UNIVERSIDAD JOSÉ ANTONIO PÁEZ

VIRTUALIZACIÓN DEL ROBOT LAB-VOLT 5250 DEL LABORATORIO DE ROBÓTICA INDUSTRIAL DE LA UNIVERSIDAD JOSÉ ANTONIO PÁEZ EN ROS

Autores:
Nazar Tovar Sahira Valentina
D'Agostini Rosario Jan Carlo

Urb. Yuma II, calle Nº 3. Municipio San Diego
Teléfono: (0241) 8714240 (master) – Fax: (0241) 8712394

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD JOSÉ ANTONIO PÁEZ
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA MECÁNICA

VIRTUALIZACIÓN DEL ROBOT LAB-VOLT 5250 DEL LABORATORIO DE ROBÓTICA INDUSTRIAL DE LA UNIVERSIDAD JOSÉ ANTONIO PÁEZ EN ROS

Proyecto del Trabajo de Grado para optar al título de

INGENIERO MECÁNICO

Autores:
Nazar Tovar Sahira Valentina
D'Agostini Rosario Jan Carlo
Tutor:
Ing. Wilmer Sanz

San Diego, octubre de 2023

UNIVERSIDAD JOSÉ ANTONIO PÁEZ
COORDINACIÓN DE PASANTÍA Y TRABAJO DE GRADO

ACTA DE APROBACIÓN

INFORME FINAL DE PASANTÍA ☐

TRABAJO DE GRADO ☑

El jurado designado por la Facultad de Ingeniería para la evaluación del **Informe Final de Pasantía o Trabajo de Grado** titulado: Virtualización del Robot Lab-VolT 5250 del laboratorio de robótica industrial de la Universidad José Antonio Páez en Ros

Realizado por el (la) Br. Sahira. V. Nazar Tovar

C.I. Nº 28.142.805 cursante de la carrera de Ing. Mecánica

hace constar después de analizar su contenido y oída la exposición oral, considera que el Informe Final o Trabajo de Grado ha obtenido la calificación de:

APROBADO ☑ **NO APROBADO** ☐

El Jurado

Tutor Académico (Coordinador)
Nombre: Wilmer Saez
C.I.: 7130476

Jurado
Nombre: FREDY BARRACHINA
C.I.:

Jurado
Nombre: Wilton Espinoza
C.I.: 9885895

Fecha: 13 / 11 / 2023

16/11/23

UNIVERSIDAD JOSÉ ANTONIO PÁEZ
COORDINACIÓN DE PASANTÍA Y TRABAJO DE GRADO

ACTA DE APROBACIÓN

INFORME FINAL DE PASANTÍA ☐

TRABAJO DE GRADO ☑

El jurado designado por la Facultad de Ingeniería para la evaluación del **Informe Final de Pasantía o Trabajo de Grado** titulado: Virtualización del Robot Lab-Volt 5250 del laboratorio de robótica industrial de la Universidad José Antonio Páez en Ros

Realizado por el (la) Br. Jan Carlo D'Agostini Rosario

C.I. Nº 25.832.803 cursante de la carrera de Ing Mecánica

hace constar después de analizar su contenido y oída la exposición oral, considera que el Informe Final o Trabajo de Grado ha obtenido la calificación de:

APROBADO ☑ **NO APROBADO** ☐

El Jurado

Tutor Académico (Coordinador)
Nombre: Wilmer Sant
C.I.: 7130496

Jurado
Nombre: FREDY BONILAGAN
C.I.: 11.151.678

Jurado
Nombre: Wiston Espinoza
C.I.: 9885895

Fecha: 13 / 11 / 2023

16/11/23

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD JOSÉ ANTONIO PÁEZ
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA MECÁNICA

CONSTANCIA DE APROBACIÓN PARA LA PRESENTACIÓN PÚBLICA DEL TRABAJO DE GRADO

Quien suscribe, Ing. Wilmer Sanz, portador de la cédula de identidad N° 7.130.496, en mi carácter de tutor del trabajo de grado presentado por la ciudadana **SAHIRA VALENTINA NAZAR TOVAR**, portador de la cédula de identidad N°28.142.805, y por el ciudadano **JAN CARLO D'AGOSTINI ROSARIO**, titular de la cédula de identidad N° 25.832.803, titulado **VIRTUALIZACIÓN DEL ROBOT LAB-VOLT 5250 DEL LABORATORIO DE ROBÓTICA INDUSTRIAL DE LA UNIVERSIDAD JOSÉ ANTONIO PÁEZ EN ROS,** presentado como requisito parcial para optar al título de **INGENIERO MECÁNICO**, considero que dicho trabajo reúne los requisitos y méritos suficientes para ser sometido a la presentación pública y evaluación por parte del jurado examinador que se designe.

En San Diego, a los 26 días del mes de octubre del año dos mil veintitrés.

Ing. Wilmer Sanz
C.I: 7.130.496

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD JOSÉ ANTONIO PÁEZ
FACULTAD DE INGENIERÍA

FI N 006 2023-1CR TG

Valencia, 04 de agosto de 2023

Ciudadanos:
D´AGOSTINI ROSARIO, JAN CARLO
25.832.803
NAZAR TOVAR, SAHIRA VALENTINA
28.142.805
Presente -

Cumplo con informarles que la comisión de Trabajo de Grado y Pasantías de la Facultad de Ingeniería en su reunión **Nº 09-2023** de fecha **14/06/2023** aprobó el proyecto de grado titulado:

Virtualización del robot Lab-Volt 5250 del laboratorio de robótica industrial de la Universidad José Antonio Páez en ROS.

Presentado por ustedes como requisito para optar al título de Ingeniero Mecánico.

Se ratifica la designación del Tutor Académico que lo asesorará en el desarrollo de este proyecto a:
Ing. Wilmer Eduardo Sanz Fernández, titular de la cédula de identidad V- 7.130.496

Atentamente

Facultad de ingeniería UJAP DECANATO

Dra. Laura Aurora Sáenz Palencia
Decana de la Facultad de Ingeniería

c.c. Coordinación de Pasantías y Trabajo de Grado de la Facultad de Ingeniería

AGRADECIMIENTO

Primeramente, le damos Gracias a Dios por habernos permitido culminar de manera exitosa nuestro trabajo de investigación, por estar siempre presente en nuestras vidas y día a día brindarnos la confianza y seguridad en los momentos más difíciles que hayamos podido presentar en nuestra formación.

A nuestros padres, por siempre estar presente en cada paso que damos y brindarnos la seguridad y confianza que nos impulsa día a día. Pero principalmente por todo el amor que fue nuestra motivación e inspiración para seguir adelante y luchar por lo que hoy en día se ha vuelto una realidad para nosotros.

Agradecemos a la Universidad José Antonio Páez por brindarnos la invaluable oportunidad de educación y crecimiento a lo largo de nuestro tiempo en la institución. Nuestra experiencia académica ha sido enriquecedora y queremos reconocer la importancia que ha tenido en nuestro desarrollo personal y profesional. Las oportunidades que hemos tenido aquí han sentado las bases para nuestro futuro y estamos profundamente agradecidos por ello.

No podemos dejar de lado a los profesores que conforman la institución, quienes han impactado de forma significativa en nuestras vidas y desarrollo académico. Sus enseñanzas, orientación y dedicación han dejado una profunda huella en nuestro camino educativo.

Expresamos nuestro agradecimiento al Ing. Wilmer Sanz, nuestro tutor, por su presencia constante durante la realización de nuestro trabajo de grado. Su orientación experta y apoyo desempeñaron un papel crucial en nuestro éxito académico. Valoramos enormemente su contribución a este logro.

A nuestros amigos y futuros colegas, por haber sido parte de este camino, en el cual, con esfuerzo y dedicación, logramos llegar hasta el final. Fueron tiempos difíciles, donde la alegría y las risas nunca faltaron, donde la angustia y las lágrimas fueron testigo de todo nuestro sacrificio y dedicación puesta en nuestra carrera universitaria para lograr el tan anhelado momento de la graduación; sabiendo que con disciplina, perseverancia, trabajo duro y constancia se pueden lograr y alcanzar grandes objetivos, porque triunfar fue, es y seguirá siendo nuestra meta.

Jan Carlo D'Agostini y Sahira Nazar

DEDICATORIA

El presente Trabajo de Grado está dedicado con profundo cariño a mis padres**, Ana Rosario y Roberto D'Agostini**, por su apoyo constante, sus valores, consejos y por motivarme a seguir adelante. Gracias a ustedes, estoy en la recta final de mi carrera estudiantil.

A mi novia, **María Alejandra**, le dedico este logro. Su apoyo incondicional ha sido invaluable para mí.

A mis compañeros de estudio, y en particular a mi compañera de tesis, **Sahira**, quiero expresar mi profundo agradecimiento por su apoyo constante y disposición para ayudar en cada etapa de este trabajo de grado.

A mi familia, no puedo poner en palabras cuánto valoré su apoyo durante mi travesía académica. Siempre estuvieron a mi lado, ofreciéndome apoyo emocional y alentándome a superar cada uno de los desafíos que se presentaron en el camino.

Por último, a todas esas personas que estuvieron a mi lado: mi familia, compañeros de estudio y amigos.

Jan Carlo D'Agostini

DEDICATORIA

Este trabajo está dedicado primeramente a **Dios**, sin él, nada de esto hubiese sido posible. Gracias por todos los días guiarme, acompañarme y nunca desampararme. Te agradezco profundamente por brindarme la fortaleza necesaria para avanzar en mi camino cotidiano y por extender tu mano llena de inquebrantable fidelidad y amor, la cual ha tomado la mía hasta hoy en día.

A mis padres, **Alberto Nazar** y **Magdiel Tovar**. En este momento tan significativo de mi vida, quiero detenerme y expresar mi más profundo agradecimiento por todo lo que han hecho por mí a lo largo de los años. Mi corazón rebosa de gratitud por el amor, la orientación y el sacrificio que han dedicado a mi crecimiento y desarrollo. Desde el primer día que entré en este mundo, ustedes han sido mis guías, mis protectores y mis modelos a seguir. Han sido testigos de mis logros y mis desaciertos y en cada paso del camino, su apoyo inquebrantable ha sido mi mayor fortaleza.

A mi bisabuela **Ángela Guedez**, a mis abuelos **Marina Guedez, Celedonio Tovar, Teresa Escurihuela y Rafael Nazar,** a mis tíos **Oscar, Julio y Douglas Tovar, Lilian Guedez, Elizabeth Nazar, Leonardo y Alida Peña**. Durante este proceso, han estado a mi lado en cada paso, ofreciendo palabras de ánimo que han sido faro de luz en momentos de duda, apoyo emocional y valiosos consejos. Esta tesis no solo representa mi esfuerzo y dedicación, sino también su influencia positiva en mi vida. Cada logro que he alcanzado ha sido posible gracias a la educación y los valores que me han inculcado. Ustedes me han enseñado la importancia del trabajo duro, el esfuerzo y la perseverancia, y eso ha sido fundamental en este camino académico y en mi vida diaria.

A mi novio, **Ricardo Fernández**. Quiero expresar mi sincero agradecimiento por tu apoyo incondicional. Tu presencia constante y tu respaldo han sido fundamentales en este reto académico, y quiero reconocer tu contribución a mi éxito; esta tesis también refleja el apoyo que he recibido de ti. Tus palabras de ánimo, tu paciencia y tu comprensión en cada momento de mi vida, han marcado la diferencia. Saber que puedo contar contigo en momentos importantes como este me ha dado la confianza para enfrentar cada obstáculo en el camino. Gracias por estar a mi lado y por ser una parte esencial de mi vida.

A **María La Veglia**, mi compañera inseparable desde hace muchos años, quien siempre ha estado dispuesta a tenderme una mano amiga y que se ha convertido en una hermana para mí. Gracias por compartir mis momentos de alegría y por ser mi apoyo incondicional en los momentos

difíciles. Esta amistad es un regalo que atesoro y que llevaré en lo más profundo de mi corazón para siempre.

A mi compañero de tesis **Jan D 'Agostini**, tu dedicación, compromiso, trabajo en equipo y disposición para enfrentar este gran reto conmigo han sido fundamentales para llevar a cabo este proyecto. Cada día de colaboración contigo ha sido una fuente de inspiración y aprendizaje. Esta tesis no solo es un reflejo de nuestro esfuerzo conjunto, sino también de la amistad y el compañerismo que hemos compartido a lo largo de este camino. Tu apoyo incondicional y tu contribución inestimable han sido esenciales para alcanzar nuestros objetivos.

Por último, pero ciertamente no menos importante, dedico este logro a una persona muy especial, **Blanca Guedez**, mi ángel y eterna compañera. La persona que dio todo por verme alcanzar esta meta que ambas anhelábamos compartir. Este trabajo va dedicado a ti, como un tributo y en honor a la persona que creyó en mi desde el primer día y luchó incansablemente por verme cumplir mis sueños. Aunque ya no estés físicamente entre nosotros, sé que desde el cielo me brindas la fuerza necesaria para seguir adelante y nunca me abandonas. Tus palabras llenas de sabiduría, tus relatos que enriquecieron mi conocimiento y, sobre todo, tu amor incondicional, dejaron una huella profunda e imborrable en mi corazón. Estoy segura que desde arriba observas y celebras con orgullo mis logros. Mi amor y respeto hacia ti son inquebrantables.

Sahira Nazar

ÍNDICE GENERAL

CONTENIDO **pp.**

ÍNDICE DE CUADROS

LISTA DE FIGURAS

LISTA DE TABLAS

TABLA **DESCRIPCIÓN** **pp.**

REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD JOSÉ ANTONIO PÁEZ
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA MECÁNICA

VIRTUALIZACIÓN DEL ROBOT LAB-VOLT 5250 DEL LABORATORIO DE ROBÓTICA INDUSTRIAL DE LA UNIVERSIDAD JOSÉ ANTONIO PÁEZ EN ROS

Autores: Sahira Valentina Nazar Tovar
Jan Carlo D'Agostini Rosario
Tutor: Prof. Wilmer Sanz
Fecha: Octubre 2023

RESUMEN

La Robótica Industrial se ha convertido en un campo de gran importancia en la actualidad, ya que permite la automatización de procesos y el aumento de la eficiencia y productividad en la industria. Los brazos robóticos industriales son una herramienta clave en este campo, ya que son capaces de realizar tareas repetitivas y precisas con rapidez y exactitud. Sin embargo, el desempeño y rendimiento de estos brazos robóticos dependen en gran medida del sistema operativo que utilizan. Debido a esto se planteó realizar la presente investigación, la cual tiene como propósito la virtualización del robot Lab-Volt 5250 del laboratorio de Robótica Industrial de la Universidad José Antonio Páez en el sistema operativo Robot Operating System (ROS) con el fin de modernizar e incorporar portabilidad al robot. Esta investigación está sustentada bajo las bases teóricas de la Robótica Industrial y la programación, que son dos disciplinas que se complementan en este trabajo. El proyecto está adscrito a la línea de investigación de Avances Tecnológicos en Tecnologías de Información y Comunicación, y la metodología empleada es de enfoque cuantitativo para un proyecto de tipo factible. Todo esto permitió obtener como resultados la virtualización completa del brazo robótico, para ser utilizado en el sistema operativo ROS, posibilitando el acceso a una amplia gama de herramientas y bibliotecas para programar, simular y controlar el robot de manera más portable, eficiente y sencilla.

Descriptores: Robot Industrial, robótica, diseño por ordenador, cibernética, ROS.

INTRODUCCIÓN

La Robótica Industrial es una disciplina que ha ganado importancia en los últimos años, debido a la necesidad de mejorar la eficiencia y productividad de los procesos industriales. Los brazos robóticos industriales son una herramienta clave en la automatización de procesos, y su desempeño y rendimiento dependen en gran medida del sistema operativo que utilizan. En este sentido, la modernización del sistema operativo de los brazos robóticos industriales se ha convertido en un problema importante en la industria y en el ámbito académico, especialmente en las universidades donde estos equipos son utilizados para la formación de futuros profesionales en el campo de la Robótica Industrial. La Universidad José Antonio Páez (UJAP) cuenta con un laboratorio de Robótica Industrial equipado con un brazo robótico, el cual es utilizado para la formación académica. A pesar de esto, el robot está limitado a su propio software y entorno de simulación, lo cual impide a sus usuarios poder experimentar con últimas tecnologías. Por estas razones, el presente trabajo de grado se enfoca en la virtualización del robot Lab-Volt 5250 del Laboratorio de Robótica de la UJAP, cuyo propósito general es desarrollar un modelo virtual del robot en un entorno más portable.

Las teorías en las que se basa el proyecto son la automatización y la computación. El trabajo de grado está estructurado en cuatro (IV) capítulos conformados de la siguiente manera: En el capítulo I, se plantea el problema, se establecen los objetivos, se justifica la realización del estudio, se delimitan los alcances y las limitaciones de este. En el capítulo II, se plantean los antecedentes, así como las bases teóricas que sustentan el desarrollo del proyecto. En el capítulo III, se describe la metodología que será utilizada para el cumplimiento de los objetivos planteados; donde se muestran aspectos como el tipo de investigación, las técnicas y recolección de datos que serán utilizados para llevar a cabo dicha investigación. Finalmente, en el capítulo IV se hace mención a los recursos con los cuales cuenta el proyecto. Este trabajo de investigación contribuirá al desarrollo y avance del brazo robótico, permitiendo una mayor accesibilidad y portabilidad en el entorno de desarrollo utilizado, con la finalidad de mejorar la formación y capacitación de los estudiantes en la UJAP. Además, la interfaz gráfica podrá ser utilizada en futuras investigaciones y proyectos en el campo de la Robótica Industrial.

CAPÍTULO I

EL PROBLEMA

1.1. Planteamiento Del Problema

La robótica, considerada un tema de investigación de gran relevancia en la actualidad, ha emergido como una disciplina multidisciplinaria que combina la ingeniería mecánica, la electrónica, la informática y la inteligencia artificial. Su objetivo principal es el diseño, construcción, programación y operación de robots, máquinas capaces de realizar tareas de manera autónoma o asistida. Si bien la robótica encuentra aplicaciones en diversos ámbitos, destaca especialmente en el sector industrial, donde los robots desempeñan un papel fundamental para mejorar la eficiencia y la productividad. Estas máquinas automatizadas permiten una mayor fabricación y producción de bienes y servicios a nivel mundial, contribuyendo a la optimización de los procesos, la reducción de costos y la mejora de la calidad en la producción industrial.

Los brazos robóticos, también conocidos como brazos articulados, son componentes esenciales en los procesos de automatización industrial. Su versatilidad los convierte en herramientas indispensables capaces de desempeñar una amplia variedad de tareas, desde soldadura y ensamblaje hasta pintura y manipulación de materiales, entre muchas otras. Estos brazos articulados pueden ser programados con precisión para ejecutar movimientos repetitivos a gran velocidad y con una exactitud milimétrica, lo que se traduce en una notoria mejora en la productividad y eficiencia de los procesos industriales. Además de sus ventajas en términos de rendimiento, la utilización de robots en la industria también ha demostrado ser una estrategia efectiva para reducir los costos de producción y elevar la calidad de los productos finales. Mediante su integración en las líneas de producción, los brazos robóticos han revolucionado el panorama industrial al optimizar la eficacia y la rentabilidad de las empresas.

En este contexto, la formación de profesionales en Robótica Industrial juega un papel fundamental en el desarrollo de la industria y la economía en general. Los laboratorios de robótica de las universidades a nivel mundial desempeñan un papel crucial en la preparación de estos profesionales y en la promoción de la investigación en el campo de la Robótica Industrial. Estos laboratorios brindan a los estudiantes la invaluable oportunidad de trabajar con equipos de última generación, tales como brazos robóticos, sensores y software especializado para la programación y control de robots, lo que les permite adquirir experiencia práctica y habilidades técnicas

relevantes para el mercado laboral. Además, los laboratorios de robótica también sirven como entornos de vanguardia para los investigadores, quienes llevan a cabo investigaciones pioneras en el ámbito de la robótica y la automatización, impulsando así el avance de esta disciplina y su impacto en diversas industrias.

Los proyectos de investigación en los laboratorios de robótica son sumamente diversos y abarcan un amplio espectro de áreas de estudio. Estos proyectos van desde la exploración de nuevos algoritmos de control, que buscan mejorar la precisión y eficiencia de los robots, hasta el desarrollo de tecnologías innovadoras que revolucionan la automatización de procesos industriales. Con el fin de lograr resultados exitosos en estas investigaciones, resulta crucial que los laboratorios estén equipados con tecnología de última generación. Esto implica contar con brazos robóticos y otros equipos en óptimas condiciones, capaces de cumplir con los objetivos académicos e investigativos establecidos. Además, la colaboración entre los laboratorios de robótica de diferentes universidades y empresas es esencial para el desarrollo de nuevas tecnologías y la solución de problemas complejos en la industria.

En el orden de las ideas anteriores, los simuladores de robots se han convertido en herramientas fundamentales para el desarrollo e investigación en el campo de la robótica, así como para el aprendizaje en el ámbito educativo. Estos simuladores permiten a los ingenieros recrear de manera precisa un producto o software en un entorno virtual controlado, que reproduce las características del dispositivo o ubicación real para el cual fue creado. De esta forma, se pueden analizar, verificar, corregir y mejorar los productos o procesos en función de los resultados obtenidos en la simulación. Esto reduce significativamente la cantidad de fallos y modificaciones posteriores necesarias una vez que el proceso se lleva a cabo en la realidad, así como los costos y el tiempo perdido asociados.

Sin embargo, uno de los principales desafíos que enfrentan tanto los investigadores como los estudiantes, es la falta de flexibilidad de estos robots para trabajar en entornos distintos a los previstos por el fabricante. El sector comercial de la robótica está dominado por sistemas cerrados que priorizan la dependencia del proveedor en lugar de fomentar la innovación. Cada robot cuenta con su propio sistema operativo y lenguaje de programación, lo que implica que se debe escribir un código que no es reutilizable en otros tipos de robots para programarlos. Mantener esta tecnología basada en sistemas cerrados obstaculiza el progreso en el ámbito académico. Todo esto ha llevado a la necesidad de trabajar en entornos de simulación adecuados y actualizados, que

permitan a los investigadores y estudiantes experimentar con diferentes configuraciones y programaciones de robots en un ambiente controlado y seguro.

