GUÍA DE LABORATORIO VIRTUAL PARA LA SIMULACIÓN Y CONTROL DE MOVIMIENTOS DE UN BRAZO ROBÓTICO.

JESÚS RAFAEL OSORIO ALVAREZ

Universidad Antonio Nariño
Facultad de Ingeniería Mecánica, Electrónica y Biomédica Cartagena, Colombia
2021

GUÍA DE LABORATORIO VIRTUAL PARA LA SIMULACIÓN Y CONTROL DE MOVIMIENTOS DE UN BRAZO ROBÓTICO.

JESÚS RAFAEL OSORIO ALVAREZ

Proyectó de grado presentado como requisito parcial para optar al título de:
Ingeniero Electrónico.

Director:
PH. D Cesar Augusto Quinayás Burgos

Línea de Investigación:
Grupo de investigación:

Universidad Antonio Nariño
Facultad de Ingeniería Mecánica, Electrónica y Biomédica Cartagena, Colombia
2021

Agradecimientos

A mis padres, mi hijo y demás personas que siempre estuvieron presente y me esforzaron a ser una mejor persona y un buen profesional.

Resumen

Este proyecto, se basó en como los recursos experimentales de un laboratorio universitario, enfocado en las materias de carrera de la Facultad de Ingeniería Mecánica, Electrónica y Biomédica (FIMEB), de la Universidad Antonio Nariño, ayudan a afianzar los conocimientos y a poner en práctica la teoría aprendida en las materias de robótica, control, algebra lineal entre otras, por medio del desarrollo de un Laboratorio Virtual, el cual, con la ayuda de una interfaz gráfica en MATLAB, permitirá que el estudiante pueda interactuar con la plataforma, procurando así un mejor desarrollo académico, puesto que los laboratorios al momento de realizar este trabajo de grado, no se podrían desarrollar de forma presencial, dadas las restricciones decretadas por el Gobierno Nacional de Colombia a raíz de la pandemia de SARS-COVID-19. Pero gracias a las TIC (Tecnología de la Información y la Comunicación) y sus novedosas características de comunicaciones y aprendizaje, tales como el Internet, las video conferencias, la mensajería instantánea, la telepresencia y la simulación. Han transformado todos los aspectos académicos tradicionales, presentando un enorme potencial para el desarrollo de la educación a distancia.

PALABRAS CLAVE: Brazo Robótico, Simulador virtual, Modelación, Sistemas de control, Laboratorio Remoto, Interfaz Gráfica.

Abstract

This project was based on how the experimental resources of a university laboratory, focused on the career subjects of the Faculty of Mechanical, Electronic and Biomedical Engineering (FIMEB), of the University Antonio Nariño, help to strengthen the knowledge and put into practice the theory learned in the subjects of robotics, control, linear algebra among others, through the development of a Virtual Laboratory, which, with the help of a graphical interface in MATLAB, will allow the student to interact with the platform, thus ensuring a better academic development, since the laboratories at the time of this degree work, could not be developed in person, given the restrictions decreed by the National Government of Colombia as a result of the SARS-COVID-19 pandemic. But thanks to ICT (Information and Communication Technology) and its novel communications and learning features, such as the Internet, video conferencing, instant messaging, telepresence and simulation. They have transformed all traditional academic aspects, presenting an enormous potential for the development of distance education.

KEY WORDS: Robotic Arm, Virtual Simulator, Modeling, Control Systems, Remote Laboratory, Graphic Interface

Contenido

Lista de figuras

Lista de tablas

Introducción

En Colombia los alumnos inscritos en la modalidad de pregrado han venido ascendiendo desde 1970, para dicha época, la población de estudiantes entre 17 y 21 años que ingresaba a las universidades era del 3.9%, a su vez, el nuevo siglo trajo desarrollo de tecnologías y una mayor accesibilidad a las entidades de educación superior, lo que derivó en un incrementó de tal porcentaje a 49% en el 2015, es decir, aproximadamente 2.142.443 estudiantes ingresaron a la universidad. No obstante, lo anterior, este porcentaje de la población aún es bajo, si lo contrastamos con el ámbito internacional, donde la cobertura de acceso a educación formal supera el 80%. (Melo Becerra, Ramos Forero, & Hernadez Santamaria, 2017)

Cada uno de estos estudiantes, ha desarrollado propuestas de grado que han servido a la sociedad, las universidades, las instituciones gubernamentales, fundaciones, entre otras, cada uno desde su énfasis académico. En el caso de los estudiantes de ingeniería electrónica, biomédica, mecatrónica, automatización y control en Colombia, se empezaron a crear modelamientos en simuladores para que académicos e instituciones tomen estos prototipos como ayuda de un desarrollo hacia un fin mayor, como lo realizado por Guardiola de León (2015), quien modeló una prótesis transtibial, con diseño 3D en el software Matlab con el fin de conocer la respuesta de las distintas fuerzas con la que la prótesis interactúa durante la marcha, logrando un modelo matemático que describe la prótesis, y en cual se pueden simular estados, y movimientos acompañados de variables físicas con el fin de observar el comportamiento de la misma ante la marcha humana y como puede afectar el desplazamiento de una persona.

Estudiantes de otros países también han realizado trabajos de grado con esta tecnología, uno de ellos fue García Pozo (2014), de la universidad Carlos III de Madrid, quien realizo una simulación de la mano humana mediante MATLAB, consiguiendo un modelo que tiene en cuenta todos los músculos involucrados en el movimiento de la mano, desarrollando las trayectorias deseadas y permitiendo ejecutar los movimientos de manera correcta.

Estos tipos de modelamientos se desarrollarían hacia un ámbito educativo donde los docentes y los estudiantes a través de la metodología CDIO (Concebir, Diseñar, Implementar y Operar), el cual se utiliza a nivel mundial en la educación de ingenierías, puedan realizar un aprendizaje mediante un laboratorio remoto y relacionar el concepto teórico con la implementación práctica.

Este concepto de modelo de interacción teórico-practica o enseñanza-aprendizaje, y la situación actual de pandemia SARS-COVID-19 a nivel internacional y la no presencialidad debido a las restricciones gubernamentales, crea la necesidad de realizar laboratorios remotos o virtuales, como lo efectuó Lorandi Medina A (2011), y su equipo de trabajo en el artículo de la revista “Revista Internacional de Educación de Ingeniería” Los laboratorios virtuales y laboratorios remotos en la enseñanza de la ingeniería, en donde vieron la necesidad de crear un sistema de apoyo al aprendizaje y de experiencias educativas, en donde trasladan el entorno de enseñanzas a espacios virtuales enriqueciendo el proceso de autoaprendizaje, mediante laboratorios remotos y virtuales que permiten a un número mayor de estudiantes simular fenómenos físicos, modelamientos matemáticos y de sistemas.

Los laboratorios han venido evolucionando gracias las TIC (Tecnología de la Información y la Comunicación), y al distanciamiento social o restricciones impartidas por los entes gubernamentales de cada país, sin embargo, no existe una metodología unificada para el desarrollo, de los laboratorios virtuales o remotos. De acuerdo con lo anterior, los tipos de laboratorios más utilizados en este modelo de aprendizaje teórico-práctico son los establecidos en la tabla 0-1, que se muestra a continuación.

Laboratorio	Tipo	Lenguaje	Interfaz Web	Ámbito
eMersion	LVR	LabView™	Java	Control
Connexions®	IV	LabView™	HTML	Filtros Digitales, Señales
MeRLab	LVR	LabView™	HTML	Mecatrónica
UNED	LVR	Ejs, LabView™	HTML	Control
WebLab	LVR	Java	AJAX	Electrónica
Aurova	LVR	Ejs, Java 3D	HTML	Brazos, V.Artificial, Redes
Robotoy	IR	Java	AJAX	Brazos
Telelabs	LR	LabView™	No	Brazos, Mecatrónica
ACT	LR	Matlab®, Simulink	HTML, Java	Control
Chattanooga	LV	Matlab®, LabView™	HTML	Control, Dinámica
LER	LVR	LabView™	HTML, PHP	Robótica

Tabla 0-1: Características de los laboratorios

Andujar Marquez, J. M. (2010). DISEÑO DE LABORATORIOS VIRTUALES Y/O REMOTOS. UN CASO PRACTICO. *Revista Iberoamericana de Automática e Informatica Industrial RIAI*, 64 -72.

Capítulo_1

1.1 Planteamiento del Problema.

Debido a la pandemia del virus SARS-COVID-19, enfrentada a nivel nacional e internacional, las universidades se han visto forzadas a realizar un cambio estructural, tecnológico y funcional en su metodología, al momento de impartir las clases a sus estudiantes; dado que, se definió un modelo virtual para dichos estudiantes, los laboratorios físicos asociados a la carrera ingeniería electrónica como: circuitos, electrónica, automatización, control, entre otros, quedaron sin poder utilizarse, y ello generó entonces la necesidad de implementar laboratorios virtuales que contribuyeran con el desarrollo de sus capacidades y competencias.

Precisamente en los laboratorios, con las guías a desarrollar, se presenta una mayor interacción entre el docente y el estudiante, donde el primero, por medio del modelo CDIO, potencia las capacidades cognoscitivas, interpretativas, argumentativas y propositivas de los alumnos, permitiendo que estos puedan desarrollar un proyecto, teniendo en cuenta las etapas de diseño, construcción y evaluación de los sistemas y modelos requeridos para la guía de laboratorio y la ejecución de la misma. Es por eso que el estudiante pone en práctica su estudio teórico, sobre los conceptos y elementos básicos de los sistemas de control analógico y afianza su conocimiento por medio de las prácticas, desarrollando las guías de los laboratorios.

Los estudiantes de Ingeniería Electrónica de la Universidad Antonio Nariño, a lo largo de sus estudios de pregrado, deben utilizar varios softwares que los ayuda a desarrollar, aprender y solucionar problemas complejos en las principales asignaturas de la carrera, tales como: Cálculo, Ecuaciones, Algebra lineal, control, programación, sistemas digitales y analógicos, lógica computacional, robótica, entre otras, para ello, el software más utilizado por los estudiantes y docentes es MATLAB, es importante resaltar, que dicho software tiene en su caja de herramienta o ToolBox, un complemento poco conocido y utilizad, que permite modelar y simular cualquier sistema, y este se denomina SIMULINK.

Retomando a la necesidad que tenemos actualmente y al software mencionado anteriormente (MATLAB-SIMULINK), que se utiliza en el 60% de las materias principales de la carrera. Se genera la siguiente incógnita.

¿Cómo contribuir en la formación de profesionales de ingeniería electrónica, utilizando entornos virtuales de aprendizaje, de la teoría de control de sistemas, diseñando, analizando e implementando un control PID, para un brazo robótico virtual?

1.2 Justificación.

La estructuración de información, hipermedia, multimedia, internet o simplemente las TIC (Tecnologías de la información y la comunicación) han venido transformado a la sociedad y mucho más en este tiempo en el cual nos estamos enfrentando a una Pandemia, que no nos permite desarrollar un cronograma de estudio normal y presencial debido al distanciamiento social establecido en el país, por este motivo las universidades y los centros educativos, se han reinventado en sus metodologías de aprendizaje, que incluyen los laboratorios virtuales – "que son instrumentos simulados contenidos en uno o más ordenadores, conectados o no entre sí, con capacidades de gestión y/o aprendizaje de contenido" (Blas & Loyarte, 2011) - como modelos de éxito implementados en diferentes partes del mundo. Colombia no es la excepción a ello, durante los últimos años, se ha querido potenciar las habilidades tecnológicas y de innovación de los estudiantes universitarios, por medio de diferentes laboratorios virtuales, ya que estos son más accesibles económicamente dados sus requerimientos de implementación, son más flexibles en temas de horarios ya que la gran mayoría de la población de universitarios en Colombia debe trabajar, potencia las habilidades de autoaprendizaje y el desarrollo de competencias en menor tiempo y los estudiantes pueden experimentar de mejor manera por medio del ensayo – error, ya que no se tendrá afectación en algún equipo o instrumento de medición.

La creación de laboratorios virtuales, puede apoyar y facilitar el desarrollo de prácticas de las habilidades y capacidades cognoscitivas, interpretativas, argumentativas y propositivas de los estudiantes de ingeniería electrónica de la

Universidad Antonio Nariño, desde el aprendizaje por el modelo CDIO, que tiene como objetivo, por medio del conocimiento, la competencia y la capacidad interpersonal e innovadora de los estudiantes plantear soluciones a problemas del entorno real de la ingeniería, es decir, donde el estudiante aplica los conceptos y elementos adquiridos teóricamente, como también puede ejecutar la práctica, modelando y simulando de una forma interactiva la solución a dichos problemas desde el diseño, la construcción y la evaluación de la efectividad de tales alternativas (Melo Becerra, Ramos Forero, & Hernadez Santamaria, 2017).

La modalidad a utilizar sería el Laboratorio Virtual (LV), en el cual, el estudiante podrá explorar las diferentes configuraciones, por medio de conceptos y elementos de la teoría, lo cual, lo lleva a obtener un útil y practico acercamiento a los sistemas de control en este caso con el desarrollo de guía de laboratorio virtual para la simulación y control de posición de un brazo robótico, la cual le permitirá al estudiante entender y verificar el

comportamiento de un sistema de control analógico, y de esa manera ponerlo en práctica diseñando e implementando un sistema de estos, en el caso de este trabajo, los movimientos de una Brazo Robótico, serán generados por motores simulados, a los cuales se le aplicara la teoría de control en mención.

Matlab/Simulink (Mathworks), es software clásico que proporciona facilidades para la construcción de modelos, de forma gráfica mediante diagramas de bloques, por lo que se ha convertido en una herramienta ampliamente utilizada en la enseñanza de gran parte de las ingenierías, sin embargo, la interactividad es proporcionada por los modelos Simulink (Garcia Pozo, 2014); teniendo en cuenta que todos, los estudiantes tienen acceso a este recurso, y dadas sus funcionalidades, la apropiación de conceptos en la práctica se hace más efectiva, este será el medio en el que se desarrollará la guía de laboratorio virtual objeto de este trabajo, permitiendo con ello evaluar las competencias adquiridas por los estudiantes en las diferentes asignaturas de la carrera por medio del modelo CDIO, para lo cual se diseñaran las rúbricas correspondientes.

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar una herramienta de simulación en Matlab-Simulink que permita implementar una guía de laboratorio para el control de un brazo robótico virtual.

1.3.2 Objetivos Específicos

- Convertir un modelo CAD de un Brazo Robótico a un modelo virtual a través del Toolbox SimsMechanics, para simular los movimientos de las articulaciones mediante el control de motores DC.
- Desarrollar un algoritmo de control PID que permita regular la posición y los movimientos de las articulaciones del prototipo de Brazo Robótico simulado.
- Diseñar una guía de laboratorio virtual de simulación de un prototipo de Brazo Robótico, que le permita al estudiante analizar, diseñar e implementar un sistema de control PID para los movimientos de las articulaciones.
- Evaluar a través de una rúbrica de evaluación la ejecución del laboratorio por el estudiante.

Capítulo_2

2.1 Estado del Arte.

Los avances tecnológicos en la industria ha desarrollo un interés en jóvenes estudiantes de secundaria, quienes optan por desarrollarse profesionalmente con posgrado con énfasis en la robótica y en las nuevas tecnologías. Esto ha permitido que universidades implementaran en su pensum académico la materia de robótica en los postgrados afines a las tecnologías.

En donde docentes y estudiantes a nivel nacional como, Gonzales P y Ospina F, desarrollaron una guía práctica de laboratorio orientada a la enseñanza de la robótica, en el que, generaron un ambiente de aprendizaje, mediante diversas técnicas de programación en diferentes robots y de diferentes complejidades.

Estos desarrollos también los encontramos a nivel internacional como es el caso de Salazar Patin, W. quien desarrollo una interfaz gráfica de control cinemático de un brazo robótico de seis grados de libertad en la universidad politécnica salesiana en la ciudad de Guayaquil en ecuador, donde los estudiantes por medio de la interfaz gráfica, establecieron una comunicación con el brazo robótico, en el cual se tomarían las posiciones indicadas por el estudiante. También se pudo evidenciar como el estudiante implemento conceptos de robótica y cálculos establecidos para las cinemáticas del robot.

No solo los estudiantes, saben la importancia de estos avances, sino también los docentes como es el caso de Pazmiño E. El cual desarrollo una mano robótica para uso docente, el cual le permitirá al docente desarrollar una adecuada intervención al momento de impartir la clase, debido a que cuenta con las herramientas necesarias para el desarrollo teórico practico.

Estos avances también lo vemos reflejado en las telecomunicaciones las cuales también han venido creciendo de forma exponencial, nos han permitido estar siempre conectados

e interactuado con las demás personas. Por lo tanto, tenemos acceso a muchos desarrollos e implantaciones en el ámbito de la robótica a nivel internacional.

De acuerdos a estos avances tecnológicos, han permitido que las universidades tengan un acercamiento u obtén por un mercado internacional, las cuales mediante las comunicaciones pueden llegar a cualquier parte del mundo, Esta metodología o forma de aprendizaje se vio acelerada debido a la pandemia del virus SARS-COVID-19 enfrentada en el 2020 y a la no presencialidad en las diferentes instituciones educativas. La cual permitió a universidades, por medio de sus estudiantes como Farias G. quien desarrollo un informe IEEE para el departamento de matemáticas de la universidad de Murcia en España, sobre laboratorios virtuales, interactivos y remotos. Donde logro vencer el paradigma educacional sobre la educación a distancia, evidenciando el beneficio y el progreso de las tecnologías, también evidencio estos laboratorios proporcionan al estudiante y al alumno herramientas que simulan la iteración de un sistema.

Estos laboratorios virtuales, se convirtieron rápidamente en un sistema eficientes para el desarrollo de las materias teóricas prácticas, como lo evidencia Andujar J. en su artículo Diseño de laboratorios virtuales y/o remotos. Un caso práctico, de la revista iberoamericana de automática e informática industrial del municipio de palos de la frontera del municipio de Huelva - España. Quien desarrolla un recorrido por todos los laboratorios desde el laboratorio tradicional hasta el laboratorio virtual, donde analizo los diferentes problemas asociados a la enseñanza tradicional y donde concluyo con un laboratorio de ensayo de robots LER, cuyo único objetivo es difundir a través del internet un laboratorio con fines educativos y de investigación.

Así de esta manera cada vez más las instituciones educativas promueven el uso de estos laboratorios para el desarrollo en sistemas reales, tal como lo desarrollo Blas M y Loyarte A. con su laboratorio virtual con acceso local y remoto, como simulación interactiva de un sistema real de controles de niveles. Donde lograron implementar un laboratorio virtual y remoto, permitiendo que el estudiante desarrollara su práctica de laboratorio en la distancia, para ensayos sobre una planta ubicada en los laboratorios pertenecientes a la universidad Nacional del Litoral en Santa Fe – Argentina.

En la actualidad con el nuevo método de aprendizaje e-learning, enseñanza y aprendizaje online o modalidad de formación online, encontramos a Calda J. donde implemento un

laboratorio virtual y remoto para la automatización industrial del e-learning. Debido a la dificultad de aprendizaje de la programación de PLC y tiempos de permanencias de los alumnos en las aulas y laboratorios presenciales, en el cual realizo una representación virtual de todos los equipos ubicados en el laboratorio y conectándolos entre sí, basado en un sistema LabView SCADA conectado a los PLC del aula.

De acuerdo a todos estos avances tecnológicos presentados y al fusionar los conceptos de robótica y de laboratorios virtuales, es el desarrollo del proyecto. El cual permite una iteración entre le estudiante y lo aprendido en clase al momento de desarrollar una guía virtual de un brazo robótico de tres grados de libertad.

Capítulo_3

3.1 Robot Industrial

De acuerdo a la RIA (Asociación de Industrias Robóticas), un robot industrial es un manipulador multifuncional reprogramable, capaz de mover diferentes piezas, materias, herramientas o dispositivos, de acuerdo a sus trayectorias variables y programadas para realizar dichas tareas.

Y según la Organización Internacional de Estándares (OIE), habla sobre un manipulador multifuncional reprogramable con varios grados de libertad, capaz de manipular materias, piezas, herramientas o dispositivos especiales según trayectorias variables programadas con el fin de realizar diversas tareas (Salazar Patin, 2015)

3.1.1 Clasificación del Robot Industrial

La Asociación Francesa de Robótica Industrial (AFRI), Califica a los Robots industriales en los siguientes tipos y generaciones:

- **Tipo A:** Manipulador de Control Manual o Telemando.
- **Tipo B:** Manipulador automático con ciclos preajustados, regulados mediante fines de carrera, controlados por PLC y de accionamientos neumático, eléctrico o hidráulico.
- **Tipo C:** Programable de trayectoria continua o punto a punto y carece de conocimiento de su entorno.
- **Tipo D:** Robot capaz de adquirir datos de su entorno, readaptando su tarea en función de estos.
- **1era Generación:** Robots Manipuladores, Repite la tarea programada secuencialmente.

- **2da Generación:** Robots De Aprendizaje, Adquiere información limitada de sus entrono y actúa en consecuencia. Capaz de clasificar, detectar esfuerzos, localizar y adaptar sus movimientos en consecuencia a lo que necesite.
- **3ra Generación:** Robots Con Control Sensorizado, Pose una programación que se realiza mediante un lenguaje neutral, el cual le permite una planificación automática de sus tareas, captando su entorno mediante sensores.

3.1.2 Estructura del Brazo Robótico.

Los brazos robóticos industriales, tienen la siguiente tipología.

Articulaciones

Es la interconexión de dos piezas u objetos que permiten una unión, permitiendo que se mueva relativa a la otra y se conocen cinco tipos de articulaciones:

- **Articulación Prismática:** Formada por dos uniones anidadas, que se desplazan dentro y a lo largo de cada una. El movimiento que presenta es relativo en las uniones, el cual produce un movimiento en línea recta (Extendiéndose o retrayéndose).
- **Articulación Rotacional:** Las articulaciones de revolución permiten que una unión gire sobre un único eje en el otro, como una puerta y una bisagra.
- **Articulación Cilíndrica:** Consiste en una articulación de rotación y una articulación de traslación y poseen dos grados de libertad.
- **Articulación Planar:** Se caracteriza por el movimiento de desplazamiento en un plano y posee dos grados de libertad.
- **Articulación Esférica:** Combina tres giros en tres direcciones perpendiculares en el espacio.

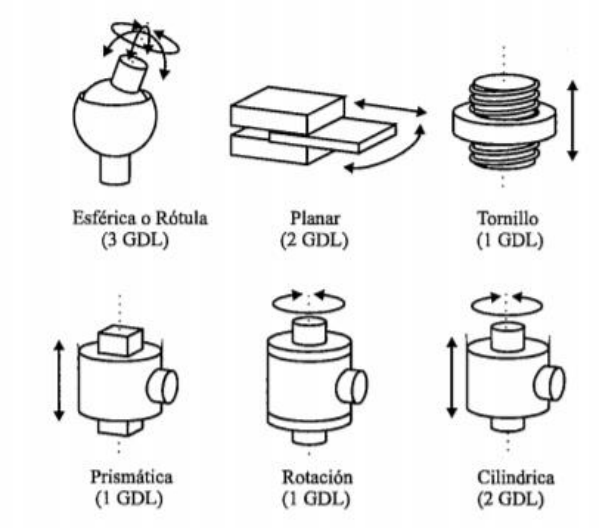


Figura 3-1: Tipos de Articulaciones.

Salazar Patin, W. G. (2015). *PROYECTO DE GRADO "DISEÑO DE UNA INTERFAZ DE USUARIO Y CONTROL CINEMATICO DE UN BRAZO ROBOTICO DE 6 GRADOS DE LOBERTAD PARA LA PLANIFICACIÓN DE TRAYECTORIAS EN SOFTWARE MATLAB Y SIMULINK" ".* Guayaquil - Ecuador: Universidad Poletecnica Salesiana.

Eslabones

Son aquellos que permiten al robot obtener una respuesta rápida del movimiento siendo su estructura lo más rígido y ligero posible

Estructura Mecánica

La estructura mecánica de un brazo robótico, se encuentra conformada por varios eslabones unidos mediante articulaciones y esta unión logra un movimiento relativo entre cada dos eslabones consecutivos, como se observa en la figura 3-2.

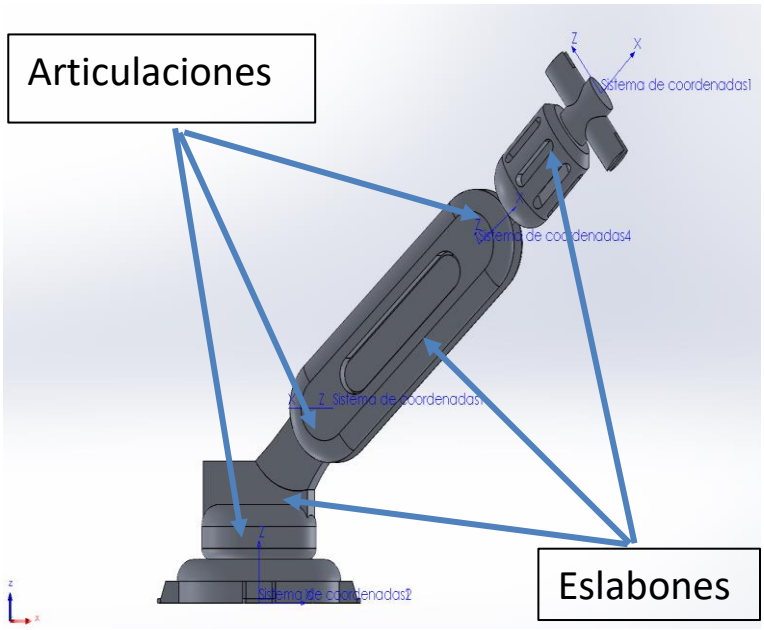


Figura 3-2: Estructura Mecánica

Fuente Propia.

Elementos Terminales

El elemento o efector final de un brazo robótico, es el que se encarga de interactuar con el entorno del brazo. Básicamente es una herramienta de sujeción, que es utilizada para coger y sostener un objeto. Un elemento muy común son las pinzas de sujeción o agarre (Salazar Patin, 2015).

3.1.3 Configuraciones del Brazo Robótico

Los diferentes tipos y mezclas de articulaciones de un robot, dan lugar a las configuraciones, el cual se utiliza dependiendo de su diseño y aplicaciones. Como se observan en las siguientes configuraciones y en la figura 3-3.

Configuración Cartesiana o Rectilínea

Esta configuración, se basa en la posición del objeto. El cual realiza sus movimientos con las articulaciones prismáticas y tienen tres movimientos lineales o tres grados de libertad que representan a los ejes X, Y e Z. Es utilizado cuando el área de trabajo es grande y debe abarcarse. Su movimiento es mediante interpolaciones lineales.

Configuración Cilíndrica

Esta configuración, se basa en la rotación sobre un base, El cual realiza sus movimientos con articulaciones prismáticas para la altura y una articulación rotacional para la base.
Sus movimientos son dos lineales y uno rotacional, presentando tres grados de libertad.

Configuración Esférica o Polar

Esta configuración, se basa en la rotación sobre un base y otro punto, El cual realiza sus movimientos con dos articulaciones rotacionales y una articulación prismática, logrando que el robot apunte a diferentes direcciones y poder extender su efector final a un poco de distancia radial.

Configuración Angular o Brazo Articulado

Esta configuración se basa en una articulación rotacional y dos articulaciones angulares, su área de trabajo es esférica.

Configuración SCARA

Selective Compliance Assembly Robot Arm, Tiene una configuración parecida a la cilíndrica solo que la rotación y el radio se logran en uno o dos eslabones. Está configuración realiza movimientos horizontales con mayor alcance por sus dos articulaciones rotacionales.

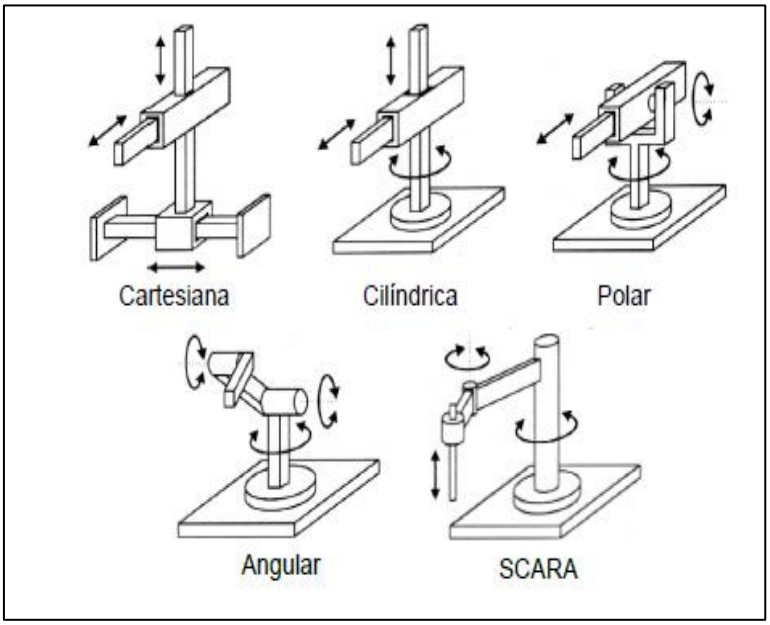


Figura 3-3: Configuraciones Brazo Robótico Industrial

Doménech Jara, J. (2020). *TRABAJO FINAL DE GRADO "Diseño, desarrollo y programación de un brazo robot de 6 grados de libertad".* Valencia, España: UNIVERSITAT POLITECNICA DE VALENCIA.

