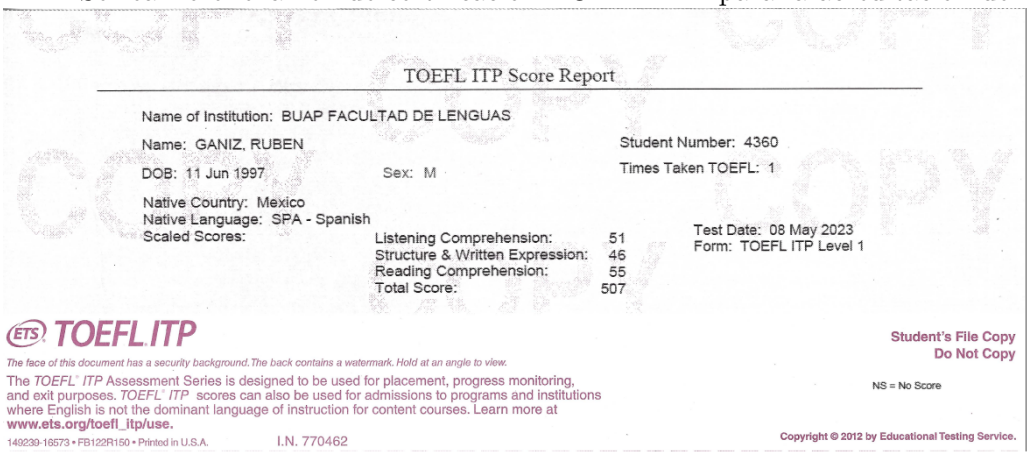
cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

JORNADA DE CIENCIA & TECNOLOGÍA APLICADA

ESPACIO 100% LIBRE DE PLÁSTICO DE UN SOLO USO

Certificación del idioma

Se realizó el examen de certificación TOEFL-ITP para la acreditación del idioma.



TOEFL ITP Score Report

Name of Institution: BUAP FACULTAD DE LENGUAS

Name: GANIZ, RUBEN
Student Number: 4360

DOB: 11 Jun 1997 Sex: M
Times Taken TOEFL: 1

Native Country: Mexico
Native Language: SPA - Spanish
Scaled Scores:

Listening Comprehension:	51
Structure & Written Expression:	46
Reading Comprehension:	55
Total Score:	507

Test Date: 08 May 2023
Form: TOEFL ITP Level 1

TOEFL ITP

The face of this document has a security background.The back contains a watermark. Hold at an angle to view.

The *TOEFL® ITP* Assessment Series is designed to be used for placement, progress monitoring, and exit purposes. *TOEFL® ITP* scores can also be used for admissions to programs and institutions where English is not the dominant language of instruction for content courses. Learn more at **www.ets.org/toefl_itp/use.**

Student's File Copy
Do Not Copy

NS = No Score

149239-16573 • FB122R150 • Printed in U.S.A. I.N. 770462

Bibliografía

[1] A. Barrientos Cruz, *Fundamentos de robótica.* McGraw-Hill, 2007.

[2] C. C. Sáenz, "Industria 4.0,," *Recuperado en Noviembre 20, 2021. [Online]. Disponible: https://biblioteca. unirioja. es/tfe_e/TFE002004. pdf.*

[3] FEMEVAL, "Robots industriales y cobots," *Recuperado en Abril 18, 2022. [Online]. Disponible: https://www.femeval.es/Guia_Robots.pdf.*

[4] Z. M. Bi, M. Luo, Z. Miao, B. Zhang, W. J. Zhang, and L. Wang, "Safety assurance mechanisms of collaborative robotic systems in manufacturing," *Robotics and Computer-Integrated Manufacturing*, vol. 67, 2021.

[5] R. Moreno Gómez *et al.*, "Modelo cinemático-dinámico dimensional de una zona de un motor de encendido provocado para el diseño basado en modelos (mbd) de la ecu en matlab/simulink," 2016.

[6] R. M. Llorente, "Practical control of electric machines: Model-based design and simulation," *Springer Nature*, 2020.