En relación con lo anterior, los sistemas operativos utilizados en los brazos robóticos industriales en los laboratorios universitarios pueden presentar problemas de obsolescencia tecnológica. Es decir, es posible que estén utilizando sistemas operativos antiguos que ya no reciben soporte ni actualizaciones por parte del fabricante. Esto puede generar problemas de compatibilidad con los softwares más recientes y limitar la capacidad de los estudiantes para experimentar con las últimas tecnologías. Por lo tanto, trabajar con un robot en un sistema operativo estándar brinda mayor flexibilidad y adaptabilidad a diferentes entornos y aplicaciones. Además, un sistema operativo especializado puede incluir herramientas de depuración y monitoreo que permiten una mejor comprensión del comportamiento del robot y una resolución más rápida de cualquier problema que pueda surgir.

Adicional a los desafíos mencionados anteriormente, otro aspecto crítico en el campo de la Robótica Industrial es la seguridad, en este sentido, los simuladores de robots también pueden ser un recurso valioso para la formación en seguridad. Aunque los robots pueden mejorar la eficiencia y productividad de las empresas, es importante tener en cuenta que al igual que cualquier herramienta industrial, pueden ser peligrosos si no se utilizan de manera adecuada. Por lo tanto, es fundamental que los estudiantes y profesionales de la robótica industrial reciban una capacitación adecuada en el manejo seguro de estos dispositivos para prevenir accidentes y garantizar un ambiente de trabajo seguro. A través de los simuladores, los estudiantes pueden practicar el manejo de los robots en un entorno virtual seguro y controlado, lo que les permite adquirir habilidades y conocimientos sin correr riesgos. Además, los simuladores de robots pueden simular situaciones de emergencia para que los estudiantes aprendan cómo responder de manera segura y efectiva.

En octubre del año 2011, la UJAP introdujo en su laboratorio de Robótica Industrial el servo-robot Lab-Volt 5250, el cual ofrece una capacitación accesible y completa sobre la programación y operación de robots industriales. Sin embargo, con el paso de los años, el entorno operativo del robot ha quedado estancado en las limitaciones impuestas por el fabricante, lo que ha ocasionado diversos problemas, incluyendo la incompatibilidad con nuevas tecnologías. Esta falta de adaptabilidad puede afectar negativamente el proceso de aprendizaje y formación de los estudiantes y profesores involucrados en el laboratorio, por lo que resulta necesario abordarlo diligentemente. La dependencia de sistemas cerrados y la obsolescencia tecnológica de los

sistemas operativos empleados limitan la capacidad de los estudiantes para experimentar con nuevas tecnologías y obstaculizan el avance de la investigación en el campo de la robótica. Es necesario trabajar en soluciones que permitan la integración de nuevos dispositivos y tecnologías en los laboratorios de robótica para fomentar la innovación y el progreso en este campo de estudio.

1.2. Formulación del Problema

Sobre la base de las consideraciones anteriores, surge la necesidad de presentar una solución para evitar la incompatibilidad e inadaptabilidad de robot Lab-Volt de la Universidad José Antonio Páez; por lo tanto, ante la problemática anteriormente expuesta, se plantea la siguiente interrogante:

¿Cómo se podría interactuar con representaciones virtuales del robot Lab-Volt 5250 del laboratorio de Robótica Industrial de la Universidad José Antonio Páez en un entorno de desarrollo más portable al ofrecido por el fabricante Festo?

1.3. Objetivos de la Investigación

1.3.1. Objetivo General

Desarrollar un modelo virtual del robot Lab-Volt 5250 del laboratorio de Robótica Industrial de la Universidad José Antonio Páez en ROS.

1.3.2. Objetivos Específicos

- Estudiar el robot Lab-Volt 5250 y sus herramientas de programación y simulación.
- Analizar información obtenida en foros de discusión con expertos en el manejo del entorno ROS.
- Efectuar el modelado tridimensional del robot Lab-Volt 5250 y de los accesorios disponibles en el Laboratorio de Robótica de la facultad de Ingeniería de la UJAP, en un software portable para ROS.
- Obtener el modelo cinemático del robot Lab-Volt 5250.
- Demostrar el uso y manejo del robot virtual y sus accesorios en la interfaz gráfica de ROS.

1.4. Justificación

De las consideraciones anteriores, se encuentra la simulación como una etapa imprescindible en casi cualquier proceso de producción de un bien o servicio. Aplicado al campo de las tareas robóticas, que es lo que se estudia en este proyecto, la simulación constituye un campo de pruebas donde se pueden recrear entornos similares al real, tanto en aspecto como en

características físicas (gravedad, campo magnético, entre otros). En este sentido, la simulación se convierte en un campo de pruebas que permite a los ingenieros y diseñadores probar y optimizar los robots en un ambiente seguro y controlado antes de ponerlos en funcionamiento en el mundo real.

El sistema operativo ROS se ha convertido en una herramienta altamente utilizada, tanto en el campo de la Robótica Industrial, como en el ámbito académico. ROS es un software libre y de código abierto que permite a los desarrolladores de robots tener acceso a una amplia gama de herramientas y bibliotecas para programar, simular y controlar robots de manera más eficiente y sencilla. Además, ROS es compatible con diversos sistemas operativos y lenguajes de programación, lo que lo hace una herramienta versátil y adaptable a diferentes tipos de robots y proyectos. Al igual que la robótica, ROS viene en ascenso y ya muchas universidades adoptaron en sus cátedras la utilización de este sistema operativo.

En el entorno de simulación ROS, el usuario puede introducir directamente los modelos de los robots con los que pretende trabajar, o bien diseñarlos y construirlos él de forma propia. Así, una vez que se cuenta con todos los modelos y elementos necesarios, se puede ir desarrollando el código de control de los robots, comprobando directamente su resultado en la plataforma de simulación, para luego proceder a su implantación en el entorno real. En la UJAP, específicamente en el laboratorio de Robótica Industrial, se trabaja con distintos proyectos para el aprendizaje en la manipulación y control del brazo robótico Lab-Volt 5250. En este caso, se parte del software que este robot incluye, RoboCIM, que a pesar de que ofrece el entorno necesario para realizar las prácticas de laboratorio, está limitado para trabajar con sistemas servo-robóticos de la serie Lab-Volt, así como sus equipos externos opcionales.

Así pues, este trabajo constituye un aporte al laboratorio de Robótica Industrial de la UJAP y a los estudiantes que cursen dicha materia basándose en el hecho de que la difusión de ROS en el entorno académico es notable y marcadamente creciente en el entorno industrial, lo que deja entrever que en un futuro cercano puede ser un elemento común e indispensable en muchas aplicaciones de la robótica, cada vez más variadas y cubriendo nuevos sectores productivos y de servicios. Evidentemente, la velocidad con que se producen cambios tecnológicos puede provocar la aparición de nuevos estándares en el campo de la robótica, pero, dada su difusión actual, es previsible que ROS marque el camino a seguir, ya sea con sus actuales características o con posibles variantes derivadas de ellas.

1.5. Alcance y Limitaciones

El alcance de este proyecto consiste en conseguir la actualización del robot Lab-Volt 5250. Los modelos tridimensionales efectuados en un software de diseño tridimensional, incluyendo el robot y los accesorios disponibles en el laboratorio de Robótica Industrial de la UJAP, van a ser portables dentro del entorno de simulación de ROS, por lo tanto, no se va a trabajar con un ambiente de simulación nuevo. Este trabajo de investigación también comprende los ensayos pertinentes de programación del robot a través de la programación orientada a objetos hasta conseguir la completa movilización del robot virtual según el modelo cinemático obtenido. De esta manera se conseguirán múltiples beneficios para el laboratorio de Robótica Industrial, estudiantes y profesores de la UJAP.

También resulta oportuno destacar que, aunque ROS es un sistema operativo ampliamente utilizado en robótica, podría tener limitaciones en términos de compatibilidad, soporte y funcionalidad. Además, la integración del robot virtual en el entorno de simulación de ROS podría presentar desafíos, especialmente porque el robot Lab-Volt 5250 utiliza una arquitectura de control diferente. Por otro lado, se ha optado por llevar a cabo el desarrollo del proyecto en el sistema operativo Windows. No obstante, se reconoce que pueden surgir desafíos o dificultades específicas en este entorno. Por tal motivo, se ha considerado una alternativa viable el sistema operativo Linux. De esta manera, se busca asegurar una mayor versatilidad y compatibilidad.

CAPÍTULO II

MARCO TEÓRICO

El marco teórico proporciona el contexto teórico y conceptual del problema de investigación que se aborda. Se trata de una revisión crítica y sistemática de la literatura relevante relacionada con el tema en cuestión. Palella y Martins (2006, p.67) definen al marco teórico como el soporte principal del estudio, y proponen que a través de él "se amplía la descripción del problema, pues permite integrar la teoría con la investigación y establecer sus interrelaciones". Es así como el marco teórico proporciona una comprensión profunda del problema, identificando los conceptos clave, las teorías y los enfoques metodológicos utilizados por otros investigadores.

2.1. Antecedentes

Los antecedentes suministran una descripción detallada de los estudios previos y relevantes que se han realizado sobre el tema en cuestión. Arias (2012, p.106) expone que "los antecedentes reflejan los avances y el estado actual del conocimiento en un área determinada y sirven de modelo o ejemplo para futuras investigaciones". También establece que los antecedentes pueden ser trabajos y tesis de grado, trabajos de ascenso, artículos e informes científicos relacionados con el problema planteado, entre otros. En conclusión, los antecedentes abarcan toda investigación realizada anteriormente y que guarde alguna investigación con el proyecto a desarrollar.

Granda (2022), realizó un trabajo de investigación titilado **"Propuesta de diseño de un brazo robótico industrial para proceso de soldadura SMAW"**. Este proyecto fue realizado como requisito para optar al título de Ingeniero Mecánico en la Universidad José Antonio Páez de Valencia, Venezuela. Este trabajo de grado tuvo como objetivo diseñar un brazo mecánico capaz de realizar eficazmente el proceso de soldadura SMAW, para ello se realizó un estudio de los pasos que se llevan a cabo en los procesos de soldadura, para así determinar los requisitos necesarios en el diseño del brazo robótico y seleccionar la opción más viable. Esta investigación constituye un gran aporte en los procesos de soldadura, ya la implementación de un brazo robótico en los procesos de soldadura, aumenta significativamente el rendimiento de la operativa. A diferencia de un operario, los brazos robóticos pueden repetir la misma tarea todo el tiempo que sea necesario como también optimizar y dotar de eficiencia a los procesos complejos ya que poseen una gran precisión disminuyendo así el margen de error.

Como resultado en la automatización de este proceso, se disminuirán los costos de operación debido a enfermedades inesperadas o urgentes por seguridad industrial, se minimizarán los costos de operación y mantenimiento; aumentando de esta manera los beneficios de la actividad industrial en la empresa, también se disminuirán problemas de calidad y desperdicios, permitiendo una producción continua para cumplir con los tiempos de entrega de los clientes. Esta investigación fue un proyecto factible con diseño de campo, documental y experimental ya que se observó el fenómeno de soldadura en su contexto natural con la finalidad de obtener información respecto al mismo. Los datos se recolectaron a través de la observación y la revisión documental de los distintos tipos de robots industriales, haciendo énfasis en el robot antropomórfico, en el cual se basó el diseño del brazo robótico. En este contexto, el cálculo de la cinemática directa del brazo robótico a través de los parámetros de Devanit – Hartenberg, al igual que todo el diseño tridimensional del brazo, contribuyen al presente trabajo de investigación.

De igual forma, Peña (2021), realizó un trabajo de investigación titulado **"Control de brazo robótico mediante ROS en plataformas de bajo coste"**. Este trabajo de investigación fue realizado como requisito para optar por un Máster Universitario en Automática y Robótica en la Universidad de Alicante, España. Se planteó como objetivo que el robot SNAM6800 sea capaz de realizar la tarea cotidiana de servir cafés en una oficina de la empresa IRYS Control a través de un brazo robótico que sea capaz de cumplir con esta tarea. Para controlar los movimientos del brazo robótico se utilizó la plataforma ROS, integrada en un dispositivo de bajo coste. Asimismo, se implementó un sistema de supervisión que permitió monitorizar aspectos relevantes del sistema, como la cantidad de agua o cápsulas disponibles.

Este proyecto surge como respuesta al auge de la automatización, especialmente en el campo de la domótica y los robots de servicio. Actualmente, la mayoría de los fabricantes de robots están invirtiendo en la adaptación de sus brazos y robots al mundo del servicio, lo que ha permitido grandes avances en este ámbito. En este contexto, el software ROS resulta muy útil para la simulación e implementación de dispositivos reales. Como resultado de este proyecto, se logró instalar un brazo robótico en una plataforma de bajo coste como es la BeagleBone Black. La interfaz realiza su función de control y monitorización del sistema. La BeagleBone Black ha sido capaz de ejecutar varias tareas a la vez, como son ROS, OpenVPN y las comunicaciones TCP/IP. Mostrando que es posible instalar un sistema robótico en una plataforma de bajo coste como puede ser esta BeagleBone. En el trabajo se pueden encontrar conceptos y técnicas sobre el control de

brazos robóticos, así como una descripción detallada de cómo se implementó ROS en plataformas de bajo costo. También se proporcionan detalles sobre la configuración del hardware y software necesarios para llevar a cabo este tipo de proyecto. Los resultados obtenidos en este proyecto pueden ser utilizados para mejorar la eficiencia y precisión del robot.

De manera similar, el trabajo de grado de Labrador y Romero (2020) es un proyecto de investigación titulado: **"Simulación de un robot hexápodo con ROS y Gazebo"**. Este trabajo fue presentado como requisito para optar por el título de Ingeniero en Automatización en la Universidad de Bogotá, Jorge Tadeo Lozano. El objetivo principal de la investigación llevada a cabo por Labrador y Romero fue el desarrollo de guías educativas para la programación de robots, utilizando como base el robot hexapobot y el entorno ROS, así como su respectiva simulación en el software Gazebo. Este proyecto fue motivado por la creciente importancia de la robótica como tema de investigación en la actualidad, así como por la popularidad de ROS en este campo. Es importante destacar que varias universidades ya han adoptado el uso de este framework en sus cátedras.

Los autores concluyeron que la simulación es un aspecto esencial en proyectos de robótica, ya que permite reducir los recursos necesarios para el proyecto (como tiempo y dinero) y reduce el riesgo de dañar el hardware del robot. Además, señalaron que la principal ventaja de utilizar ROS es que es una plataforma open source, lo que significa que existen numerosos paquetes desarrollados por la comunidad que se pueden utilizar en el proyecto para agregar valor, y robustez. Asimismo, la comunidad de ROS es bastante amplia y está en constante crecimiento, por lo que ofrece soporte a cada uno de los paquetes desarrollados. El trabajo de Labrador y Romero representa un importante aporte al campo de la robótica educativa, ya que facilitan el proceso de aprendizaje a través de guías educativas. Al utilizar la información proporcionada por estos autores, se podrá tener un enfoque práctico y detallado en la simulación de sistemas robóticos utilizando ROS, lo que permitirá desarrollar un proyecto educativo de alta calidad. Es una valiosa herramienta para proyectos educativos en el área de la robótica y la automatización.

Igualmente, el trabajo de grado de Escobar (2019) **titulado "Diseño de sistemas de control industrial de robots basados en la industria 4.0"**, fue presentado como requisito para obtener el título de Ingeniero Industrial en Procesos de Automatización en la Universidad Técnica de Ambato, Quito. Esta investigación tuvo como objetivo implementar métodos de control utilizando tarjetas empotradas de bajo costo para el brazo robótico manipulador Kuka youBot. Para

esto se realizó una simulación habilitó la comunicación a través de ROS, lo que permitió el envío de mensajes hacia el robot simulado de manera similar al sistema de control desarrollado para el brazo robótico y de forma efectiva. La relevancia del diseño radica en que proporciona a investigadores y empresas dedicadas a la robótica más opciones para el control de robots y la programación de sus aplicaciones, sin tener que encontrarse con problemas en la compatibilidad del hardware, contando además con una amplia variedad de herramientas como bibliotecas y paquetes que se encuentran disponibles para las tarjetas empotradas.

Esta investigación fue un proyecto factible con diseño de campo, documental y experimental. Los datos se recolectaron directamente del robot en estudio, además de obtener información a través de artículos científicos, tesis, libros y páginas web. Se realizaron pruebas de funcionamiento del sistema de control implementado sobre el brazo robótico del Kuka youBot hasta obtener el funcionamiento correcto. El aporte de la investigación realizada por Escobar radica en la utilización de herramientas de software libre y de código abierto, así como en la implementación de tecnologías avanzadas como parte de la Industria 4.0. El trabajo de Escobar es relevante porque proporciona una solución viable y de bajo costo para el control de robots utilizando tarjetas empotradas, lo que puede permitir una mayor accesibilidad y difusión de la robótica en diferentes ámbitos. Además, la utilización de tecnologías avanzadas como parte de la Industria 4.0 puede mejorar la eficiencia y precisión del control de robots, permitiendo su integración en procesos industriales complejos y avanzados.

Por último, Morales, Hoyos, García, J. (2019). Egresados de la Universidad nacional experimental del Táchira, Venezuela. Realizaron una investigación titulada **“Diseño y optimización de la estructura mecánica de un brazo robótico antropomórfico desarrollado con fines educativos”** para optar al título de Ingeniero Mecánico. El objetivo fue presentar un diseño de la estructura mecánica del brazo robótico de 4 grados de libertad (GDL) para cubrir parte de las necesidades presentes en el Laboratorio de Prototipos de la Universidad Nacional Experimental del Táchira, que requería de brazos robóticos de bajo presupuesto para desarrollar sus proyectos de investigación tales como: desarrollo y mejoramiento de sistemas de control monoarticular y multiarticular para seguimiento de trayectorias, control de vibraciones y desarrollo de pequeñas celdas de trabajo utilizando el brazo como elemento actuador. Adicionalmente, se planeaba utilizarlo en prácticas de laboratorio como: caracterización del modelado cinemático y

dinámico del robot, seguimiento de trayectorias en el espacio cartesiano, programación y definición de la precisión, repetitividad y resolución del manipulador.

Se presentó el diseño de un brazo robótico ejecutado siguiendo una metodología que incluyó la definición de doce especificaciones de diseño fundamentadas en las necesidades del cliente. A partir de ello se generaron seis conceptos de diseño los cuales se valoraron de acuerdo a dos evaluaciones: una cualitativa y otra cuantitativa, obteniendo el concepto que mejor satisfacía las necesitadas del cliente planteadas inicialmente. Por otro lado, se hizo el modelado cinemático directo utilizando la formulación de Denavit–Hartenberg y el modelado cinemático inverso utilizando el método geométrico. Las ecuaciones obtenidas fueron validadas comparando los resultados de su aplicación con aquellos obtenidos utilizando simulación computacional que permitía configurar el modelo CAD.

2.2. Bases Teóricas

2.2.1. Teoría Central de la Investigación

El trabajo de grado se basa en dos teorías centrales: la teoría de la automatización y la teoría de la computación. La teoría de la automatización proporciona los fundamentos para el diseño y la implementación de sistemas que realizan tareas de manera autónoma. Por su parte, la teoría de la computación se enfoca en la creación de un modelo virtual del brazo, lo que permite simular su comportamiento y realizar pruebas antes de su implementación práctica. Estas teorías se combinan para desarrollar un sistema robótico virtual eficiente y funcional. La virtualización e implementación del robot Lab-Volt 5250 del laboratorio de Robótica Industrial de la UJAP en ROS puede aportar en la formación de los estudiantes de Robótica Industrial, así como a la investigación y desarrollo de tecnologías y soluciones.

2.2.2. Robótica Industrial

La Robótica Industrial es una disciplina de la ingeniería que se enfoca en el diseño, fabricación y control de robots que pueden realizar tareas específicas en ambientes industriales. Desde su aparición a mediados del siglo XX, la Robótica Industrial ha ganado especial importancia por su impacto significativo en la industria manufacturera, ya que permite la automatización de procesos repetitivos y peligrosos, mejorando la calidad y precisión de los productos, aumentando la eficiencia y productividad. En la actualidad, la Robótica Industrial tiene una gran variedad de aplicaciones en distintos sectores, como la automoción, la electrónica, la alimentación, la farmacéutica, la aeronáutica, entre otros.

2.2.2.1. Robots Industriales (RI)

Los robots industriales son el elemento más común e importante en la Robótica Industrial, considerando que son una herramienta clave en la automatización de procesos. Según D'Addario se puede considerar un robot industrial como:

> Una máquina de manipulación automática reprogramable y multifuncional con tres o más ejes que pueden posicionar y orientar materias, piezas, herramientas o dispositivos especiales para la ejecución de trabajos diversos en las diferentes etapas de la producción industrial, ya sea en una posición fija o en movimiento. (2016, p. 44)

Hay que mencionar, además, la definición oficial por parte de la Organización Internacional de Estándares (ISO, por sus siglas en inglés) que define al robot industrial como "un manipulador multipropósito reprogramable y controlado automáticamente, programable en tres o más ejes, que puede fijarse en su lugar o fijarse a una plataforma móvil para su uso en aplicaciones de automatización en un entorno industrial" (ISO 8373:2021).

2.2.2.2. Clasificación de los Robots Industriales

De acuerdo con la Federación Internacional de Robótica (IFR) los robots industriales se pueden clasificar según su estructura mecánica como se describe en el cuadro 1.

Cuadro 1: Clasificación de los robots industriales.