3.1.4 Grados de Libertad

Los grados de libertad de un brazo robótico, básicamente son los movimientos en un espacio tridimensional, es decir la capacidad de moverse mediante una traslación en tres ejes perpendiculares y una rotación sobre sobre tres ejes perpendiculares, como se observa en la figura 3-4. El movimiento a lo largo de los ejes X, Y e Z es independiente de los otros, y cada uno de ellos es independiente de la rotación sobre cualquiera de sus ejes, el movimiento de hecho tiene seis grados de libertad. (Barrientos, Antonio; Peñin , Luis Felipe; Balaguer, Carlos; Aracil, Rafael;, 1997)

- Delante – Atrás (Forward -Back).
- Arriba – Abajo (Up -Down).
- Izquierda – Derecha. (Left – Right)
- Guiñada (Yaw).
- Cabecear (Pitch).
- Alabeo (Roll).

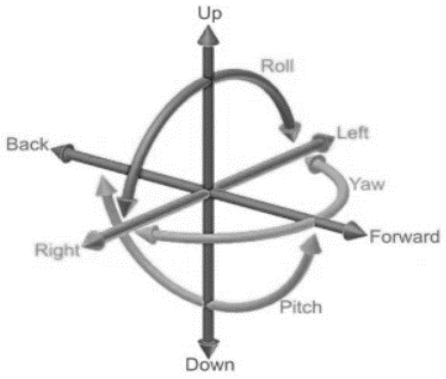


Figura 3-4: Grados de libertad

Barrientos, Antonio; Peñin , Luis Felipe; Balaguer, Carlos; Aracil, Rafael;. (1997). *FUNDAMENTOS DE ROBOTICA.* Madrir - España: McGraw-Hill.

Se Denominan GDL (Grados de Libertad) a cada una de las coordenadas referenciadas independientes, las cuales son necesarias para reseñar o determinar la posición y orientación en el sistema mecánico del robot. Normalmente, en los movimientos de cinemáticas abiertas, se tiene un solo grado de libertad por cada par eslabón y articulación, ya sea de rotación o de traslación. Pero una sola articulación podría tener dos o más grados de libertad. Que al operar sobre ejes que se cortan entre sí. Para describir y controlar las capacidades de movimientos de un brazo robótico industrial es preciso determinar:

- El sistema mundo. Se denomina cuando la posición del punto terminal (o un punto especifico) con respecto a un sistema de coordenadas externo y/o fijo.
- El movimiento del brazo teniendo en cuando los elementos actuadores que aplican sus fuerzas y momentos.

Este análisis se realiza desde el punto de vista mecánico de un robot el cual es atendiendo exclusivamente por sus movimientos cinemáticos. Atendiendo el momento y las fuerzas que actúan sobre sus partes dinámicas, debido a cada elemento, a los actuadores y a la carga transportada por el efector final.

3.2 Cinemáticas del Robot

Por medio de la cinemática examinaremos, el movimiento del brazo robótico con respecto a un sistema de referencia o coordenadas, el cual se sitúa en la base o en eslabón 0, para conseguir una descripción analítica del movimiento y, en particular, de la orientación y la posición del efector o eslabón final del robot.
Las cinemáticas se dividen de acuerdo a las necesidades que se requieren para el brazo robótico, en donde encontramos las siguientes (Rodriguez Zambrana, 2012):

- **Cinemática directa:** determina la posición y orientación del extremo final del robot, con respecto a un sistema de coordenadas de referencia, conociendo los valores de los eslabones y de las articulaciones.
- **Cinemática inversa:** determina la configuración que debe seguir el robot para una orientación y posición del extremo conocido.

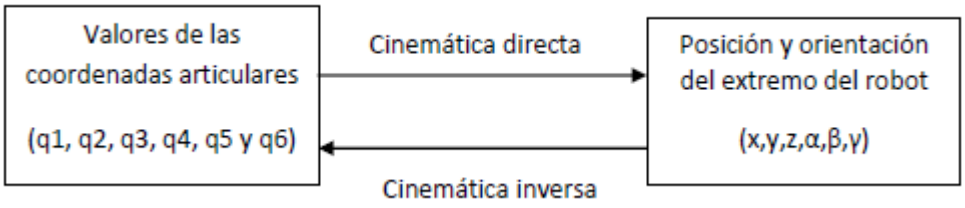


Figura 3-5: Relación entre Cinemática directa e inversa

Rodriguez Zambrana, J. C. (2012). *TRABAJO FIN DE GRADO "Modelo cinemático y control de un brazo robótico imprimible".* Madrid - España: Universidad Carlos III de Madrid.

En la figura 3-5, se puede observar cómo, podemos tener una relación entre las dos cinemáticas y cómo podemos cruzar de la directa a la inversa y viceversa, teniendo los valores necesarios.

3.2.1 Cinemática Directa

El modelado cinemático directo, se basa en determinar la posición y orientación del extremo final del brazo robótico y para determinar dichos elementos nos basamos en los

parámetros de Denavit-Hartenberg. Consiste en originar una matriz sistemática que ayude a establecer un sistema de referencias o coordenadas ligado a cada eslabón del mecanismo, y así determinar la cinemática completa del brazo robótico (Aguirre Gil, Arismendi, & Luis, 2011).

una serie de transformaciones básicas de rotaciones y de traslaciones, que dependen única y exclusivamente de las características geométricas de cada eslabón; pudiendo así asociar un sistema de coordenadas con otro. Las transformaciones básicas que se llevan a cabo son las siguientes:

- Rotación alrededor del eje Z_i un ángulo de θ_i
- Traslación en el eje Z_i una distancia d_i. Vector (0,0, d_i)
- Traslación en el eje X_i una distancia a_i. Vector (a_i,0,0)
- Rotación alrededor del eje X_i un ángulo de α_i

Donde:

- $\boldsymbol{\theta_i}$**:** Es el ángulo que forman los ejes X_{i-1} y X_i medido en un plano perpendicular al eje Z_i, mediante la regla de la mano derecha. Es un parámetro para articulaciones giratorias.
- $\boldsymbol{d_i}$**:** Es la distancia a lo largo del eje Z_{i-1} desde el origen del sistema de coordenadas (i-ésimo – 1), hasta la intersección del eje Z_i con el eje X_i.
- $\boldsymbol{a_i}$**:** Es la distancia a lo largo del eje X_i que va desde la intersección del eje Z_{i-1}con el eje X_i hasta el origen del sistema i-ésimo, en el caso de articulaciones giratorias. En las articulaciones prismáticas, se evalua como la distancia más corta entre los ejes Z_{i-1} y Z_i.
- $\boldsymbol{\alpha_i}$**:** Es el ángulo de separación del eje Z_{i-1} y el eje Z_i, medido en un plano perpendicular al eje X_i, con la regla de la mano derecha. (Barrientos, Antonio; Peñin , Luis Felipe; Balaguer, Carlos; Aracil, Rafael;, 1997)

Una Vez calculada la tabla de los parámetros de Denavit – Hartenberg, se obtiene la matriz de transformación, como se evidencia en la siguiente ecuación.

$$^{i-1}A_{i-1} = RotZ(\theta_i) * Tras([0,0,d_i]) * Tras([a_i,0,0]) * RotX(\alpha_i) \qquad (3\text{-}1)$$

La ecuación (3-1), la podemos convertir en una única matriz, donde el $\mathrm{Cos}\,(\theta_i)$ lo representamos por la expresión C_i y el $\mathrm{Sen}\,(\theta_i)$ se represente con la expresión S_i.

$$ {}^{i-1}A_{i-1} = \begin{pmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_1 S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3\text{-}2) $$

Con la matriz de transformación D-H, se puede calcular las matrices de transformación de cada sistema o de cada articulación.

La matriz de transformación T, la cual relaciona la posición y la orientación del extremo del robot con respecto a un sistema coordenado de referencia fija, que por lo general es la base. Dicha relación es realizada a través de la matriz de transformaciones en donde se asocian cada uno de los sistemas de coordenadas de los eslabones y las articulaciones. Para este caso la transformación viene dada por la siguiente ecuación:

$$ T = {}^0A_1 * \quad {}^1A_2 * \quad {}^2A_3 = \quad {}^0A_3 = \begin{pmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (3\text{-}3) $$

Se observa en la ecuación 3-2 una sub matriz (n, o, a), la cual representa la orientación del manipulador y los vectores P que representan la posición.

3.2.2 Cinemática Inversa

La cinemática inversa se basa, en hallar los valores que adquieren las coordenadas y las articulaciones de un robot, para que su extremo se posicione y se oriente según una determinada localización espacial en el área de trabajo del mismo. El cálculo de la cinemática inversa no es tan sencillo como el caso de la cinemática directa, ya que depende de la configuración del robot. Pese a que existen métodos comunes de programación, para que un computador a partir de la cinemática directa, calcule la

cinemática inversa; los mismos son métodos iterativos que muchas veces suelen ser lentos e incluso no se garantiza la convergencia del resultado (Ogata, 2010).

3.3 Control en lazo abierto y lazo cerrado

Un sistema que mantiene una determinada relación entre la entrada y la salida de referencia, comparándolas y tomando su diferencia como un medio de control, se denomina SISTEMA DE CONTROL RETROALIMENTADO. Estos sistemas no se limitan solamente a la ingeniería, sino que también los podemos encontrar en diversos campos. Un ejemplo de un sistema de control retroalimentado, se encuentra en el cuerpo humano, cuando se hace alguna actividad física o se encuentra expuesto a altas temperaturas el cuerpo inmediatamente realiza un proceso de transpiración para poder regular la temperatura corporal.

Los sistemas de control retroalimentados se dividen en dos clases.

3.3.1 Sistema control en lazo cerrado.

Un sistema de control de lazo cerrado, se caracteriza por que su salida o señal controlada, es re alimentada y comparada con la señal de referencia, de esta forma el sistema es capaz de modificar la señal de referencia en función de la señal de salida, es decir la señal de salida tiene un efecto directo sobre la acción de control, denominando la diferencia entre la señal de entrada y la señal de salida como error del sistema; esta señal es la que actúa sobre el sistema. Es decir, el término lazo cerrado implica el uso de acción de realimentación negativa para reducir el error del sistema.(Ogata, 2010) y (Kuo, 1996).

Ejemplo de un sistema de control de lazo cerrado, es el utilizado en los sistemas de alumbrados públicos, en donde para no dejar funcionando el alumbrado las 24 horas, se instala un sensor y determinar la ausencia de luz y de esta manera encender el alumbrado como se observa en la figura 3-6 a continuación.

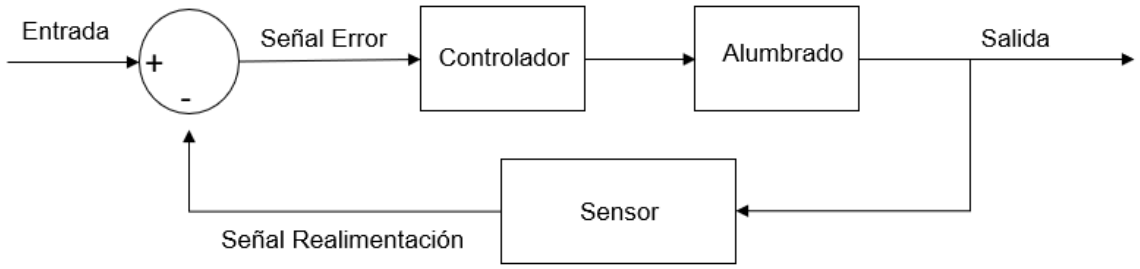


Figura 3-6: Sistema de lazo Cerrado

Fuente Propia

3.3.2 Sistema de control en lazo abierto.

Los sistemas en donde la salida no tiene efecto directo sobre la acción de control, se denominan sistemas de control en lazo abierto. En otras palabras, un sistema de control en lazo abierto no se puede medir la salida y tampoco se realimenta. Por lo tanto, no se confrontan la entrada Vs la salida. (Ogata, 2010) y (Kuo, 1996).

La señal de salida no se compara con la señal de entrada, quiere decir que cada entrada de referencia le corresponde una condición de actividad u operación fija, lo cual nos da como resultado, que la precisión de este sistema depende única y exclusivamente a la calibración.

Los sistemas de control de lazo abierto, se pueden dividir en dos partes: el controlador, y el proceso controlado. Una señal de entrada es aplicada al controlador, cuya salida actúa como una señal de control o señal actuante, la cual regula el proceso controlado, de tal forma que la variable de salida pueda desempeñarse de acuerdo a ciertas especificaciones o estándares establecidos. En los casos más sencillos, el controlador se comporta como un filtro, una unión mecánica, un amplificador u otro elemento de control.

3.4 Diseño de un sistema de control

Al realizar un diseño, de un sistema de control a cualquier sistema deseado, debe tener presente estas tres especificaciones o características:

3.4.1 Especificaciones y Comportamiento del sistema.

Las especificaciones y comportamientos de un sistema de control, tienen en cuenta la precisión, la estabilidad y la velocidad de respuesta, para cada aplicación en que se proporcionan o se manifiestan a través de valores numéricos precisos, pero en algunos casos se muestran en valores cualitativos. En este caso es necesario modificar las especificaciones en el proceso de diseño, debido a que el sistema a controlar puede variar o no presentar un comportamiento estimado.

Las especificaciones no deben ser más restrictivas de lo necesario, para la realización de una tarea específica, debido a que un sistema depende de cómo se presenten las especificaciones del mismo. Ejemplo un sistema de única entrada y única salida, el método más sencillo a utilizar, es el del tanteo y modificación, donde se diseña un modelo capaz de ejercer una tarea específica y posteriormente realizar medidas, para compararlas con las especificaciones iniciales.

3.4.2 Redes de Compensación del sistema.

Existen ocasiones en donde algunas partes de los sistemas no cambian y no se puede actuar, lo cual obliga a modificar o compensar el sistema, de tal manera que permita la utilización de elementos que se adapten al sistema de control diseñado como se muestra en la figura 3-7. Como primer paso se necesita obtener la variación de ganancia del sistema, sin embargo, este método no siempre es válido, debido a que presenta muchas limitaciones como el de empeorar el comportamiento respecto a la exactitud y la estabilidad, por lo cual es necesario introducir nuevos elementos que compensen las derivaciones y se adapten a las especificaciones requeridas.

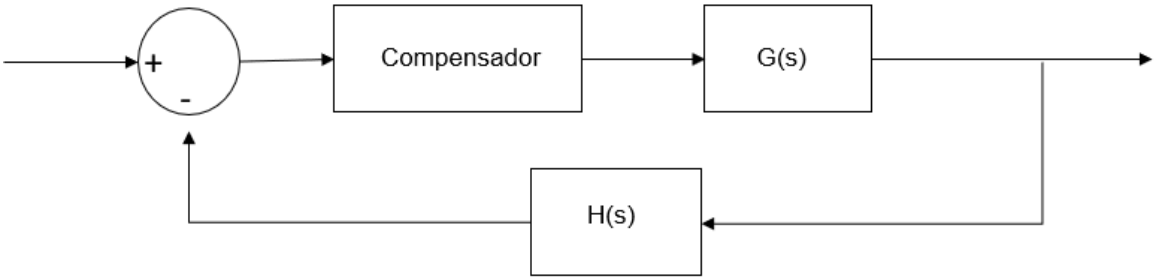


Figura 3-7: Sistema de Compensación en Serie

Fuente Propia

Existen tres formas para compensar el sistema:

- Compensación en Adelanto: La salida en estado estable presentan un adelanto de fase con respecto a la entrada, mejorando la estabilidad del sistema sin variar la exactitud.
- Compensación en Atraso: La Salida en estado estable presenta un atraso con respecto a la entrada, aumentando el orden del sistema, mejorando la respuesta en el estado estable.
- Compensación en Adelanto-Atraso: Contienen un adelanto y un atraso de fase, pero en distintas regiones de frecuencia, aumentando en dos el orden del sistema.

3.4.3 Diseño.

En un diseño de un sistema de control se puede realizar en los siguientes dominios:

- Dominio del Tiempo, Se implementa para poder realizar diseños en sistemas de segundo orden y encontramos lo siguientes:
    - Error estado estable, el cual se especifica mediante una entrada Rampa o entrada escalón.
    - Sobre impulso, tiempo de subida y tiempo de estabilización, el cual se especifica mediante una entrada de Escalón Unitario.
- Dominio de la Frecuencia, Se utilizan sin necesidad de realizar un dibujo detallado y sin conocer la planta. En el cual permite diseñar un control, aplicando especificaciones en el dominio de la frecuencia como lo son: el margen de ganancia y/o el margen de fase. Algunos diseños de frecuencia se basan en los siguientes métodos.
    - Trazas de Bode.
    - Nyquist.
    - Black.
    - Carta de Nichols.

3.5 Sistemas de controles automáticos.

3.5.1 Diagrama de Bloques

Son un sistema de representación gráfica de las funciones que lleva a cabo cada componente y el flujo de señales, como se observa en la figura 3-6. Empleado por los ingenieros de control y de otras áreas, para modelar todo tipo de sistemas, donde se puede describir la composición e interconexión de un sistema.

En un diagrama de bloques, todas las variables se enlazan unas con otras mediante bloques funcionales, cuya finalidad es representar la operación matemática de la señal de entrada hacia el bloque para producir la salida. A su vez las funciones de transferencia de los componentes se introducen en los bloques correspondientes, que se interconectan entre si mediante flechas y de esta manera indicar la dirección del flujo de las señales (Ogata, 2010)

Figura 3-8: Diagrama de Bloques

Ogata, K. (2010). *Ingeniería de Control Moderna.* Madrid: PEARSON.

3.5.2 Dispositivos Detectores

Los dispositivos detectores son los que realizan operaciones matemáticas simples, tales como suma, resta y multiplicación y en alguna de las ocasiones combinaciones de las mismas, como se observa en la figura 3-7. Cabe resaltar que las cantidades que se requieran en las operaciones antes mencionadas deben contener las mismas unidades y las mismas dimensiones.

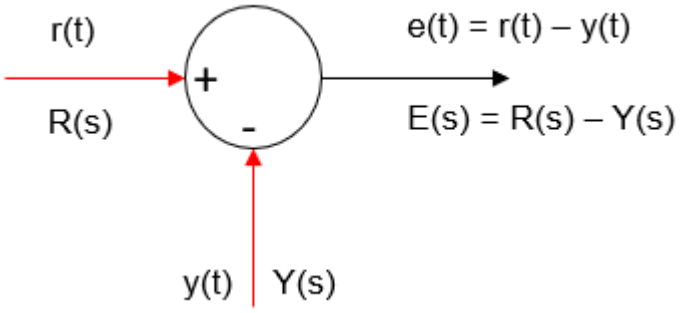
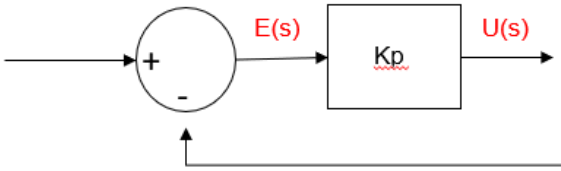
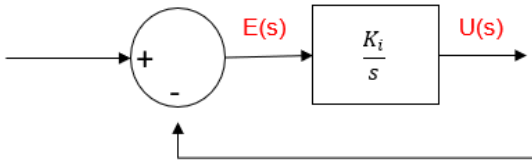


Figura 3-9: Dispositivos Detectores

Kuo, B. C. (1996). *SISTEMA DE CONTROL AUTOMATICO, 7ma Edición.* Mexico: PRNTICE HALL.

3.5.3 Tipos de Controles

De acuerdo a su respectiva acción de control, los controladores automáticos más utilizados son los presentados en la tabla 3-1.

CONTROL.	DIAGRAMA	FUNCIÓN DE TRANSFERENCIA
P		$\frac{U(s)}{E(s)} = Kp$
I		$\frac{U(s)}{E(s)} = \frac{K_i}{s}$

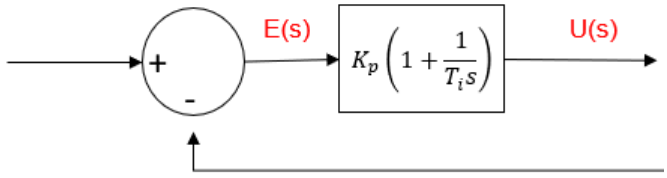
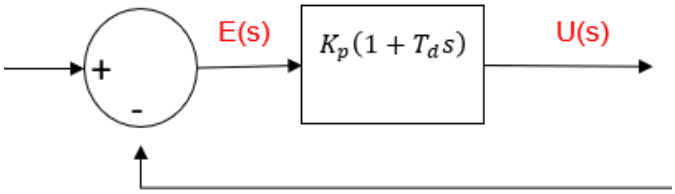
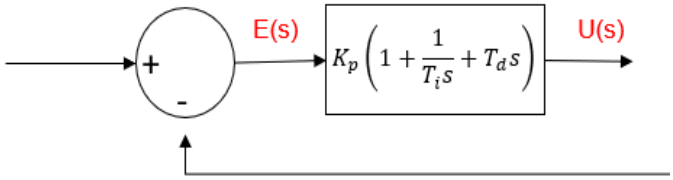
PI		$\frac{U(s)}{E(s)} = K_p\left(1+\frac{1}{T_i s}\right)$
PD		$\frac{U(s)}{E(s)} = K_p(1+T_d s)$
PID		$\frac{U(s)}{E(s)} = K_p\left(1+\frac{1}{T_i s} + T_d s\right)$

Tabla 3-1: Tipos de controladores Automáticos

Fuente Propia

3.6 **Métodos de Sintonización.**

En la sintonización es una herramienta, posible de aplicar cuando la planta es tan complicada, que no es fácil obtener su modelo matemático y tampoco es posible un modelo analítico para diseñar un controlador PID.

En la actualidad existen diferentes métodos de sintonización, que se utilizan para sistemas con lazo abierto y sistemas con lazo cerrado. Para nuestro caso en particular nos centraremos en 3 métodos de sintonización los cuales uno ellos funcionan para ambos sistemas y los otros solo funcionan dependiendo de la configuración del sistema.

3.6.1 Método Ziegler-Nichols

Desarrollado por los Ingenieros Jhon Ziegler y e Nathaniel Nichols aproximadamente en los años 1942 y desde entonces es uno de los métodos más utilizados y difundidos. Su característica principal, es determinar los valores de ganancia proporcional (Kp), el tiempo integral (Ti) y el tiempo derivativo (Td), basándose en la respuesta transitoria de una planta dada.

Existe un método para sistema de lazo abierto y un método para sistemas de lazo cerrado (sistemas mayores de segundo orden) los métodos de sintonización de Ziegler y Nichols. En ambos métodos el objetivo es conseguir que el valor máximo del sobre impulso sea menor al 25% para una entrada escalón, Difícilmente se cumpla esta condición en plantas complejas, pero asegura la estabilidad en un sistema de lazo cerrado.

Capítulo_4

El desarrollo metodológico, se basa en el siguiente diagrama de flujo descrito en la figura 4-1, el cual se complementará de acuerdo al avance del mismo.

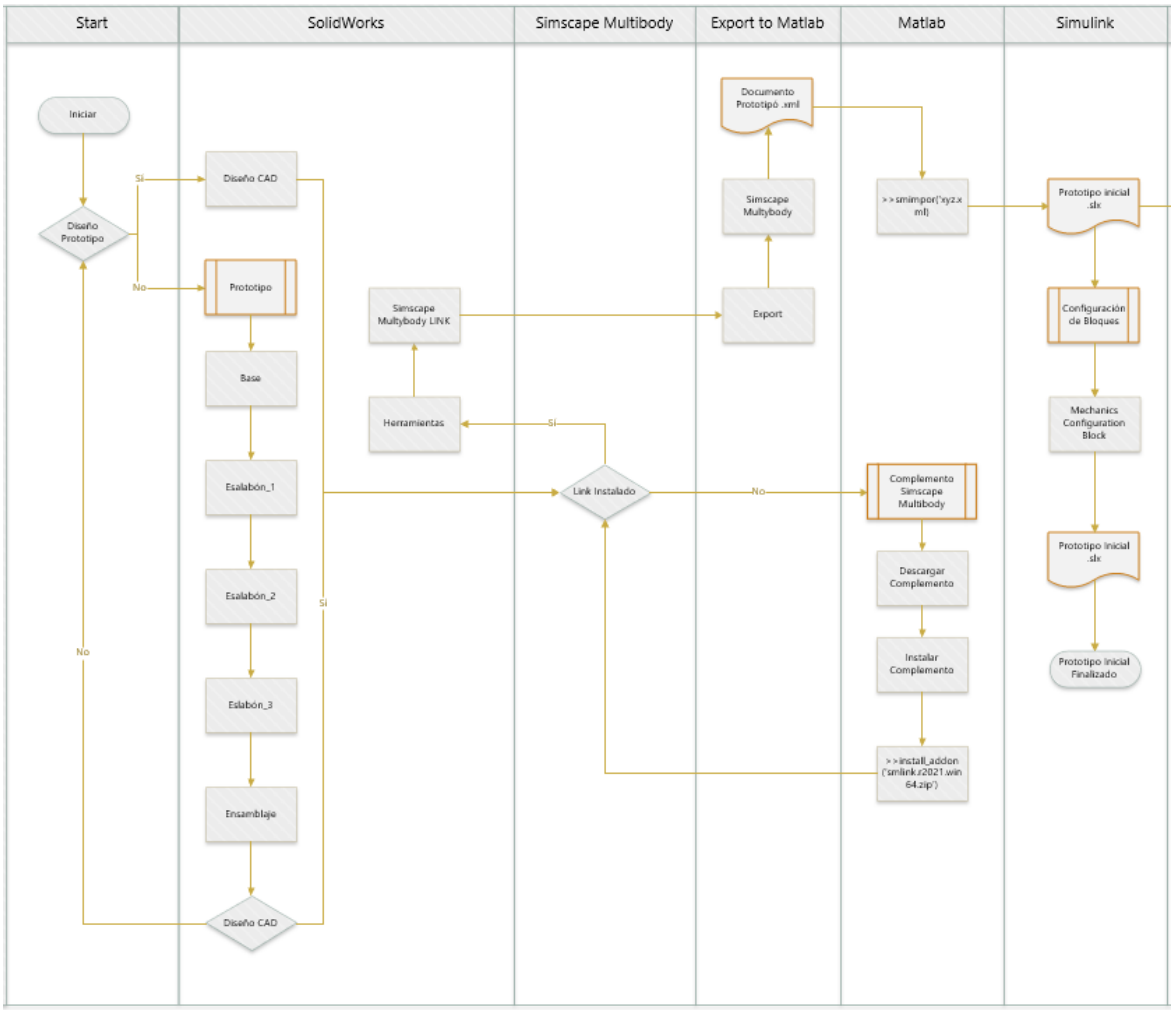


Figura 4-1: Diagrama de Flujo desde inicio hasta el Prototipo Inicial

Fuente Propia

4.1 Software SolidWorks.

SOLIDWORKS es un software de diseño CAD 3D (diseño asistido por computadora) para modelar piezas y ensamblajes en 3D y planos en 2D. El software ofrece un abanico de soluciones para cubrir los aspectos implicados en el proceso de desarrollo del producto. Sus productos ofrecen la posibilidad de crear, diseñar, simular, fabricar, publicar y gestionar los datos del proceso de diseño (SOLIDWORKS, 2015).

SOLIDWORKS ofrece soluciones intuitivas para cada fase de diseño. Cuenta con un completo conjunto de herramientas que le ayudan a ser más eficaz y productivo en el desarrollo de sus productos en todos los pasos del proceso de diseño. La sencillez que es parte de su propuesta de valor, es decisiva para lograr el éxito de muchos clientes.

- La solución de SOLIDWORKS incluye cinco líneas de productos diferentes:
- Herramientas de diseño para crear modelos y ensamblajes
- Herramientas de diseño para la fabricación mecánica, que automatiza documentos de inspección y genera documentación sin planos 2D.
- Herramientas de simulación para evaluar el diseño y garantizar que es el mejor posible
- Herramientas que evalúan el impacto medioambiental del diseño durante su ciclo de vida.
- Herramientas que reutilizan los datos de CAD en 3D para simplificar el modo en que las empresas crean, conservan y utilizan contenidos para la comunicación técnica.
- Finalmente, todas estas herramientas están respaldadas por SolidWorks PDM para gestionar y controlar de forma segura los datos mediante una única fuente de datos reales de sus diseños y SOLIDWORKS Manage, una herramienta que gestiona los procesos y proyectos implicados en todo el desarrollo del producto y está conectado al proceso de diseño.

4.2 Diseño Prototipo.

Para determinar el prototipo de brazo robótico a realizar, es muy importante identificar los grados de libertad del robot a utilizar. Debido a que estos nos determinan movimientos finales del efector y de cada uno de los eslabones.