[7] Universal-Robots, "Robot UR10," *Recuperado en Noviembre 13, 2021. [Online]. Disponible: https://www.universal-robots.com/es/productos/robot-ur10/.*

[8] Sitio Web de Mathworks. *Recuperado en: Junio 24 2021. [Online]. Disponible: https://www.mathworks.com/products/matlab.html.*

[9] I. E. Makrini, K. Merckaert, D. Lefeber, and B. Vanderborght, "Design of a collaborative architecture for human-robot assembly tasks," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-Septe, pp. 1624–1629, 2017.

[10] A. Bauer, D. Wollherr, and M. Buss, "Human-robot collaboration: A survey," *International Journal of Humanoid Robotics*, vol. 5, no. 1, pp. 47–66, 2008.

[11] A. Kusiak, "Smart manufacturing," *International Journal of Production Research*, vol. 56, no. 1-2, pp. 508–517, 2018.

[12] "ISO / TS 15066 Robots and robotic devices —Collaborative robots," *International Organization for Standarization (ISO)*, 2016.

[13] P. Gross, "10 of the best collaborative robotics companies that everyone should know about," *Recuperado en: Junio 24, 2021. [Online]. Disponible: https://www.rg-robotics.com/10-best-collaborative-robotics-companies-everyone-know/*, 2017.

[14] G. G. Wang, "Definition and review of virtual prototyping," *Journal of Computing and Information Science in Engineering*, vol. 2, no. 3, pp. 232–236, 2002.

[15] L. Haitao, L. Yuwang, C. Zhengcang, and Y. Leng, "Co-simulation control of robot arm dynamics in adams and matlab," *Research Journal of Applied Sciences, Engineering and Technology*, vol. 6, pp. 3778–3783, 10 2013.

[16] ADAMS, "Multibody dynamics for functional virtual prototyping," *folleto recuperado en: Junio 21 2021. [Online]. Disponible: https://documents.mscsoftware.com/cdn/farfuture/ad0VlEGXj5xF6gRY_AQ7pD-3agPZsWZm7CELZ050X9s/mtime:1489169578/sites/default/files/br_adams_ltr_w.pdf.*

[17] J. Xing, "Getting Started Using Adams / Controls," *MSC ADAMS Training Manual*, 2012.

[18] J. Jacinto, "Robot Standard Improves Safety, Permits Collaboration,," *Recuperado en Junio 24, 2021. [Online]. Disponible: https://www.totallyintegratedautomation.com/2015/09/robot-standard-improves-safety-permits-collaboration/.*

[19] Y.-H. Lee and K.-T. Song, "Real-time obstacle avoidance with a virtual torque approach for a robotic tool in the end effector," in *IEEE, International Conference on Robotics and Automation (ICRA)*, pp. 8436–8442, IEEE, 2021.

[20] N. Van Oosterwyck, *Real Time Human-Robot Interactions and Speed Control of a Robotic Arm for Collaborative Operations.* PhD thesis, PhD thesis, Ph. D. dissertation, 05 2018, 2018.

[21] G. Michalos, S. Makris, P. Tsarouchi, T. Guasch, D. Kontovrakis, and G. Chryssolouris, "Design considerations for safe human-robot collaborative workplaces," *Procedia CIrP*, vol. 37, pp. 248–253, 2015.

[22] Y. Dmytriyev, A. M. A. Zaki, M. Carnevale, F. Insero, and H. Giberti, "Brain computer interface for human-cobot interaction in industrial applications," in *International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pp. 1–6, IEEE, 2021.

[23] W. Li, "Reinforcement learning-based learning from demonstrations for collaborative robots," in *International Conference on Automation Science and Engineering (CASE)*, pp. 1642–1647, IEEE, 2021.

[24] A. Palleschi, M. Hamad, S. Abdolshah, M. Garabini, S. Haddadin, and L. Pallottino, "Fast and safe trajectory planning: Solving the cobot performance/safety trade-off in human-robot shared environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5445–5452, 2021.

[25] M. Ratiu, A. Rus, and M. L. Balas, "Modeling in adams of a 6r industrial robot," in *MATEC Web of Conferences*, vol. 184, p. 02006, EDP Sciences, 2018.