Tipo	Elementos y Funcionamiento
Robot cartesiano	Manipulador que tiene tres articulaciones prismáticas, cuyos ejes forman un sistema de coordenadas cartesianas.
Robot SCARA	Manipulador que tiene dos juntas rotativas paralelas para proporcionar cumplimiento en un plano seleccionado.
Robot articulado	Manipulador que tiene tres o más juntas rotativas.
Robot Parallelo/Delta	Manipulador cuyos brazos tienen enlaces que forman una estructura de bucle cerrado.
Robot cilíndrico	Manipulador que tiene al menos una junta rotatoria y al menos una junta prismática, cuyos ejes forman un sistema de coordenadas cilíndrico.
Robot polar (robot esférico)	Manipulador que tiene dos articulaciones rotatorias y una prismática, cuyos ejes forman un sistema de coordenadas polares.

Fuente: Federación Internacional de Robótica (2022)

2.2.2.3. Estructura de los Robots Industriales

Los RI están compuestos por una serie de elementos que les permiten llevar a cabo tareas específicas de acuerdo con las necesidades del proceso de producción. A continuación, se describen algunos de los componentes principales de los robots industriales:

a) Brazo mecánico: El brazo mecánico es el elemento que realiza los movimientos necesarios para llevar a cabo las tareas asignadas.
b) Actuadores: Los actuadores son los elementos que permiten convertir la energía eléctrica en movimiento mecánico.
c) Sensores: Los sensores son elementos que permiten a los robots industriales recopilar información sobre su entorno y ajustar su comportamiento en consecuencia.
d) Controladores: Los controladores son los dispositivos que permiten programar y controlar el comportamiento del robot industrial.
e) Herramienta terminal: La herramienta es el elemento que se coloca en el extremo del brazo mecánico y que permite al robot interactuar con los objetos en su entorno.

Para simplificar, la estructura de los robots industriales está diseñada para permitir el control preciso de los movimientos del brazo mecánico, la activación de los sensores y actuadores y la interacción con objetos en el entorno de producción. Todos estos elementos están integrados en un entorno de desarrollo que se puede programar y controlar para realizar una variedad de tareas específicas, de modo que dependen en gran medida del sistema operativo que utilizan.

2.2.3. Robot Lab-Volt 5250

El sistema servo robot Lab-Volt serie 5250 es un robot educativo desarrollado por la compañía Lab-Volt de Festo (ver figura 1). Este robot está diseñado para ser utilizado en entornos educativos y de formación como laboratorios, para enseñar a los estudiantes sobre Robótica Industrial y su programación, además de brindarles una experiencia práctica y enriquecedora en este mismo ámbito.

Este sistema incluye un robot servoaccionado de 5 grados de libertad con controlador integrado, un panel de control, una estación de programación y un conjunto de herramientas de programación y simulación. Como se expresa en el Manual de usuario del robot “el servo robot cuenta con cinco ejes de rotación, más una pinza, y es capaz de utilizar todas las articulaciones simultáneamente para realizar una determinada secuencia de movimientos” (2011, p. 1).

Figura 1. Robot Lab-Volt 5250

Fuente: Festo Didactic Inc

2.2.3.1. Características del robot Lab-Volt 5250

- Servomotores con alto par.
- Construcción de acero y aluminio.
- Tiene una carga útil máxima de 1.5 kg (10 lb) y un alcance máximo de 480 mm (18,9 in).
- Velocidad Máxima: 584 mm/s (23 in/s)

2.2.4 Sistemas operativos

Los sistemas operativos son el software fundamental que gestiona y controla el hardware de una computadora, permitiendo que los programas se ejecuten y se comuniquen con los recursos del sistema. Además, gestionan la asignación de recursos del sistema, como memoria y CPU, para garantizar un rendimiento óptimo de las aplicaciones ROS, lo que es fundamental en entornos robóticos en tiempo real. En adición, los sistemas operativos proporcionan medidas de seguridad y estabilidad para prevenir fallos y asegurar la integridad del sistema robótico. Por otro lado, la mayoría de los paquetes y bibliotecas de ROS están diseñados y probados en sistemas operativos específicos, lo que asegura que funcionen correctamente en entornos compatibles. Resumiendo todo lo anteriormente descrito, los sistemas operativos son la base sobre la que ROS y las aplicaciones robóticas se ejecutan efectivamente, garantizando un entorno coherente y confiable para el desarrollo y operación de sistemas robóticos.

2.2.4.1 Windows

Windows es un sistema operativo desarrollado por Microsoft que es ampliamente utilizado en computadoras personales y servidores en todo el mundo. La compatibilidad de ROS con Windows directamente, sin el uso de WSL (Windows Subsystem for Linux) o máquinas virtuales, ha sido un desafío en el pasado debido a diferencias fundamentales en la arquitectura y herramientas entre Windows y los sistemas Linux. ROS ha estado principalmente orientado a sistemas basados en Linux. Aunque ha habido esfuerzos para mejorar la compatibilidad de ROS con Windows, como ROS 2, que tiene una mejor integración con Windows, aún es más común y recomendable utilizar un entorno Linux para desarrollar con ROS. Esto se debe a que muchas de las bibliotecas, herramientas y paquetes de ROS están diseñados y optimizados para sistemas Linux, lo que facilita el desarrollo y la resolución de problemas.

2.2.4.2 Virtual Box

VirtualBox es una aplicación de virtualización que permite a los usuarios crear máquinas virtuales (VM) en sus sistemas para ejecutar sistemas operativos adicionales. Estas máquinas virtuales son ambientes virtuales aislados y completamente funcionales que operan como si fueran una computadora independiente dentro de la computadora anfitriona. Los sistemas operativos adicionales que se ejecutan en estas máquinas virtuales pueden ser diferentes de la plataforma del sistema anfitrión.

Utilizar VirtualBox con ROS permite tener un entorno de desarrollo ROS independiente y aislado en tu sistema anfitrión. Esto puede ser útil si se desea trabajar con ROS en una plataforma que no sea compatible con ROS de manera nativa o si se prefiere mantener un sistema anfitrión sin cambios para otros propósitos. Sin embargo, se debe tener en cuenta que el rendimiento de ROS en una máquina virtual puede ser inferior al de una instalación nativa, especialmente en aplicaciones que requieren alta velocidad de procesamiento o interacción con hardware físico.

2.2.4.3 WSL

WSL, o Windows Subsystem for Linux, es una característica de Windows que permite ejecutar un sistema operativo Linux en paralelo con Windows. Esto significa que se puede tener un entorno Linux en una máquina con Windows, lo que es particularmente útil si se desea utilizar herramientas y software diseñados originalmente para Linux, como ROS (Robot Operating System). La ventaja de utilizar WSL con ROS es que se puede ejecutar ROS directamente en el sistema Windows sin la necesidad de una máquina virtual o un sistema dual. Sin embargo, hay que

tener en cuenta que la comunicación entre aplicaciones ROS en WSL y Windows puede requerir configuraciones adicionales y no es tan transparente como en un entorno Linux nativo.

2.2.5 Entornos de desarrollo

Como se ha venido mencionando, la Robótica Industrial es una rama de la ingeniería que se enfoca en el diseño, programación y control de brazos robóticos industriales utilizados en la automatización de procesos en la industria. Uno de los aspectos más importantes en esta área es el desarrollo de software y hardware para el control y programación de los robots. Entre los entornos de desarrollo de software utilizados para la programación de la Robótica Industrial se encuentra a Octave, el cual es un lenguaje de programación y un entorno de desarrollo numérico de código abierto.

Octave ofrece herramientas para la simulación y el control de robots, permitiendo el desarrollo de modelos matemáticos para el control de los robots y la simulación de su comportamiento en diferentes situaciones. Como alternativa gratuita y de código abierto, Octave brinda a los desarrolladores acceso a una amplia gama de funciones y bibliotecas para el procesamiento de señales, cálculo numérico y análisis de datos, siendo una opción popular en el campo de la Robótica Industrial.

Otro software recurrente es el LabVIEW que se utiliza en la Robótica Industrial. Es un sistema de programación gráfica utilizado para la automatización de procesos y el control de dispositivos electrónicos, incluyendo brazos robóticos. LabVIEW permite la creación de interfaces gráficas para el control y monitoreo de los robots, así como también la simulación de su comportamiento. De igual forma Python es otro lenguaje de programación muy utilizado en la Robótica Industrial. Es un lenguaje fácil de aprender y su sintaxis es simple, lo que lo hace muy útil para el desarrollo de aplicaciones robóticas.

2.2.5.1. ROS

ROS (Robot Operating System) es un sistema operativo de código abierto diseñado específicamente para robots. Fue desarrollado por la empresa de robótica estadounidense Willow Garage en 2007 y actualmente es mantenido por la Fundación ROS. Este entorno operativo se ha convertido en una de las herramientas más utilizadas en la Robótica Industrial debido a su flexibilidad, modularidad y capacidad para integrar diferentes componentes de hardware y software. En ROS, se ofrece un conjunto de bibliotecas, herramientas y convenciones para el desarrollo de software para robots, que permite el control, programación y simulación de robots.

Otro rasgo de ROS, es que los desarrolladores pueden crear programas y algoritmos para el control de los robots, así como también simular su comportamiento en diferentes situaciones y entornos.

ROS está diseñado para trabajar con diferentes plataformas y sistemas operativos, lo que lo hace compatible con una amplia variedad de robots y dispositivos electrónicos. Además, ROS permite la integración de diferentes componentes de hardware y software, como sensores, actuadores, cámaras y algoritmos de visión, lo que lo hace muy útil para el desarrollo de aplicaciones robóticas complejas. Una de las principales ventajas de ROS es su comunidad de desarrollo activa y comprometida, donde durante más de 10 años, el proyecto ROS ha producido un vasto ecosistema de software para robótica al nutrir una comunidad global de millones de desarrolladores y usuarios que contribuyen y mejoran dicho software.

A lo largo de su evolución, ROS ha pasado por varias versiones, cada una con sus propias características, mejoras y enfoques. Estas versiones reflejan el continuo desarrollo de la plataforma de robótica de código abierto y sus esfuerzos por mejorar la eficiencia, la estabilidad y la usabilidad. Desde ROS Electric Emys hasta ROS Noetic Ninjemys, estas iteraciones han sido fundamentales para satisfacer las necesidades de la creciente comunidad robótica y proporcionar soluciones sólidas en sistemas operativos específicos. A continuación, se describirán estas versiones de ROS 1:

- ROS Electric Emys (2011): Fue una versión temprana de ROS 1 que marcó un hito en la estabilidad y la interoperabilidad. A medida que ROS ganaba tracción en la comunidad de robótica, Electric Emys ofreció una base más sólida para el desarrollo de aplicaciones robóticas. Se centró en la mejora de la arquitectura y la compatibilidad de las bibliotecas.
- ROS Fuerte (2012): Continuó construyendo sobre el trabajo de sus predecesores. Su enfoque se centró en la mejora de la instalación, documentación y compatibilidad con múltiples plataformas. Introdujo nuevas bibliotecas y características para el desarrollo de aplicaciones robóticas.
- ROS Groovy Galapagos (2012): Puso énfasis en la interoperabilidad, introduciendo nuevas características como ROS Java y mejorando la documentación. Estas mejoras permitieron a los desarrolladores de robótica trabajar más eficazmente con componentes de ROS y aprovechar su flexibilidad.
- ROS Hydro Medusa (2013): Trajo importantes mejoras en eficiencia y rendimiento. Esto fue fundamental para las aplicaciones robóticas, ya que permitió un funcionamiento más

suave de los sistemas. Además, incluyó nuevas bibliotecas y características que hicieron que el desarrollo en ROS fuera aún más efectivo.

- ROS Indigo Igloo (2014): Mejoró aún más la calidad del software y la facilidad de uso. ROS se convirtió en una plataforma más madura, con numerosas bibliotecas y herramientas que facilitaron el desarrollo de aplicaciones robóticas más avanzadas.
- ROS Jade Turtle (2015): Representó una versión de transición en el desarrollo de ROS. Aunque no introdujo cambios revolucionarios, fue esencial para estabilizar la plataforma y mejorar la documentación, allanando el camino para futuras iteraciones. Jade fue compatible principalmente con sistemas Ubuntu y ayudó a fortalecer la base de ROS.
- ROS Kinetic Kame (2016): Se convirtió en la versión estándar de ROS para muchas aplicaciones robóticas. Se centró en la eficiencia, la calidad y la facilidad de uso. Kinetic brindó un soporte a largo plazo (LTS) y fue especialmente relevante para la robótica en sistemas Ubuntu 16.04. Durante su tiempo, se convirtió en la elección principal para la comunidad de robótica.
- ROS Lunar Loggerhead (2017): Continuó la evolución de ROS 1 con mejoras en eficiencia y calidad del código. Diseñado principalmente para sistemas Ubuntu, ofreció a la comunidad robótica una plataforma confiable y estable. La versión Lunar contribuyó a la consolidación de ROS como el marco de trabajo preferido para la programación robótica en diversos dominios.
- ROS Melodic Morenia (2018): Se centró en la estabilidad y calidad del código. Diseñado para funcionar en sistemas Ubuntu 18.04 LTS, proporcionó una base sólida para una amplia variedad de proyectos de robótica. La comunidad robótica confió en Melodic para sus aplicaciones debido a su naturaleza LTS.
- ROS Noetic Ninjemys (2020): Fue la última versión oficial de ROS 1. Diseñado para funcionar en Ubuntu 20.04 LTS, ROS Noetic enfatizó la estabilidad y la calidad del código. Aunque marcó el fin de la serie ROS 1, sigue siendo una elección sólida para aquellos con proyectos en sistemas Ubuntu 20.04 que requieren soporte a largo plazo.

ROS 2, la evolución de ROS 1, se perfila como el futuro del desarrollo robótico y la automatización. A medida que los sistemas robóticos y autónomos se vuelven más sofisticados y se integran en diversos campos, ROS 2 se presenta como la respuesta a los desafíos en evolución. Con su enfoque en la escalabilidad, la comunicación eficiente y la seguridad, ROS 2 se está

convirtiendo en la elección preferida para una amplia gama de aplicaciones robóticas, desde la manufactura automatizada hasta la exploración espacial. Su capacidad para operar en múltiples plataformas y entornos, junto con su creciente ecosistema de paquetes, lo convierte en un candidato sólido para liderar la próxima generación de soluciones robóticas.

2.2.5.2. RViz

RVIZ es una herramienta fundamental en el desarrollo de robots y aplicaciones en el contexto de ROS. Se trata de un visor 3D interactivo que proporciona a los ingenieros y desarrolladores de robótica una interfaz gráfica para visualizar datos, modelos y representaciones en tiempo real de robots y entornos simulados. Su flexibilidad y capacidad para la visualización y análisis de datos lo convierten en una herramienta esencial para la depuración, pruebas y desarrollo de sistemas robóticos. Una de las características más notables de RViz es su capacidad para mostrar una amplia variedad de tipos de datos, incluyendo nubes de puntos, mallas 3D, datos de sensores, trayectorias de robots y modelos cinemáticos. Esto permite a los desarrolladores inspeccionar de manera eficiente la información generada por sensores, así como la interacción de un robot con su entorno.

RViz es altamente personalizable y ofrece una amplia gama de paneles de visualización que se pueden configurar para adaptarse a las necesidades específicas de un proyecto. Los usuarios pueden crear configuraciones personalizadas para mostrar información relevante y ocultar elementos innecesarios, lo que facilita el enfoque en áreas específicas del sistema robótico en desarrollo. Además, RVIZ es valioso en la etapa de depuración, ya que permite identificar problemas en el comportamiento del robot al proporcionar una visión detallada de su estado y acciones. Los ingenieros pueden rastrear el movimiento del robot, verificar las salidas de los sensores y observar cómo el robot interactúa con su entorno, lo que facilita la identificación de errores o desviaciones inesperadas en el comportamiento.

2.2.5.3. Gazebo

Gazebo es un potente simulador 3D de código abierto ampliamente utilizado en el campo de la robótica y la automatización. Este software desempeña un papel crucial en la simulación, prueba y validación de robots y entornos virtuales, siendo esencial para el desarrollo de sistemas robóticos. Una de las características más destacadas de Gazebo es su capacidad para crear modelos precisos de robots, incluyendo aspectos como cinemática, dinámica y sensores. Además, permite

diseñar entornos virtuales realistas que simulan situaciones del mundo real. Esto es invaluable para probar algoritmos y comportamientos de robots antes de la implementación física.

La integración de Gazebo con ROS es una de sus ventajas más importantes. Esto facilita la simulación y control de robots que utilizan ROS, lo que se ha convertido en un estándar en la industria de la robótica. La integración nativa con ROS permite a los desarrolladores probar y depurar algoritmos de control, planificación de trayectorias y percepción en un entorno de simulación antes de ejecutarlos en un robot real. Otro aspecto fundamental de Gazebo es su amplia variedad de sensores compatibles. Los usuarios pueden simular cámaras, lidar, sensores inerciales y más, lo que les permite probar y calibrar algoritmos de percepción utilizando datos generados en la simulación. La interfaz gráfica 3D de Gazebo es esencial para visualizar la simulación en tiempo real, lo que ayuda a comprender cómo el robot interactúa con su entorno y a identificar problemas de colisión o comportamiento anómalo.

La personalización es una característica destacada de Gazebo. Los usuarios pueden adaptar la simulación agregando modelos de robots, entornos y objetos personalizados. Además, pueden implementar controladores específicos para los robots y ajustar los parámetros de simulación según sea necesario. La robustez y escalabilidad de Gazebo son notables, ya que puede manejar simulaciones de robots complejas y entornos detallados, lo que lo hace adecuado para una amplia variedad de aplicaciones, desde la investigación en robótica hasta la simulación de sistemas de automatización industrial.

2.2.5.4. MoveIt

MoveIt es un conjunto integral de herramientas de código abierto diseñado para facilitar la planificación y control de movimiento en robots. Fue desarrollado específicamente para robots móviles y manipuladores robóticos y se ha convertido en un componente crucial en la industria de la robótica, ya que ofrece soluciones para una variedad de tareas relacionadas con el movimiento y la interacción de robots. Una de las principales características de MoveIt es su capacidad para planificar trayectorias de movimiento seguras y eficientes. Los algoritmos de planificación de movimiento incluidos permiten a los usuarios definir trayectorias para sus robots teniendo en cuenta restricciones de movimiento y obstáculos en el entorno. Esto es fundamental para garantizar la seguridad en aplicaciones robóticas y evitar colisiones.

MoveIt proporciona una interfaz intuitiva para la programación de movimientos, lo que facilita la creación de secuencias de movimiento y tareas específicas. Los usuarios pueden diseñar

fácilmente acciones y comportamientos complejos para sus robots sin necesidad de conocimientos avanzados de programación. El sistema de percepción de MoveIt también es destacable, ya que puede integrarse con varios sensores, como cámaras y lidar, para detectar objetos y obstáculos en el entorno del robot. Esto permite una planificación de movimiento más precisa y una mayor autonomía en robots móviles.

2.2.6 Estudio de la cinemática directa

El objetivo principal al realizar un análisis de cinemática directa es determinar la posición y la orientación del extremo del manipulador en relación con cada posición articular. En otras palabras, partiendo de la posición inicial del efector final del robot, aplicamos movimientos de traslación y rotación que dependen únicamente de la configuración específica de las articulaciones, y luego tratamos de determinar la posición final. Por lo tanto, el objetivo es encontrar una matriz de transformación T que pueda calcular los resultados de las rotaciones y traslaciones realizadas por cada articulación. Esta matriz se basa en parámetros conocidos como parámetros D-H.

2.2.6.1. Parámetros Denavit-Hartenberg (D-H)

Los parámetros de Denavit-Hartenberg (D-H) son un conjunto de valores utilizados en robótica para describir la geometría y la cinemática de un manipulador. Estos parámetros definen las características de cada articulación y la relación entre ellas, permitiendo calcular la matriz de transformación que relaciona la posición y orientación del extremo del manipulador con las variables de articulación. Los parámetros D-H incluyen los siguientes parámetros:

a: Representa la longitud del brazo entre dos articulaciones consecutivas, medida a lo largo del eje perpendicular al giro. Indica la longitud del enlace o brazo del robot y se mide en unidades de longitud.

θ: Este parámetro representa la rotación alrededor del eje común entre dos eslabones consecutivos. Indica la orientación relativa de un eslabón con respecto al anterior. La rotación se mide en radianes.

d: Representa la distancia a lo largo del eje común entre dos eslabones consecutivos. Indica la traslación a lo largo del eje común y se mide en unidades de longitud.

α: Este parámetro representa la rotación alrededor del eje perpendicular al brazo entre dos eslabones consecutivos. Indica la inclinación del eslabón con respecto al anterior y se mide en radianes.

Gracias a estos parámetros, se puede modelar y controlar el movimiento del manipulador de forma precisa y eficiente. Uno de los aspectos más importantes de los parámetros D-H es que permiten describir la cinemática directa del manipulador, es decir, calcular la posición y orientación del extremo del manipulador a partir de las variables de articulación. Además, también permiten describir la cinemática inversa del manipulador, es decir, calcular las variables de articulación necesarias para alcanzar una posición y orientación deseada del extremo del manipulador. Esto es esencial en aplicaciones de robótica, donde se requiere que el manipulador realice tareas precisas y repetitivas.

2.3. Bases Legales

A continuación, se presentan algunas de las principales bases legales que rigen este proyecto de investigación:

1. Constitución de la República Bolivariana de Venezuela: La Constitución establece los principios y valores fundamentales del Estado venezolano, incluyendo el derecho al acceso y uso de las tecnologías de la información y la comunicación (TIC) como un derecho humano básico.
2. Ley Orgánica de Ciencia, Tecnología e Innovación (LOCTI): Esta ley establece el marco legal para la promoción, el desarrollo y la difusión de la ciencia, la tecnología y la innovación en Venezuela. La LOCTI establece mecanismos de financiamiento y estímulos para la investigación y el desarrollo tecnológico, incluyendo el desarrollo de proyectos de robótica e inteligencia artificial.
3. Ley Orgánica de Educación (LOE): Esta ley establece los principios y objetivos de la educación en Venezuela, incluyendo la formación de ciudadanos críticos y creativos capaces de utilizar las TIC para el desarrollo personal y social. La LOE también establece las bases para la formación en ciencia, tecnología e innovación en todos los niveles educativos, desde la educación básica hasta la educación superior.
4. Normas Técnicas Venezolanas (NTV): Estas normas establecen los estándares y requisitos técnicos para la fabricación, uso y mantenimiento de los equipos y sistemas robóticos en Venezuela. Las NTV incluyen normas relacionadas con la seguridad, la eficiencia energética y la interoperabilidad de los sistemas robóticos.
5. La Organización Internacional de Normalización (ISO) ha desarrollado una serie de normas, como la ISO 8373, que establece la terminología, clasificación y especificaciones

técnicas para robots industriales, y la ISO 10218, que establece los requisitos de seguridad para robots industriales.