Las articulaciones del brazo, puede coincidir con los grados de libertad, pero también varían de acuerdo a las necesidades del diseñador, Teniendo en cuenta las funciones, actividades y movimientos que realiza el brazo robótico. Para este proyecto se diseña un prototipo de brazo robótico de tres grados de libertad, el cual le realizaremos un control de posición y de movimientos a cada una de las articulaciones (Hombro, Codo, Muñeca).

4.2.1 Base.

Para el diseño de la base, se escoge el sistema de medidas, para este caso se tomó el sistema en milímetros.

Paso_1

- Se referencia el plano a trabajar.
- Se realiza una circunferencia de 100 mm
- Se utiliza la función extraer o saliente, para crear un cilindro.

Paso_2

- Se Escoge la base superior del cilindro.
- Se aplica la función redondeo de 10 mm

Paso_3

- Se utiliza la función rectangular con centro, para crear un rectángulo.
- Se utiliza la función extraer o saliente, para crear una plancha.
- Se utiliza crea un nuevo croquis con la base del cilindro y la plancha, para poder eliminar las aristas innecesario.
- Se aplica la función redondeo, para mejorar la estética.
- Se funciona el resultado, dejando una base con una salida.

Paso_4

- Se crea la pared de la base, para simular la ubicación de pernos.
- Se utiliza la función extraer o saliente para crear una pared lateral.
- Se utiliza la función simetría para duplicar la pared lateral en la base de la plancha anterior.

Paso_5

- Se utiliza la función Matriz, para replicar una acción escogida.
- Una vez escogida la base principal del diseño, en donde se replicarán las opciones seleccionadas y la cantidad de copias

Paso_6

- Se realiza la conexión de la base con el eslabón 1.
- Se realiza una circunferencia y se utiliza la función extraer o saliente, para dejar un cilindro.

Todos los pasos anteriormente descritos se observan en la figura 4-2, que se muestra a continuación.

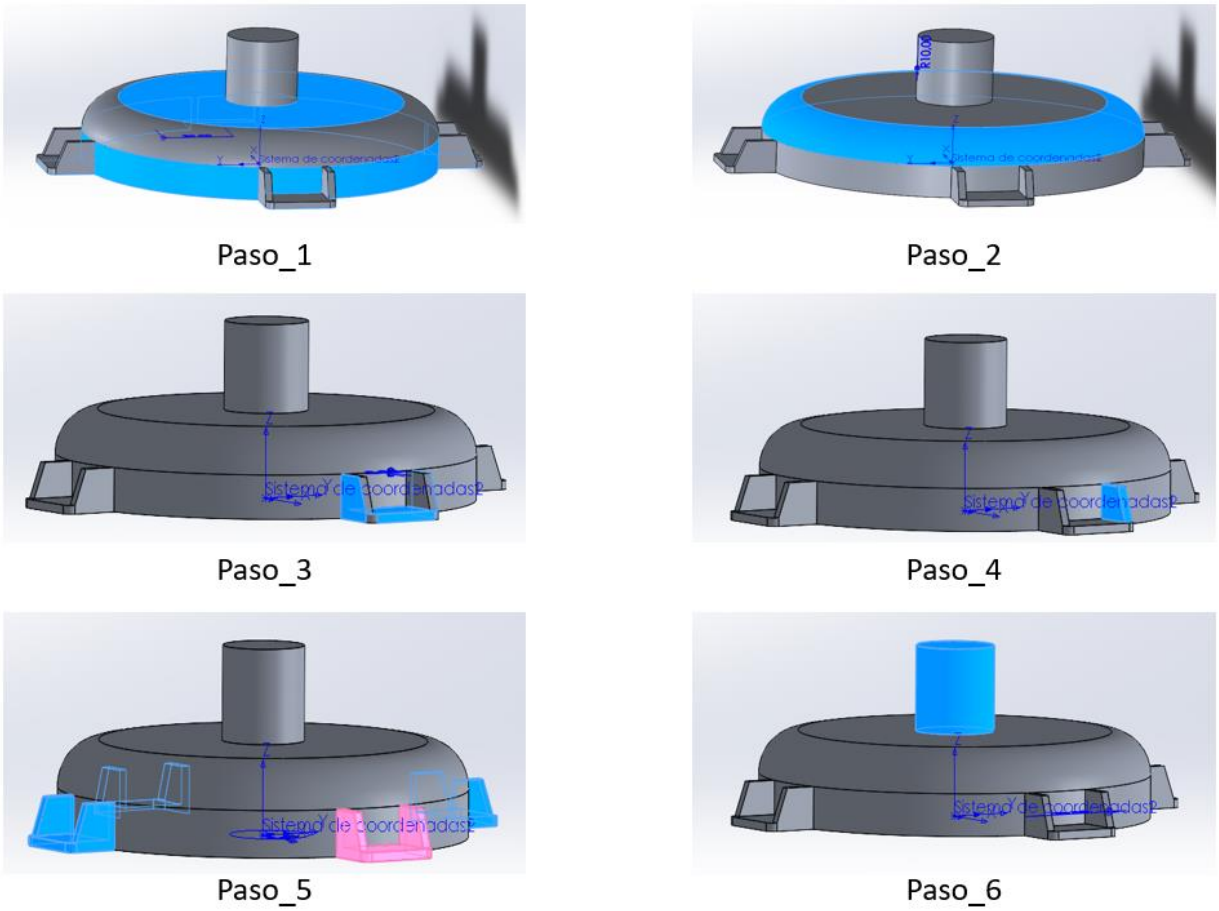


Figura 4-2: Diseño Base

Fuente Propia.

4.2.2 Eslabón_1.

Para el diseño del eslabón 1, se escoge el sistema de medidas, para este caso se tomó el sistema en milímetros.

Paso_1

- Se referencia el plano a trabajar.
- Se realiza una circunferencia.
- Se utiliza la función extraer o saliente, para crear un cilindro.
- Se utiliza la función redondeo, en la parte inferior y superior del cilindro creado.

Paso_2

- Se realiza la conexión del eslabón 1con la base.
- Se realiza una circunferencia y se utiliza la función corte o extruir, para dejar un orificio en forma cilindro.

Paso_3

- Se utiliza el plano alzado, para poder generar la parte superior del eslabón 1.
- Se diseña con un círculo, un rectángulo y unión entre estas formas geométricas. Para poder generar un nuevo croquis o una nueva figura completa.

Paso_4

- Se selecciona la nueva forma generada en el paso 3.
- Se utiliza la función extraer o saliente para generar un objeto solido de tres dimensiones

Paso_5

- Se realiza la conexión del eslabón 1 con el eslabón 2.
- Se realiza una circunferencia y se utiliza la función extraer o saliente, para dejar un cilindro.

Todos los pasos anteriormente descritos se observan en la figura 4-3, que se muestra a continuación.

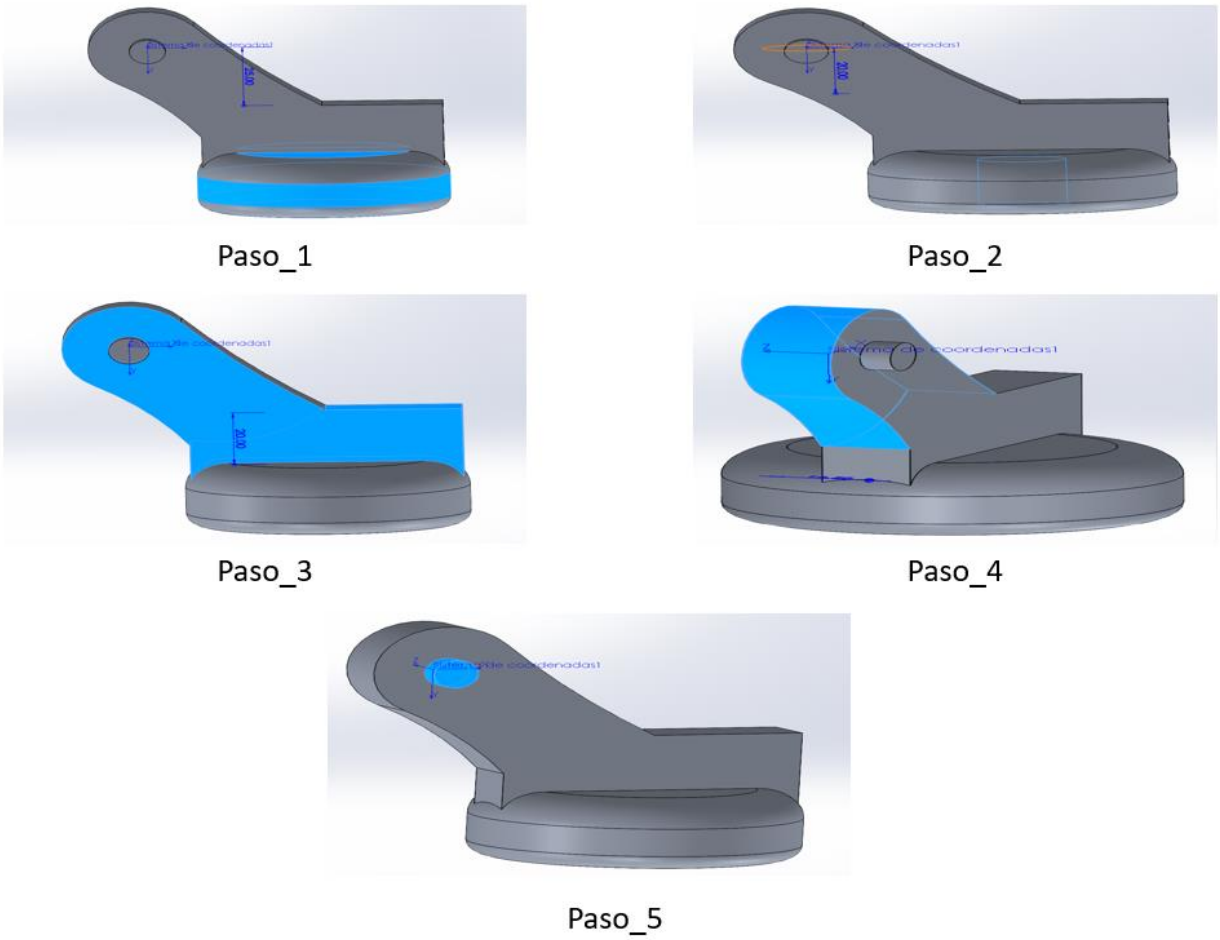


Figura 4-3**:** Diseño Eslabón 1

Fuente Propia.

4.2.3 Eslabón_2.

Para el diseño del eslabón 2, se escoge el sistema de medidas, para este caso se tomó el sistema en milímetros.

Paso_1

- Se referencia el plano a trabajar.
- Se genera un objeto nuevo, a partir de dos figuras geométricas (Circulo y Rectángulo)
- Se utiliza la función extraer o saliente para generar un objeto solido de tres dimensiones

- Se utiliza la función redondeo, en la parte inferior y superior del objeto creado.

Paso_2

- Se realiza una circunferencia en la parte superior del objeto creado en el paso 1.
- Se utiliza la función extraer o saliente, para crear un cilindro.

Paso_3

- Se escoge un nuevo plano, con respecto al cilindro creado en el punto 2.
- El plano escogido se referencia a una distancia de la cara del cilindro, para poder generar un nuevo cilindro, el cual se le aplican diferentes técnicas y se genera un nuevo croquis.
- El croquis escogido se basa en una serie de uniones y de figuras geométricas, las cuales se seleccionan y se eliminan las aristas innecesarias.
- Se utiliza la función simetría del nuevo croquis creado, para que nos replique el objeto en la parte inferior del objeto principal.

Paso_4

- Se utiliza la función ranura recta, en la mitad del objeto creado en el paso 1.
- Se utiliza la función cortar o extruir, para generar la forma de orificio.
- Se utiliza la función simetría de planos, para generar el lado opuesto del objeto creado en el paso 1.

Paso_5

- Se realiza la conexión del eslabón 2 con el eslabón 1.
- Se realiza una circunferencia y se utiliza la función corte o extruir, para dejar un orificio en forma cilindro.

Paso_6

- Se realiza la conexión del eslabón 2 con el eslabón 3.
- Se realiza una circunferencia y se utiliza la función extraer o saliente, para dejar un cilindro.

Todos los pasos anteriormente descritos se observan en la figura 4-4, que se muestra a continuación.

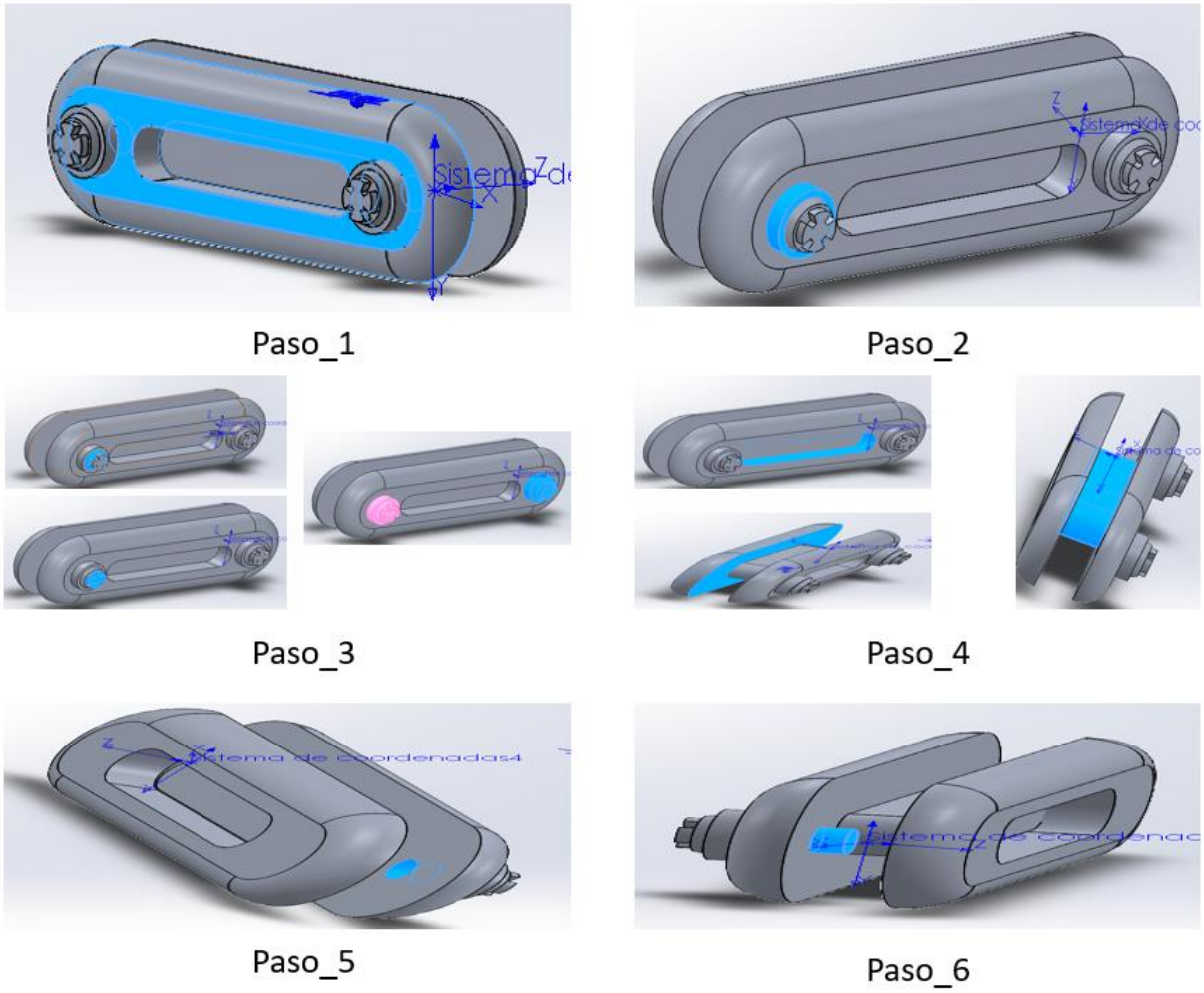


Figura 4-4: Diseño Eslabón 2

Fuente Propia.

4.2.4 Eslabón_3.

Para el diseño del eslabón 3, se escoge el sistema de medidas, para este caso se tomó el sistema en milímetros.

Paso_1

- Se realiza una circunferencia.
- Se utiliza la función extraer o saliente, para crear un cilindro.
- Se utiliza la función redondeo, en la parte inferior y superior del cilindro creado.

Paso_2

- Se realiza una circunferencia en la parte inferior del objeto creado en el punto 1.
- Se utiliza la función extraer o saliente, para crear un cilindro.
- Se utiliza la función redondeo, en la parte inferior y superior del cilindro creado.

Paso_3

- Se escoge un nuevo plano, con respecto al cilindro creado en el punto 2.
- El plano escogido se referencia a una distancia de la cara del cilindro, para poder generar un nuevo cilindro.

Paso_4

- Se utiliza la función ranura recta, en la mitad del objeto creado en el paso 1.
- Se utiliza la función cortar o extruir, para generar la forma de orificio.
- Se utiliza la función matriz de base, para generar diferentes orificios.

Paso_5

- Se realiza una circunferencia en la parte superior del objeto creado en el punto 1.
- Se utiliza la función extraer o saliente, para crear un cilindro.

Paso_6

- Se realiza la conexión del eslabón 2 con las pinzas u otro objeto que se pueda conectar.
- Se realiza una circunferencia y se utiliza la función extraer o saliente, para dejar un cilindro.
- Se realiza la conexión del eslabón 3 con el eslabón 2.
- Se realiza una circunferencia y se utiliza la función corte o extruir, para dejar un orificio en forma cilindro.

Todos los pasos anteriormente descritos se observan en la figura 4-5, que se muestra a continuación.

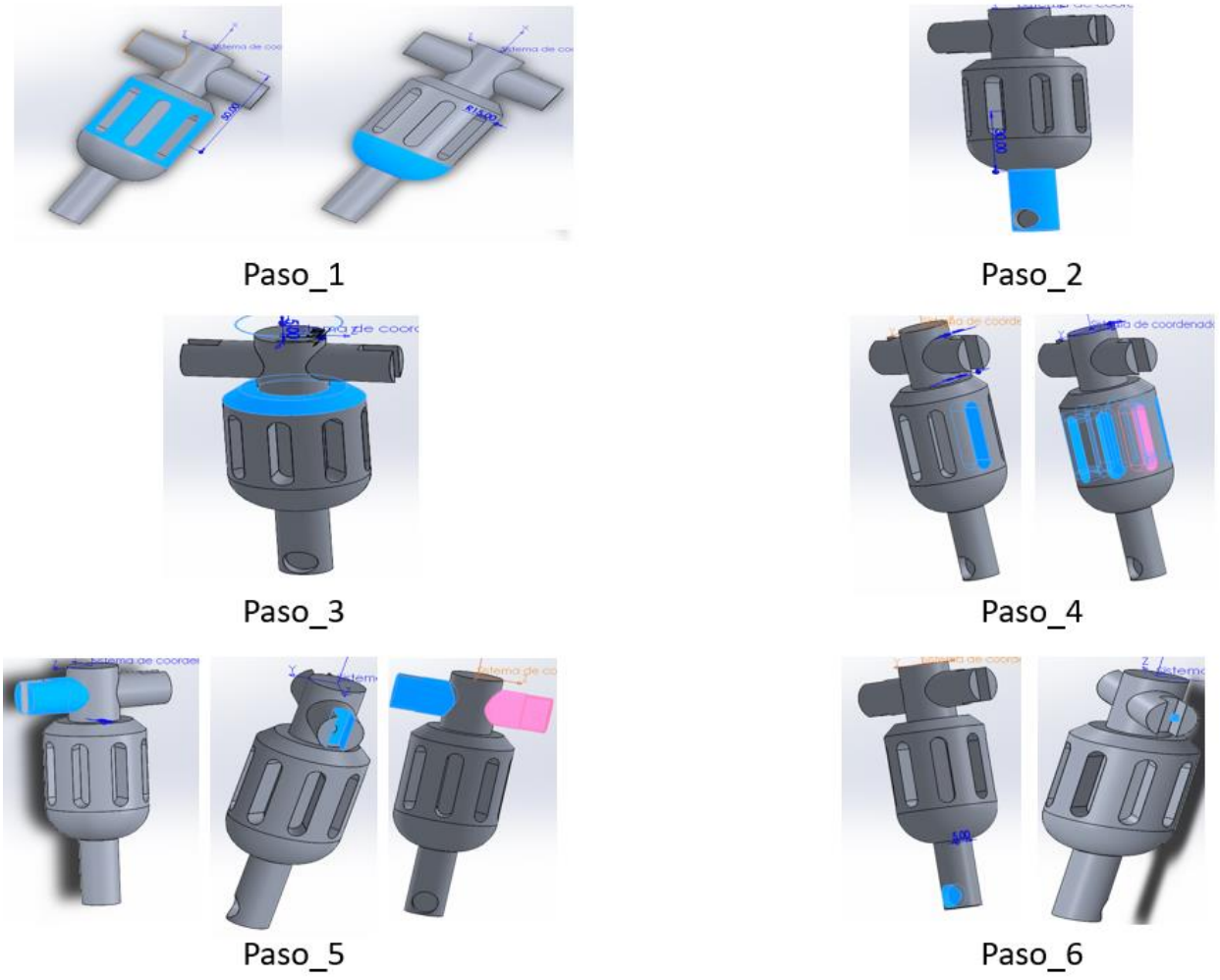


Figura 4-5**:** Diseño Eslabón 3

Fuente Propia.

4.2.5 Ensamblaje.

Para el ensamblaje, se escoge el sistema de medidas, para este caso se tomó el sistema en milímetros y se despliegan en la zona de graficas los diseños creados anteriormente (Base, Eslabón 1, Eslabón 2 y Eslabón 3)

Paso_1

- En la barra de herramientas del sistema de información, se escoge la función relación de posición.

- Se seleccionan las piezas correspondientes y necesarias de la base y del eslabón 1 para realizar el movimiento o la articulación del hombro
- Una vez escogidas las piezas se aplica las relaciones de posiciones concéntrica y paralela.

Paso_2

- En la barra de herramientas del sistema de información, se escoge la función relación de posición.
- Se seleccionan las piezas correspondientes y necesarias del eslabón 1 y del eslabón 2 para realizar el movimiento o la articulación del codo.
- Una vez escogidas las piezas se aplica las relaciones de posiciones concéntrica y paralela.

Paso_3

- En la barra de herramientas del sistema de información, se escoge la función relación de posición.
- Se seleccionan las piezas correspondientes y necesarias del eslabón 2 y del eslabón 3 para realizar el movimiento o la articulación de la muñeca. (no en sentido rotacional)
- Una vez escogidas las piezas se aplica las relaciones de posiciones concéntrica y paralela.

Todos los pasos anteriormente descritos se observan en la figura 4-6, que se muestra a continuación.

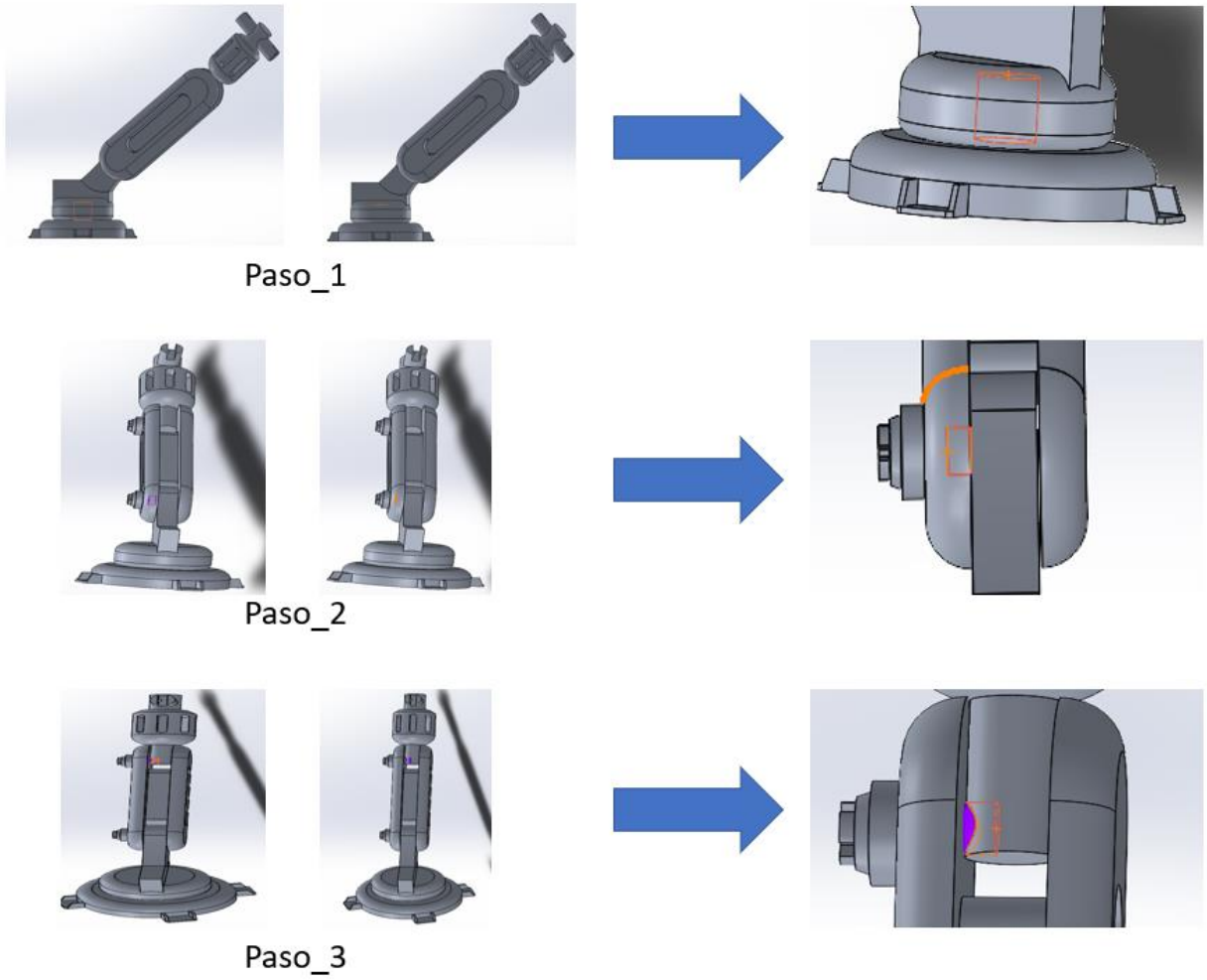


Figura 4-6: Diseño Ensamblado

Fuente Propia.

4.3 **Software Matlab**

Matlab cuya abreviatura es (MATrix LABoratory), como un sistema de cómputo numérico, que maneja un lenguaje propio y de alto nivel (lenguaje M) para cálculos científicos y de ingeniería, sirve para analizar, implementar y diseñar sistemas y productos, que están presentes en sistemas de seguridad, naves espaciales, dispositivos de monitorización de salud y también es utilizado para el procesamiento de señales, imágenes, comunicaciones, robótica, entre otros campos (MathWorks Inc, 2021).

Está disponible para plataformas como Unix, Windows, Mac, Linux.

Requisitos del sistema:

- Windows 7 en adelante.
- Procesador Intel o AMD
- Disco duro de 2GB para el programa y 6GB mínimo para instalación típica.
- RAM 1GB mínimo, 4GB recomendado.
- Tarjeta gráfica, soporte para OpenGL 3.3 recomendado con 1GB en GPU.

4.3.1 SimScape Multybodi.

Add Ons o complemento del Software Matlab, que proporciona un entorno de simulación multicuerpo para sistemas 3D mecánicos, tales como robots, suspensiones de vehículos, equipos de construcción, y el tren de aterrizaje del avión. Permite modelar el sistema usando bloques que representan los órganos, articulaciones, restricciones y elementos de fuerza y, tras ello, SimMechanics formula y resuelve las ecuaciones de movimiento para el sistema mecánico completo. Modelos de sistemas CAD, incluyendo la masa, la inercia, las articulaciones, limitación, y la geometría 3D, se pueden importar en SimMechanics. Una animación 3D generada automáticamente le permite visualizar la dinámica del sistema.
En el ámbito de este proyecto, se utilizará SimMechanic o Simscape Multybody para importar el sistema generado en SolidWorks creando el conjunto de bloques que representen el modelo dinámico.

- SimMechanics: Matlab de modelos Inferiores e igual 2019b
- Simscape Multybody: Matlab de modelos Superiores e igual 2020ª

La forma de instalar este complemento, en caso de no tener instalado y descargado es el siguiente. Abrir el Software Matlab, ubicar en las opciones de la barra de herramientas “Home”, el icono de referencia a los complementos o “*Add-Ons*”, desplegar el listado del icono e ingresar a obtener complementos o “*Get Add-Ons*”, tal como se muestra en la figura 4-7.