[26] Z. Hu, D. Xu, G. Liu, and C. Jia, "Study on co-simulation technology based on adams and matlab," in *2015 International Symposium on Material, Energy and Environment Engineering*, pp. 595–598, Atlantis Press, 2015.

[27] L. Ángel, M. Pérez, C. Díaz-Quintero, and C. Mendoza, "Adams/matlab co-simulation: dynamic systems analysis and control tool," in *Applied Mechanics and Materials*, vol. 232, pp. 527–531, Trans Tech Publ, 2012.

[28] C. A. M. Montoya, M. J. H. López, and G. A. H. Londoño, "Prototipado virtual y cosimulación aplicado a un manipulador paralelo tipo delta de tres grados de libertad para estudios de comportamiento cinemático y cinético," *Scientia et Technica*, vol. 23, no. 2, pp. 175–186, 2018.

[29] M. P. Perez Sandoval, "Co-simuación adams-matlab para el control de posición del robot gryphon," 2012.

[30] F. Cheraghpour, M. Vaezi, H. E. S. Jazeh, and S. A. A. Moosavian, "Dynamic modeling and kinematic simulation of stäubli© tx40 robot using matlab/adams co-simulation," in *2011 IEEE International Conference on Mechatronics*, pp. 386–391, IEEE, 2011.

[31] K. S. Fu, *Robótica, control, detección, visión e inteligencia*. 1989.

[32] R. M. Colorado, *Cinemática y dinámica de robots manipuladores*. Alfaomega, 2016.

[33] F. Torres, J. Pomares, and P. Gil, *Robots y sistemas sensoriales*. Automática y Robótica, Pearson Educación, 2002.

[34] F. Reyes, *Robótica control de robots manipuladores*. Alfaomega, 2011.

[35] A. I. Basco, G. Beliz, D. Coatz, and P. Garnero, *Industria 4.0: fabricando el futuro*, vol. 647. Inter-American Development Bank, 2018.

[36] V. Tapia, "Industria 4.0-internet de las cosas," *UTCiencia Ciencia y Tecnología al servicio del pueblo*, vol. 1, no. 1, pp. 51–60, 2017.

[37] C. A. González Moreno, "Desarrollo de aplicaciones industriales con robots colaborativos utilizando el middleware de control de robots robot operating system," 2018.

[38] M. Vagaš, A. Galajdová, and D. Šimšík, "Techniques for secure automated operation with cobots participation," in *International Carpathian Control Conference (ICCC)*, pp. 1–4, IEEE, 2020.

[39] E. Zakharyan, "Desarrollo de aplicaciones mediante robots colaborativos basadas en interfaces naturales hombre-máquina," 2018.

[40] KUKA, "Ventajas con la colaboración hombre-robot,," *Recuperado en Noviembre 20, 2021. [Online]. Disponible: https://www.kuka.com/es-mx/la-producción-del-futuro/cooperación-hombre-robot*.

[41] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the ACM (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.

[42] G. Buttazzo, *Hard real-time computing systems: predictable scheduling algorithms and applications*. Springer Science & Business Media, 2005.

[43] J. A. Stankovic, "Hard real-time computing systems: Predictable scheduling algorithms and applications," in *Proceedings of the IEEE*, vol. 91, pp. 1097–1117, IEEE, 2011.

[44] M. B. A. Jiménez, *Real-time systems: design principles for distributed embedded applications*. Springer, 2015.

[45] C. Cerrada and F.-J. Fernández, "Real-time sensor fusion for safe human-robot interaction," in *International Conference on Robotics and Automation (ICRA)*, pp. 3477–3482, IEEE, 2016.

[46] T.-Y. Shin and J.-M. Huh, "Real-time control for a pneumatic artificial muscle driven arm considering inherent nonlinearities," in *International Conference on Control, Automation and Systems (ICCAS)*, pp. 1667–1672, IEEE, 2009.

[47] R. Jammalamadaka and K. M. Kavi, "Quality-of-control considerations in real-time and cyber-physical systems: A scheduling perspective," in *International Conference on Real-Time Systems (RTSS)*, pp. 203–214, IEEE, 2014.