Existen distintas bases legales que regulan la robótica y la virtualización de brazos robóticos en entornos de laboratorio. Es importante que el proyecto tenga en cuenta estas leyes y normativas para garantizar el cumplimiento de las regulaciones y evitar posibles problemas legales o de responsabilidad.

2.4. Definición de Términos

Automatización: Se refiere al uso de tecnología para realizar tareas sin la intervención humana. Puede implicar la utilización de maquinaria, software o sistemas para automatizar procesos y mejorar la eficiencia.

Interfaz Gráfica: Es un medio que permite la comunicación entre el usuario y una aplicación informática, a través de elementos visuales como botones, iconos y ventanas. Es una forma intuitiva de interactuar con un software o sistema.

Manipulación: En el contexto de la robótica, se refiere al conjunto de técnicas utilizadas para controlar los movimientos de un robot, ya sea para realizar tareas específicas o para permitir una interacción con su entorno.

Modelación 3D: Es la creación de modelos digitales tridimensionales que representan objetos, espacios o diseños. Puede ser utilizado en una variedad de industrias, como la arquitectura, ingeniería, diseño de productos, entre otros.

Programación: Es el proceso de escribir código para crear software o sistemas informáticos. Los lenguajes de programación se utilizan para crear instrucciones que un ordenador puede interpretar y ejecutar.

Robot: Es una máquina programable que puede realizar tareas de forma autónoma o controlada por un operador humano. Los robots pueden ser diseñados para una variedad de aplicaciones, desde la fabricación y la construcción hasta la exploración espacial y la asistencia médica.

Servo-robot: Es un tipo de robot que utiliza motores eléctricos llamados servomotores para controlar el movimiento y la posición de sus componentes mecánicos.

Sistema Operativo: Es el software que controla el funcionamiento de un ordenador o dispositivo móvil. Permite a los usuarios interactuar con los componentes de hardware y con otras aplicaciones, proporcionando una interfaz y administrando los recursos del sistema.

Software: Es un conjunto de programas, datos y archivos utilizados por un ordenador o dispositivo móvil para realizar una variedad de tareas. Puede incluir sistemas operativos, aplicaciones, juegos o herramientas de productividad.

Virtualización: La virtualización en el contexto de un robot se refiere a la creación de una representación digital del robot y su entorno físico, que se puede simular y controlar en un entorno virtual.

Repositorios: Son espacios de almacenamiento donde se guardan y comparten los paquetes de ROS. Los repositorios pueden ser locales (en la máquina del usuario) o en línea (como GitHub o el repositorio oficial de ROS). Los repositorios facilitan la distribución, colaboración y gestión de paquetes en un sistema ROS.

Espacio de trabajo: Es un directorio donde se organizan y compilan los paquetes que se utilizarán en un proyecto específico. Los espacios de trabajo permiten aislar los paquetes necesarios para un proyecto particular y compilarlos juntos para que funcionen correctamente. También proporcionan un entorno de desarrollo separado del sistema operativo principal para evitar conflictos.

Paquetes: Son la unidad básica de organización en ROS. Contienen todos los archivos necesarios para llevar a cabo una tarea específica. Estos archivos pueden incluir códigos fuente, bibliotecas, archivos de configuración, datos y más.

Nodos: Son procesos individuales que realizan tareas específicas en un sistema ROS. Cada nodo se encarga de una función particular, como la adquisición de datos de sensores, el procesamiento de información, el control de actuadores o la comunicación con otros nodos. Los nodos se ejecutan de manera independiente y se comunican entre sí a través de mensajes y servicios.

CAPÍTULO III

MARCO METODOLÓGICO

3.1. Paradigma de investigación

El vertiginoso avance alcanzado por las tecnologías en los últimos años, consolidó un nuevo paradigma que generó las denominadas nuevas tecnologías de información y comunicación (NTIC) las que plantean a los países con un menor desarrollo nuevos retos, precisos de aceptar para garantizar el acceso al enorme potencial de información existente. Este paradigma es el tecnológico, el cual se reconoce por su creciente papel en las investigaciones tecnológicas, el aumento de la demanda de información y nuevos conocimientos, la tendencia a la comercialización del nuevo conocimiento, el auge en la generación y difusión de las nuevas tecnologías. Sobre la base de las consideraciones anteriores, esta investigación pertenece al paradigma tecnológico debido a que se basa en la aplicación de conocimientos y técnicas tecnológicas para desarrollar una solución práctica y funcional en el ámbito de la robótica industrial.

3.2. Tipo de investigación

Palella y Martins (2006, p.107) plantean que el proyecto factible consiste en "elaborar una propuesta viable destinada a atender necesidades específicas, determinadas a partir de una base diagnóstica" y consideran que el propósito fundamental de esta modalidad es presentar proposiciones y planteamientos que se puedan ejecutar. Es evidente entonces que esta investigación se limitará en una modalidad de tipo factible ya que, en primer lugar, existe una amplia literatura y estudios previos sobre la virtualización de robots y simulación en escenarios de robótica como ROS, por lo tanto, se puede realizar una revisión bibliográfica exhaustiva para recopilar información relevante y actualizada del tema. En segundo lugar, el hecho de que el robot Lab-Volt 5250 esté diseñado con una arquitectura modular y programable, lo hace más adecuado para la virtualización. Esto permitirá separar la lógica de control del hardware físico y crear una representación del robot que funcione igual al robot físico.

3.3. Diseño de investigación

El diseño de investigación es el plan o la estrategia general determinada por el investigador luego de tener una visión clara de su problema. Esta estrategia le brindará las herramientas necesarias para definir cómo abordará el problema y cómo obtendrá la información necesaria para

la investigación, de este modo se podrá proponer un modelo de verificación que permita relacionar lo hechos con la teoría.

Palella y Martins (2006, p.97) proponen cuatro tipos de investigación, pre-experimental, cuasiexperimental, documental y de campo. Exponen que la investigación documental "se concentra exclusivamente en la recopilación en diversas fuentes. Indaga sobre un tema en documentos escritos u orales"; y plantean que la investigación documental "consiste en la recolección de datos directamente de la realidad donde ocurren los hechos, sin manipular o controlar variables. Estudia los fenómenos sociales en su ambiente natural".

Significa entonces que el diseño de investigación utilizado para la virtualización del robot Lab-Volt 5250 en el framework ROS, estará sustentado en una investigación de campo ya que habrá contacto directo con el objeto de estudio, se recopilarán datos y se extraerá información directamente de la realidad que está siendo estudiada con el fin de obtener nuevos medios para afrontar la problemática planteada y encontrar la mejor solución. También se apoyará en una investigación documental ya que será necesaria la revisión del manual del robot Lab-Volt 5250, manuales que faciliten la utilización de ROS, investigaciones previas donde se haya utilizado a ROS como entorno de simulación, foros de la comunidad de ROS, entre otros.

3.4. Nivel de investigación

Arias (2012, p.23) define al nivel de la investigación como "el grado de profundidad con que se aborda un fenómeno u objeto de estudio". Entre los niveles de investigación se encuentra el interactivo, el cual busca realizar una intervención diseñada para modificar el evento bajo estudio. Este proceso comienza con la exploración y descripción del evento, y a través del análisis se identifican los cambios necesarios y se proponen acciones específicas para generar cambios. Se observa claramente que el proyecto en desarrollo posee un nivel de investigación interactivo. Este proceso de investigación implica una exploración y descripción del robot, así como el análisis de los cambios necesarios para lograr mantener actualizado el robot y lograr una mejor enseñanza. Posteriormente se identifican las acciones requeridas para lograr el objetivo planteado, las cuales constituyen la virtualización del robot, el diseño de la interfaz gráfica a través del software de código abierto para luego implementarla en la simulación del robot Lab-Volt 5250.

3.5. Población

En el ámbito de la investigación, es fundamental definir con precisión la población objeto de estudio. Según Palella y Martins (2006, p.115), la población puede ser definida como "el

conjunto finito o infinito de elementos, personas o cosas pertenecientes a una investigación y que generalmente suele ser inaccesible". En el caso concreto del proyecto en cuestión, la población se limita al robot Lab-Volt 5250 de la Universidad José Antonio Páez, ya que este es el objeto de estudio en específico.

3.6. Muestra

Arias (2012, p.83) establece que la muestra es "un subconjunto representativo y finito que se extrae de la población accesible" y define a la muestra representativa como "aquella que por su tamaño y características similares a las del conjunto, permite hacer inferencias o generalizar los resultados al resto de la población con un margen de error conocido". Sin embargo, en el caso de la presenta investigación, conociendo la población de estudio, y atendiendo a las definiciones de muestra, se puede deducir que la población y la muestra coincides. En otras palabras, la investigación posee una muestra censal, donde todas las unidades de investigación son consideradas como muestra. En este sentido, la muestra se comprende como el robot Lab-Volt 5250 de la Universidad José Antonio Páez, en el cual se centrará la investigación.

3.7. Técnicas e instrumentos de recolección de datos

Tal como se ha visto, para dar respuesta a las interrogantes formuladas, es necesario utilizar ciertas técnicas e instrumentos que permitirán ampliar la información que se conoce del objeto de estudio. Según Arias (2012, p.68), la aplicación de técnicas de recolección de datos "conduce a la obtención de información, la cual debe ser guardada en un medio material de manera que los datos puedan ser recuperados, procesados, analizados e interpretados posteriormente. A dicho soporte se le denomina instrumento".

3.8. Técnicas de recolección de datos

Con relación a lo anterior, Palella y Martins (2006, p.126) plantean que las técnicas de recolección de datos deben ser aplicadas al entrar en contacto directo con el objeto de investigación y las definen como "las distintas formas o maneras de obtener información". Las técnicas utilizadas para el desarrollo del proyecto son:

3.8.1. Observación: "Consiste en visualizar o captar mediante la vista, en forma sistemática, cualquier hecho, fenómeno o situación que se produzca en la naturaleza o en la sociedad, en función de unos objetivos de investigación preestablecidos". (Arias, 2012, p.69). Para la presente investigación se utilizará la observación directa al observar y recopilar información

dentro del conjunto de elementos que se quiere conocer e investigar, como el comportamiento del robot, su posición inicial, entre otros aspectos.

3.8.2. Entrevista: "Es una técnica basada en un diálogo o conversación cara a cara, entre el entrevistador y el entrevistado acerca de un tema previamente determinado, de tal manera que el entrevistado pueda obtener la información requerida". (Arias, 2012, p.73). En referencia al presente proyecto, las entrevistas fueron realizadas directamente a expertos en el manejo del entorno ROS, lo que permitió aclara dudas, solicitar ayuda en la resolución de algún inconveniente y obtener información precisa y actualizada sobre las mejores prácticas y recomendaciones para la simulación del robot en ROS. Además, estas entrevistas generaron nuevas perspectivas para enriquecer la investigación con diferentes puntos de vista.

3.8.3. Revisión analítica de la literatura: "Implica detectar, consultar y obtener la bibliografía y otros materiales que sean útiles para los propósitos del estudio, de donde se tiene que extraer y recopilar la información relevante y necesaria para enmarcar nuestro problema de investigación". (Sampieri, 2018, p.61). Para el desarrollo de la investigación, se revisó tanto el manual del robot Lab-Volt 5250, como manuales que faciliten la utilización de ROS para el desarrollo del proyecto. También fue necesaria la revisión de trabajos de investigación realizados previamente donde se utilice el software de código abierto ROS como entorno de simulación y sustenten la solución que se planteó.

3.9. Instrumentos de recolección de datos

Arias (2012, p.68) define a los instrumentos de recolección de datos como "cualquier recurso, dispositivo o formato (en papel o digital) que se utiliza para obtener, registrar o almacenar información". Para el presente trabajo de investigación se utilizaron los siguientes instrumentos para la recolección de datos:

Registro descriptivo: Esta herramienta tiene como finalidad registrar por escrito información de competencias observables y determinadas a través de criterios específicos de una actividad, en un tiempo y lugar determinado. El registro no debe contener apreciaciones personales ni subjetivas. Esta herramienta se utilizó para registrar todas las observaciones realizadas al robot, referentes a su comportamiento.

Entrevista no estructurada: En este estudio se aplicaron entrevistas no estructuradas, donde las preguntas serán realizadas a través de foros de discusión con expertos en el manejo del entorno ROS. Palella y Martins (2006, p.141) definen al guion de entrevista no estructurada como

“aquel en que no existe una estandarización formal dejando, por lo tanto, un margen más o menos grande de libertad para formular las preguntas y proporcionar las respuestas”. Un tipo de entrevista no estructurada es la informal. Palella y Martins (2006), en su obra Metodología de la Investigación afirman que:

> La entrevista informal es la modalidad menos estructurada posible de entrevista. Se reduce a una simple conversación sobre el tema en estudio. Lo importante aquí no es definir los límites de lo tratado ni ceñirse a ningún esquema previo, sino “hacer hablar” al entrevistado, con el fin de obtener un panorama de los problemas más resaltantes de los mecanismos lógicos y mentales del respondente, de los puntos que él considera básicos. (p.141).

Hecha la observación anterior, el presente trabajo de investigación utilizó la entrevista informal para recolectar información de los foros de comunicación con expertos en ROS. Este tipo de entrevista es una técnica flexible, que permite adaptarse a las necesidades y características del entrevistador, lo que puede facilitar la obtención de información de forma precisa, mejorar la calidad de los datos y la confiabilidad de los resultados obtenidos. Se obtuvo información valiosa y detallada sobre el uso de ROS, lo que contribuyo significativamente al desarrollo de la investigación.

Computadora y sus unidades de almacenaje: Es una herramienta clave en la realización de una revisión bibliográfica exhaustiva sobre la virtualización y control de robots en ROS. En primer lugar, se utilizaron motores de búsqueda especializados y repositorios digitales para recopilar información relevante. Una vez obtenidos los resultados, estos se organizaron y almacenaron para su posterior utilización en la investigación. Además, las entrevistas con expertos en ROS fueron almacenadas en la computadora para su posterior análisis y utilización en la investigación. De esta manera, se aseguró una adecuada gestión y organización de la información recopilada, permitiendo una evaluación rigurosa y precisa de los resultados obtenidos.

3.10. Técnicas de análisis de datos

La información proveniente de la observación directa, de las fuentes bibliográficas y de la entrevista no estructurada, se recolectará de manera cuidadosa y sistemática, se estudiará y se seleccionará de forma tal que sea de gran relevancia para la investigación. Para lograr esto, es necesario clasificar la información según su contenido y organizarla de acuerdo al grado de conocimiento teórico que contenga la misma. De esta manera, se pueden identificar las fuentes de información más relevantes y útiles para el estudio, lo que a su vez permitirá enfocar los esfuerzos en la adquisición de información de alta calidad y relevancia. Además, las clasificación y

organización de la información, también facilita el proceso de análisis y la identificación de patrones y relaciones entre los datos recopilados.

3.11. Fases Metodológicas

Fase I Estudio del robot Lab-Volt 5250 y sus herramientas de programación y simulación.

Se realizó una observación directa del robot en el laboratorio para obtener una información relevante sobre su comportamiento y operación en el entorno real. También se revisó la documentación técnica del robot y se recopiló información relevante de otras fuentes bibliográficas. De igual forma, se llevó a cabo un análisis riguroso del software del fabricante Festo utilizado por el robot y todas las herramientas que este posee, evaluando la facilidad de uso y la eficacia de estas herramientas para programar y simular el robot en distintos escenarios. Toda la información recopilada fue clasificada y organizada en función de su contenido y relevancia para la investigación.

Fase II Análisis de información obtenida en foros de discusión con expertos en el manejo del entorno ROS.

En esta fase se recopilarán las opiniones y sugerencias de los expertos, y se identificarán las mejores prácticas y recomendaciones para la virtualización del robot Lab-Volt 5250 en ROS: Se utilizarán herramientas de análisis de datos y se aplicará la “minería de datos” para identificar patrones y tendencias relevantes en la información recopilada, para luego ser verificada a través de la práctica.

Fase III Realización del modelado tridimensional del robot Lab-Volt 5250 y sus accesorios.

En este proyecto, se utilizará SolidWorks para crear un modelo virtual del robot y sus accesorios, lo que permitirá una simulación y virtualización precisa del robot en un entorno controlado y seguro en ROS. Se prestará especial atención a la precisión y la fidelidad del modelo virtual para garantizar su adecuación para la simulación en ROS. Esto implica que se deben considerar los detalles más pequeños del robot, como las dimensiones de las piezas y la geometría del sistema de articulaciones, así como los accesorios y herramientas que se utilizarán en el robot.

Fase IV Obtención del modelo cinemático del robot Lab-Volt 5250.

Se llevará a cabo un análisis detallado de los movimientos y las características del robot, empleando herramientas matemáticas y de estudio cinemático. Para ello, se utilizarán los

parámetros de Denavit - Hartenberg (D-H) y la matriz de transformación, que permite obtener el resultado de las rotaciones o traslaciones efectuadas por cada articulación, del robot, el cual será posteriormente utilizado en la simulación del mismo en el entorno virtual de ROS. Es importante destacar que el uso de estas herramientas y técnicas permitirá una adecuada representación de los movimientos y operaciones del robot en el entorno virtual, lo que contribuirá a una simulación más precisa y eficiente del mismo en dicho entorno.

Fase V Demostración del manejo del robot virtual y sus accesorios en la interfaz gráfica de ROS.

Se utilizarán herramientas de programación para desarrollar y simular el modelo tridimensional del robot Lab-Volt 5250 en una interfaz clara, intuitiva y fácil de usar del entorno virtual de ROS. La interfaz permitirá la interacción con el robot virtual y el control de sus movimientos y operaciones. Además, se mostrará el funcionamiento del robot virtual en el entorno virtual de ROS, y se demostrará su capacidad para realizar diversas tareas y operaciones utilizando sus accesorios. Se prestará especial atención a la precisión y la fidelidad de la simulación, así como la facilidad de uso y eficiencia de la interfaz gráfica. La demostración permitirá validar la eficiencia y utilidad del robot virtual para la investigación y el desarrollo en robótica, y permitirá obtener retroalimentación valiosa para futuras mejoras y optimizaciones del sistema. Además, la demostración también servirá como herramienta de enseñanza y divulgación para aquellos interesados en el aprendizaje de la robótica y su aplicación en el mundo real, especialmente, a los estudiantes de la Universidad José Antonio Páez.

CAPÍTULO IV

RESULTADOS

Fase I Estudio del robot Lab-Volt 5250 y sus herramientas de programación

La primera fase de este proyecto se enfocó en el estudio detallado del robot Lab-Volt 5250 y sus componentes esenciales a través de la observación directa y la revisión documental de sus manuales. Esta fase tiene como objetivo principal comprender a fondo las características y capacidades de este robot, junto con sus accesorios educativos, como el carrusel y la banda transportadora. Asimismo, se hizo énfasis en la herramienta de programación y simulación utilizada por el robot, el software RoboCIM, evaluando sus funciones y aplicaciones en el contexto del Lab-Volt 5250. Esta fase inicial sienta las bases para una exploración exhaustiva del robot, sus accesorios y su software de simulación.

4.1.1 Robot Lab-Volt 5250

El Servo Robot en estudio, el cual se observa en la figura 2, es un sistema versátil compuesto por cinco ejes de rotación y una pinza, conocida como gripper. Estas articulaciones se sincronizan para realizar secuencias de movimientos que pueden ser programadas previamente para cumplir con distintas tareas. El robot tiene la capacidad de operar en dos modos: el modo articular, donde cada articulación puede ser controlada de manera independiente, y el modo cartesiano, que permite el movimiento lineal del gripper a lo largo de un eje específico. Para programar y controlar el robot, existen dos opciones. Una de ellas es a través de un práctico dispositivo portátil denominado "teach pendant". La otra opción es utilizar una computadora que se conecta al robot mediante el software RoboCIM.

El equipo proporcionado junto al sistema de servo robot, modelo 5250-1, incluye el propio servo robot, un módulo controlador del robot, un teach pendant, un módulo de botón de parada de emergencia, el software RoboCIM 5250, rejillas, herramientas de calibración del robot, manuales tanto para el estudiante como para el instructor, guías del usuario y todos los cables necesarios para operar el sistema.

Figura 2. Lab-Volt 5052

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.1.2 Módulo Controlador del Robot

El módulo controlador del robot está equipado con una serie de características clave, así como se muestra en la figura 3. En primer lugar, dispone de entradas TTL (Transistor-Transistor Logic) que se utilizan para el monitoreo de dispositivos de entrada. Además, incorpora salidas TTL que permiten la comunicación con otras unidades robóticas y la capacidad de controlar accesorios externos, como una cinta transportadora. También, incluye salidas de relé destinadas al control de accesorios externos, como sirenas o luces giratorias (aunque estos accesorios no están incluidos en el sistema principal).

Para controlar dispositivos externos, el módulo presenta puertos de salida específicos, por ejemplo, para el manejo del carrusel giratorio. Por otro lado, el puerto de comunicación serial desempeña un papel crucial, ya que facilita las operaciones de control desde una computadora o desde el teach pendant, y permite la carga y descarga de programas de manera eficiente. Esta amplia variedad de opciones de control proporciona una flexibilidad excepcional al sistema y lo convierte en una herramienta versátil para diversas aplicaciones.

Figura 3. Módulo Controlador

Fuente: D'Agostini, J. y Nazar, S. (2023)

Un aspecto significativo es la presencia de puertos CNC (Control Numérico Computarizado) en el módulo de control, lo que facilita la comunicación con las máquinas CNC de Lab-Volt Automation. Esta integración permite un sistema de fabricación altamente coordinado. Además, el controlador de robot está equipado con una unidad de disquete diseñada especialmente para el almacenamiento de programas.