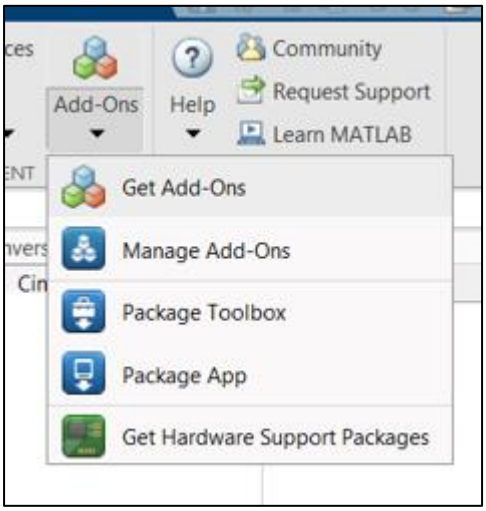


Figura 4-7**:** Descargar Simscape Multybody

Fuente Propia.

Se desplegará una ventana de exploración de complementos o "*Add-Ons Explorer*" y en el buscador digitamos el complemento a descargar, para este caso en particular debemos digitar simscape multybody.

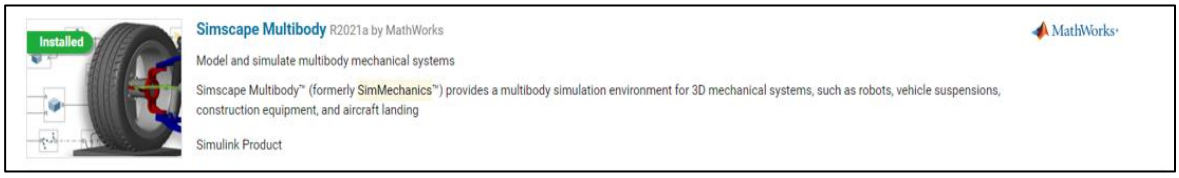


Figura 4-8: Explorador de Complementos

Fuente Propia.

Debe aparecer el complemento en la parte superior o de primero en la lista de complementos, se procese a descargar el complemento evidenciado en la figura 4-8. Tener muy presente la carpeta donde se guardará el complemento, por lo general se recomienda que se guarde en donde se encuentran el instalador del software.

Al finalizar la descarga, validamos en la carpeta donde se solicitó descargar y se encuentran dos archivos, fichero .m (install_addon.m) y un fichero comprimido (smlink.r2021.win64.zip), que en realidad es un fichero de instalación.

En el Comand Windows de Matlab se digita el siguiente comando, teniendo en cuenta que la carpeta donde se encuentra el archivo .zip y el archivo .m deben estar desplegada en el Current Folder de Matlab.

>> install_addon (‘smlink.r2021.win64.zip’)

4.3.2 Export Cad a Matlab.

Una vez finalizado el ensamble de las piezas diseñadas, que se abordó en el capítulo IV del presente libro, se exporta de nuestro diseño a Matlab, la herramienta en la que se llevara a cabo el entorno de simulación mediante el Tollbox Simulink, en donde ingresaran y comenzaran a funcionar los bloques de SimMechanics (sistema mecánico multicuerpo), en el cual se pueden desarrollar trabajos y simulaciones deseadas o específicas, en nuestro caso, se diseñara un sistema de control PID para un brazo robótico.

Al momento de realizar la exportación de cualquier diseño del Software SolidWorks al Software Matlab, se debe verificar que si cuenten con el complemento SimMechanics o Simscape Multybody link. La prueba rápida para identificar si contamos con el complemento instalado es ingresar al Software SolidWorks, ubicar en la barra de herramientas o control de mandos “*Herramientas*” y verificar que en el listado desplegable cuente con la opción Simscape Multybody Link, ver figura 4-9.

Si no se cuenta con el complemento instalado, al que se hace mención en este libro, exactamente en el capítulo IV. Se especifica como se puede descargar e instalar este complemento.

Una vez instalado el complemento, seguimos la siguiente ruta. Barra de herramientas o control de mandos, ubicamos “*Herramientas*”, en el listado desplegable escogemos la opción “*Simscape Multybody Link*”, nuevamente nos muestra otro listado desplegable, donde debemos escoger la opción “*Export*” y por último nos arroja una única opción “*Simscape Multybodi*”, como se observa a continuación.

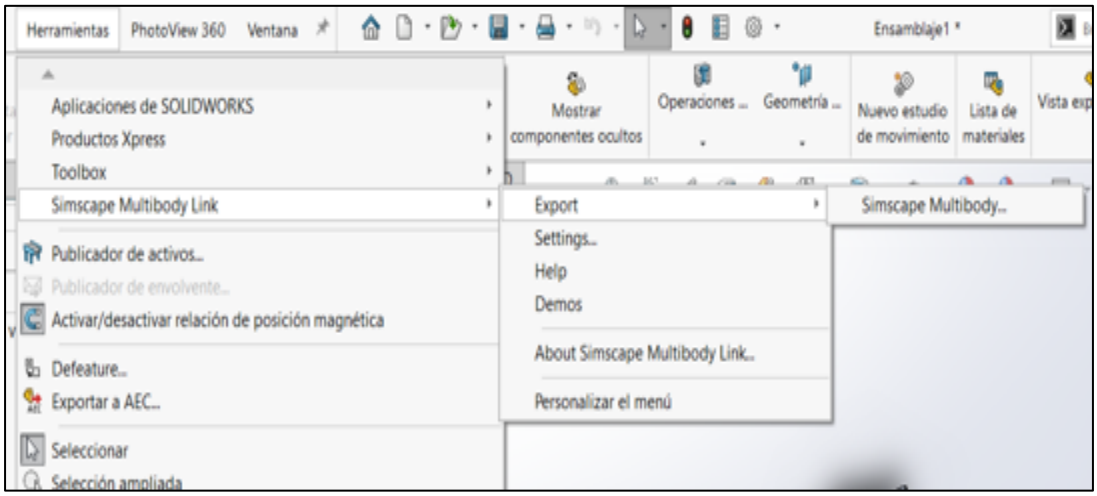


Figura 4-9**:** Exportar Diseño

Fuente Propia.

Culminada la acción anterior, el sistema activa una ventana emergente para guardar el archivo a exportar con la extensión .xml. Dicho archivo contiene toda la información necesaria para poder exportar el diseño de un software a otro, cabe resaltar que se tiene que tener presente la carpeta donde se archivara dicho fichero. Ver figura 4-10.

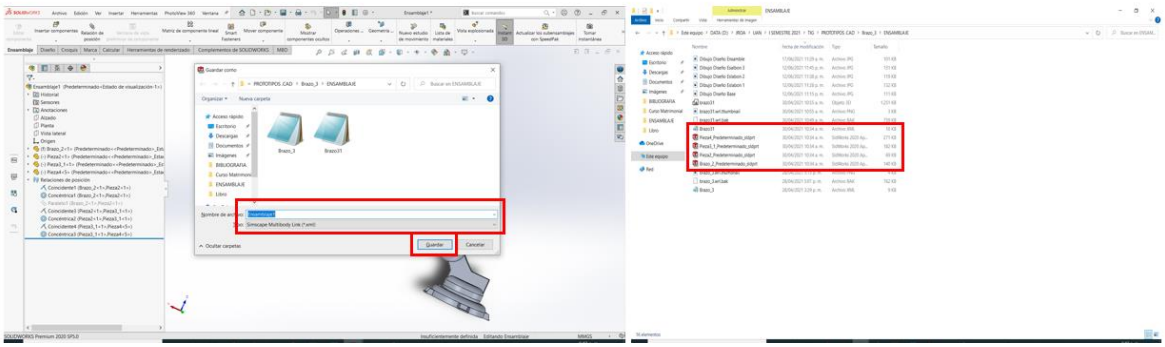


Figura 4-10**:** Guardar archivo .xml

Fuente Propia.

Es importe ubicar la carpeta donde se guardó el archivo con la extensión .xml, porque en esa misma carpeta se guardan de forma automática cada uno de los elementos diseñados y que se utilizaron en el ensamblado del mismo.

Para finalizar la exportación del diseño, se debe tener desplegada la carpeta donde se guardó la extensión .xml en el Current Folder de Matlab, para este caso en particular el

archivo se llama Brazo_31.xml y por lo tanto digito el comando “smimport” y el nombre del archivo escogido, en el Comand Wimdow de Matlab

>> smimport (‘xxxxx.xml’)

>> smimport (‘Brazo_31.xml)

Cuando se ejecuta el comando, automáticamente se abre una ventana emergente del simulador Simulink, con nuestro modelo exportado, mediante un diagrama de bloques de la aplicación Simscape Multybodi, como se observa en la figura 4-11.

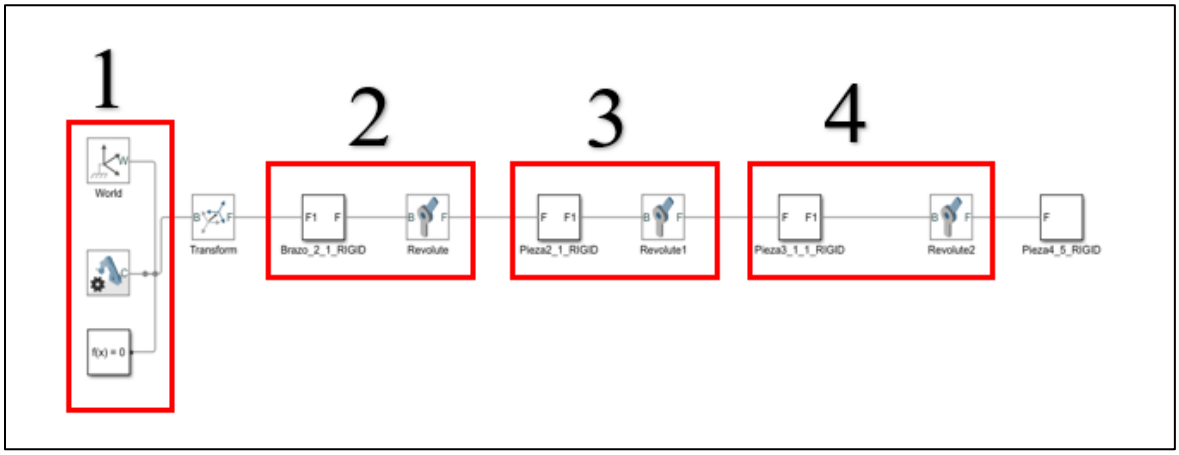
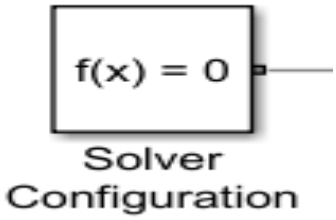


Figura 4-11**:** Simulink – Diagrama de Bloques Simscape Multybodi

Fuente Propia.

1er Grupo de bloques

También denominados bloques de referencias Proporciona acceso al marco de coordenadas Mundo o Tierra, un único marco de coordenadas inmóvil, ortogonal y derecho predefinido en cualquier modelo mecánico. El marco mundial es la base de todas las redes de marcos en un modelo mecánico (MathWorks Inc, 2021).



Establece los parámetros mecánicos y de simulación que se aplican a toda una máquina, la máquina de destino a la que está conectado el bloque. En la sección Propiedades, puede especificar la gravedad uniforme para todo el mecanismo (MathWorks Inc, 2021).

A este bloque se puede modificar las propiedades iniciales, para nuestro caso en particular realizamos un cambio en el vector de la gravedad [*X Y Z*]. El cual por defecto se encuentra situado en eje *Y* [0 -9.80665 0]; nuestro diseño la gravedad debe estar en eje *Z* [0 0 -9.80665]. ver figura 4-12.

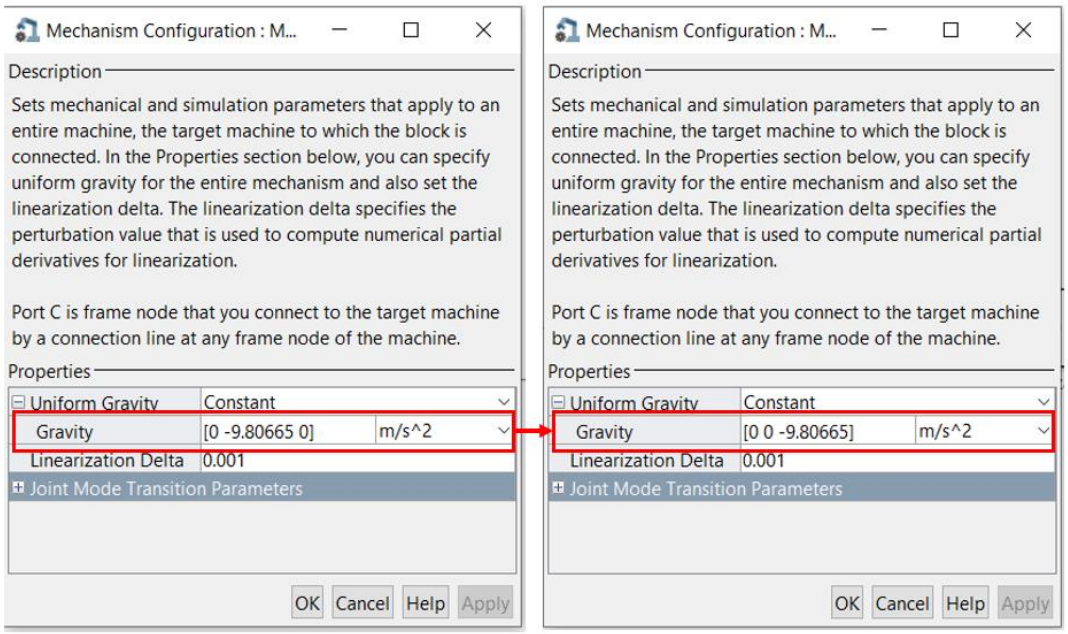


Figura 4-12**:** Simulink – Diagrama de Bloques Simscape Multybodi

Fuente Propia.

Define la configuración del solucionador que se utilizará para la simulación.

2do al 4to Grupo de bloques

Son los bloques que simulan los eslabones y las articulaciones

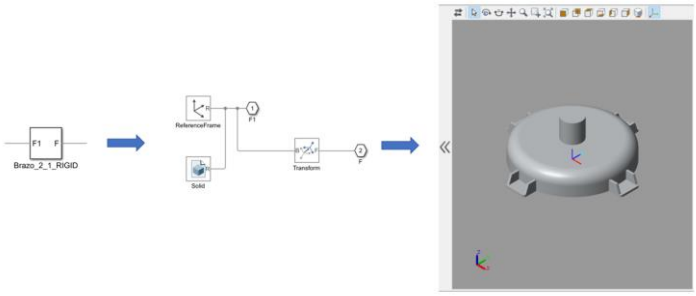
Representa una articulación revoluta que actúa entre dos cuadros. Esta junta tiene un grado de libertad rotacional.

El bloque revolute que simula una articulación, contiene un "*nodo de propiedades*" el cual se puede modificar algunas propiedades de la articulación, para mejorar la simulación. Tales como:

- Objetivo de estado, Especificación de Posición o de velocidad.
- Mecanismos internos como la fricción.
- Limites o fronteras.
- Actuadores de torque o actuadores de movimientos, los cuales funcionan como datos de entrada.
- Sensores de posición, de velocidad, aceleración, de torque, de torque máximo y torque mínimo, los cuales funcionan como datos de salidas.

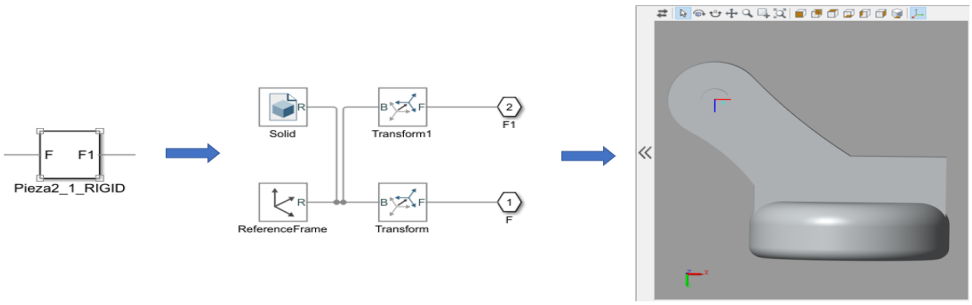
Los bloques que representan los eslabones ver figura 4-13, contienen un sistema interno de bloques. En donde se caracterizan el bloque de referencia mundo y el bloque del solido a simular.

Base



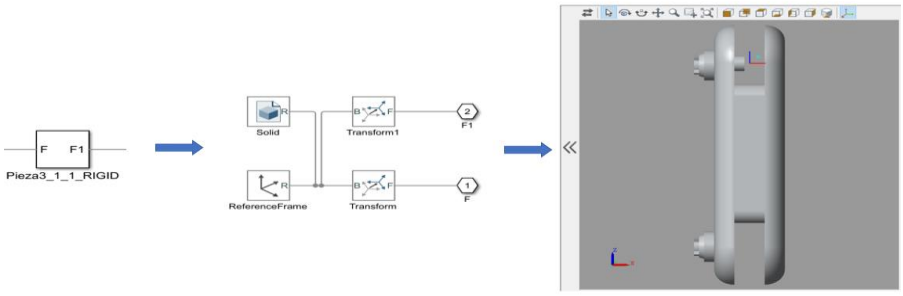
(a)

Eslabón 1



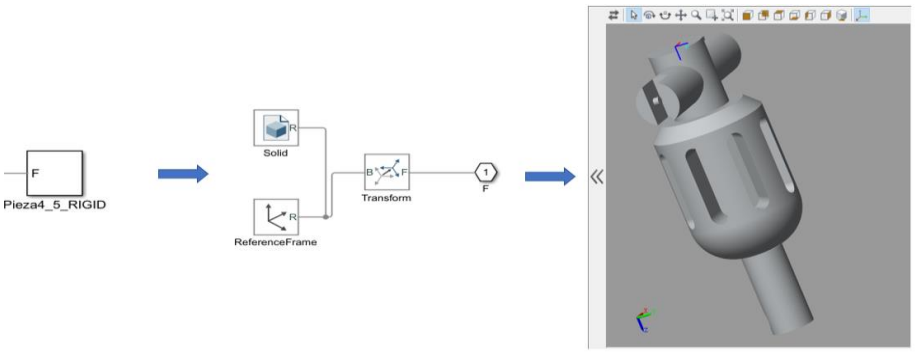
(b)

Eslabón 2



(c)

Eslabón 3



(d)

Figura 4-13**: (a), (b), (c) y (d)**. Estructura interna del bloque pieza

Fuente Propia.

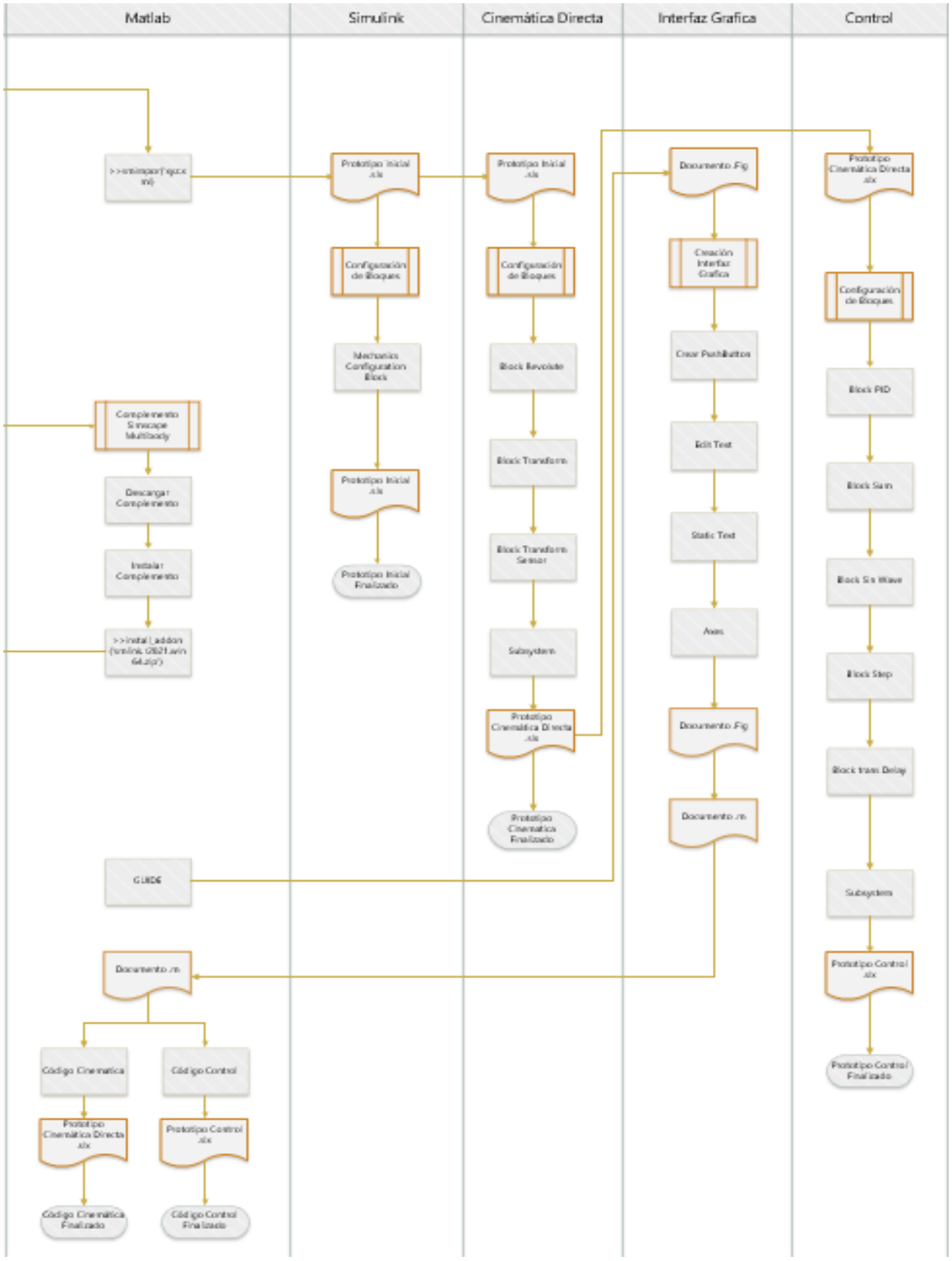


Figura 4-14: Diagrama de Flujo prototipo inicial y los diseños

Fuente Propia

4.4 **Cinemática Directa**

Se utilizará el método de matrices de transformación homogénea para un robot de 3 grados de libertad, conformado por 4 eslabones y 3 articulaciones, en donde a cada eslabón se le asigna un sistema de coordenadas, el cual por medio de las matrices de transformación pueden representar las traslaciones y las rotaciones relativas entre los eslabones. De acuerdo a lo anterior es posible utilizar la representación de Denavit-Hartenberg (depende de la geometría y construcción del braso), que permite calcular de forma matricial, estableciendo un sistema de coordenadas a cada eslabón ligado a una cadena articulada, obteniendo las ecuaciones cinemáticas.

Como se abordó en el inciso 3.2.1, lo teórico de la cinemática directa, del presente libro. Colocaremos en práctica lo abordado, el desarrollo de todos los pasos para utilizar la representación D-H en el anexo E, de donde extraemos la siguiente tabla 4-1 y ecuaciones (4-1), (4-2) y (4-3).

Articulación	$\boldsymbol{\theta}$	**d**	**a**	$\boldsymbol{\alpha}$
1	$\boldsymbol{\theta_1}$	0.01258	0.00125	90°
2	$\boldsymbol{\theta_2}$	0	0.0170	0
3	$\boldsymbol{\theta_3}$	0	0.011	180°

Tabla 4-1: Parámetros Denavit - Hartenberg.

Fuente Propia.

De acuerdo a la ecuación (3-2) Matriz D-H, se puede calcular las matrices de transformación de cada sistema o de cada articulación.

$$ {}^{0}A_1 = \begin{pmatrix} C_1 & 0 & S_1 & 0 \\ S_1 & 0 & -C_1 & 0 \\ 0 & 1 & 0 & 0.0125 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (4\text{-}1) $$

$$ {}^{1}A_2 = \begin{pmatrix} C_2 & -S_2 & 0 & (0.017 * C_2) \\ S_2 & C_2 & 0 & (0.017 * S_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (4\text{-}2) $$

$$ {}^{2}A_3 = \begin{pmatrix} C_3 & -S_3 & 0 & (0.011 * C_3) \\ S_3 & C_3 & 0 & (0.011 * S_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4\text{-}3) $$

La matriz de transformación T, la cual relaciona la posición y la orientación del extremo del robot con respecto a un sistema coordenado de referencia fija, que por lo general es la base. Esta relación se realiza a través de la matriz de transformaciones en donde se asocian cada uno de los sistemas coordenados de cada uno de los eslabones y articulaciones. Se desarrollo en un código el cual se encuentra en el Anexo A y de forma analítica en el anexo E, de donde se traen las siguientes ecuaciones de las posiciones del efector final, de acuerdo a la matriz de transformación homogénea.

Px = $C\theta_1$/800 + (0.017*$C\theta_1$*$C\theta_2$) + (0.011*$C\theta_1$*$C\theta_2$*$C\theta_3$) - (0.011*$C\theta_1$*$S\theta_2$*$S\theta_3$) (4-4)

Py = $S\theta_1$/800 + (0.017*$C\theta_2$*$S\theta_1$) + (0.011*$C\theta_2$*$C\theta_3$*$S\theta_1$) - (0.011*$S\theta_1$*$S\theta_2$*$S\theta_3$) (4-5)

Pz = (0.017*$S\theta_2$) + (0.011*$C\theta_2$*$S\theta_3$) + (0.011*$C\theta_3$*$S\theta_2$) + 629/50000 (4-6)

4.4.1 Área de trabajo

El área de trabajo del robot, es quien permite generar los movimientos necesarios para llegar a diferentes puntos dentro su área delimitada, para de esta forma llevar a cabo su respetiva tarea. Teniendo en cuenta, que el robot al estar en su espacio de trabajo delimitado, no puede acceder a diferentes puntos debido a su orientación, Ejemplo los puntos retirados y más cercano del área de trabajo, se debe llegar con una configuración específica del brazo.

Para determinar una buena área de trabajo se debe tener en cuenta los grados de libertad, ya que este determina la accesibilidad y la capacidad de orientación del mismo. Teniendo en cuenta la mecánica de del robot industrial (cumpla la característica de eslabones con articulaciones).

- Articulación 1: Tiene capacidad de moverse en los 360° del eje.

- Articulación 2: Tiene capacidad de moverse en un rango comprendido entre -35° y 195°.
- Articulación 3: Tiene la capacidad de moverse en un rango comprendido entre -90° y 90°

Las restricciones mencionadas, se realizaron en base a la estructura geométrica de cada eslabón, el cual permita el libre movimiento, si algún tipo de obstáculo por su misma geometría.

4.4.2 Simulink

Una vez obtenido, el modelo visto en la figura 4-11, al exportar nuestro diseño, se realizan las siguientes intervenciones para mejorar nuestra simulación. Agregando bloques como se observa en la figura 4-15

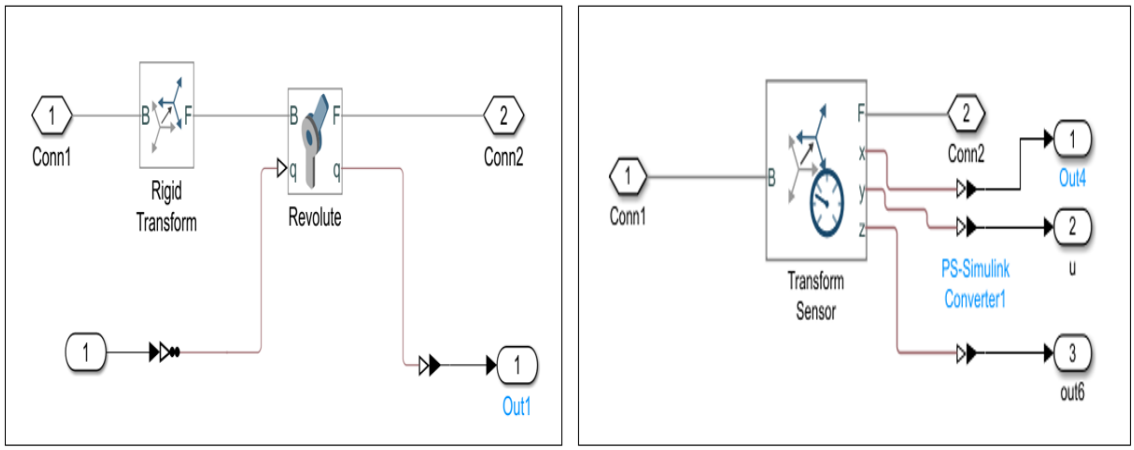


Figura 4-15**:** Diseño Simulink

Fuente Propia.

El bloque utilizado para simular las articulaciones, denominado revolute, presentado en la figura 4-16, se debe ingresar al "*nodo de propiedades*" y se activan las propiedades del actuador y de sensor.

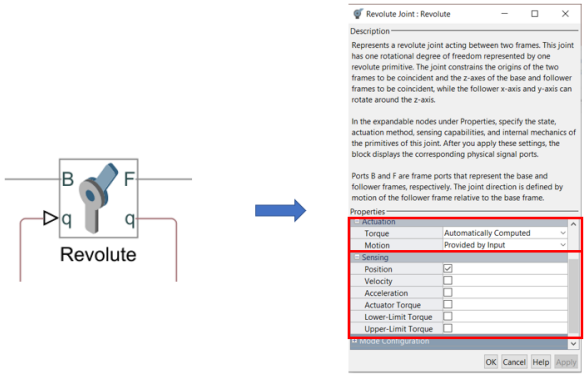


Figura 4-16: Configuración bloque Revolute

Fuente Propia.