[48] A. Soror, "Real-time behavior modeling for soft real-time systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 18, no. 5s, pp. 1–26, 2019.

[49] J. Mina, Z. Flores, E. López, A. Pérez, and J.-H. Calleja, "Processor-in-the-loop and hardware-in-the-loop simulation of electric systems based in fpga," in *2016 13th International Conference on Power Electronics (CIEP)*, pp. 172–177, IEEE, 2016.

[50] Sitio Web de Mathworks. *Recuperado en: Noviembre 21 2021. [Online]. Disponible: https://la.mathworks.com/help/physmod/simscape/ug/what-is-hardware-in-the-loop-simulation.html.*

[51] F. Zorriassatine, C. Wykes, R. Parkin, and N. Gindy, "A survey of virtual prototyping techniques for mechanical product development," *Proceedings of the institution of mechanical engineers, Part B: Journal of engineering manufacture*, vol. 217, no. 4, pp. 513–530, 2003.

[52] P. Song, V. Krovi, V. Kumar, and R. Mahoney, "Design and virtual prototyping of human-worn manipulation devices," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, vol. 19722, pp. 551–559, American Society of Mechanical Engineers, 1999.

[53] A. G. de Sá and G. Zachmann, "Integrating virtual reality for virtual prototyping," 1998.

[54] J. D. Cardona, M. Hidalgo, H. Castán, F. Rojas, D. Borro, and H. Jaramillo, "Realidad virtual y procesos de manufactura," *Cali: Universidad Autónoma de Occidente*, 2007.

[55] A. G. Higuera, *CIM: el computador en la automatización de la producción.*, vol. 50. Universidad de Castilla La Mancha, 2007.

[56] M. Software, "Ingeniería asistida por computadora (cae) y software de simulación para la validación de diseño," *Recuperado en Abril 17, 2022. [Online]. Disponible: https://www.mscsoftware.com/product/adams.*

[57] Universal-Robots, "Se cumplen 10 años desde la venta del primer cobot de Universal Robots,," *Recuperado en Noviembre 13, 2021. [Online]. Disponible: https://www.universal-robots.com/es/acerca-de-universal-robots/noticias/se-cumplen-10-años-desde-la-venta-del-primer-cobot-de-universal-robots/.*

[58] M. W. Spong and M. Vidyasagar, *Robot dynamics and control.* John Wiley & Sons, 2008.

[59] K. Kufieta, "Force estimation in robotic manipulators: Modeling, simulation and experiments," *Department of Engineering Cybernetics NTNU Norwegian University of Science and Technology*, 2014.

[60] C. P. Montufar, "Plataforma mecatrónica de arquitectura abierta implementada en robot industriales para la intercepción de trayectorias en tiempo real," 2008.

[61] W. E. Ford, "What is an open architecture robot controller?," in *Proceedings of 1994 9th IEEE International Symposium on Intelligent Control*, pp. 27–32, IEEE, 1994.

[62] J. Park, *Control strategies for robots in contact.* Stanford University, 2006.

[63] W. Gao and M. Cheng, "A new control strategy for robotic systems in tracking tasks," *Mechatronics*, vol. 1, no. 3, pp. 353–366, 1991.

[64] K. Ogata, *Ingeniería de control moderna.* Pearson Educación, 2003.

[65] D. A. S. Bueno and R. M. Angel, "Planeación de trayectorias para un robot en una celda de manufactura," *Revista UIS Ingenierías*, vol. 1, no. 2, pp. 31–41, 2002.

[66] A. B. Torres, "Sistema de planeación de trayectorias de mínimo tiempo para robots bajo el criterio de fault tolerance," 1995.

[67] BIS-Research, "Focus on Payload, Application Sales Channel, Component, and Industry – Analysis & Forecast, 2020-2025," tech. rep., 2020.

[68] A. C. Solé, *Instrumentación industrial.* Marcombo, 2005.

[69] A. S. Tanenbaum, *Redes de computadoras.* Pearson educación, 2003.

[70] S. William, *Comunicaciones y redes de computadores.* Prentice Hall, México, 2004.