4.1.3 Teach Pendant

El teach pendant o terminal portátil se conecta al controlador del robot a través de un puerto de comunicación serial. Funciona como una interfaz de usuario que permite a los operadores interactuar de manera directa con los robots, facilitando tareas como la programación de movimientos, la depuración de programas y el control en tiempo real. Esta herramienta está equipada con una pantalla, botones y controles manuales, tal como se puede visualizar en la figura 4, por lo cual, ofrece una experiencia similar a la de trabajar desde una computadora, pero con la ventaja de la inmediatez.

No obstante, es importante destacar que, a diferencia de una computadora, el teach pendant no proporciona la capacidad de simular movimientos antes de su ejecución; en su lugar, permite observar y ajustar las acciones del robot en tiempo real. Esta característica lo convierte en una herramienta esencial para lograr operaciones eficientes y seguras en entornos industriales.

Figura 4. Teach Pendant

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.1.4 Almacén/Superficie de trabajo – Modelo 6309

En la figura 5 se observa la superficie de trabajo o almacén, la cual es una placa metálica perforada, de 38 x 584 x 584 mm, sobre la que se coloca el equipo. Se pueden unir dos superficies de trabajo utilizando separadores, modelo 39035.

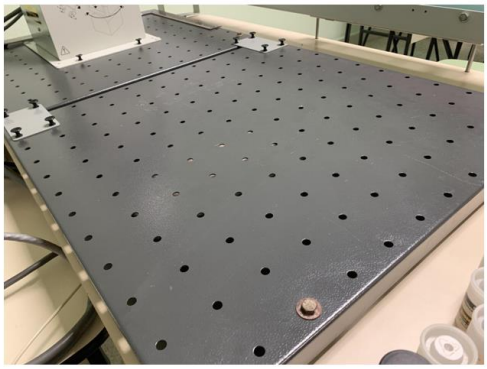


Figura 5. Superficie de trabajo

Fuente: D'Agostini, J. y Nazar, S. (2023)

Pueden añadirse muchos dispositivos opcionales al sistema para realizar procesos más complejos. Los dispositivos opcionales incluyen un carrusel giratorio, un transportador de banda, alimentadores por gravedad, alimentadores neumáticos, una torre de señales y un tobogán lineal. Dentro del laboratorio de Robótica Industrial de Universidad José Antonio Páez se pueden encontrar 4 dispositivos adicionales: un alimentador por gravedad para piezas cuadradas, un alimentador por gravedad para piezas cilíndricas, un carrusel giratorio y una banda transportadora.

Estos dispositivos son anclados a la superficie mediante un acople de accesorio, tal como muestra la figura 6.

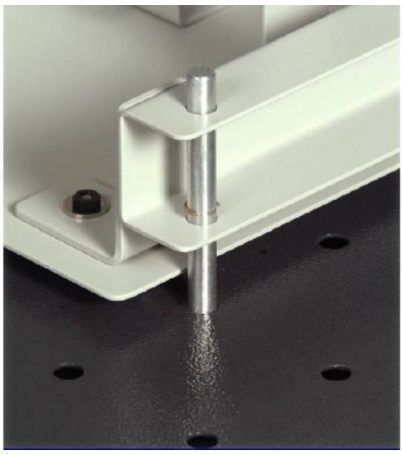


Figura 6. Acople de accesorio

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.1.5 Alimentador por gravedad (Piezas cuadradas) - Modelo 5119

El alimentador por gravedad para piezas cuadradas está diseñado para alimentar piezas cuadradas de 51 x 51 mm (2 x 2 pulg.) con un grosor de entre 13 y 38 mm (0,5 y 1,5 pulg.), este se encuentra en la figura 7. El alimentador dispone de un interruptor sensor y cables de realimentación para la conexión al controlador del robot. Además, puede variar su posición en el espacio de trabajo ya que se encuentra sobre la superficie microperforada, por lo tanto, se puede desplazar sobre esta y luego se fija a través de los separadores 39035.

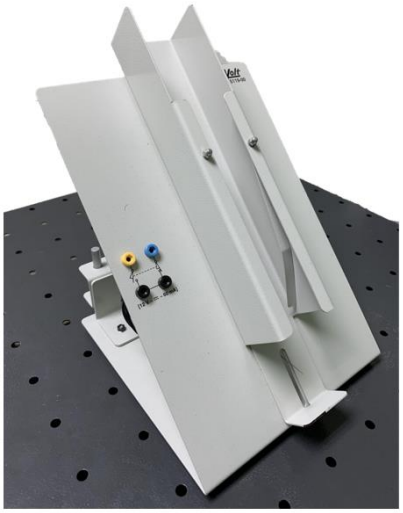


Figura 7. Alimentador por gravedad cuadrado

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.1.6 Alimentador por gravedad (piezas cilíndricas) - Modelo 5121

El alimentador por gravedad (piezas cilíndricas), expuesto en la figura 8, está diseñado para alimentar piezas cilíndricas similares en forma y dimensiones a los recipientes de película,. Dispone de un interruptor sensor y cables de realimentación para su conexión al controlador del robot. Al igual que el alimentador por gravedad para piezas cuadradas, puede variar su posición ya que se encuentra sobre la superficie microperforada, pudiendo desplazarse sobre ella para posteriormente fijarlo a través de los separadores 39035.

Figura 8. Alimentador por gravedad cilíndrico

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.1.7 Carrusel giratorio - Modelo 5208-1

El Carrusel Rotativo (figura 9) se utiliza para demostrar cómo las piezas pueden ser transferidas hacia y desde un robot en un patrón repetitivo y rotativo. Su plato de 457 mm (18 pulgadas) es accionado por un servomotor de corriente continua. El carrusel tiene un sistema de retroalimentación de bucle cerrado, este sistema verifica continuamente la posición del carrusel y hace correcciones para garantizar que se detenga o gire exactamente donde se espera. También cuenta con un interruptor de límite de retroalimentación para las capacidades de hard home, el cual está diseñado para garantizar que, cuando el sistema se desplace hacia su posición de referencia o "home", sepa cuándo ha alcanzado con precisión ese punto. Cuando el interruptor detecta que el

dispositivo ha alcanzado su posición de "home", envía una señal al controlador para indicar que se ha completado con éxito la operación de "hard home". Al igual que los alimentadores por gravedad, el funcionamiento del carrusel giratorio está controlado por el controlador del robot.

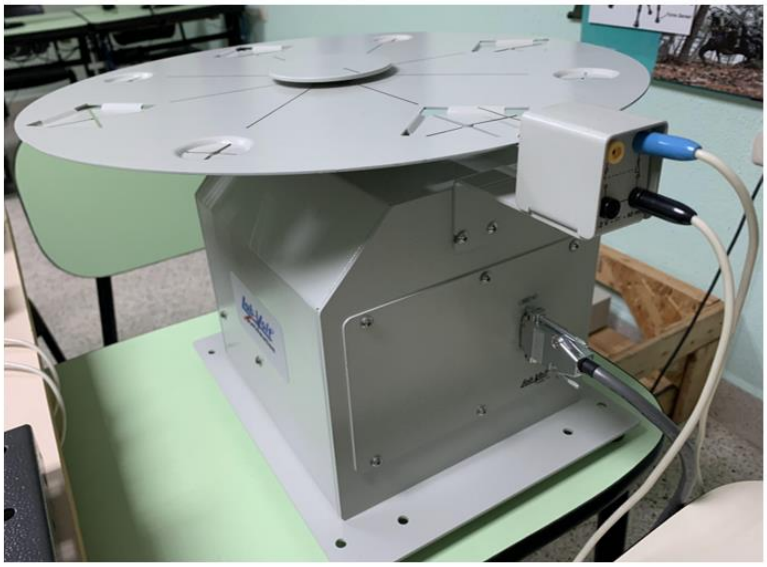


Figura 9. Carrusel giratorio

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.1.8 Banda transportadora - Modelo 5210

El transportador de cinta es utilizado en experimentos de manipulación de materiales. Cuenta con una fuente de alimentación autónoma y una interfaz electrónica intuitiva. En cuanto a su integración con el robot, el panel de control ofrece múltiples entradas (tal como se aprecia en la figura 10) que permiten una interacción efectiva con el controlador del robot, brindando así la capacidad de controlar el transportador de forma remota. Sin embargo, es importante destacar que este versátil dispositivo también puede operar de manera independiente. Además, para garantizar una detección de piezas en la cinta, tiene incorporado un final de carrera móvil. La cinta transportadora en sí presenta unas dimensiones generosas, con una longitud de 1.880 mm y una anchura de 127 mm, lo que la hace adecuada para una amplia gama de aplicaciones.

Figura 10. Banda transportadora

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.1.9 Software RoboCIM

RoboCIM es un programa de software que simula y controla el funcionamiento de sistemas robóticos y sistemas servo-robóticos de la serie Lab-Volt, así como sus equipos externos opcionales, como alimentadores por gravedad, cintas transportadoras o carruseles giratorios. Este software se basa en cuatro pasos básicos para configurar un espacio de trabajo, los cuales son los mismos pasos que se usarían para crear el sistema real. Secuencialmente, los pasos se pueden definir de la siguiente manera:

1. Agregar objeto al espacio del trabajo.
2. Aplicar movimiento a la articulación controlable.
3. Crear y ejecutar el programa.
4. Conectarse al sistema real.

4.1.10 Workspace

El "workspace" o "espacio de trabajo" del software RoboCIM se refiere al espacio virtual o entorno de trabajo en el cual se realiza la programación y simulación del robot, es decir, proporciona una interfaz gráfica que permite a los usuarios visualizar y modificar las configuraciones de los robots, añadir elementos y objetos, así como diseñar y planificar las tareas que el robot realizará en el mundo virtual. Es una parte esencial de la programación y simulación

de robots en RoboCIM, ya que facilita la creación y prueba de programas antes de implementarlos en un espacio real.

4.1.11 Movimientos

El software RoboCIM permite el control y la visualización de los movimientos del sistema de forma interactiva. El robot Lab-Volt 5250 utiliza dos tipos de sistemas de coordenadas para el control de sus movimientos, estos son:

4.1.11.1 Sistema de coordenadas articulares

Controla el movimiento en cada una de las ocho articulaciones del robot, desde la base, el hombro, el codo, la muñeca, el giro de la muñeca, el motor de la pinza, hasta de los dispositivos externos que están conectados a los puertos del controlador. Al utilizar el sistema de coordenadas articular, el movimiento no seguirá una línea recta entre dos puntos, además, es mucho más rápido, por lo que no se recomienda utilizar este sistema de coordenadas cuando se necesita un movimiento preciso entre dos puntos. En la figura 11 se muestra el panel de control de las coordenadas articulares.

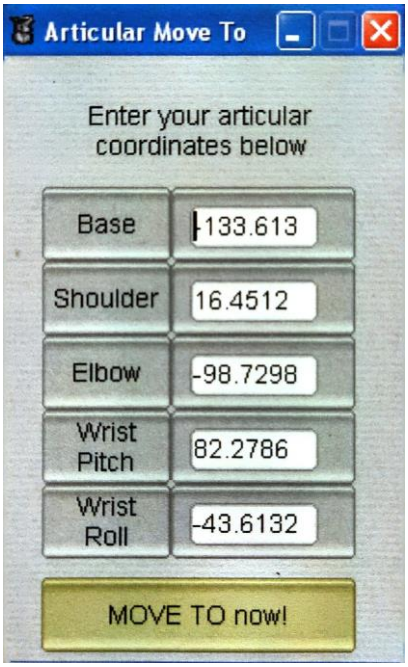


Figura 11. Panel de control de las coordenadas articulares

Fuente: D'Agostini, J. y Nazar, S. (2023)

En el panel de control del sistema articular, se encuentran dos secciones distintas para manejar las articulaciones del robot. La sección superior está dedicada a los controles individuales de las cinco articulaciones, que son la base, el hombro, el codo, la muñeca y el giro de muñeca.

Mientras tanto, en la sección inferior se ubica el control para el movimiento del gripper, que permite abrir y cerrar la pinza y los controles para los equipos externos.

Para ajustar los valores de las coordenadas, se utilizan las flechas izquierda y derecha. En el caso de las cinco articulaciones, el valor ubicado entre las flechas representa los ángulos con respecto al eje de rotación. Esta disposición de controles proporciona una manera intuitiva para que el operador pueda controlar con precisión los movimientos del robot. También se puede encontrar en la sección media de este mismo panel, el botón Move To, el cual permite mover varias articulaciones al mismo tiempo para llegar a un punto determinado.

4.1.11.2 Sistema de coordenadas cartesianas

Controla el movimiento en el efector final del robot. Al utilizar el sistema de coordenadas cartesianas para mover el robot, la trayectoria entre las posición inicial y final será lineal, por lo tanto, es recomendable utilizarlo cuando se requieran movimientos precisos y no tan rápidos. Este sistema no puede ser utilizado para el gripper o los dispositivos externos ya que solo comprende movimientos lineales. Podemos visualizar el panel de control de las coordenadas cartesianas en la figura 12.

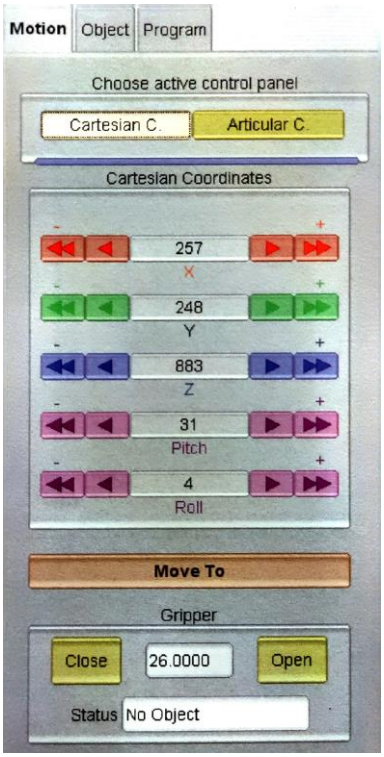


Figura 12. Panel de control de las coordenadas cartesianas

Fuente: D'Agostini, J. y Nazar, S. (2023)

La sección superior del panel de control presenta cinco valores de coordenadas que definen la posición del efector final del robot. Estas coordenadas se dividen en dos grupos. El primero consta de las tres primeras coordenadas (X, Y, Z), que se emplean para ubicar el efector final en un punto específico en relación con los ejes de referencia de la base del robot. Para ser más preciso, la coordenada X mueve el efector final de adelante hacia atrás o viceversa, manteniendo constantes las otras coordenadas. Mientras tanto, la coordenada Y desplaza el efector final de izquierda a derecha o viceversa, y la coordenada Z lo eleva o desciende.

En cuanto al segundo grupo, formado por las coordenadas Pitch y Roll, se utilizan para orientar el efector final alrededor del punto definido anteriormente. Pitch gira el efector final de arriba hacia abajo o viceversa, mientras que Roll lo rota en sentido horario o antihorario, sin afectar las demás articulaciones. Además, en la sección del gripper se encuentra un control que permite abrir o cerrar la pinza de manera similar al sistema de coordenadas articulares. La combinación de todas estas coordenadas ofrece un control exhaustivo y preciso sobre la posición y orientación del efector final del robot.

De manera adicional, en el panel se encuentra el botón "Move To", que desempeña la misma función que el sistema de coordenadas articulares, permitiendo mover múltiples articulaciones simultáneamente para alcanzar un punto específico.

4.1.12 Panel registrador de puntos

RoboCIM ofrece una funcionalidad fundamental a través de su panel de registro de puntos. Este panel exhibido en la figura 13, posibilita el registro de ubicaciones clave a las que se desea que el robot se desplace. Cada punto registrado recibe un nombre descriptivo y se le asigna un color para simplificar su identificación posterior. Los puntos registrados desempeñan un papel crucial en la definición de los movimientos necesarios para el robot. Al establecer estos puntos estratégicos a lo largo del recorrido deseado, se puede crear un programa detallado que guíe al robot a través de una serie de movimientos específicos. Este enfoque basado en puntos proporciona una manera intuitiva y eficiente de programar las acciones del robot de acuerdo con la ruta deseada.

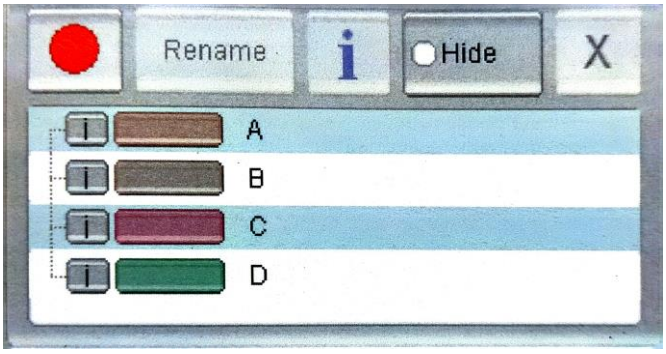


Figura 13. Panel registrador de puntos

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.1.13 Programación de Tareas

RoboCIM ofrece a los usuarios la flexibilidad de crear y ejecutar dos tipos distintos de programas, lo que amplía su capacidad para abordar una variedad de tareas y proyectos:

4.1.13.1 Programa de Iconos

Este enfoque se destaca por su simplicidad y es ideal para tareas de naturaleza más sencilla. Se basa en el uso de iconos y herramientas gráficas para construir programas y rutinas.

4.1.13.2 Programa de Texto

Esta es una opción más completa que se emplea para diseñar y ejecutar tanto tareas simples como complejas. Este método se basa en la utilización de comandos de texto para construir programas específicos. Ofrece un mayor grado de control y personalización.

Una ventaja destacada de RoboCIM es que no impone límites en cuanto a la cantidad de programas que se pueden crear y utilizar. Esto significa que los usuarios pueden gestionar múltiples programas de iconos y de texto simultáneamente. Esta flexibilidad es esencial para adaptarse a una amplia gama de proyectos y requisitos.

4.1.14 Conexión al sistema real

Una vez que el espacio de trabajo esté debidamente preparado y todos los elementos necesarios para la simulación estén en su lugar, se procede a establecer la conexión entre el sistema real y el entorno de simulación. Este paso implica cambiar del modo "simulación" al modo "control", en el cual la computadora que ejecuta RoboCIM se comunica directamente con el controlador del robot. De esta manera, se pueden enviar comandos desde la computadora al controlador del robot para que el robot realice movimientos que correspondan exactamente a las posiciones simuladas en el entorno virtual.

Fase II: Análisis de información obtenida en foros de discusión con expertos en el manejo del entorno ROS.

Tras completar exhaustivamente la Fase I de este estudio, donde se realizó un profundo análisis del robot Lab-Volt 5250 y se exploraron sus herramientas de programación y simulación, así como el de sus accesorios, damos inicio a la Fase II, que marca una nueva etapa fundamental en nuestra investigación. El propósito fundamental de esta fase radica en la extracción de conocimiento a partir de una fuente de información invaluable: los foros de discusión con expertos en el entorno Robot Operating System.

La Fase II no solo representa un proceso de análisis, sino también la oportunidad de establecer un puente entre la teoría y la práctica. Estos hallazgos se validarán mediante experimentación y pruebas en un entorno ROS real, lo que enriquecerá aún más nuestro entendimiento y capacidad de aplicación en el mundo de la robótica.

4.2.1 Selección de Foros de Discusión

La elección de los foros de discusión se realizó mediante un proceso minucioso y meticuloso, con el propósito de asegurar la calidad y pertinencia de los datos que serían recopilados. Entre los foros seleccionados, destacan ROS Discourse, ROS Answers y ROS GitHub Issues, siendo figuras notables en la comunidad de ROS gracias a su continuada actividad y a la diversidad de temas abordados. Estos foros no solo son reconocidos por su reputación como fuentes confiables de información, su relevancia en el contexto de la investigación y su demostrada capacidad para brindar conocimientos actualizados y experiencia experta en el ámbito de ROS y la virtualización de robots, sino también por su compromiso en la resolución de problemas de los usuarios, lo que añade un valor significativo a la información recopilada.

4.2.2 Recopilación de Datos

Se procedió a la recopilación de una muestra representativa de discusiones y debates que se encontraban directamente relacionados con el proceso de virtualización de robots en los foros seleccionados.

4.2.3 Limpieza de Datos

Se realizó un proceso de depuración exhaustivo en los datos, con el objetivo de eliminar cualquier información no deseada, lo cual incluyó la identificación y eliminación de duplicados y contenido incorrecto.

4.2.4 Análisis de Datos

Se llevó a cabo un análisis de frecuencia de palabras para identificar patrones clave en las discusiones. El análisis de frecuencia de palabras es una herramienta fundamental en el estudio de textos y análisis de datos lingüísticos. Esta técnica nos permite examinar la distribución y ocurrencia de palabras específicas dentro de un texto, revelando patrones y tendencias (Ver tabla 1).

Tabla 1. Frecuencia de palabras clave.

Palabra	Frecuencia
Linux	110
ROS	98
Gazebo	77
Rviz	63
MoveIt	41
URDF	39

Fuente: D'Agostini, J. y Nazar, S. (2023)

La tabla de frecuencia de palabras clave revela patrones significativos en las discusiones de la comunidad de ROS con respecto a la virtualización de los robots. Las palabras clave más mencionadas, como ROS, Gazebo, RViz y MoveIt, son indicativas de la importancia de estas herramientas y plataformas en el contexto de la virtualización de robots.

4.2.5 Resultados del Análisis

Después de analizar detenidamente las discusiones en los foros de ROS, se identificó una estrategia sólida para llevar a cabo la virtualización del robot Lab-Volt 5250 en el entorno ROS. Esta estrategia, respaldada por la comunidad de expertos, sigue un enfoque estructurado y efectivo que involucra el siguiente procedimiento:

- Iniciar el proceso con la creación de un modelo tridimensional (3D) del robot para obtener una representación visual completa.
- A continuación, transformar este modelo al formato URDF (Unified Robot Description Format), compatible con ROS.
- Para configurar el control de movimiento y la planificación, se emplea la herramienta MoveIt.
- RViz se utiliza como plataforma de visualización para verificar y validar los movimientos planificados.