Se agrega también un bloque llamado Transform, presentado en la figura 4-17, antes de cada articulación, el cual Define una transformación rígida 3D fija entre dos marcos. Dos componentes especifican independientemente las partes de traslación y rotación de la transformación. Se pueden combinar libremente diferentes traslaciones y rotaciones.

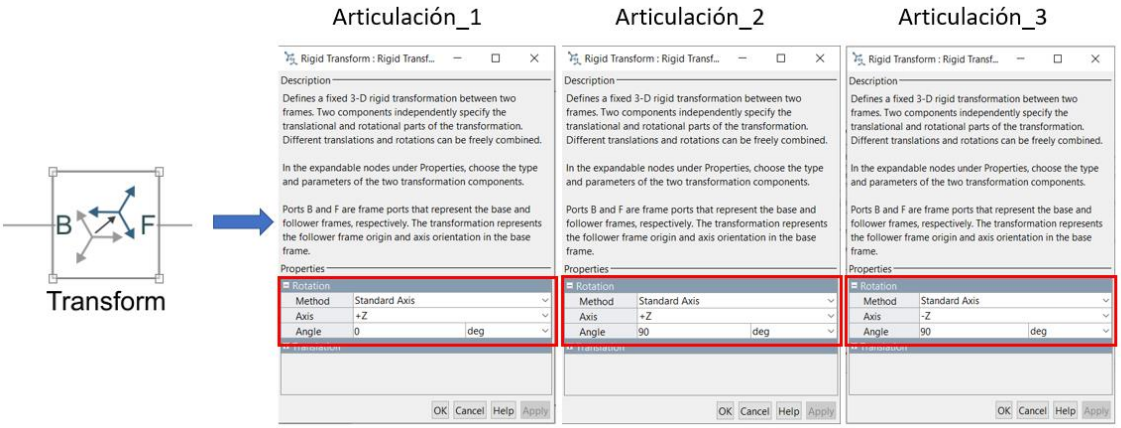


Figura 4-17: Configuración bloque Transform

Fuente Propia.

Por último, se agrega el bloque Transform Sensor, presentado en la figura 4-18, el cual nos permite medir la relación dependiente del tiempo entre dos fotogramas. Un sensor de

transformación detecta pasivamente esta transformación tridimensional que varía en el tiempo, y sus derivadas, entre los dos fotogramas.

Además, que, en el nodo desplegable de Propiedades, podemos seleccionar qué relaciones de rotación y traslación, incluidas las velocidades y aceleraciones, deseamos medir. Después de aplicar estos ajustes, el bloque muestra los puertos de señal física de salida correspondientes.

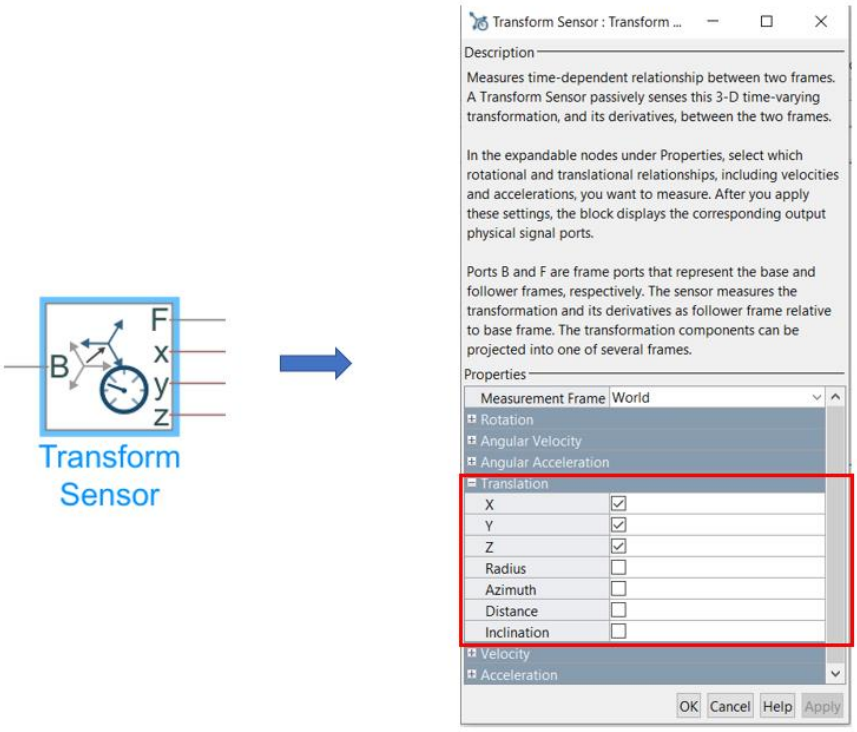
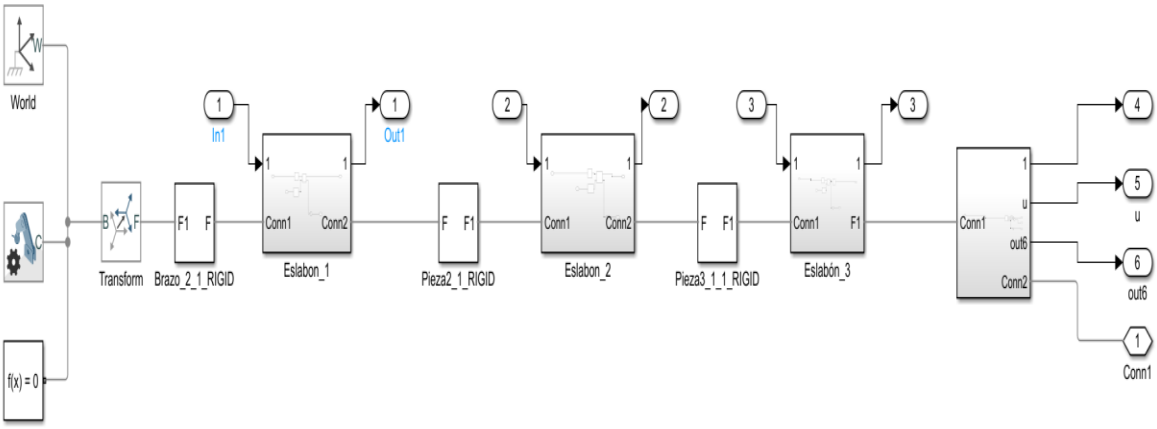


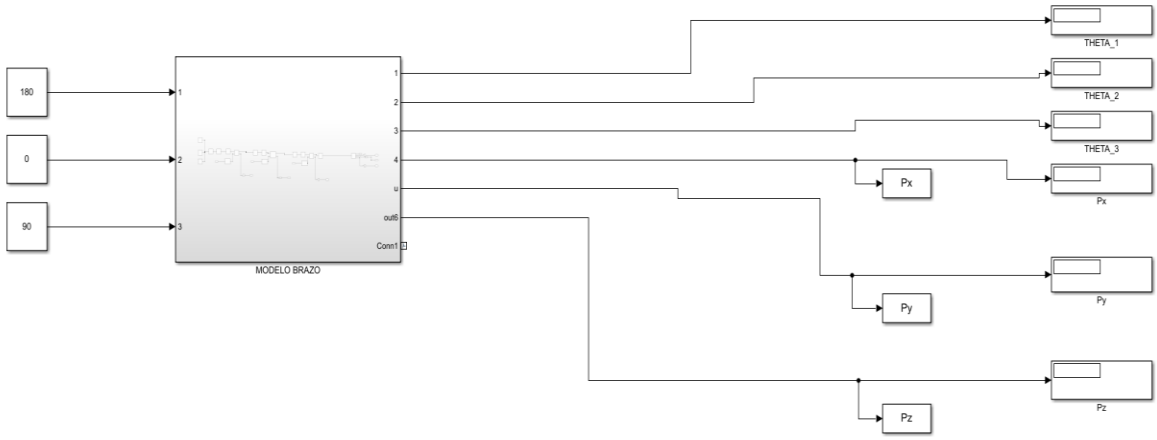
Figura 4-18**:** Configuración bloque Transform Sensor

Fuente Propia.

Una vez finalizado todos estos procesos mencionados, se crea un único bloque o subsistema, que comprende todos los bloques del modelo a simular, para minimizar el espacio y todo pertenezca un mismo bloque, como se observa en la figura 4-18 que representa el modelo de la siguiente manera.



(a) Subsistema del Brazo



(b) Complemento

Figura 4-19**:** Estructura final para la Cinemática Directa

Fuente Propia.

4.4.3 Guide – Interfaz Gráfica. (Anexo D)

GUI (también conocidas como interfaces gráficas de usuario o interfaces de usuario) permiten un control sencillo (con uso de ratón) de las aplicaciones de software, lo cual elimina la necesidad de aprender un lenguaje y escribir comandos a fin de ejecutar una aplicación.

Los programas autónomos de MATLAB con un frontal gráfico de usuario GUI que automatizan una tarea o un cálculo. Por lo general, la GUI incluye controles tales como menús, barras de herramientas, botones y controles deslizantes (Interfaz de Usuario Personalizada).

Creación de una GUI de MATLAB de forma interactiva.

GUIDE (entorno de desarrollo de GUI) proporciona herramientas para diseñar interfaces de usuario para Apps personalizadas. Mediante el editor de diseño de GUIDE, es posible diseñar gráficamente la interfaz de usuario. GUIDE genera entonces de manera automática el código de MATLAB para construir la interfaz, el cual se puede modificar para programar el comportamiento de la app, como se observa en la figura 4-20.

Creación de una GUI de MATLAB de forma programática.

A fin de ejercer un mayor control sobre el diseño y el desarrollo, también se puede crear código de MATLAB que defina las propiedades y los comportamientos de todos los componentes. MATLAB contiene funcionalidad integrada que le ayudará a crear la GUI para su app de forma programática. Cabe la posibilidad de agregar cuadros de diálogo, controles de interfaz de usuario (como botones y controles deslizantes) y contenedores (como paneles y grupos de botones).

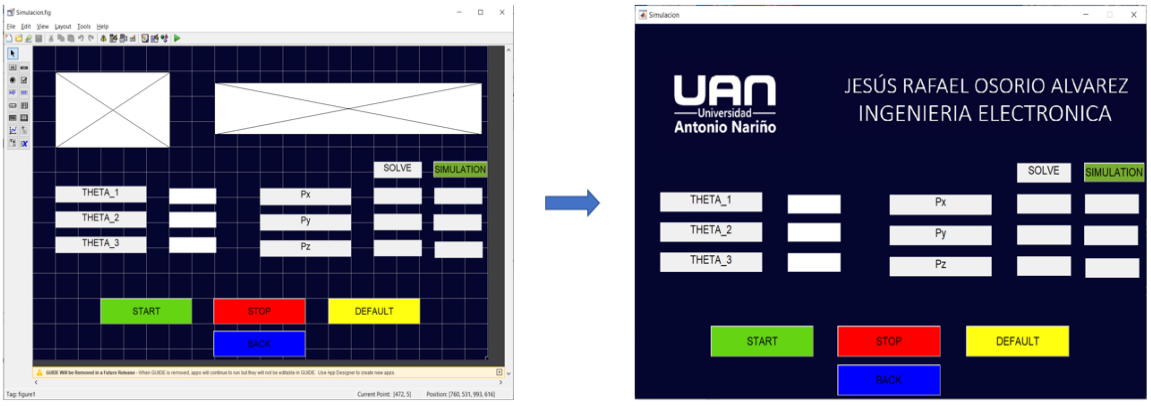


Figura 4-20**:** Interfaz gráfica Cinemática Directa

Fuente Propia.

4.5 Control PID Brazo Robótico

4.5.1 Simulink

Partimos de una base ya diseñada y realizada, que sería el bloque principal del modelo de la cinemática directa. A Dicho bloque le conectaremos el controlador de nuestro diseño.

Se utiliza el bloque PID controler, ingresamos a sus propiedades de bloque y se configura dependiendo del tipo de control a utilizar (P, PI, PID), en nuestro caso en particular escogemos el controlador PID, el cual tiene la capacidad de representar cada uno de los controles mencionados en el marco teórico. Luego se configura la forma del controlador (ideal o paralelo) en donde utilizaremos el controlador PARALELO y por último configuramos el bloque (Externa o Interna), a una fuente EXTERNA para de esta manera ingresar los datos manualmente al controlador y se pueda escoger el control que se necesite. Cada uno de estas modificaciones son realizadas de acuerdo al requerimiento del diseño.

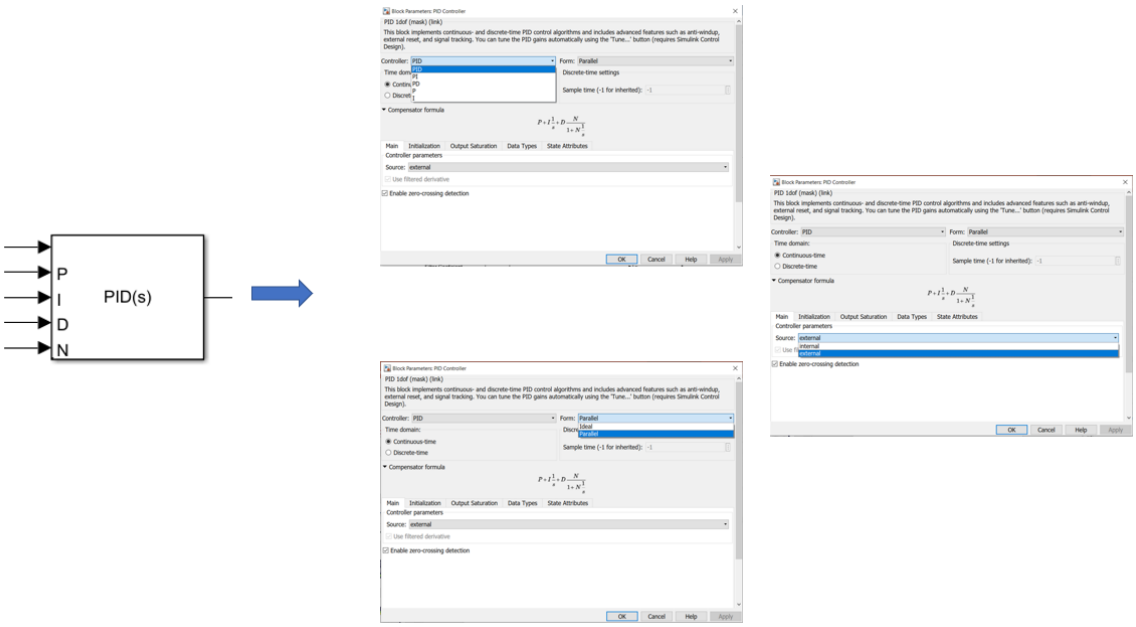


Figura 4-21**:** Configuración bloque PID Controller

Fuente Propia.

En la figura 4-21, se observa cómo se configura el bloque de PID con las características mencionadas en el párrafo anterior y en la figura 4-22 se crea el subsistema con la configuración externa de ingreso de los datos.

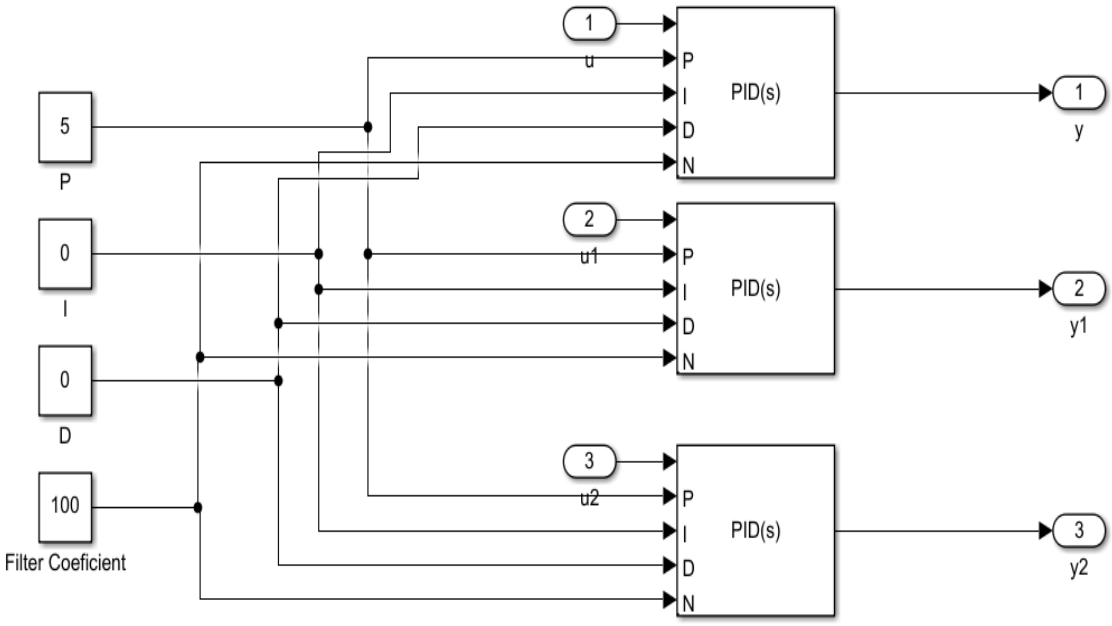


Figura 4-22: Subsistema del control PID

Fuente Propia

Para el siguiente paso, se utilizó el bloque sum que viene siendo un dispositivo detector para realizar la realimentación o el lazo cerrado del controlador, ingresamos a las propiedades del bloque y colocamos + y –. Ver figura 4-23

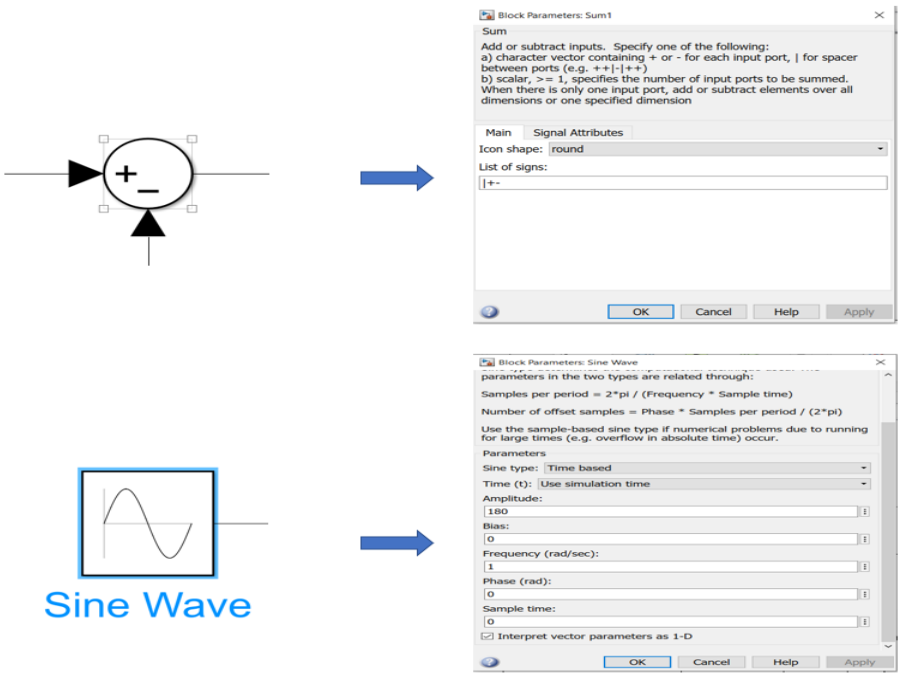
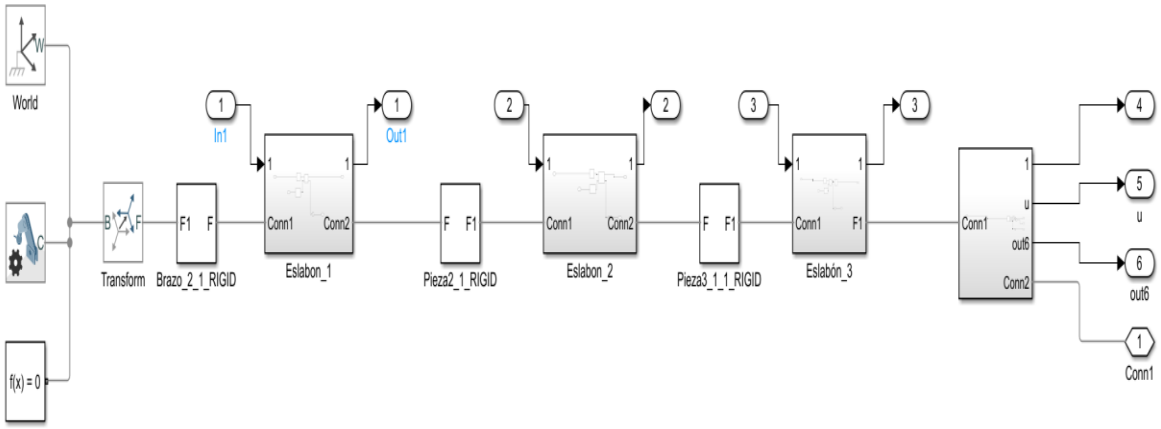


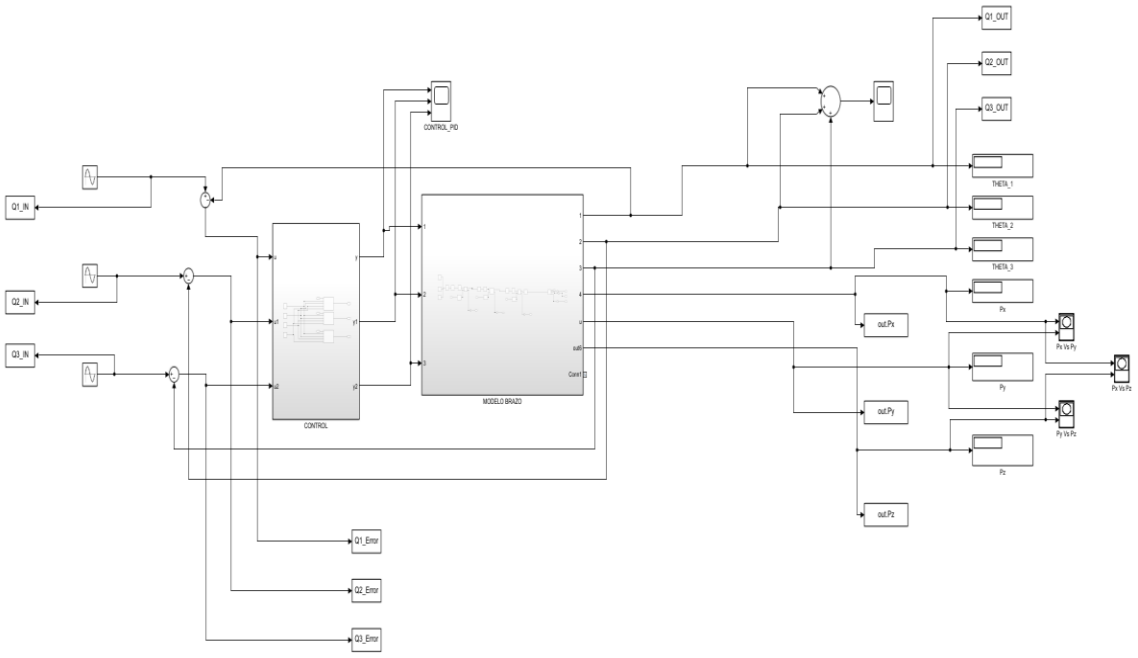
Figura 4-23**:** Configuración bloques Sum y Sine Wave

Fuente Propia.

Luego debemos tomar una fuente de donde provenga la señal, para realizar los movimientos del brazo robótico, entonces utilizamos el bloque Sine Wave, el cual no es más que una fuente de onda sinusoidal la cual modificamos de acuerdo a los parámetros y requisitos de nuestro brazo robótico. Ver figura 4-23.



(a) Subsistema del Brazo



(b) Diseño Final

Figura 4-24**:** Control Brazo robótico.

Fuente Propia.

Como se debe obtener la respuesta del controlador, se crea una nueva ventana en simulink de Matlab y se tiene como base el subsistema del brazo de la figura 4-24 (a), el subsistema del control aplicado al brazo mencionado en la figura 4-22. Se adicionan los siguientes bloques mencionados en la figura 4-25, con sus respectivas configuraciones.

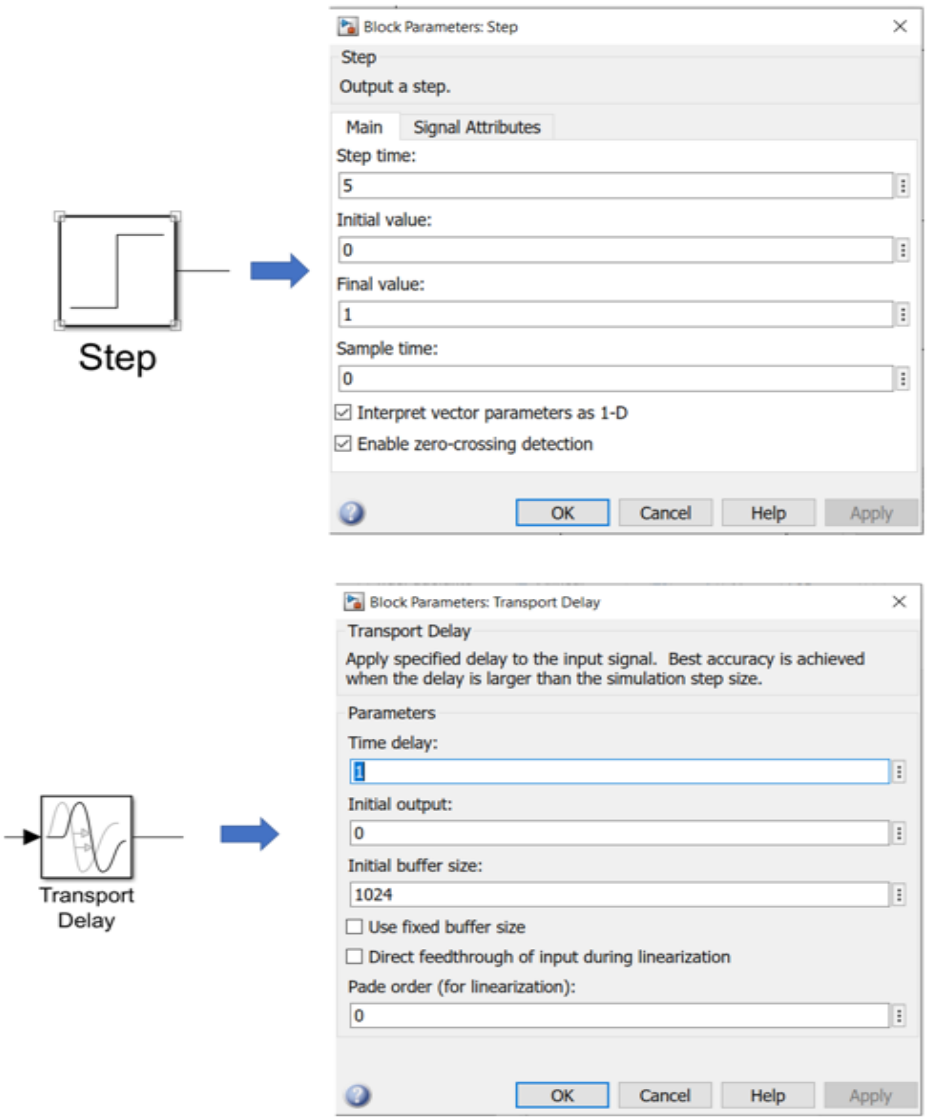


Figura 4-25: Configuración de Bloques Step y Transport Delay

Fuente Propia

4.5.2 Guide – Interfaz Gráfica (Anexo D)

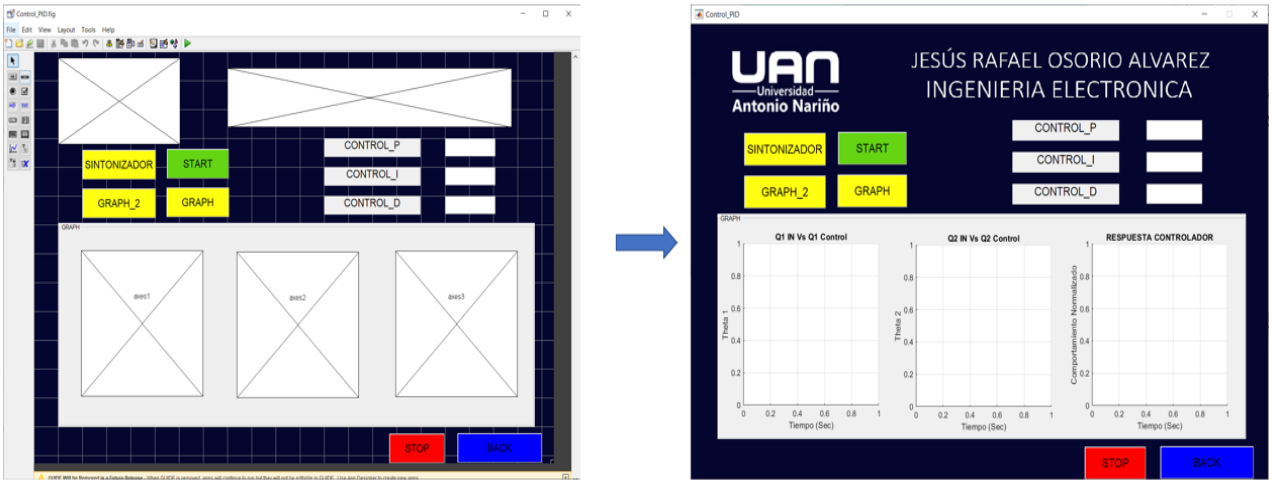


Figura 4-26:. Interfaz gráfica Controlador

Fuente Propia.