- Por último, Gazebo ofrece un entorno de simulación completo y físico que permite probar el robot en un contexto virtual antes de su despliegue práctico en el mundo real.

El procedimiento descrito anteriormente se presenta de manera más clara y visual en el siguiente flujograma:

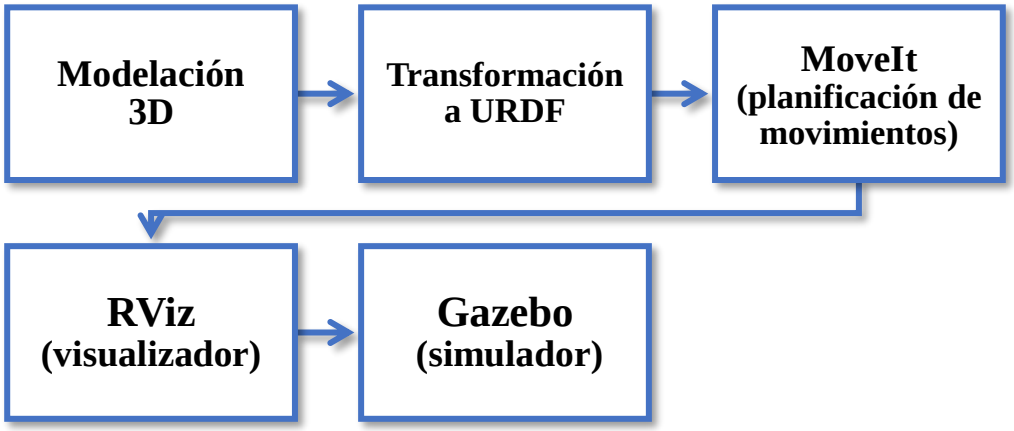


Figura 14. Flujograma de la virtualización.

Fuente: D'Agostini, J. y Nazar, S. (2023)

Al concluir esta fase preliminar de la investigación, hemos derivado conclusiones significativas del análisis de datos en los foros de ROS. Las tendencias identificadas indican de manera destacada un proceso sólido para la virtualización del robot Lab-Volt 5250. Estos resultados ofrecen una guía clara respaldada por la comunidad de ROS, que orienta de manera efectiva el desarrollo exitoso del proyecto de virtualización.

Fase III: Realización del modelado tridimensional del robot Lab-Volt 5250 y sus accesorios.

Iniciando con el proceso establecido en la fase anterior, la etapa actual de modelado tridimensional del robot Lab-Volt 5250 y sus accesorios se enfoca en aspectos cruciales, como las dimensiones de las piezas y su geometría, evitando detalles innecesarios que puedan afectar el rendimiento del modelo posteriormente en la simulación en ROS. Para lograr este objetivo, se empleó el software SolidWorks, una herramienta poderosa y versátil en el diseño y modelado tridimensional.

4.3.1 Recolección de datos

Para la realización de los modelos tridimensionales tanto del robot como de sus accesorios, se requirió en primera instancia recopilar todas las dimensiones necesarias para una representación precisa. Este proceso de adquisición de medidas se llevó a cabo a través de la creación de planos

detallados. La precisión y minuciosidad desempeñaron un papel crítico en esta etapa, ya que resultaron fundamentales para lograr una representación precisa de cada componente.

La elaboración detallada de planos requirió una estrategia que dividió sistemáticamente el robot en segmentos fácilmente distinguibles. En un principio, se abordó la base, seguida de la cintura y luego se continuó con el brazo y el antebrazo, como se muestra de la figura 15 a la 18. Posteriormente, se examinó cuidadosamente el gripper, el cual se subdividió en varias subpartes debido a su complejidad y abundancia de elementos, como se pueden apreciar en las figuras de la 10 a la 23. Una vez completado este proceso de dimensionamiento, se procedió al ensamblaje de todas las partes, tal como se exhibe en la figura 24, para obtener una visión completa y precisa del robot en su conjunto.

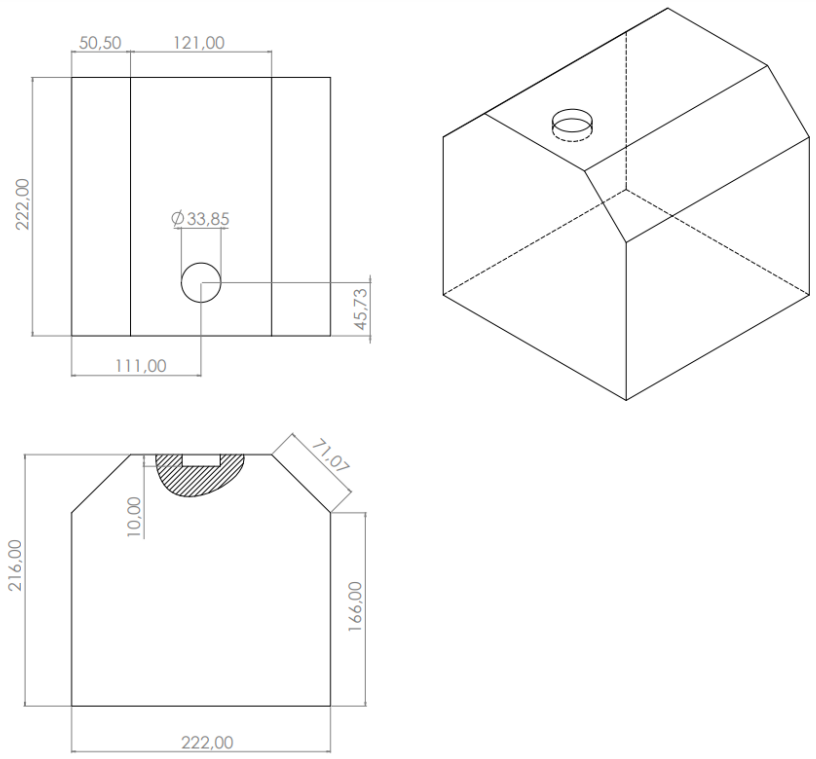


Figura 15. Base del robot Lab-Volt 5250 en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

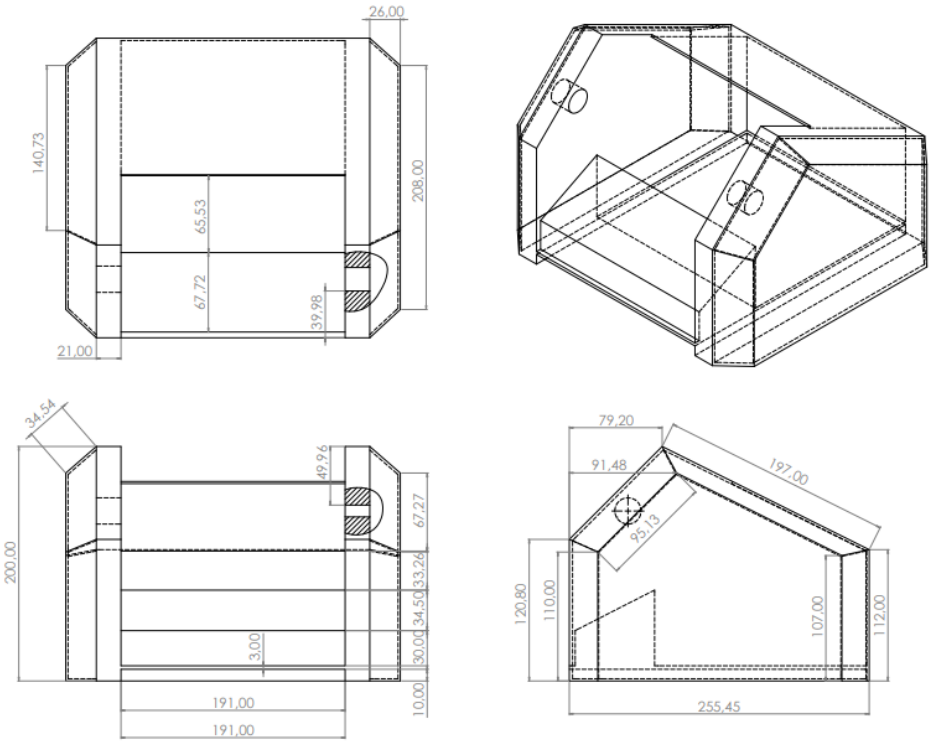


Figura 16. Cintura del robot Lab-Volt 5250 en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

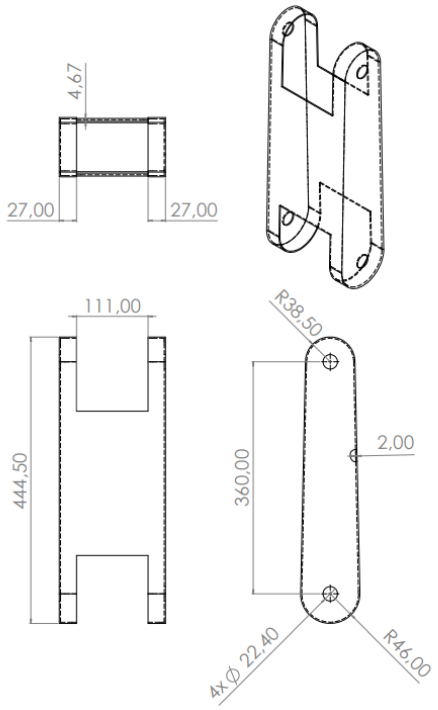


Figura 17. Brazo del robot Lab-Volt 5250 en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

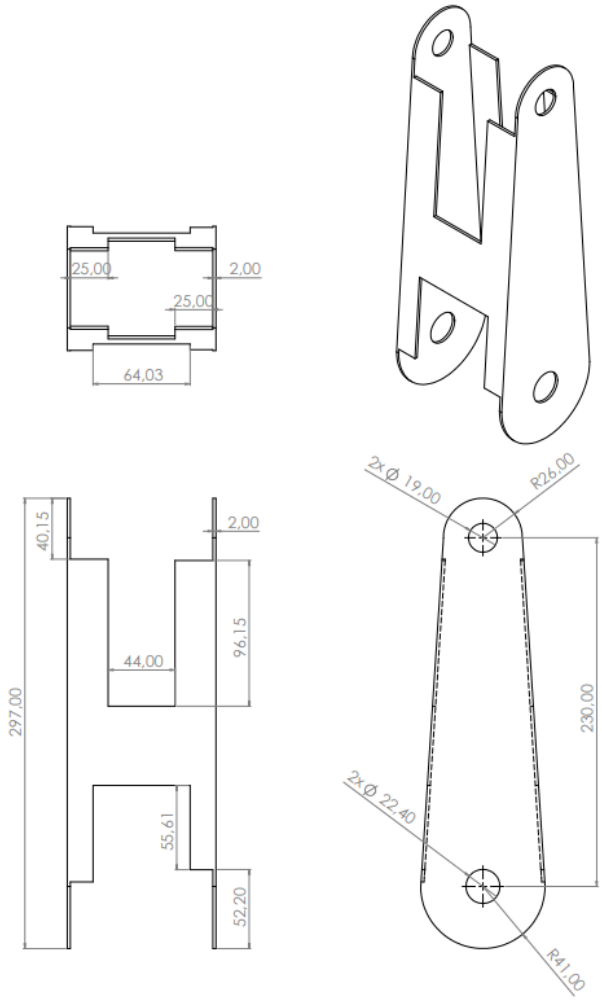


Figura 18. Antebrazo del robot Lab-Volt 5250 en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

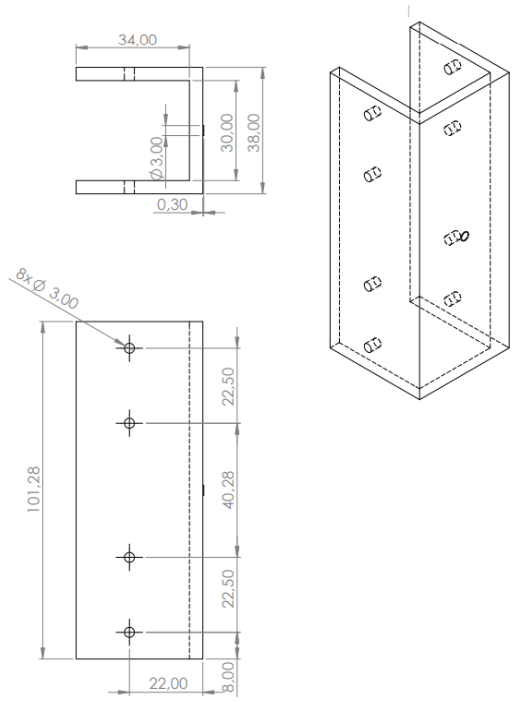


Figura 19. Base del gripper del robot Lab-Volt 5250 en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

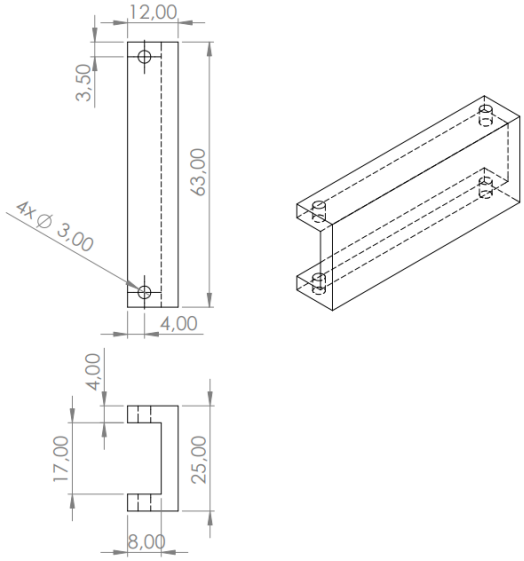


Figura 20. Uniones del gripper del robot Lab-Volt 5250 en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

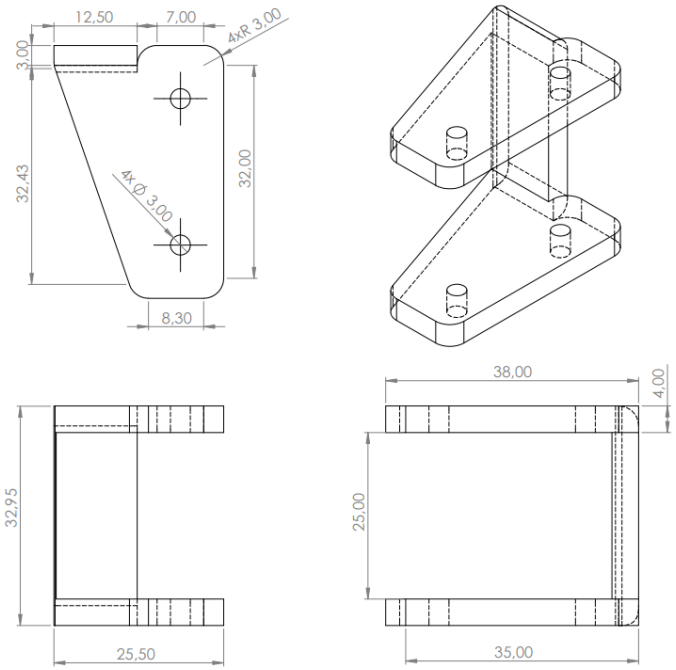


Figura 21. Base de terminal del gripper del robot Lab-Volt 5250 en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

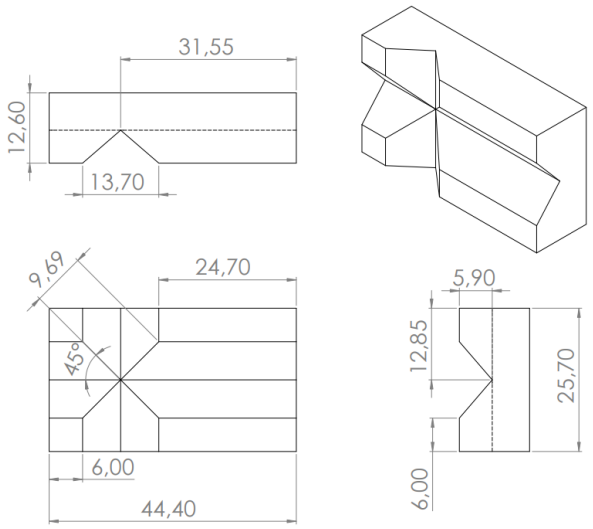


Figura 22. Terminal del gripper del robot Lab-Volt 5250 en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

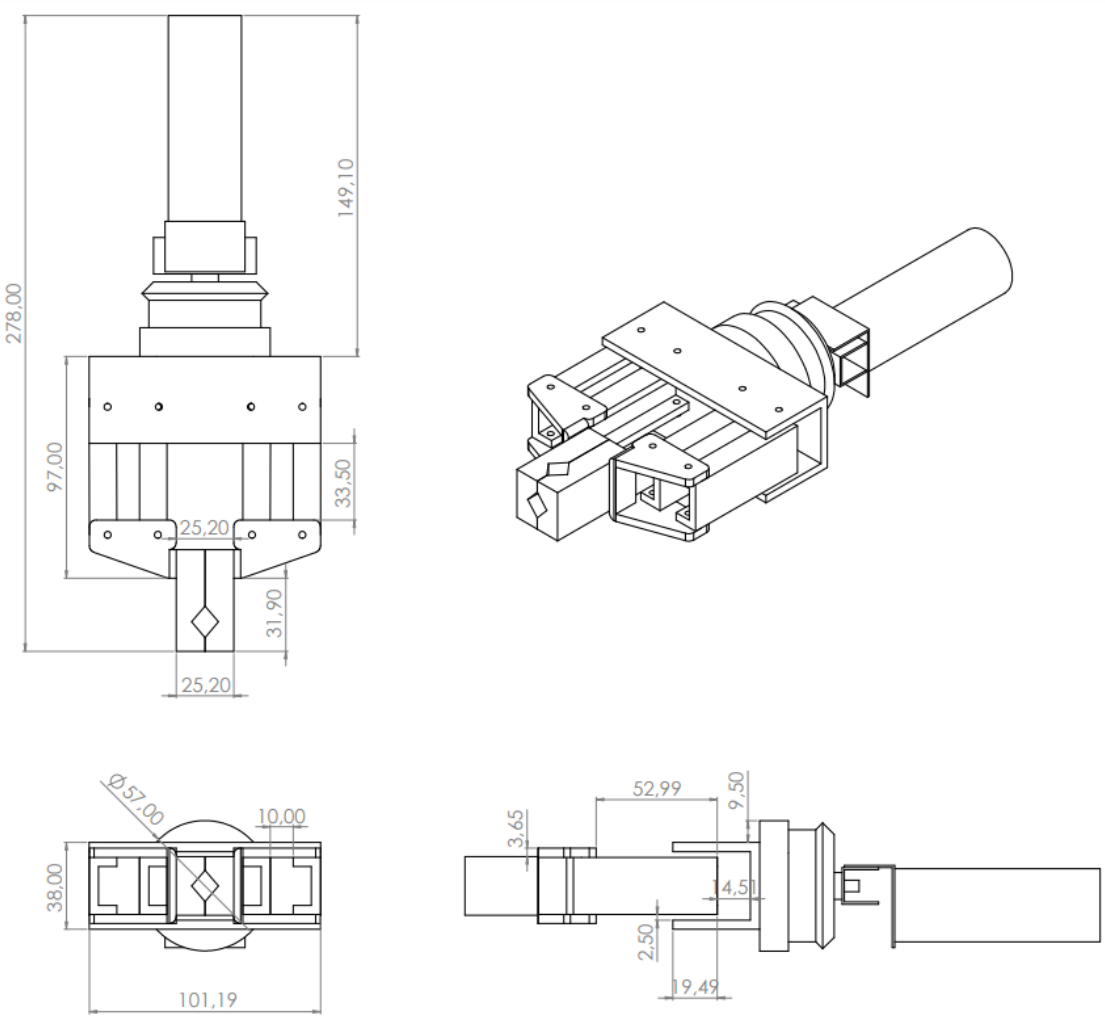


Figura 23. Ensamblaje del gripper del robot Lab-Volt 5250 en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

Durante el proceso de elaboración del plano de ensamblaje del robot Lab-Volt 5250, se puso un énfasis particular en la medición precisa de las distancias entre las articulaciones. Esta atención meticulosa a las distancias entre las articulaciones se justifica por su relevancia crítica en el desarrollo del modelo cinemático, donde cada valor es esencial para garantizar un funcionamiento preciso y coherente del robot en su conjunto.