Las pruebas se realizaron en dos secciones. La primera es la que se observa en la figura 4-27, en donde se inicia con un valor mínimo de un controlador proporcional kp = 1 para observar las respuestas del controlador con una entrada sinusoidal (Diferencia entre ángulos de entrada y ángulos de salida de una señal sinusoidal) y con una entrada tipo escalón.

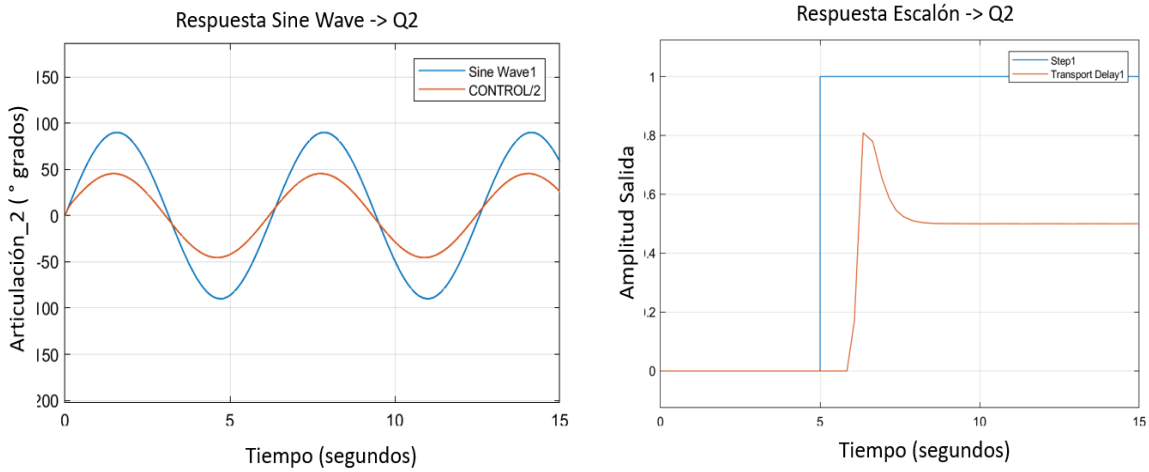


Figura 4-27: Respuesta del Sistema

Fuente Propia

En esta misma sección, se continua con el modelo de sintonización de Zieger – Nichols, aumentando la ganancia proporcional hasta Kp = 24.15, generando una oscilación en la gráfica de respuesta de entrada escalón, como se observa en la figura 4-28.

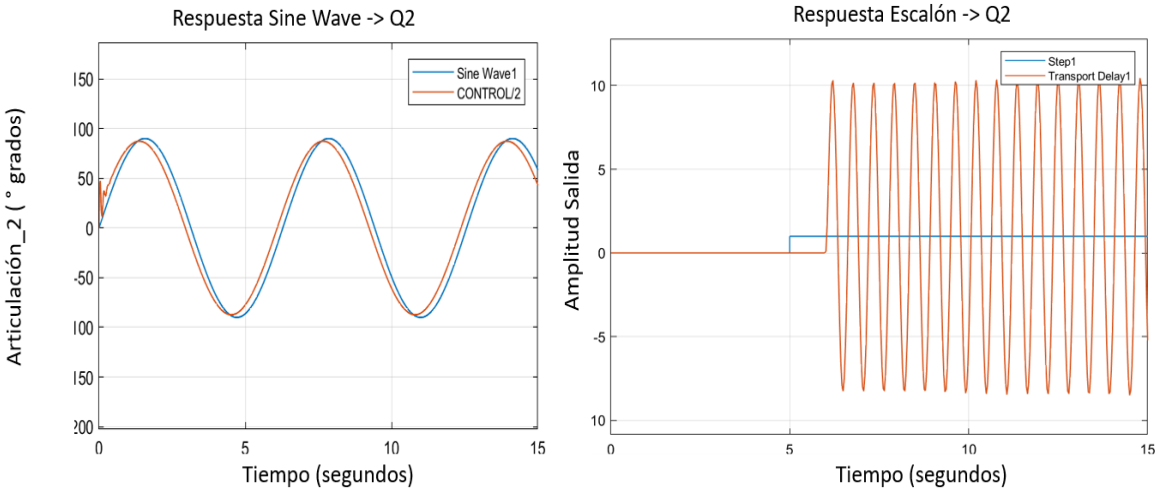


Figura 4-28: Respuesta Oscilatoria

Fuente Propia

El sistema se vuelve oscilatorio, al momento de encontrar la ganancia critica debido a que el sistema nos está indicando que se encuentra al borde de la inestabilidad. Por lo que, en la gráfica de los polos y ceros del sistema del plano s, se evidencia que los polos se encuentran justo sobre el eje imaginario jw, a lo cual un pequeño incremento en la ganancia, pasarían al semiplano derecho del plano s, el cual nos muestra inestabilidad. Que en un controlador proporcional se refleja cómo se observa en la figura 4-29

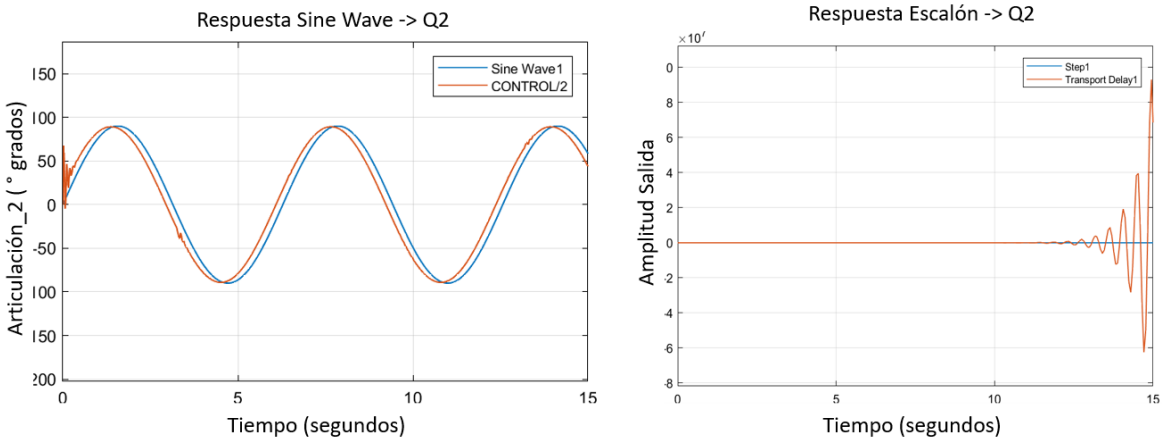
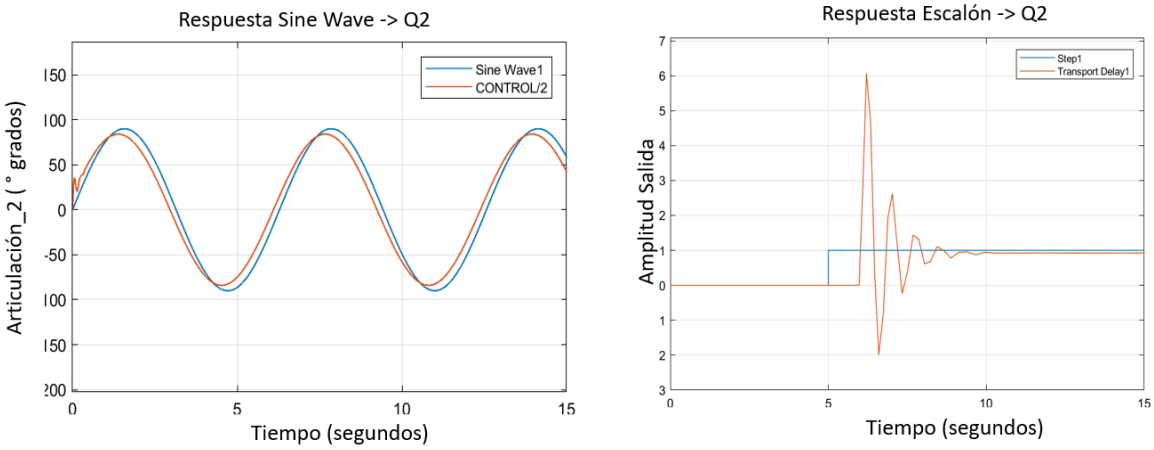


Figura 4-29: Respuesta Inestable

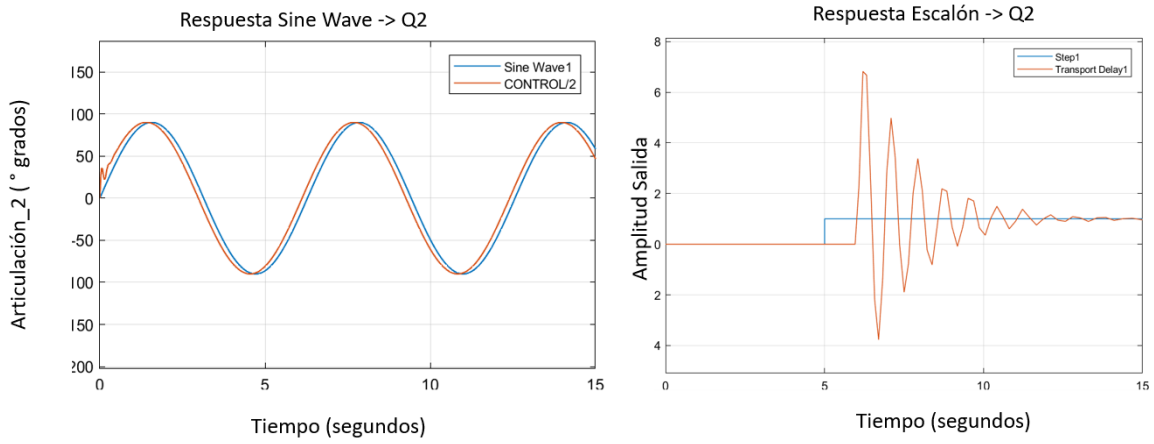
Fuente Propia

Analizando las gráficas de las respuestas sinusoidales, las cuales muestra la diferencia de la entrada del controlador con la salida del controlador de acuerdo a los ángulos de la articulación se observa como la señal de salida empieza a presentar un adelanto con respeto a la señal de entrada, debido a un ruido o inestabilidad presentada en el controlador.

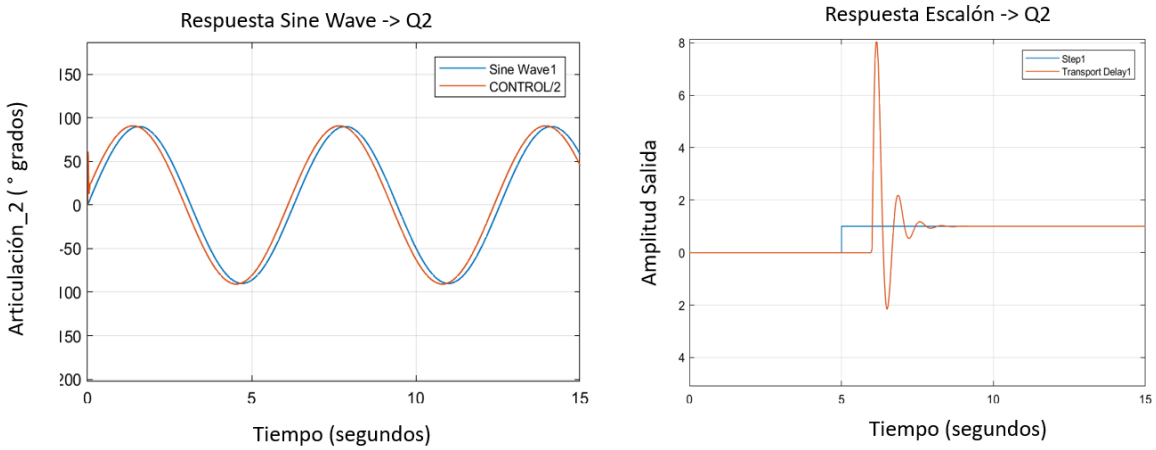
La segunda prueba se observa en la figura 4-30, donde se aplica el 2do método de sintonización de ZIEGER – NICHLOS, y se puede apreciar el comportamiento de cada uno de los controladores.



(a)



(b)



(c)

Figura 4-30: Respuesta de los controladores (a) Control P, (b) Control PI y (c) Control PID

Fuente Propia.

- **Control P**, Se observa el error del OFFSET
- **Control PI**, Se evidencia que el error del OFFSET ya no se encuentra, pero el sistema tiene una oscilación con amplitud directa y lenta.
- **Control PID**, Se evidencia que mejora la estabilidad del sistema y la respuesta del sistema.

4.6 Evaluación al Estudiante (Rubrica de Evaluación)

Como parte de los objetivos se desarrolló un rubrica de evaluación la cual evalúa los conocimientos previos del estudiante y el desarrollo de la misma. Como se observa en la siguiente tabla 4-2.

	Porcentaje	0 a 2.0	2.1 a 4.0	4.1 a 5.0	NOTA
CONOCIMIENTO	30%	Conocimiento deficiente de los siguientes fundamentos teóricos: -Características de los controladores tipo P -Características de los controladores tipo PI. -Características de los controladores tipo PD. -Características de los controladores tipo PID.	Conocimiento y explicación incompleta de los fundamentos teóricos. -Características de los controladores tipo P -Características de los controladores tipo PI. -Características de los controladores tipo PD. -Características de los controladores tipo PID.	Conocimiento completo y explicación clara de los fundamentos teóricos. -Características de los controladores tipo P -Características de los controladores tipo PI. -Características de los controladores tipo PD. -Características de los controladores tipo PID.	
APLICACIÓN Y DESARROLLO DE LA GUÍA	50%	Cumple con UNO de los siguientes criterios: -Puede simular e identificar cada bloque del sistema de control P, PI y PID con Simulink. -Identifica las características de cada controlador a partir de las gráficas. -Calibra correctamente un controlador P, PI, PD y PID.	Cumple con DOS de los siguientes criterios: -Puede simular e identificar cada bloque del sistema de control P, PI y PID con Simulink. -Identifica las características de cada controlador a partir de las gráficas. -Calibra correctamente un controlador P, PI, PD y PID.	Cumple con los TRES de los siguientes criterios: -Puede simular e identificar cada bloque del sistema de control P, PI y PID con Simulink. -Identifica las características de cada controlador a partir de las gráficas. -Calibra correctamente un controlador P, PI, PD y PID.	
VALIDACIÓN	20%	Su análisis, es deficiente en la interpretación de los fundamentos teóricos: -Controladores tipo P. -Controladores tipo PI. -Controladores tipo PD.	Su análisis es incompleto en la interpretación de los fundamentos teóricos: -Controladores tipo P. -Controladores tipo PI. -Controladores tipo PD.	Interpreta correctamente los fundamentos teóricos: -Controladores tipo P. -Controladores tipo PI. -Controladores tipo PD. -Controladores tipo PID.	

		-Controladores tipo PID.	-Controladores tipo PID.		
TOTAL	100%				

Tabla 4-2: Rubrica Para los Estudiantes.

Fuente Propia.

El laboratorio fue presentado y enviado por correo a una cantidad resumida de estudiantes, en los cuales se evaluaron sus competencias teóricas prácticas y su resultado fue el siguiente.

NOMBRE	UNIVERSIDAD	SEMESTRE	Conocimiento	Desarrollo y aplicación	Validación	Nota Final
			30%	50%	20%	100%
Michael Ibbran	UAN	10	3,5	4	3,5	3,75
Felpe Castro	UAN	9	3	3,5	3,8	3,41
Josepth Vargas	UAN	9	3,8	4,5	4,2	4,23

Tabla 4-3: Nota de los Estudiantes.

Fuente Propia.

Los estudiantes, pertenecientes a la UAN, desarrollaron la Guía de laboratorio de manera virtual en sus ordenadores y luego me enviaron respuesta de la guía completamente diligenciada y desarrollada. De acuerdo a los criterios de evaluación mencionados en la rúbrica de la tabla 4-2, se calculó las notas correspondientes a cada estudiante y se evidencio en la tabla 4-3, el cual para mostrar un mejor resultado de las pruebas se importó los datos de la tabla en mención a una gráfica presentada en la figura 4-29.

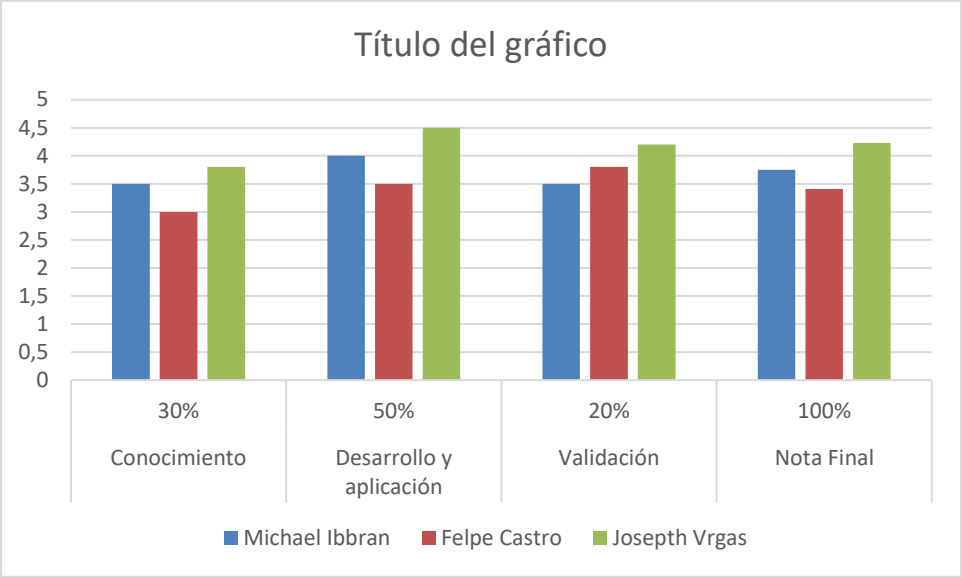


Figura 4-31 Grafico de las notas de los estudiantes.

Fuente Propia.

Conclusiones y recomendaciones

4.7 Conclusiones

La característica principal de este libro y de la guía de laboratorio es que el estudiante pueda utilizar todas las herramientas necesarias, para poder implementar la metodología CDIO de lo aprendido en la materia de robótica y de sistemas de control analógico.

Al enfocarnos en la materia de robótica, se cumple con el primer objetivo específico en donde se concibe y se diseña un prototipo de brazo robótico, de tres grados de libertad y articulaciones fijas rotativas, con el fin de implementar y operarlo. Sirviendo como base para, que el estudiante, pueda diseñar e implementar su propio diseño, mediante la herramienta computacional. La cual le permitirá al estudiante y al docente poder exportar cualquier diseño creado en sus clases o laboratorios y de esta manera realizar las pruebas correspondientes y necesarias.

La operación del diseño creado, también nos permite el uso de herramientas computacionales como MATLAB, para los cálculos matemáticos y la programación necesaria en el desarrollo de nuestro brazo robótico. Esta herramienta fue quien nos permitió realizar las simulaciones adecuadas y poder observar los movimientos esperados, mediante la programación necesaria en el desarrollo de la cinemática directa del brazo robótico descrito. Permitiendo que el estudiante implemente un código o programación de la cinemática, con base a los estudios básicos de la asignatura robótica, una vez desarrollado el código se observa en la simulación, la cinemática directa del robot y sus posiciones de acuerdo a los valores ingresados y a la programación realizada por el estudiante.

Con base al diseño o prototipo robótico, el cual facilita la implementación de un sistema de control automático, mediante la utilización y configuración de bloques de control, de bloques generadores de señales y de bloques transporte de señales. Diseñe una interfaz

gráfica de interacción con ingresos de datos, con el fin que el estudiante pueda elegir el control a utilizar, ingresando los respectivos valores y modificando el diseño. Una vez se pueda ingresar los valores y de observar el comportamiento del sistema, el estudiante puede realizar el segundo método de sintonización de Zieger-Nichols para encontrar las ganancia proporcional, integral y derivativa. Observando el comportamiento de cada controlador con sus respectivos valores asignados y calculados.

Este diseño, permite observar los comportamientos en la respuesta del sistema, identificando el estado del controlador, exactamente si se encuentra estable u oscilatorio, si se encuentra sobre amortiguado o sub amortiguado. Ya que de esto depende la sintonización por el método Ziegler & Nichols, el cual le damos un valor a la ganancia proporcional, aumentándola paulatinamente. Teniendo dos respuestas: la primera es la diferencia entre los ángulos de entrada del sistema con los ángulos de salida del controlador y de la planta. La segunda es en la curva de respuesta del sistema donde se evidencia el estado del controlador, el cual se debe llevar hasta una ganancia critica o hasta que el sistema quede estable u oscilante.

Cuando se implementa el método de sintonización descrito en este libro, en donde encontramos la ganancia proporcional critica, establece un modelo, en el cual nos permite desarrollar y operar controladores Proporcionales, Proporcionales – Integrativos y Proporcionales – integrativos y Derivativos. Obteniendo como resultado la respuesta del sistema, donde se evidencia el tipo de controlador utilizado, lo cual nos permite tomar la decisión correcta sobre el mejor controlador a implementar.

Una vez desarrollado todo el prototipo con sus respectivas interfaces de usuario y sus respectivo funcionamiento, se realiza una Guía de laboratorio virtual, en donde comprende el manual de usuario, una introducción, un marco teórico, sus respectivos objetivos de desarrollo de la misma, preguntas introductorias que el estudiante debe conocer antes de realizar la actividad y por último el desarrollo de la guía por medio de la interfaz gráfica, en donde el estudiante debe responder y argumentar sus respuestas de acuerdo a la interpretación de las gráficas obtenidas.

De esta manera, el estudiante debe llegar con conocimientos teóricos básicos antes de realizar la guía de laboratorio virtual, para poder evaluar su intelecto antes del desarrollo y

después del mismo. Esta evaluación se realiza mediante una rubrica diseñada por mi persona donde se evalúa los conceptos de la metodología CDIO

- Concebir: Conocimiento Previo.
- Diseñar: Desarrollo de la guía.
- Implementar: Aplicación de cada uno de los conceptos descritos y aprendidos.
- Operar: Validación e interpretación de las gráficas y de las tablas descritas en la guía.

4.8 **Recomendaciones**

En este libro se desarrolló un brazo robótico de tres grados de libertad, al cual se le implemento un control y todo con su respetiva simulación. Pero esto es solo el inicio de muchos otros proyectos a futuro.

- Implementar al brazo, los elementos terminales. El cual se encuentra diseñado para soportar dos pinzas individuales en la parte superior del último eslabón y de esta manera programar el brazo robótico con su respectivo control, para que recoja y mueva objetos.
- Gracias a la tecnología existente y al recurso tan amplio del software Matlab, se puede descargar el complemento Arduino e instalar en simulink los bloques necesarios para poder manejar el robot desde una planta externa, mediante potenciómetros y evidenciar el comportamiento del control implementado en tiempo real, por ejemplo.
- Gracias al software SolidWorks y su extensión de guardado de las piezas, se puede imprimir el modelo 3-D del brazo diseñado, donde se le pueden agregar servomotores.
- Una vez realizada la impresión e instalados los servomotores, se podría cambiar a una guía de laboratorio remoto, donde cada una de las modificaciones realizadas al robot se puedan observar físicamente o en sus defectos que varios estudiantes en distintas ciudades puedan tener acceso al robot.

Anexo A: Código Cinemática Directa

Cinemática Directa en Matlab.

```
clc
clear all
syms c1 c2 c3 s1 s2 s3

          %MATRIZ DENAWIT-HARTENBERG
%Ai=[ci  -cosd(alphai)*si   sind(alphai)*s1   ai*ci;
%    si   cosd(alphai)*ci  -sind(aphai)*ci    ai*si;
%    0    sind(alphai)      cosd(alphai)       di;
%    0       0              0                  1];

          %PARAMETROS DENAWIT- HARTENBERG
disp ('ARTICULACIÓN  THETA   D(LONGITUD)  A(LONGITUD) ALPHA');
disp ('1          q1        0.01258      0     90°');
disp ('2          q2         0         0.0170    0');
disp ('3          q3         0         0.011     0')

%longitud de los brazos
d1=0.01258; d2=0;d3=0;
alpha1=90; alpha2=0;alpha3=0;
aa1=0; aa2=0.0170;aa3=0.011;

A1=[c1 -cosd(alpha1)*s1 sind(alpha1)*s1 (aa1*c1);
    s1 cosd(alpha1)*c1 -sind(alpha1)*c1 (aa1*s1);
    0  sind(alpha1)  cosd(alpha1)  d1;
    0  0  0   1];

A2=[c2 -cosd(alpha2)*s2 sind(alpha2)*s2 (aa2*c2);
    s2 cosd(alpha2)*c2 -sind(alpha2)*c2 (aa2*s2);
    0 sind(alpha2) cosd(alpha2) d2;
    0 0 0 1];

A3=[c3 -cosd(alpha3)*s3 sind(alpha3)*s3 (aa3*c3);
    s3 cosd(alpha3)*c3 -sind(alpha3)*c3 (aa3*s3);
    0 sind(alpha3) cosd(alpha3) d3;
    0 0 0 1];

T=A1*A2*A3
```

```
nx=T(1,1)
ny=T(2,1)
nz=T(3,1)

ox=T(1,2)
oy=T(2,2)
oz=T(3,2)

ax=T(1,3)
ay=T(2,3)
az=T(3,3)

Px=T(1,4)
Py=T(2,4)
Pz=T(3,4)
```