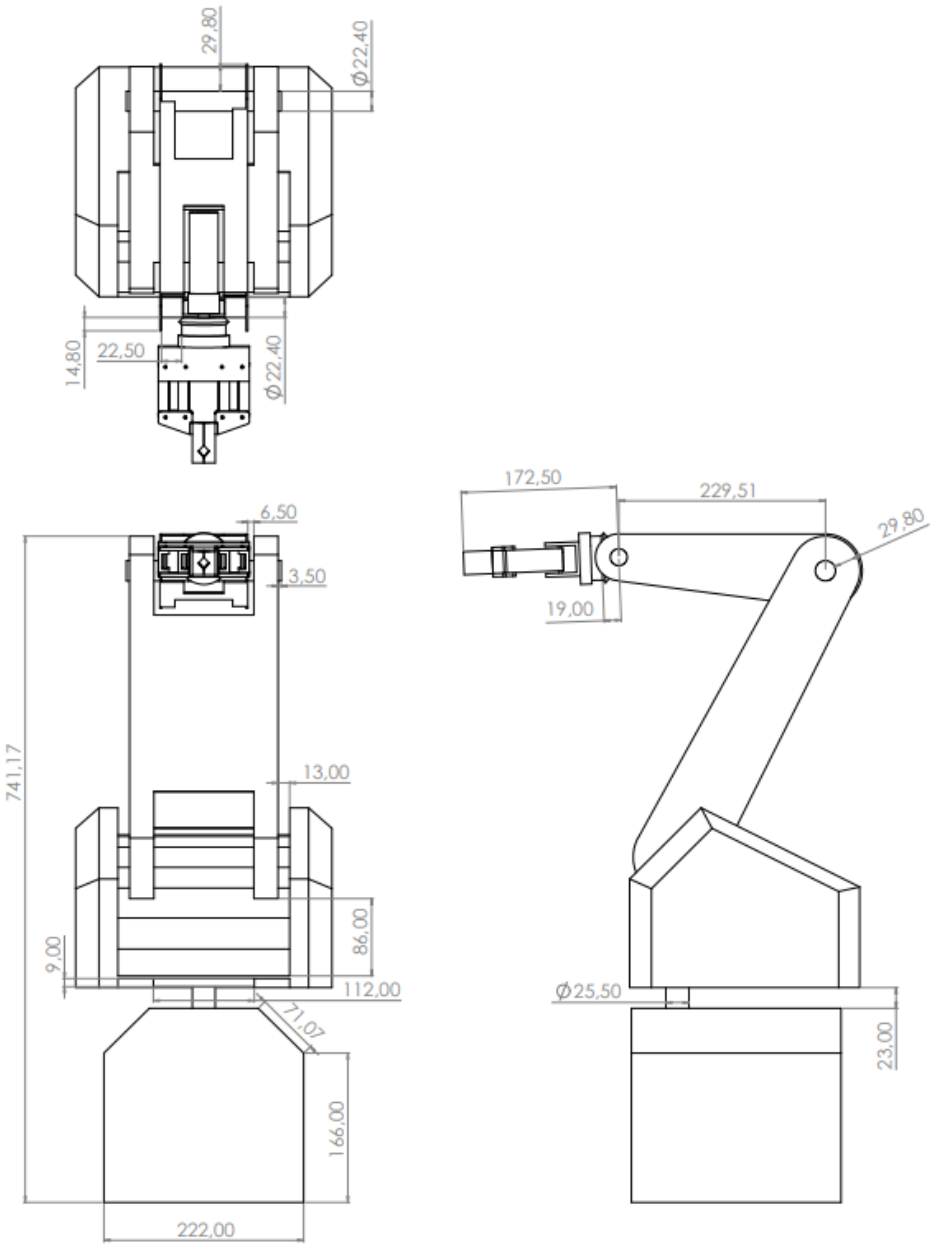


Figura 24. Ensamblaje del robot Lab-Volt 5250 en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

Para elaborar los planos de los accesorios, se inició con los alimentadores por gravedad, observando que tanto el alimentador cilíndrico como el cuadrado comparten una base idéntica, presentada en la figura 25. Además, se estudiaron los componentes por los cuales los objetos se desplazan, como se ilustra en las figuras 26 y 27. En contraste, la base del carrusel giratorio se reveló como idéntica a la del robot, mientras que su rueda contiene detalles específicos para acomodar objetos cilíndricos y cuadrados, como puede observarse en la figura 28. Finalmente, se

prestó especial atención a la longitud y el ancho de la banda transportadora, como se puede apreciar en la figura 29.

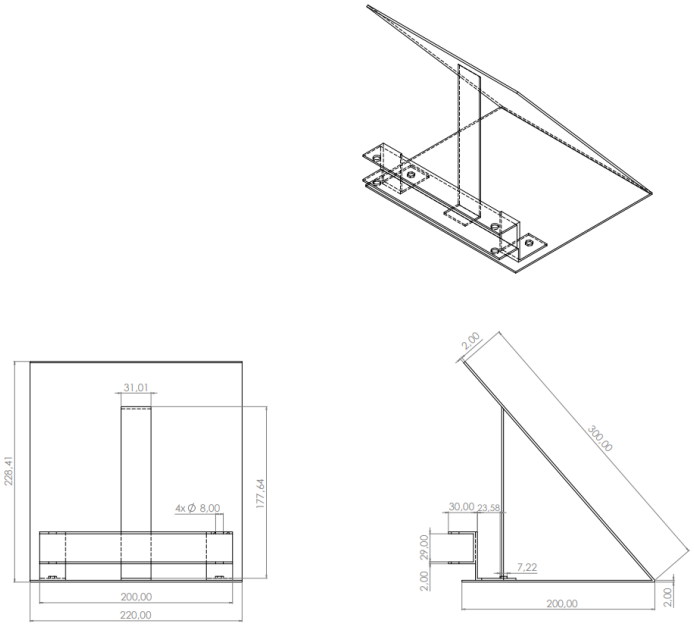


Figura 25. Base de los alimentadores por gravedad en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

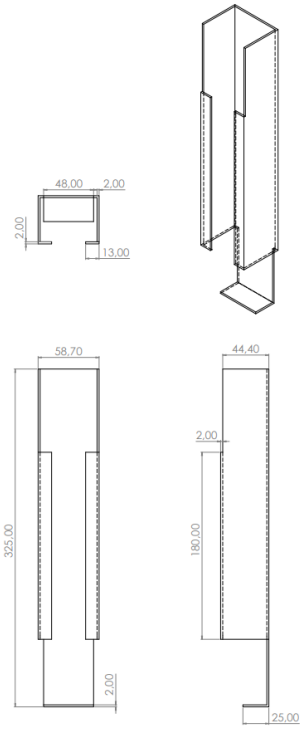


Figura 26. Alimentador por gravedad cuadrado en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

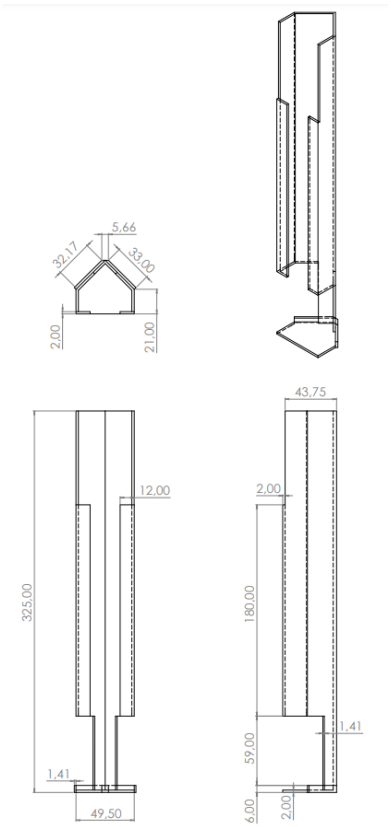


Figura 27. Alimentador por gravedad cilíndrico en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

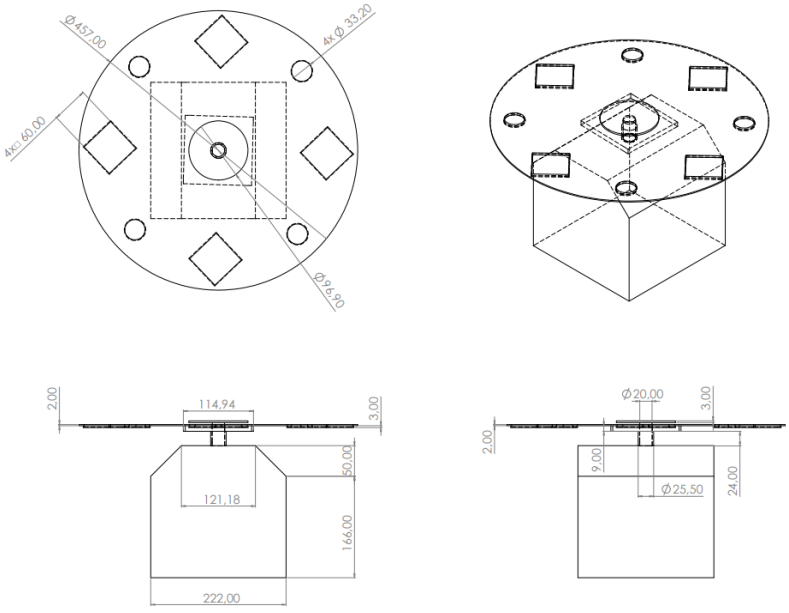


Figura 28. Carrusel giratorio en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

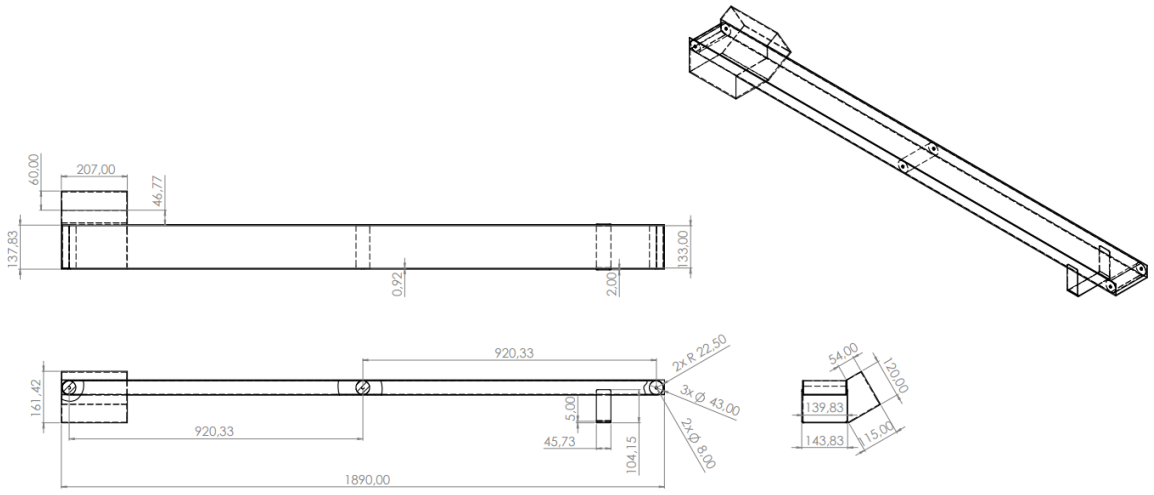


Figura 29. Banda transportadora en 2D

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.3.2 Modelado 3D del robot Lab-Volt 5250 y sus accesorios

Una vez reunidos con sumo cuidado todos los datos y dimensiones requeridos para la construcción del modelo tridimensional del robot Lab-Volt 5250 y sus respectivos accesorios, se procedió a utilizar con gran precisión el software SolidWorks. En esta fase, se plasmó de manera minuciosa cada uno de los datos recolectados con el propósito de lograr modelos que se asemejaran lo más posible a la realidad. Los resultados de este proceso pueden apreciarse detalladamente en las figuras del 30 al 34.

Figura 30. Modelo tridimensional del robot Lab-Volt 5250

Fuente: D'Agostini, J. y Nazar, S. (2023)

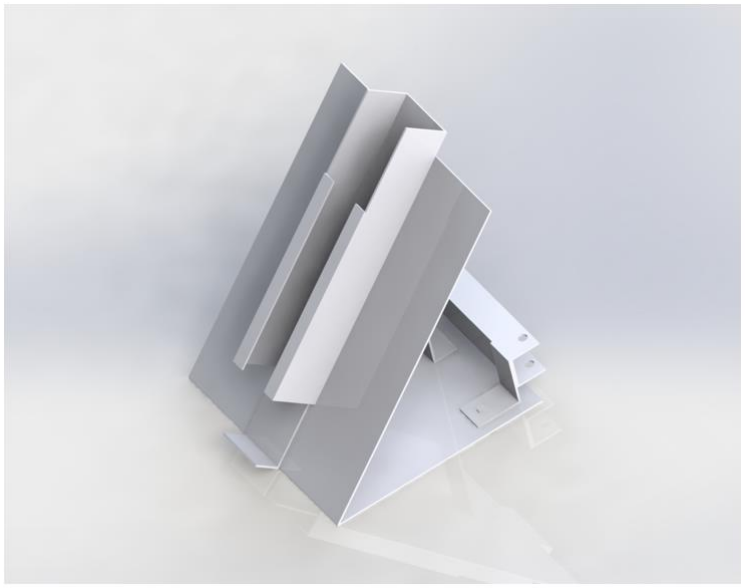


Figura 31. Modelo tridimensional del alimentador por gravedad cuadrado

Fuente: D'Agostini, J. y Nazar, S. (2023)

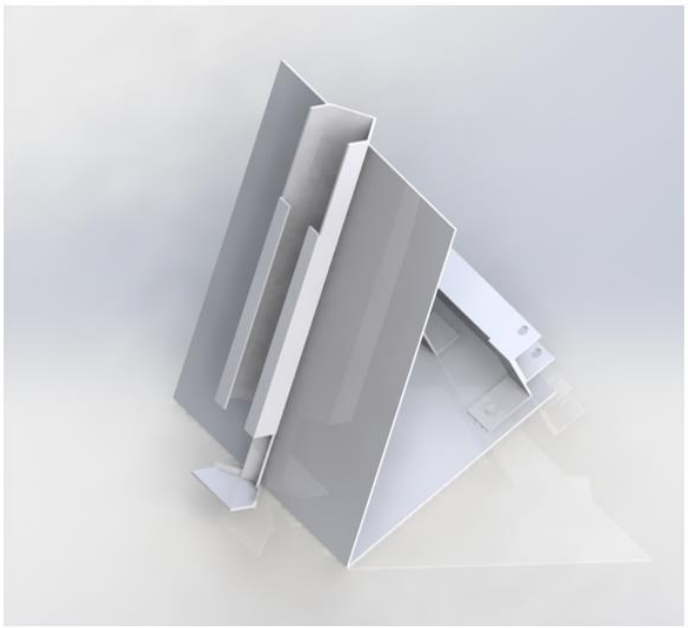


Figura 32. Modelo tridimensional del alimentador por gravedad cilíndrico

Fuente: D'Agostini, J. y Nazar, S. (2023)

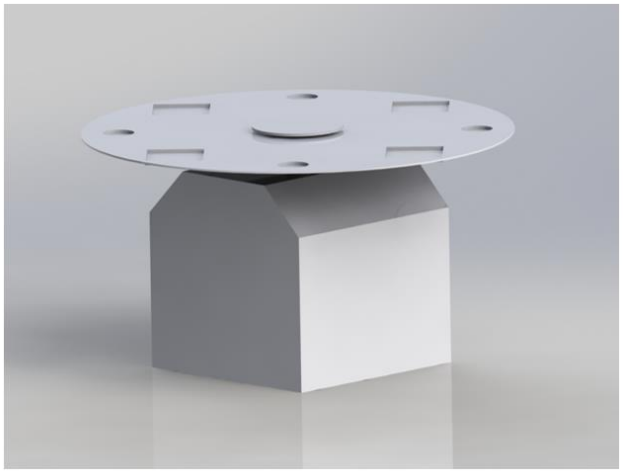


Figura 33. Modelo tridimensional del carrusel giratorio

Fuente: D'Agostini, J. y Nazar, S. (2023)

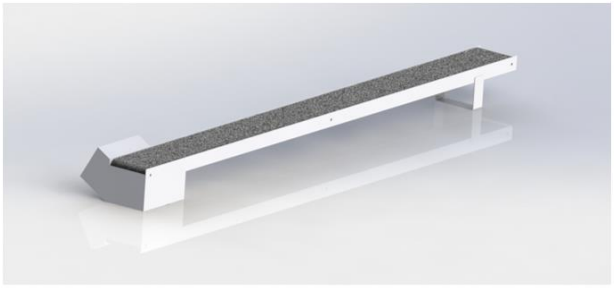


Figura 34. Modelo tridimensional de la banda transportadora

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.3.3 Exportación a URDF

El URDF, o "Unified Robot Description Format" en inglés, desempeña un papel esencial en el Robot Operating System (ROS). Es un formato de archivo utilizado para describir detalladamente la estructura, cinemática y dinámica de un robot. Esta descripción se realiza en un formato XML (Extensible Markup Language) que resulta fácilmente interpretable y procesable por las aplicaciones de ROS. Una de las principales razones para utilizar URDF en ROS es que permite la representación completa del robot. Esto incluye no solo sus enlaces (los eslabones del robot) sino también las articulaciones que los conectan. Esta representación es crucial para obtener una descripción precisa de la geometría y la cinemática del robot.

Otra función importante de URDF es la visualización. La descripción del robot en URDF se utiliza para crear modelos visuales del robot en aplicaciones como RViz (el Visualizador de ROS) y Gazebo (un simulador de robots). Esta representación visual es fundamental para la

planificación de movimientos y el análisis. URDF también desempeña un papel crucial en la planificación de trayectorias. Las bibliotecas de planificación de trayectorias en ROS emplean la información de URDF para calcular trayectorias seguras y eficientes para el robot. La precisión en la representación de la cinemática y la geometría del robot resulta esencial para este propósito.

4.3.3.1 Exportador de SolidWorks a URDF

El exportador de SolidWorks a URDF es una extensión de SolidWorks que simplifica el proceso de exportación de piezas y ensamblajes a un archivo URDF. Este complemento genera un paquete similar a ROS, que incluye un directorio para mallas, texturas y archivos URDF. Cuando se trata de piezas individuales, el exportador de piezas extrae automáticamente las propiedades del material y las integra en un único enlace dentro del URDF. Cuando se trabaja con ensamblajes en SolidWorks, el exportador despliega una estructura de enlaces y un árbol jerárquico que se basa en la configuración del ensamblaje. Además, el exportador tiene la capacidad de identificar automáticamente el tipo de unión necesario, determinar las transformaciones de unión adecuadas y establecer los ejes de referencia apropiados en el modelo URDF resultante, sin embargo, a través de la práctica se determinó que se obtienen mejores resultados al establecer los ejes de referencia de forma manual.

4.3.3.2 Pasos seguidos para exportar de SolidWorks a URDF

Paso 1: Preparación del Modelo en SolidWorks

Antes de iniciar el proceso de exportación, se requirió preparar detalladamente el modelo en SolidWorks. Esto incluyó asegurarse de que el diseño estaba completo y correctamente configurado, con todas las piezas y ensamblajes en su lugar. Además, fue fundamental que las propiedades de los materiales estuviesen asignadas adecuadamente a las piezas del modelo, ya que esto influye en la simulación y el comportamiento del robot.

Paso 2: Instalación del Complemento SW2URDF

Se instaló el complemento SW2URDF, que es una herramienta esencial para la exportación exitosa. Esto implicó descargar el complemento desde su sitio web oficial y seguir las instrucciones de instalación proporcionadas. Es importante que el complemento esté correctamente configurado en SolidWorks antes de proceder. Una vez instalado, se activó el complemento SW2URDF en SolidWorks. Esto se logró accediendo a la pestaña "Complementos" en el software y seleccionando "SW2URDF" para activar el complemento. Después de esta acción, una nueva pestaña llamada

"SW2URDF" o “Export as URDF” apareció en la opción “File” de la pestaña de herramientas de SolidWorks.

Paso 3: Establecimiento de la posición inicial

Para empezar el proceso de creación del URDF, se requirió la previa colocación de todos los enlaces en su posición inicial. Esto se logró mediante relaciones de posición que restringieron el modelo a la posición cero.

Paso 4: Establecimiento de los sistemas de referencia

Un sistema de referencia para cada articulación permite definir la posición y orientación relativa de cada articulación en el robot. Esto es esencial para realizar cálculos de cinemática directa e inversa, lo que implica determinar la posición y orientación de los extremos del robot (end-effectors) o de otros eslabones en función de las variables articulares. Durante la exportación de modelos tridimensionales a URDF desde SolidWorks, se ofrece la opción de establecer automáticamente los sistemas de referencia para cada articulación. Sin embargo, en este proyecto de investigación, se determinó que dicha opción no era óptima, ya que distorsionaba la posición de los eslabones del modelo. Por lo tanto, se optó por establecer manualmente un sistema de referencia en el punto medio de cada articulación y en el centro de la cara inferior de la base del robot, como se muestra en la figura 35, siguiendo las mismas orientaciones para los ejes x, y, z en todos los sistemas de referencia con el fin de garantizar una representación precisa del robot.

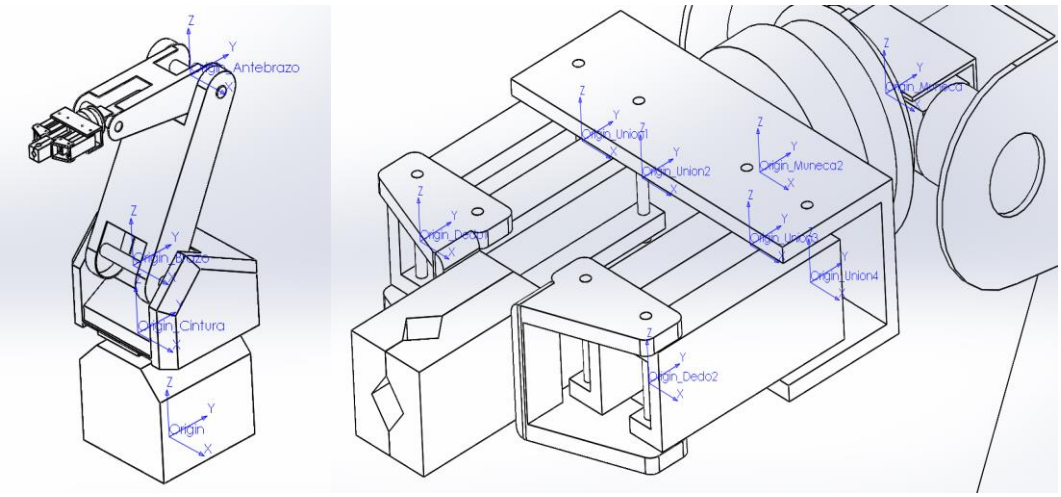


Figura 35. Sistemas de referencia en SolidWorks

Fuente: D’Agostini, J. y Nazar, S. (2023)

Paso 5: Establecimiento de los ejes de rotación

Tanto los sistemas de referencia como los ejes de rotación desempeñan un papel esencial en la descripción y cinemática de un robot. En el caso específico del robot Lab-Volt 5250, se compone de cinco articulaciones rotatorias generales, según se ilustra en la figura 36. Además, el gripper del robot también presenta articulaciones rotatorias que controlan su apertura y cierre, las cuales se visualizan en la figura 37. Para garantizar una representación precisa, se estableció un eje de rotación en el centro de cada una de estas articulaciones, atravesándolas completamente. Esto permitió definir claramente la dirección en la que estas articulaciones pueden girar, lo que resulta crucial para la cinemática y el control preciso del robot.