Anexo B: Código Interfaz Gráfica Cinemática Directa

```
function varargout = Simulacion(varargin)

gui_Singleton = 1;

gui_State = struct('gui_Name',       mfilename, ...

                   'gui_Singleton',  gui_Singleton, ...

                   'gui_OpeningFcn', @Simulacion_OpeningFcn, ...

                   'gui_OutputFcn',  @Simulacion_OutputFcn, ...

                   'gui_LayoutFcn',  [] , ...

                   'gui_Callback',   []);

if nargin && ischar(varargin{1})

    gui_State.gui_Callback = str2func(varargin{1});

end

 if nargout

    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});

else

    gui_mainfcn(gui_State, varargin{:});

end

% End initialization code - DO NOT EDIT

  % --- Executes just before Simulacion is made visible.

function Simulacion_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject    handle to figure

% eventdata  reserved - to be defined in a future version of MATLAB

% handles    structure with handles and user data (see GUIDATA)
```

```
% varargin   command line arguments to Simulacion (see VARARGIN)
 % Choose default command line output for Simulacion
handles.output = hObject;
 % Update handles structure
guidata(hObject, handles);
 % UIWAIT makes Simulacion wait for user response (see UIRESUME)
% uiwait(handles.figure1);
  % --- Outputs from this function are returned to the command line.
function varargout = Simulacion_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Get default command line output from handles structure
varargout{1} = handles.output;
 function edit1_Callback(hObject, eventdata, handles)
ModelName='Brazo31';


 %longitud de los brazos
d1=0.01258; d2=0;d3=0;
alpha1=90; alpha2=0;alpha3=0;
aa1=0; aa2=0.0170;aa3=0.011;


%obtención de Angulos
theta_1=str2num(get(handles.edit1,'string'));
```

```
theta_2=str2num(get(handles.edit2,'string'));

theta_3=str2num(get(handles.edit3,'string'));


 %simulink

set_param('Brazo31/Constant','value',num2str(theta_1));

set_param('Brazo31/Constant1','value',num2str(theta_2));

set_param('Brazo31/Constant2','value',num2str(theta_3));


A1=[cosd(theta_1) -cosd(alpha1)*sind(theta_1) sind(alpha1)*sind(theta_1)
(aa1*cosd(theta_1));

   sind(theta_1) cosd(alpha1)*cosd(theta_1) -sind(alpha1)*cosd(theta_1)
(aa1*sind(theta_1));

   0  sind(alpha1)  cosd(alpha1)  d1;

   0  0  0   1];


A2=[cosd(theta_2) -cosd(alpha2)*sind(theta_2) sind(alpha2)*sind(theta_2)
(aa2*cosd(theta_2));

   sind(theta_2) cosd(alpha2)*cosd(theta_2) -sind(alpha2)*cosd(theta_2)
(aa2*sind(theta_2));

   0 sind(alpha2) cosd(alpha2) d2;

   0 0 0 1];


A3=[cosd(theta_3) -cosd(alpha3)*sind(theta_3) sind(alpha3)*sind(theta_3)
(aa3*cosd(theta_3));

   sind(theta_3) cosd(alpha3)*cosd(theta_3) -sind(alpha3)*cosd(theta_3)
(aa3*sind(theta_3));

   0 sind(alpha3) cosd(alpha3) d3;

   0 0 0 1];
```

```
T=A1*A2*A3;
 Px=T(1,4);
set(handles.text13,'string',num2str(Px));
Py=T(2,4);
set(handles.text14,'string',num2str(Py));
Pz=T(3,4);
set(handles.text15,'string',num2str(Pz));
 % --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
 % Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in START.
function START_Callback(hObject, eventdata, handles)
ModelName='Brazo31';
 open_system(ModelName);
 set_param(ModelName,'BlockReduction','off');
set_param(ModelName,'StopTime','5');
```

```
set_param(ModelName,'simulationMode','normal');
set_param(ModelName,'StartFcn','1');
set_param(ModelName,'simulationCommand','start');

% --- Executes on button press in STOP.
function STOP_Callback(hObject, eventdata, handles)
ModelName='Brazo31';
 open_system(ModelName);
 set_param(ModelName,'BlockReduction','off');
set_param(ModelName,'StopTime','inf');
set_param(ModelName,'simulationMode','normal');
set_param(ModelName,'StartFcn','1');
set_param(ModelName,'simulationCommand','stop');

 % --- Executes on button press in DEFAULT.
function DEFAULT_Callback(hObject, eventdata, handles)
ModelName='Brazo31';

%longitud de los brazos
d1=0.01258; d2=0;d3=0;
alpha1=90; alpha2=0;alpha3=0;
aa1=0; aa2=0.0170;aa3=0.011;

theta_1=0;
theta_2=0;
```

```
theta_3=0;


set(handles.edit1,'string',num2str(theta_1));

set(handles.edit2,'string',num2str(theta_2));

set(handles.edit3,'string',num2str(theta_3));

 %simulink

set_param('Brazo31/Constant','value',num2str(theta_1));

set_param('Brazo31/Constant1','value',num2str(theta_2));

set_param('Brazo31/Constant2','value',num2str(theta_3));


A1=[cosd(theta_1) -cosd(alpha1)*sind(theta_1) sind(alpha1)*sind(theta_1)
(aa1*cosd(theta_1));

   sind(theta_1) cosd(alpha1)*cosd(theta_1) -sind(alpha1)*cosd(theta_1)
(aa1*sind(theta_1));

   0  sind(alpha1)  cosd(alpha1)  d1;

   0  0  0   1];


A2=[cosd(theta_2) -cosd(alpha2)*sind(theta_2) sind(alpha2)*sind(theta_2)
(aa2*cosd(theta_2));

   sind(theta_2) cosd(alpha2)*cosd(theta_2) -sind(alpha2)*cosd(theta_2)
(aa2*sind(theta_2));

   0 sind(alpha2) cosd(alpha2) d2;

   0 0 0 1];


A3=[cosd(theta_3) -cosd(alpha3)*sind(theta_3) sind(alpha3)*sind(theta_3)
(aa3*cosd(theta_3));
```

```
    sind(theta_3) cosd(alpha3)*cosd(theta_3) -sind(alpha3)*cosd(theta_3)
(aa3*sind(theta_3));

    0 sind(alpha3) cosd(alpha3) d3;

    0 0 0 1];

T=A1*A2*A3;

 Px=T(1,4);

set(handles.text13,'string',num2str(Px));

Py=T(2,4);

set(handles.text14,'string',num2str(Py));

Pz=T(3,4);

set(handles.text15,'string',num2str(Pz));


% --- Executes on button press in BACK.

function BACK_Callback(hObject, eventdata, handles)

close(Simulacion);

Control_Brazo


% --- Executes during object creation, after setting all properties.

function Logo_3_CreateFcn(hObject, eventdata, handles)

Logo_3=imread('Logo_3.jpg');

image(Logo_3)

axis off


% --- Executes during object creation, after setting all properties.

function Logo_4_CreateFcn(hObject, eventdata, handles)

Logo_4=imread('Logo_4.jpg');
```

```
image(Logo_4)
axis off


% --- Executes on button press in SIMULATION.
function SIMULATION_Callback(hObject, eventdata, handles)
load('Px')
load('Py')
load('Pz')
set(handles.text16,'String',Px(2,1));
set(handles.text17,'String',Py(2,1));
set(handles.text18,'String',Pz(2,1));
```

Anexo C: Código Interfaz gráfica Control PID

```
function varargout = Control_PID(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Control_PID_OpeningFcn, ...
                   'gui_OutputFcn',  @Control_PID_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

 % --- Executes just before Control_PID is made visible.
function Control_PID_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Control_PID (see VARARGIN)

% Choose default command line output for Control_PID
handles.output = hObject;
```

```
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Control_PID wait for user response (see UIRESUME)
% uiwait(handles.figure1);

 % --- Outputs from this function are returned to the command line.
function varargout = Control_PID_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

 % --- Executes on button press in BACK.
function BACK_Callback(hObject, eventdata, handles)
close(Control_PID);
Control_Brazo

 function edit1_Callback(hObject, eventdata, handles)

ModelName='Brazo31_PID'

%obtención del Control_PID
Control_P=str2num(get(handles.edit1,'string'));
Control_I=str2num(get(handles.edit2,'string'));
Control_D=str2num(get(handles.edit3,'string'));

%simulink
```

```
set_param('Brazo31_PID/CONTROL/P','value',num2str(Control_P));
set_param('Brazo31_PID/CONTROL/I','value',num2str(Control_I));
set_param('Brazo31_PID/CONTROL/D','value',num2str(Control_D));

% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

 % --- Executes during object creation, after setting all properties.
function Logo_3_CreateFcn(hObject, eventdata, handles)
Logo_3=imread('Logo_3.jpg');
image(Logo_3)
axis off

% --- Executes during object creation, after setting all properties.
function Logo_4_CreateFcn(hObject, eventdata, handles)
Logo_4=imread('Logo_4.jpg');
image(Logo_4)
axis off

 % --- Executes on button press in START.
function START_Callback(hObject, eventdata, handles)
ModelName='Brazo31_PID';
 open_system(ModelName);
```

```
 set_param(ModelName,'BlockReduction','off');
set_param(ModelName,'StopTime','15');
set_param(ModelName,'simulationMode','normal');
set_param(ModelName,'StartFcn','1');
set_param(ModelName,'simulationCommand','start');

% --- Executes on button press in STOP.
function STOP_Callback(hObject, eventdata, handles)
ModelName='Brazo31_PID';
 open_system(ModelName);
 set_param(ModelName,'BlockReduction','off');
set_param(ModelName,'StopTime','15');
set_param(ModelName,'simulationMode','normal');
set_param(ModelName,'StartFcn','1');
set_param(ModelName,'simulationCommand','stop');



% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
grid on
title('Q1 IN Vs Q1 Control')
xlabel('Tiempo (Sec)')
ylabel('Theta 1')

 % --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
grid on
title('Q2 IN Vs Q2 Control')
xlabel('Tiempo (Sec)')
ylabel('Theta 2')

% --- Executes during object creation, after setting all properties.
```

```
function axes3_CreateFcn(hObject, eventdata, handles)
grid on
title('RESPUESTA CONTROLADOR')
xlabel('Tiempo (Sec)')
ylabel('Comportamiento Normalizado')

% --- Executes on button press in GRAPH.
function GRAPH_Callback(hObject, eventdata, handles)

%CARGAMOS LA INFORMACIÓN GUARDADA .m
load('Q1_IN');
load('Q2_IN');
load('Q3_IN');
load('Q1_Error');
load('Q2_Error');
load('Q3_Error');
load('Q1_OUT');
load('Q2_OUT');
load('Q3_OUT');

%Transponemos los vectores.
q1_in=(Q1_IN).';
q2_in=(Q2_IN).';
q3_in=(Q3_IN).';
q1_out=(Q1_OUT).';
q2_out=(Q2_OUT).';
q3_out=(Q3_OUT).';

%Vectorizamos cada elemento
t=q1_in(:,1);
q1=q1_in(:,2);
q2=q2_in(:,2);
q3=q3_in(:,2);
```

```
Q1=q1_out(:,2);
Q2=q2_out(:,2);
Q3=q3_out(:,2);

%Graficamos
axes(handles.axes1)
plot(t,q1,'r',t,Q1,'b')
grid on
title('Q1 IN Vs Q1 Control')
xlabel('Tiempo (Sec)')
ylabel('Theta 1')

axes(handles.axes2)
plot(t,q2,'r',t,Q2,'b')
grid on
title('Q2 IN Vs Q2 Control')
xlabel('Tiempo (Sec)')
ylabel('Theta 2')

% axes(handles.axes3)
% plot(t,q3,'r',t,Q3,'b')
% grid on
% title('Q3 IN Vs Q3 Control')
% xlabel('Tiempo (Sec)')
% ylabel('Theta 3')

 function edit2_Callback(hObject, eventdata, handles)
ModelName='Brazo31_PID'

%obtención del Control_PID
Control_P=str2num(get(handles.edit1,'string'));
```

```
Control_I=str2num(get(handles.edit2,'string'));
Control_D=str2num(get(handles.edit3,'string'));

%simulink
set_param('Brazo31_PID/CONTROL/P','value',num2str(Control_P));
set_param('Brazo31_PID/CONTROL/I','value',num2str(Control_I));
set_param('Brazo31_PID/CONTROL/D','value',num2str(Control_D));

% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

 function edit3_Callback(hObject, eventdata, handles)
ModelName='Brazo31_PID'

%obtención del Control_PID
Control_P=str2num(get(handles.edit1,'string'));
Control_I=str2num(get(handles.edit2,'string'));
Control_D=str2num(get(handles.edit3,'string'));

%simulink
set_param('Brazo31_PID/CONTROL/P','value',num2str(Control_P));
set_param('Brazo31_PID/CONTROL/I','value',num2str(Control_I));
set_param('Brazo31_PID/CONTROL/D','value',num2str(Control_D));
```

```
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
ModelName='Brazo31_PID_1'
open_system(ModelName);

set_param(ModelName,'BlockReduction','off');
set_param(ModelName,'StopTime','15');
set_param(ModelName,'simulationMode','normal');
set_param(ModelName,'StartFcn','1');
set_param(ModelName,'simulationCommand','start');

%obtención del Control_PID
Control_P=str2num(get(handles.edit1,'string'));
Control_I=str2num(get(handles.edit2,'string'));
Control_D=str2num(get(handles.edit3,'string'));

 %simulink
set_param('Brazo31_PID_1/CONTROL/P','value',num2str(Control_P));
```

```
set_param('Brazo31_PID_1/CONTROL/I','value',num2str(Control_I));
set_param('Brazo31_PID_1/CONTROL/D','value',num2str(Control_D));

 % --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
%CARGAMOS LA INFORMACIÓN GUARDADA .m
load('step1_IN');
load('step2_IN');
load('step3_IN');
load('step1_OUT');
load('step2_OUT');
load('step3_OUT');

%Transponemos los vectores.
step1_in=(step1_IN).';
step2_in=(step2_IN).';
step3_in=(step3_IN).';
step1_out=(step1_OUT).';
step2_out=(step2_OUT).';
step3_out=(step3_OUT).';

%Vectorizamos cada elemento
t=step1_in(:,1);
step1=step1_in(:,2);
step2=step2_in(:,2);
step3=step3_in(:,2);

STEP1=step1_out(:,2);
STEP2=step2_out(:,2);
STEP3=step3_out(:,2);

%Graficamos
axes(handles.axes3)
```

```
plot(t,step1,'r',t,STEP1,'b')
grid on
title('RESPUESTA CONTROLADOR')
xlabel('Tiempo (Sec)')
ylabel('Comp. Normalizado')
```

Anexo D: Manual De la Interfaz Grafica

Manual

Abrir Matlab y descargar los siguientes Archivos (entregados comprimidos):

- Ficheros .m
- Ficheros .fig
- Ficheros .slx

Simulink Brazo.

Abrir los siguientes archivos

- Brazo31.slx
- Brazo31_PID.slx
- Brazo31_PID_1.slx

Interfaz Gráfica.

Abrir el archivo con el nombre Control_Brazo.fig y seguir las siguientes instrucciones:

- Panel de Control Principal.
- Se Puede escoger entre:
  - Control a realizar.
  - Cinemática Directa del Robot (Brazo a Simular).

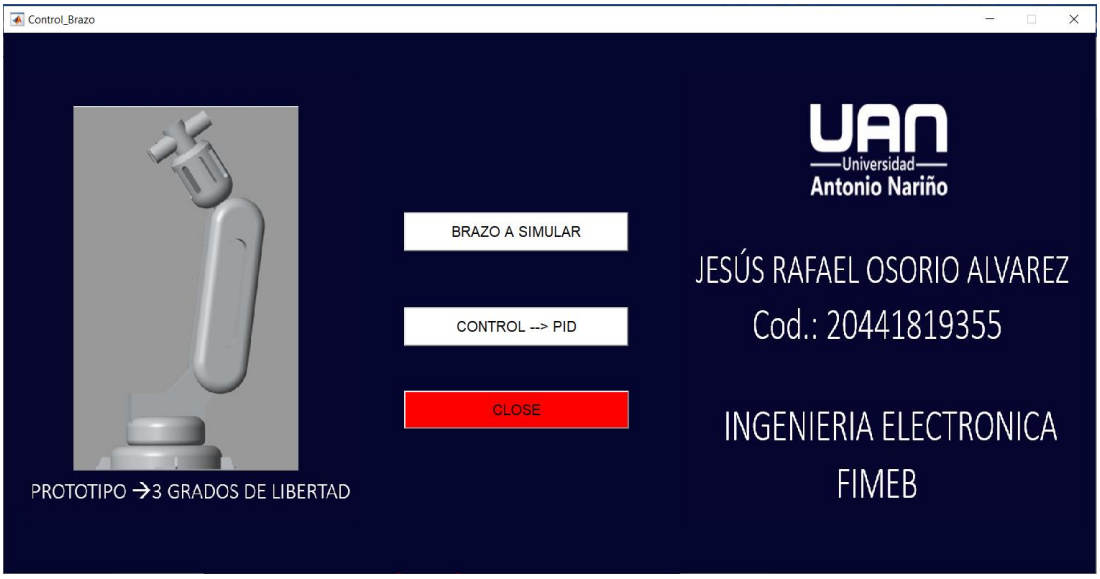


Figura 3. Interfaz Gráfica. Fuente Propia

Cinemática Directa

Paso_1: Escoger la Opción Brazo a Simular.

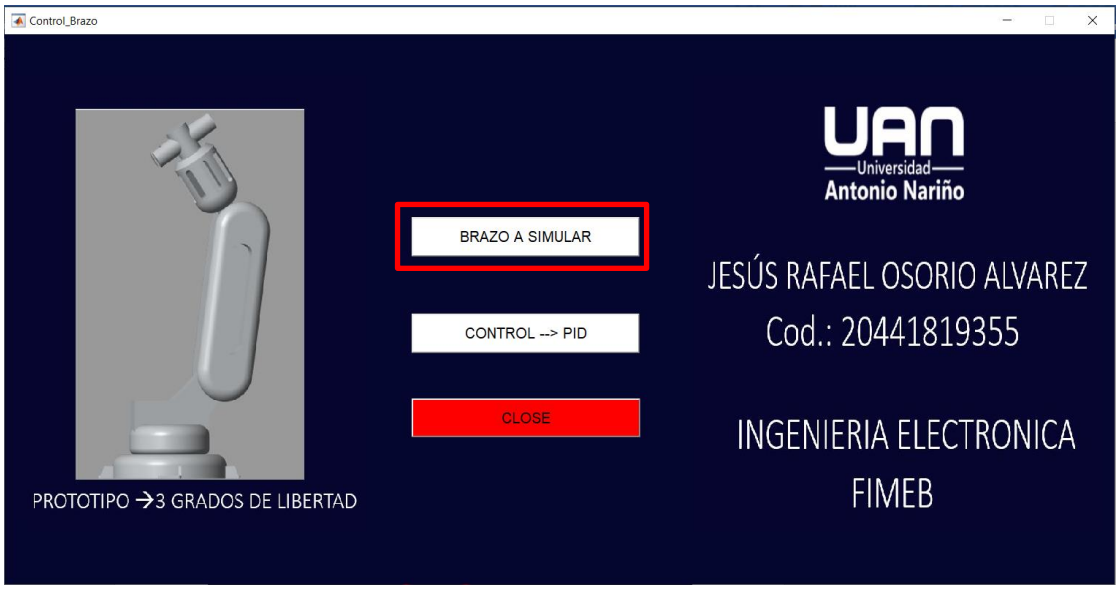


Figura 4. Interfaz Gráfica. Fuente Propia

Paso_2: Llenar las casillas seleccionadas, Correspondiente a los ángulos.

Paso_3: Oprimir el botón START.

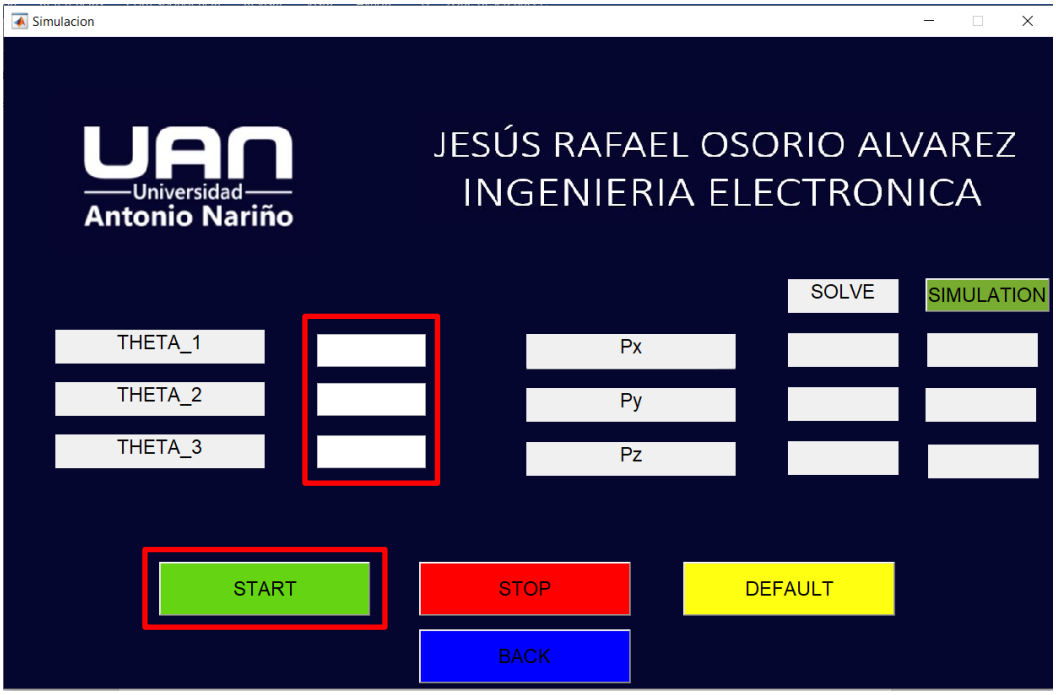


Figura 5. Interfaz Gráfica. Fuente Propia

Paso_2: Verificar Posición de los ángulos en el SIMULADOR.

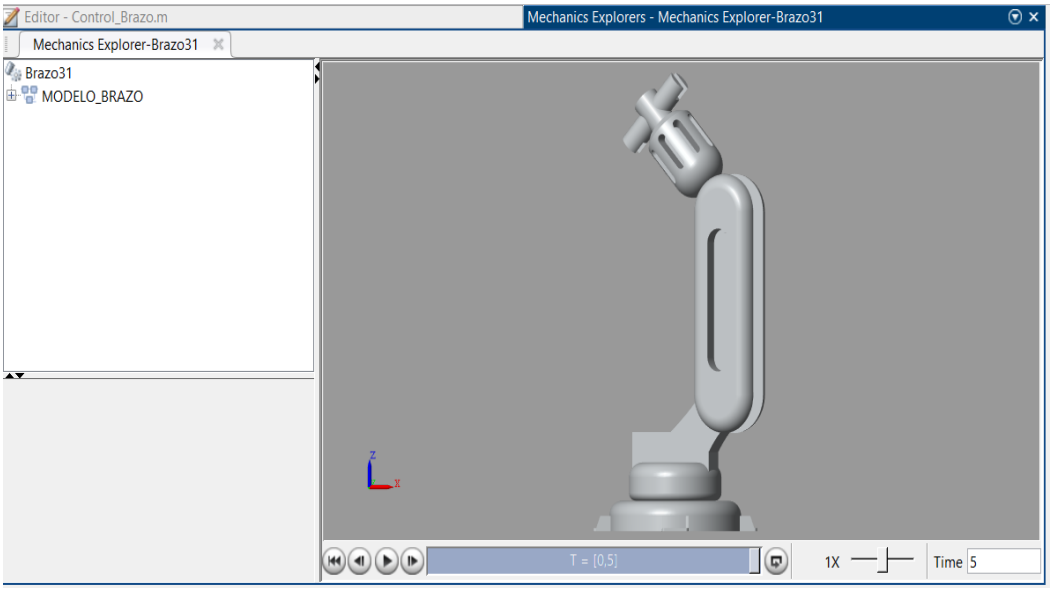


Figura 6. Interfaz Gráfica. Fuente Propia

Paso_3: Verificar el Cálculo de las posiciones.

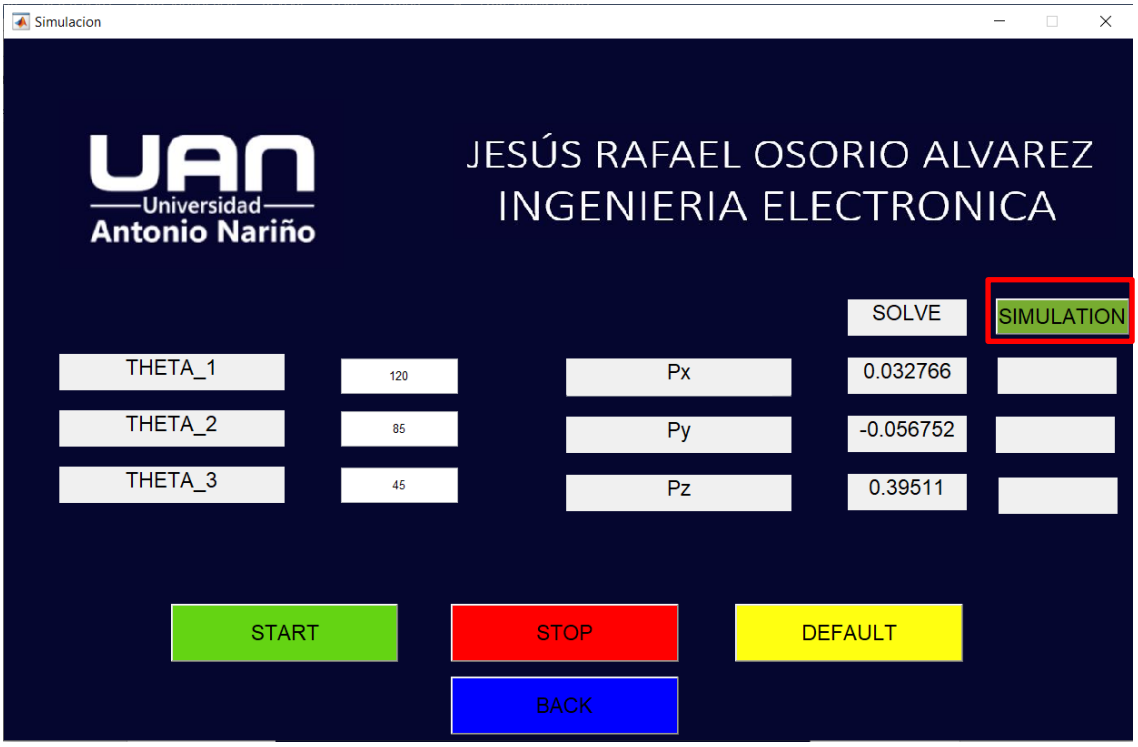


Figura 7. Interfaz Gráfica. Fuente Propia

Paso_4: Oprimir el botón SIMULATION, para observar las posiciones reales del brazo

Paso_5: El Botón Default, Coloca el Robot en su posición original.

Paso_6: Oprimir el Botón BACK. Para devolver al panel inicial

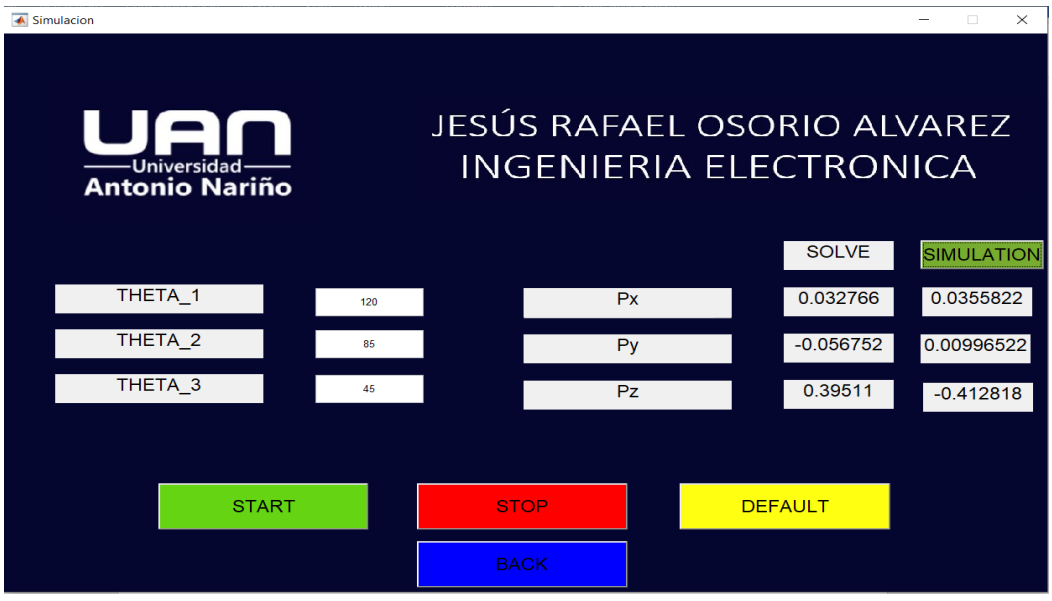


Figura 8. Interfaz Gráfica. Fuente Propia

Control a Realizar

Paso_1: Escoger el Control.

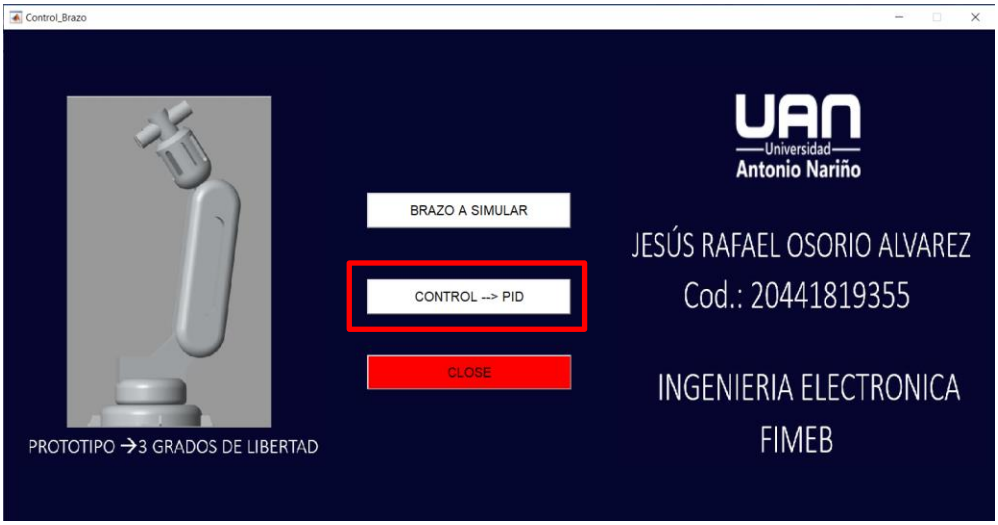


Figura 9. Interfaz Gráfica. Fuente Propia

Paso_2: Llenar las casillas seleccionadas, Correspondiente a los Coeficientes de Control.

Paso_3: Oprimir el Botón START.

Paso_4: Oprimir el Botón Sintonizador.

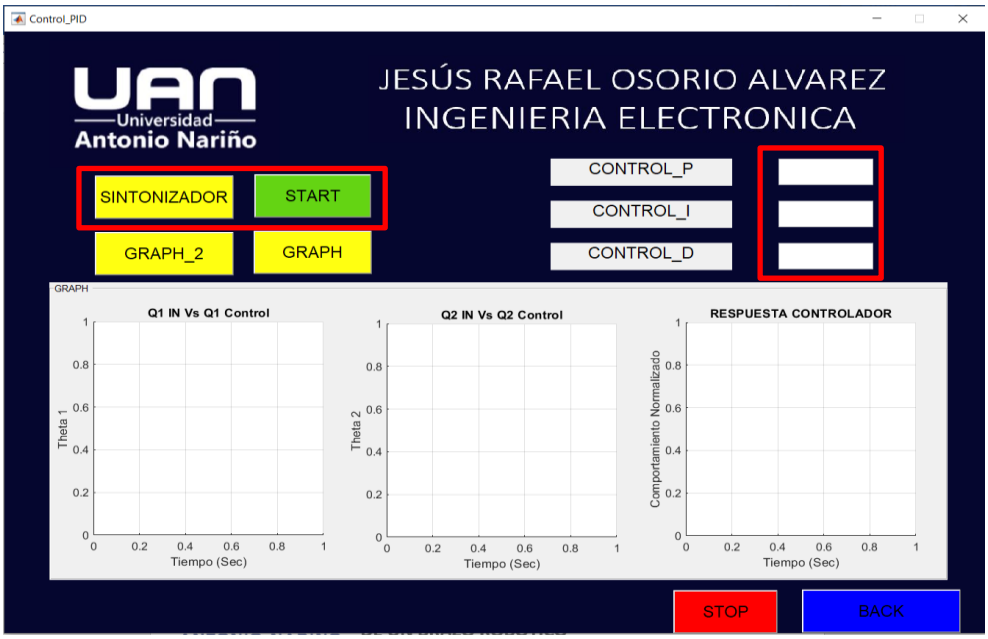


Figura 10. Interfaz Gráfica. Fuente Propia

Paso_5: Verificar el movimiento del Brazo en el SIMULADOR y la trayectoria.

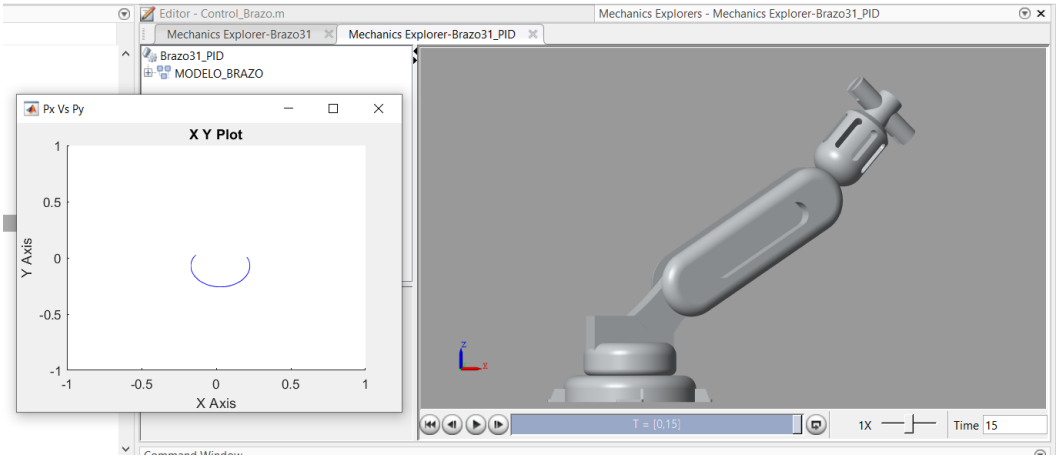


Figura 11. Interfaz Gráfica. Fuente Propia

Paso_3: Oprimir el Botón GRAPH y GRAPH_1

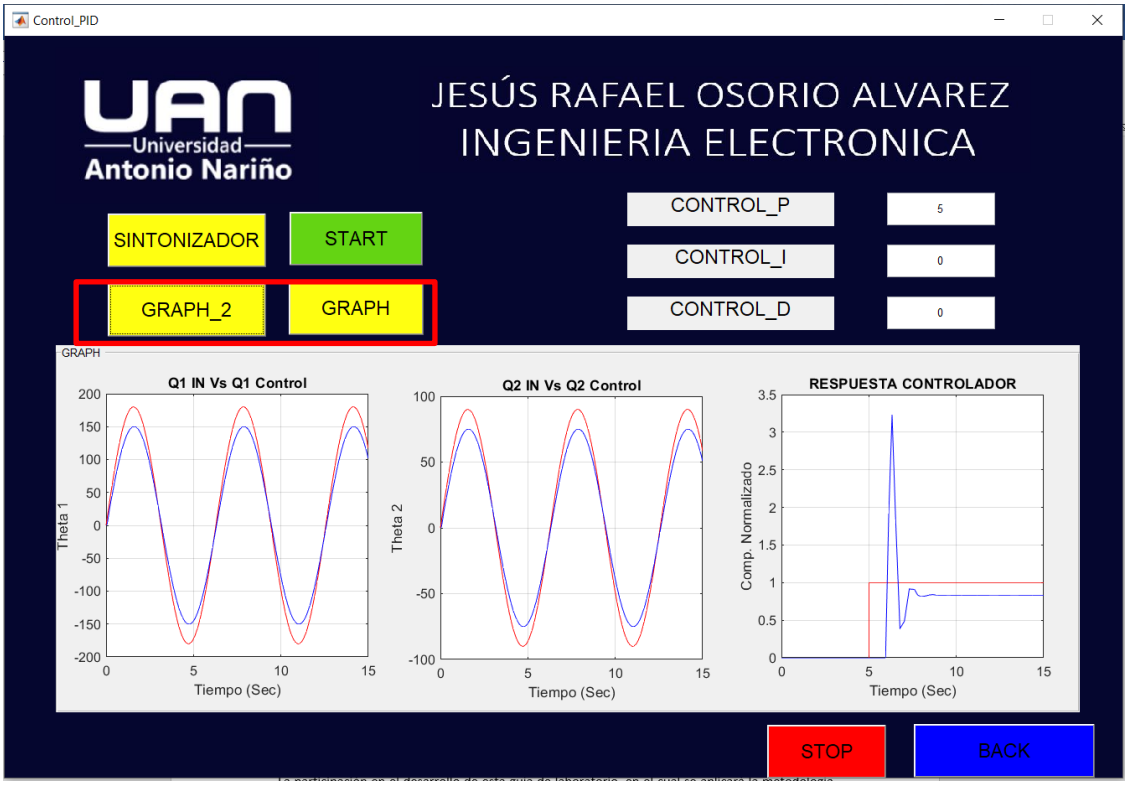


Figura 12. Interfaz Gráfica. Fuente Propia

Paso_6: Observar las Gráficas In Vs Out (posiciones) y Respuesta del Controlador.

Paso_7: Oprimir el Botón BACK. Para devolver al panel inicial

Anexo E: Calculo Tabla DENAVIT – HARTENBERG y matriz de transformación homogénea.

Paso_1: Se llena la siguiente tabla, la cual se encuentra milímetros. De acuerdo a las medidas de nuestro prototipo.

ROBOT	ESLABÓN_0	ESLABÓN_1	ESLABÓN_2	ESLABÓN_3	ALCANCE
PROTOTIPO	4 mm	8.58 mm	17 mm	11 mm	40.58 mm

Tabla 1. Medidas y alcance del prototipo.

Paso_2: Definir los Eslabones, de la siguiente manera. Identificando el eslabón_0 como la base, que esta estático y luego seguimos identificando los demás eslabones, dándole nombres desde el eslabón_1 hasta el eslabón_n.

Paso_3: Definir las Articulaciones, Es decir los puntos de movimientos del prototipo. La 1ra articulación será el 1er grado de libertad del brazo robótico y n el ultimo grado de libertad, en nuestro caso es la 3ra articulación, lo cual indica que nuestro robot tiene 3 grados de libertad.

Paso_4. Definimos un sistema de referencia a cada Articulación. Teniendo en cuenta lo siguiente:

- No se puede desplazar en el eje Y.
- No se puede rotar con respecto al eje Y.
- La rotación en el eje Z, debe coincidir con la traslación del eje Z.
- Deben cumplir la propiedad de doble perpendicularidad
    - Xi perpendicular Zi
    - Xi perpendicular Zi-1

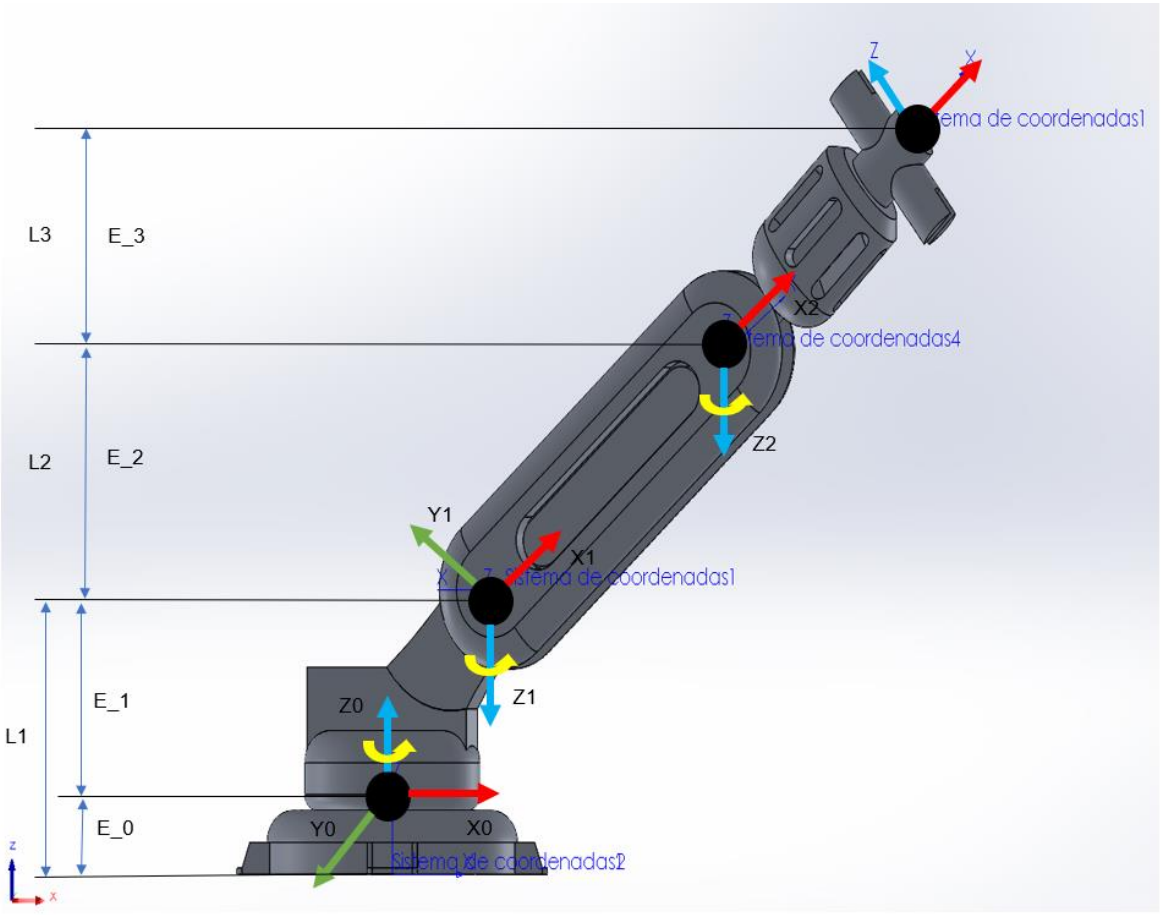


Figura_1 Definición de Sistemas de Referencia

Paso_5: Se calcula los valores para la 1ra Articulación.

- $\boldsymbol{\theta_1}$**:** Angulo entre Xo y X1.
- **d1**: Distancia entre Xo y X1, siempre y cuando se encuentren en la misma dirección. La cual sería L1, que a su vez es la suma del Eslabón_0 y el Eslabón_1.
- **a1**: Distancia ente Zo y Z1.
- $\boldsymbol{\alpha 1}$: Ángulo que existe entre Zo y Z1.

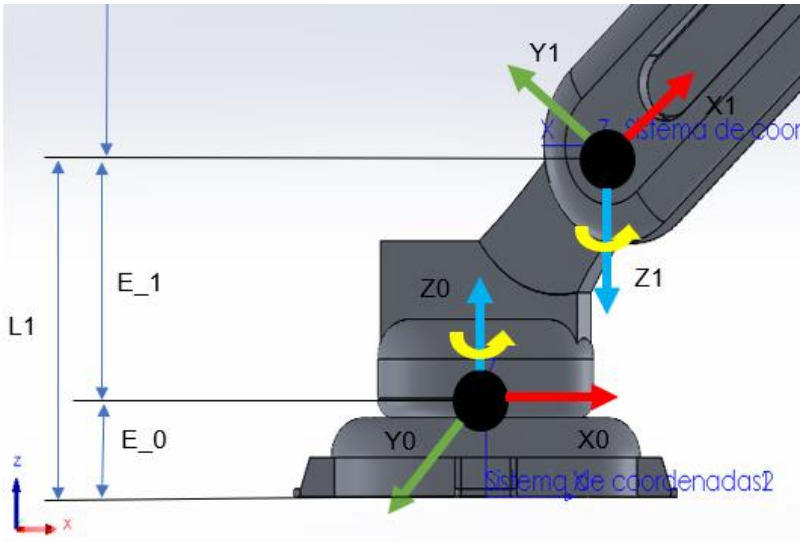


Figura_2 Imagen de las articulaciones 1 y 2

Articulación	$\boldsymbol{\theta}$	**d**	**a**	$\boldsymbol{\alpha}$
1	$\boldsymbol{\theta_1}$	12.58 mm	1.25mm	90°

Tabla 2. Datos Articulación 1

Paso_6: Se calcula los valores para la 1ra Articulación.

- $\boldsymbol{\theta_2}$**:** Angulo entre X1 y X2.
- **d2**: Distancia entre X1 y X2. Se encuentran en la misma posición.
- **a2**: Distancia ente Z1 y Z2. La cual sería L2
- $\boldsymbol{\alpha 2}$: Ángulo que existe entre Z1 y Z2, Pero como tiene la misma dirección es 0

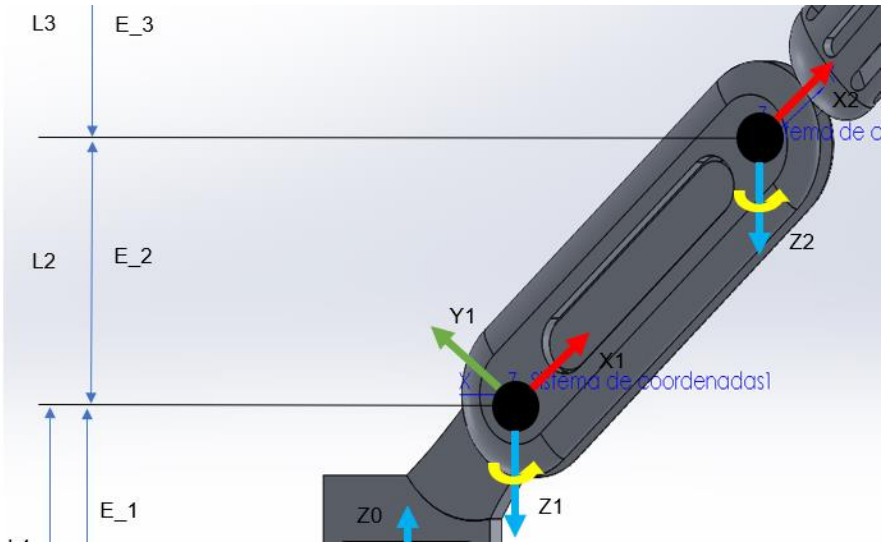


Figura_3 Imagen de las articulaciones 1 y 2

Articulación	$\boldsymbol{\theta}$	**d**	**a**	$\boldsymbol{\alpha}$
2	$\boldsymbol{\theta_2}$	0	17 mm	0

Tabla 3. Datos Articulación 2

Paso_7: Se calcula los valores para la 1ra Articulación.

- $\boldsymbol{\theta_3}$**:** Angulo entre X1 y X2.
- **D3**: Distancia entre X1 y X2. Se encuentran en la misma posición.
- **A3**: Distancia ente Z1 y Z2. La cual sería L3
- $\boldsymbol{\alpha 3}$: Ángulo que existe entre Z1 y Z2, Pero como tiene la misma dirección es 0

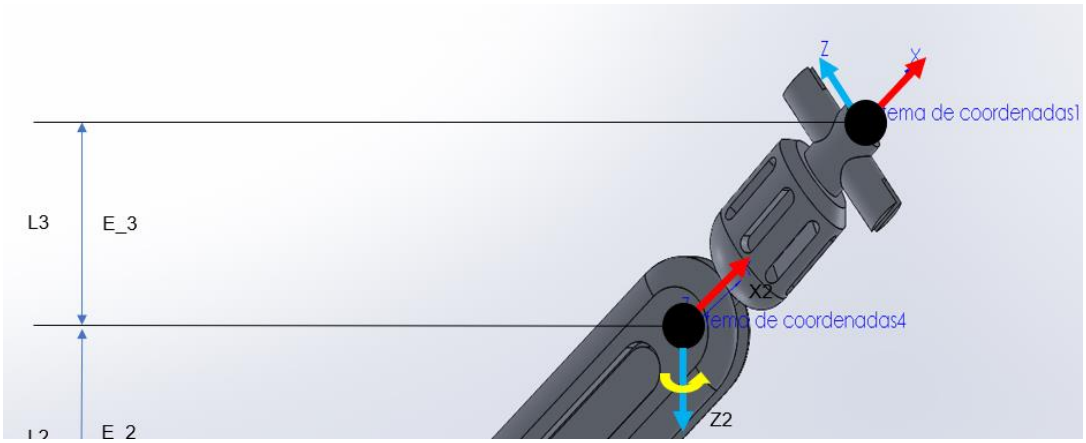


Figura_4 Imagen de las articulaciones 1 y 2

Articulación	$\boldsymbol{\theta}$	**d**	**a**	$\boldsymbol{\alpha}$
3	$\boldsymbol{\theta_3}$	0	11 mm	180

Tabla 4. Datos Articulación 3

Paso_8: Completar tabla de D-H.

Articulación	$\boldsymbol{\theta}$	**d**	**a**	$\boldsymbol{\alpha}$
1	$\boldsymbol{\theta_1}$	12.58 mm	1.25 mm	90°
2	$\boldsymbol{\theta_2}$	0	17 mm	0
3	$\boldsymbol{\theta_3}$	0	11 mm	180°

Tabla 5. Tabla D-H

Paso_9: Una vez Calculada y finalizada la tabla 5. Se procede a realizar la matriz homogénea (ecuación 3-2 del presente libro) para cada articulación.

$$ {}^{i-1}A_{i-1} = \begin{pmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3\text{-}2) $$

$$ {}^{0}A_1 = \begin{pmatrix} C\theta_1 & -C(90°)S\theta_1 & S(90°)S\theta_1 & (0.00125)C\theta_1 \\ S\theta_1 & C(90°)C\theta_1 & -S(90°)C\theta_1 & (0.00125)S\theta_1 \\ 0 & S(90°) & C(90°) & 0.01258 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{0}A_1 = \begin{pmatrix} C\theta_1 & 0 & S\theta_1 & C\theta_1/800 \\ S\theta_1 & 0 & -C\theta_1 & S\theta_1/800 \\ 0 & 1 & 0 & 0.01258 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (AE\text{-}1) $$

$$ {}^{1}A_2 = \begin{pmatrix} C\theta_2 & -C(0°)S\theta_2 & S(0°)S\theta_2 & (0.0170)C\theta_2 \\ S\theta_2 & C(0°)C\theta_2 & -S(0°)C\theta_2 & (0.0170)S\theta_2 \\ 0 & S(0°) & C(0°) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{1}A_2 = \begin{pmatrix} C\theta_2 & -S\theta_2 & 0 & (0.0170)C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & (0.0170)S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (AE\text{-}2) $$

$$ {}^{2}A_3 = \begin{pmatrix} C\theta_3 & -C(180°)S\theta_3 & S(180°)S\theta_3 & (0.011)C\theta_3 \\ S\theta_3 & C(180°)C\theta_3 & -S(180°)C\theta_3 & (0.011)S\theta_3 \\ 0 & S(180°) & C(180°) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{2}A_3 = \begin{pmatrix} C\theta_3 & S\theta_3 & 0 & (0.011)C\theta_3 \\ S\theta_3 & -C\theta_3 & 0 & (0.011)S\theta_3 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{(AE-3)} $$

Paso_10: Obtenemos la matriz de transformación (ecuación 3-3 del presente libro) que relaciona el sistema base con el extremo.

$$ T = {}^{0}A_1 * {}^{1}A_2 * {}^{2}A_3 = {}^{0}A_3 = \begin{pmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad \text{(3-3)} $$

$$ \text{T=} \begin{pmatrix} C\theta_1 & 0 & S\theta_1 & C\theta_1/800 \\ S\theta_1 & 0 & -C\theta_1 & S\theta_1/800 \\ 0 & 1 & 0 & 0.01258 \\ 0 & 0 & 0 & 1 \end{pmatrix} * \begin{pmatrix} C & -S\theta_2 & 0 & (0.0170)C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & (0.0170)S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} * $$

$$ \begin{pmatrix} C\theta_3 & S\theta_3 & 0 & (0.011)C\theta_3 \\ S\theta_3 & -C\theta_3 & 0 & (0.011)S\theta_3 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} n_x & o_x & a_x & P_x \\ n_y & o_y & a_y & P_y \\ n_z & o_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

T= [$C\theta_1$*$C\theta_2$*$C\theta_3$ - $C\theta_1$*$S\theta_2$*$S\theta_3$, $C\theta_1$*$C\theta_2$*$S\theta_3$ + $C\theta_1$*$C\theta_3$*$S\theta_2$, -$S\theta_1$, $C\theta_1$/800 + (0.017*$C\theta_1$*$C\theta_2$) + (0.011*$C\theta_1$*$C\theta_2$*$C\theta_3$) - (0.011*$C\theta_1$*$S\theta_2$*$S\theta_3$)]

[$C\theta_2$*$C\theta_3$*$S\theta_1$ - $S\theta_1$*$S\theta_2$*$S\theta_3$, $C\theta_2$*$S\theta_1$*$S\theta_3$ + $C\theta_3$*$S\theta_1$*$S\theta_2$, $C\theta_1$, $S\theta_1$/800 + (0.017*$C\theta_2$*$S\theta_1$) + (0.011*$C\theta_2$*$C\theta_3$*$S\theta_1$) - (0.011*$S\theta_1$*$S\theta_2$*$S\theta_3$)]

[$C\theta_2$*$S\theta_3$ + $C\theta_3$*$S\theta_2$, $S\theta_2$*$S\theta_3$ - $C\theta_2$*$C\theta_3$, 0, (0.017*$S\theta_2$) + (0.011*$C\theta_2$*$S\theta_3$) + (0.011*$C\theta_3$*$S\theta_2$) + 629/50000]

[0, 0, 0, 1] (AE – 4)

Paso 11: De la ecuación AE-4, Se obtiene la información de las posiciones finales.

Px = $C\theta_1$/800 + (0.017*$C\theta_1$*$C\theta_2$) + (0.011*$C\theta_1$*$C\theta_2$*$C\theta_3$) - (0.011*$C\theta_1$*$S\theta_2$*$S\theta_3$) (AE-5)

Py = $S\theta_1$/800 + (0.017*$C\theta_2$*$S\theta_1$) + (0.011*$C\theta_2$*$C\theta_3$*$S\theta_1$) - (0.011*$S\theta_1$*$S\theta_2$*$S\theta_3$) (AE-6)

Pz = (0.017*$S\theta_2$) + (0.011*$C\theta_2$*$S\theta_3$) + (0.011*$C\theta_3$*$S\theta_2$) + 629/50000 (AE-7)

Bibliografía

Aguirre Gil, I., Arismendi, C., & Luis, A. (2011). Sistema manipulador antropomórfico de tres grados de libertad. *ITECKNE*, 87 - 95.

Andujar Marquez, J. M. (2010). DISEÑO DE LABORATORIOS VIRTUALES Y/O REMOTOS. UN CASO PRACTICO. *Revista Iberoamericana de Automática e Informatica Industrial RIAI*, 64 -72.

Barrientos, Antonio; Peñin , Luis Felipe; Balaguer, Carlos; Aracil, Rafael;. (1997). *FUNDAMENTOS DE ROBOTICA.* Madrir - España: McGraw-Hill.

Blas, M. J., & Loyarte, A. S. (11 - 12 de 10 de 2011). *LABORATORIO VIRTUAL CON ACCESO LOCAL Y REMOTO COMO SIMULACIÓN.* Obtenido de RESEARCH GATE: https://www.researchgate.net/publication/265126268

Caldas Pinto, J. R. (2013). VIRTUAL AND REMOTE LABORATORIES FOR INDUSTRIAL AUTOMATION E-LEARNING. *10th Symposium Advances in Control Education* (págs. 286 - 290). Shefield, UK: The International Federation of Autmatic Control.

Cruz Ramirez, S. R., & Tapias Rodriguez, F. (2016). Diseño y Construcción de una Mano Robótica para el Proceso de Enseñanza – Aprendizaje de la Mecatrónica. *La Mecatrónica en México*, 56 - 62.

Doménech Jara, J. (2020). *TRABAJO FINAL DE GRADO "Diseño, desarrollo y programación de un brazo robot de 6 grados de libertad".* Valencia, España: UNIVERSITAT POLITECNICA DE VALENCIA.

Farias, G. (2006). *DESARROLLO DE LABORATORIOS VIRTUALES, INTERACTIVOS Y REMOTO UTILIZANDO EASY JAVA SIMULATIONS Y MODELOS SIMULINK.* Salvador - Bahia, Brazil: XII Latin Congress on Automatic Control.

Garcia Pozo, J. C. (2014). *PROYECTO FIN DE GRADO - SIMULACIÓN DE LA MANO HUMANA MEDIANTE MATLAB .* Madrir, España: Universidad Carlos III Madrid.

GONZÁLEZ ALZATE, P. F., & OSPINA DUQUE, J. F. (2010). *PROYECTO FIN DE GRADO "ELABORACIÓN DE GUÍA PRÁCTICA DE LABORATORIO ORIENTADA A LA ENSEÑANZA DE LA ROBÓTICA".* Pereira: UNIVERSIDAD TECNOLÓGICA DE PEREIRA.

Guardiola de Leon, D. (2015). Modelamiento en MatLab de prótesis transtibial. *Revista Tekhne, Vol. 2, No. 2 - Univerisidad Distrital Francisco José de Caldas*, 13-22.

Huiqing , Z., Qingtang, L., Yixue , H., & Hong , W. (2018). *The Application and Research on CDIO-Based Project Teaching Method - Taking.* Wuhan - China: 2018 7th International Conference on Industrial Technology and Management.

Kuo, B. C. (1996). *SISTEMA DE CONTROL AUTOMATICO, 7ma Edición.* Mexico: PRNTICE HALL.

Lorandi Medina, A. P. (2011). LOS LABORATORIOS VIRTUALES Y LABORATORIOS REMOTOS EN LA ENSEÑANZA DE LA INGENIERÍA . *Revista Internacional de Educación de Ingeniería*, 24 - 30.

MathWorks Inc. (2021). *MATLAB "Primer".* Massachusetts: MathWorks.

MathWorks Inc. (2021). *MATLAB & SIMULINK "Simulink Getting Started Guide".* Massachusetts: MathWorks.

Melo Becerra, L. A., Ramos Forero, J. E., & Hernadez Santamaria, P. O. (2017). LA EDUCACIÓN SUPERIOR EN COLOMBIA: SITUACIÓN ACTUAL Y ANALISIS DE EFICIENCIA. *Revista de Desarrollo y Sociedad* , 59 - 111.

Mendes Samper, A. (2016). *Trabajo de Grado "Implementación y simulación del algoritmo de sincronización p-PLL mediante un DSP LF2407A.".* Habana - Cuba: Instituo Superior Politecnico Jose Antonio Echeverria.

Ogata, K. (2010). *Ingeniería de Control Moderna.* Madrid: PEARSON.

Pazmiño Espinoza, J. (2016). *PROYECTO FIN DE GRADO - DISEÑO DE UNA MANO ROBOTICA PARA USO DOCENTE.* Madrid, España: Universidad Carlos III Madrid.

Rodriguez Zambrana, J. C. (2012). *TRABAJO FIN DE GRADO "Modelo cinemático y control de un brazo robótico imprimible".* Madrid - España: Universidad Carlos III de Madrid.

Salazar Patin, W. G. (2015). *PROYECTO DE GRADO "DISEÑO DE UNA INTERFAZ DE USUARIO Y CONTROL CINEMATICO DE UN BRAZO ROBOTICO DE 6 GRADOS DE LOBERTAD PARA LA PLANIFICACIÓN DE TRAYECTORIAS EN SOFTWARE MATLAB Y SIMULINK" ".* Guayaquil - Ecuador: Universidad Poletecnica Salesiana.

Solarte Rosas, C. J., & Muñoz Ordoñez, J. E. (15 de Marzo de 2015). Guía para la sintonización de un controlador de Par, Posiion y velocidad de un motor DC de iman permanente. Popayan, Nariño, Colombia.

SOLIDWORKS. (2015). *INTRODUCCIÓN A SOLIDWORKS.* Massachusetts: Dassault Systemes SolidWorks Corporation.