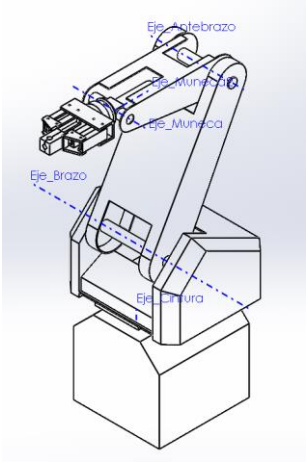


Figura 36. Ejes de rotación en el robot

Fuente: D'Agostini, J. y Nazar, S. (2023)

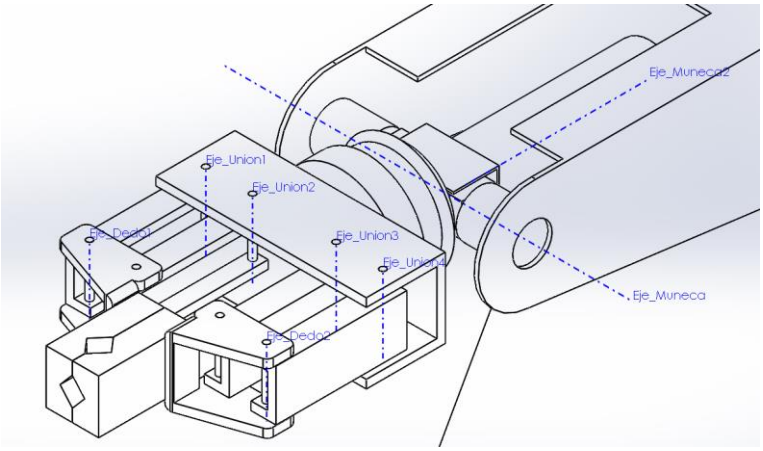


Figura 37. Ejes de rotación en el gripper

Fuente: D'Agostini, J. y Nazar, S. (2023)

Paso 6: Configuración de los eslabones y articulaciones

Dentro de la pestaña "Export as URDF", se configuraron y organizaron los eslabones y articulaciones. En las figuras 38 y 39, se presentan como referencia los nombres asignados a cada eslabón y articulación, lo que resulta esencial para comprender el procedimiento posterior.

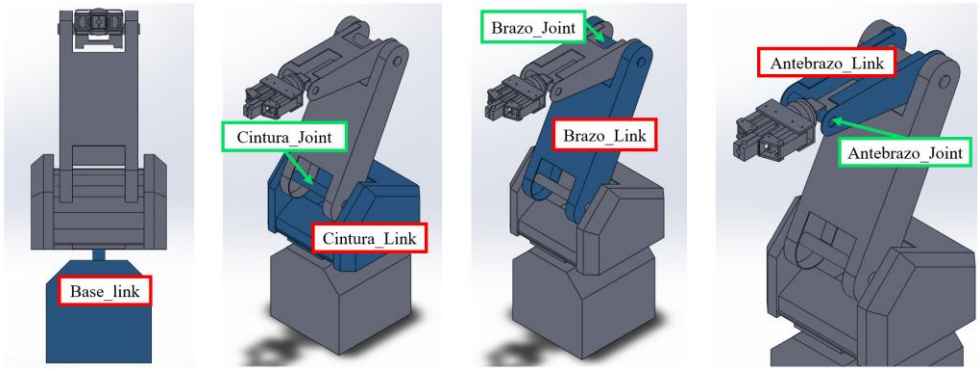


Figura 38. Nombres eslabones y articulaciones del robot

Fuente: D'Agostini, J. y Nazar, S. (2023)

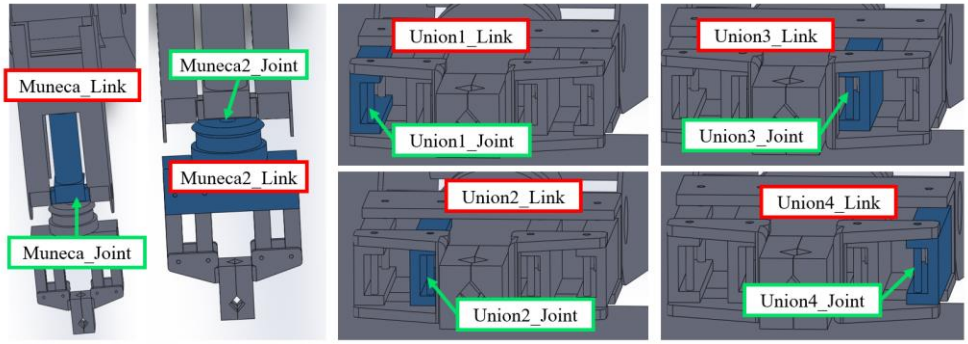


Figura 39. Nombres eslabones y articulaciones del gripper (1)

Fuente: D'Agostini, J. y Nazar, S. (2023)

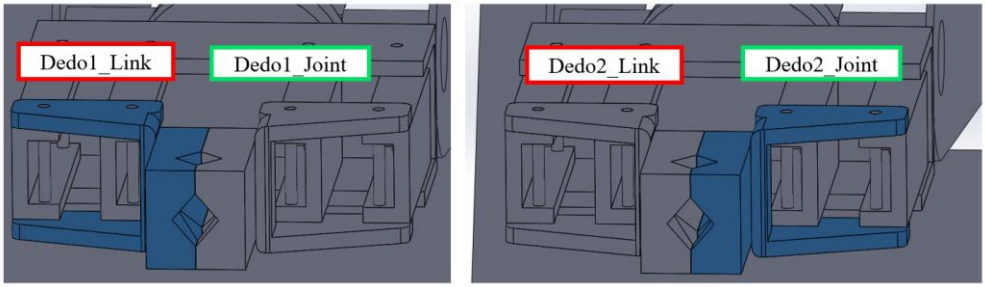


Figura 40. Nombres eslabones y articulaciones del gripper (2)

Fuente: D'Agostini, J. y Nazar, S. (2023)

En el panel "URDF Exporter," se dividen en 8 segmentos que deben completarse de acuerdo con las necesidades del modelo, tal como se muestra en la figura 40.

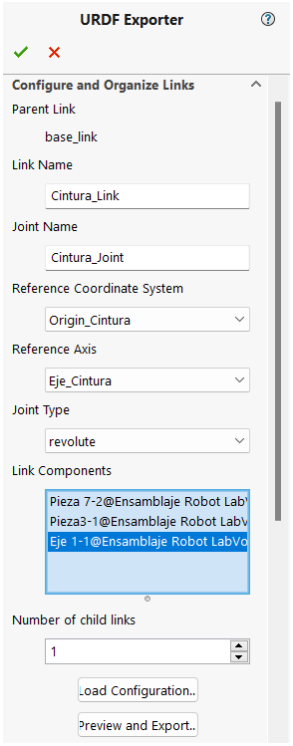


Figura 41. Segmentos del URDF Exporter

Fuente: D'Agostini, J. y Nazar, S. (2023)

El primer segmento, "Parent Link" o enlace padre, se refiere al eslabón que precede a otro en la cadena cinemática y se conecta mediante una articulación. Este segmento se llena automáticamente según la información proporcionada. A continuación, se encuentran los segmentos "Link Name" y "Joint Name," donde se asignaron los nombres deseados a los eslabones y articulaciones seleccionados. Luego, los segmentos "Reference Coordinate System" y "Reference Axis" permitieron la selección de los sistemas y ejes de referencia previamente establecidos, mostrados en la figura 35, correspondientes a los elementos seleccionados.

Posteriormente, el segmento "Joint Type" permitió elegir el tipo de articulación requerida en función del movimiento que el conjunto debe realizar. En el caso del robot Lab-Volt 5250, todas las articulaciones son del tipo "revolute," lo que posibilita el movimiento de rotación alrededor del eje de referencia seleccionado. Después, en el segmento "Link Components," se seleccionaron todos los componentes que formarán parte del conjunto que se está estudiando. Finalmente, en el segmento "Number of Children," se registraron el número de elementos que se derivan del eslabón que se está configurando. Una vez configurados y organizados los eslabones y las articulaciones, en la parte inferior del panel "URDF Exporter," se muestra la estructura jerárquica resultante que

exhibe todos los eslabones, sus precedentes y los componentes que derivan de ellos. En la figura 42 se muestra la estructura jerárquica del robot Lab-Volt 5250 definida en el presente trabajo de investigación.

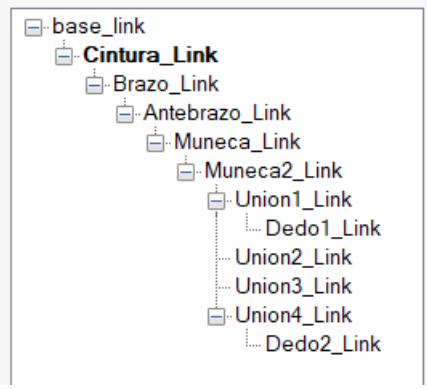


Figura 42. Estructura jerárquica de eslabones

Fuente: D'Agostini, J. y Nazar, S. (2023)

Paso 7: Previsualizar

La previsualización del archivo URDF que se generó posteriormente, se llevó a cabo mediante la opción "Preview and Export". Esta función permite una inspección visual y, en caso necesario, corrección de la información que estará contenida en el archivo URDF, tanto para los eslabones como para las articulaciones. En el presente trabajo de investigación se optó por realizar las correcciones directamente en el archivo URDF generado.

Paso 8: Generación del Archivo URDF

Una vez que las configuraciones de exportación fueron ajustadas, se seleccionó la opción "Export URDF and Meshes" (Exportar URDF y mallas) para que el complemento SW2URDF comenzara a procesar el modelo de SolidWorks y generara un archivo URDF que describiera el robot. Es importante revisar el archivo resultante para verificar que todo se haya exportado correctamente.

Para el resto de los modelos tridimensionales que componen el entorno de simulación del robot Lab-Volt 5250, se aplicaron procedimientos similares. En cuanto a los componentes estáticos, se definió un sistema de referencia en el centro de sus superficies inferiores, un ejemplo de ello es la banda transportadora, los alimentadores por gravedad y la superficie perforada. Estos elementos se representaron con un solo eslabón y, debido a su inmovilidad, no se requirió definir ejes de rotación, ya que carecen de articulaciones. Esta metodología asegura una correcta

descripción de los modelos en el contexto de la simulación y contribuye a la representación precisa de su disposición espacial y cinemática.

4.3.4 Descripción del paquete de ROS exportado desde SolidWorks

El paquete de ROS exportado desde SolidWorks es un conjunto de archivos que contiene la información necesaria para representar un robot en ROS. El paquete consta de un archivo URDF, un directorio de mallas, un directorio de texturas y un directorio de lanzamientos como se muestra en la figura 43.

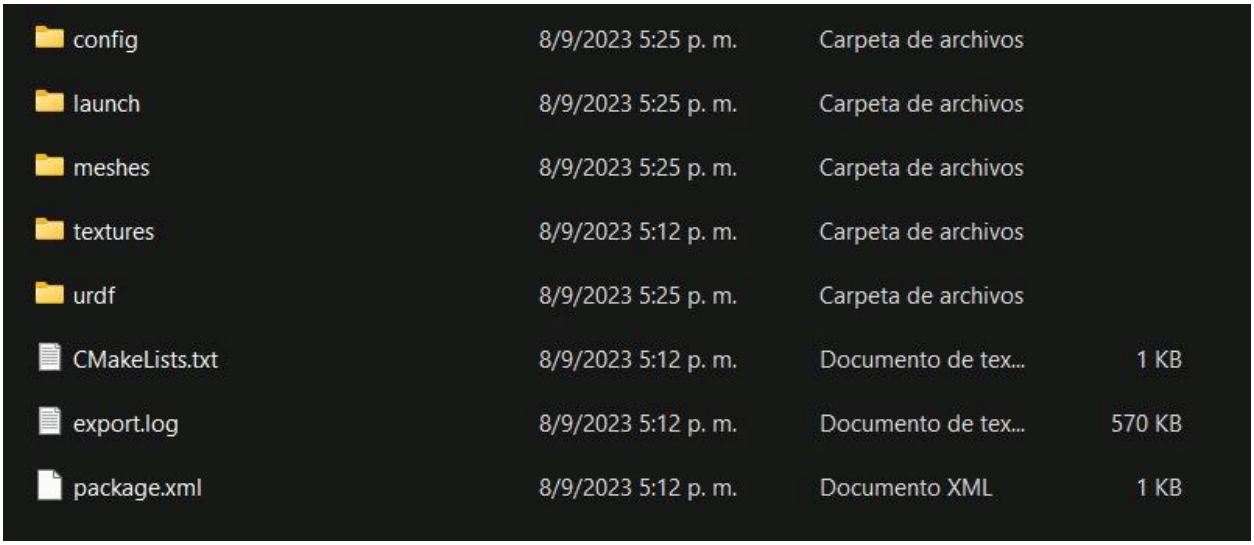


Figura 43. Directorio del paquete exportado de SolidWorks

Fuente: D'Agostini, J. y Nazar, S. (2023)

- **config:** La carpeta se compone de un archivo llamado "joint_names_NombreDelArchivo.yaml". Este archivo contiene los nombres de las articulaciones presentes en el URDF generado. Estas articulaciones son esenciales para el control del robot y, por lo tanto, se denominan "controller_joint_names" tal como se exhibe en la figura 44.

```
controller_joint_names: ['', 'Cintura_Joint', 'Brazo_Joint', 'Antebrazo_Joint', 'Muneca_Joint', 'Muneca2_Joint',
'Union1_Joint', 'Dedo1_Joint', 'Union2_Joint', 'Union3_Joint', 'Union4_Joint', 'Dedo2_Joint', 'Carrusel_Joint',
'Bandainv_joint']
```

Figura 44. Código de la carpeta **"**controller_joint_names"

Fuente: D'Agostini, J. y Nazar, S. (2023)

- **launch:** Está formada por dos archivos. "display.launch" que se utiliza para abrir el modelo tridimensional en Rviz, y "gazebo.launch", el cual se utiliza para lanzar el modelo tridimensional en Gazebo.

- **meshes:** Contiene las mallas de los componentes del modelo tridimensional. Las mallas son objetos tridimensionales que se utilizan para representar la forma geométrica de los componentes del robot. Pueden estar representadas en diferentes formatos, como STL, OBJ o PLY. En el contexto de este proyecto de investigación, se han incluido archivos ".stl" correspondientes a todos los enlaces del modelo.
- **textures:** Comprende las texturas de los componentes del robot. Las texturas son imágenes que se utilizan para representar el color y la textura de los componentes del robot. Pueden estar representadas en diferentes formatos, como PNG, JPEG o TIFF. En la presente investigación, dicha carpeta con contiene ningún archivo.
- **URDF:** Esta carpeta contiene un archivo URDF del modelo. Como se describió anteriormente, este archivo es una descripción del modelo tridimensional que está compuesto por información de eslabones, articulaciones, posiciones y rangos de movimiento de las articulaciones, sus propiedades de masa, propiedades inerciales, entre otros. Además, esta carpeta contiene un archivo nombre_del_archivo.csv que contiene información de los enlaces de tu robot.
- **CMakeLists.txt:** Este archivo es la entrada al sistema de construcción CMake para construir paquetes de software. Describe cómo construir el código y dónde instalarlo, además, define dependencias, configuraciones de construcción y otros ajustes necesarios para compilar y construir el paquete del robot. No se debe renombrar o cambiar la secuencia de código en este archivo.
- **export:** Contiene los registros y archivos temporales o intermedios generados durante la creación del paquete en SolidWorks, como archivos STL utilizados en la creación de las mallas visuales, entre otros.
- **package.xml:** Este archivo define propiedades sobre el paquete como el nombre del paquete, números de versión, descripción, autores, mantenedores, dependencias y otras etiquetas importantes que son utilizadas por el sistema de construcción de ROS (Catkin). Si las dependencias faltan o son incorrectas, el paquete puede o no funcionar.

4.3.4 Modelo URDF

A pesar de que SolidWorks genera automáticamente el archivo URDF, es esencial profundizar en la comprensión de los componentes que lo conforman. En este contexto, se llevará a cabo una descripción detallada del modelo del robot Lab-Volt 5250 en formato URDF, iniciando

con la primera línea de código: <robot name="nombre_del_archivo">. Esta línea establece el elemento <robot>, que funciona como el nodo principal del archivo URDF, y el atributo "name" asigna un nombre al modelo.

El enfoque adoptado para representar el robot implica considerarlo como un conjunto de eslabones o cuerpos (links) conectados a través de articulaciones (joints), como se evidencia en las figuras 38, 39 y 40. Cada articulación o elemento link proporciona una descripción detallada de un cuerpo rígido en términos de su inercia, representación visual y representación de colisión. En consecuencia, cada eslabón se descompone en tres bloques independientes, cada uno caracterizado por una serie de parámetros y especificaciones. La figura 45 brinda una visión completa de todos los parámetros que definen el eslabón correspondiente a la cintura del robot Lab-Volt 5250.

```
<link name="Cintura_Link">
  <inertial>
    <origin xyz="-8.3267E-17 0.052188 0.063397" rpy="0 0 0" />
    <mass value="2.8951" />
    <inertia ixx="0.020805" ixy="2.1115E-18" ixz="4.1873E-18" iyy="0.031845" iyz="-0.00098546" izz="0.040126" />
  </inertial>
  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://labvolt/meshes/Cintura_Link.STL" />
    </geometry>
    <material name="">
      <color rgba="0.79216 0.81961 0.93333 1" />
    </material>
  </visual>
  <collision>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <mesh filename="package://labvolt/meshes/Cintura_Link.STL" />
    </geometry>
  </collision>
</link>
```

Figura 45. Código para los eslabones en el URDF

Fuente: D'Agostini, J. y Nazar, S. (2023)

Bloque inercial

<inertial>: En esta sección se definieron las propiedades inerciales del eslabón, incluyendo masa y tensor de inercia. Estas propiedades son esenciales para la dinámica del robot.

<origin>: Indica la posición y orientación del sistema de referencia de inercia en relación con el sistema de referencia del eslabón. Obligatoriamente debe estar en el centro de masa del cuerpo. La orientación es representada por rpy (roll, pitch e yaw) en radianes.

<mass>: Valor de la masa total del cuerpo.

<inertia>: Matriz de inercia medida en el centro de masa y respecto al sistema de referencia. ixx, ixy, ixz, iyy, iyz e izz representan los componentes del tensor de inercia.

Bloque visual

<visual>: Define cómo se ve el eslabón en la representación visual.

<origin>: La posición y orientación del sistema de referencia visual en relación con el sistema de referencia del eslabón. Para el caso en estudio, el sistema de referencia visual coincide con el sistema de referencia del eslabón para todos los eslabones.

<geometry>: La representación visual del cuerpo se logra mediante el uso de comandos como "box," "cylinder," o "sphere" para definir una geometría simple. No obstante, en el contexto de este proyecto, se optó por reemplazar estas formas geométricas por mallas tridimensionales (meshes) que están contenidas en el paquete exportado desde SolidWorks.

<material>: Define el material y el color del cuerpo en formato rgba, siendo la última letra referente a la transparencia (α) entre 0 y 1.

Bloque de colisión

<collision>: Describe la representación de colisión del eslabón. Los parámetros tienen significados equivalentes a los del bloque visual, e igualmente es posible incluir una malla (mesh). Téngase en cuenta que es posible definir un cuerpo cuyo bloque de colisión sea totalmente diferente al bloque visual, por lo que se podrían hacer cuerpos con una malla de colisión más grande al de la visual para considerar posibles componentes que no se han incluido en el visual como se muestra en la figura 46 o, por el contrario, hacer un cuerpo que nunca colisione porque no es de interés por algún motivo.

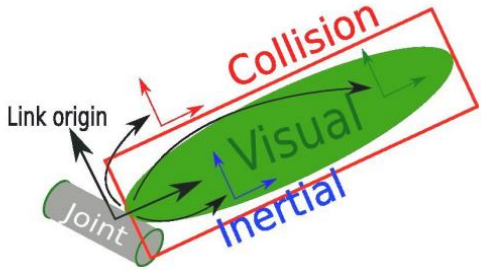


Figura 46. Componentes del URDF

Fuente: D'Agostini, J. y Nazar, S. (2023)

Por otro lado, se describen las articulaciones o joints que especifican el tipo de unión entre dos links, así como sus límites y propiedades dinámicas. En la figura 47 se observa la definición de los parámetros para la articulación que permite el movimiento de la cintura del robot Lab-Volt 5250.

```
59  <joint name="Cintura_Joint" type="revolute">
60    <origin xyz="0 -0.065274 0.243" rpy="0 0 0" />
61    <parent link="base_link" />
62    <child link="Cintura_Link" />
63    <axis xyz="0 0 1" />
64    <limit lower="-3.142" upper="3.142" effort="300" velocity="3" />
65  </joint>
```

Figura 47. Código de las articulaciones en el URDF

Fuente: D'Agostini, J. y Nazar, S. (2023)

En la etiqueta que abre el joint hay que especificar, además del nombre que se le dé a la articulación, el tipo de ésta, pudiendo ser:

- fixed: Articulación fija en la que no existe ningún grado de libertad.
- revolute: Articulación con un grado de libertad de revolución en torno a un eje con un rango limitado de giro.
- continuous: Equivalente al tipo revolute pero con un rango de giro sin límites.
- prismatic: Articulación de un grado de libertad de desplazamiento sobre un eje dentro de un rango limitado.
- planar: Articulación con 2 grados de libertad que permite el movimiento en un plano perpendicular al eje que se especifique.
- floating: Esta realmente no sería una articulación pues permitiría el movimiento libre en los 6 grados de libertad.

Las etiquetas internas de cada joint tienen los siguientes significados y argumentos:

<origin>: Indica la posición y orientación relativa de la articulación con respecto al eslabón padre.

<parent> y <child>: Links padre e hijo que une la articulación.

<axis>: Eje rotación, de translación o perpendicular al plano de la articulación especificado en el sistema de referencia de ésta.

<limit>: Limites superior e inferior del rango de movimiento (en m para las prismáticas y rad para las de revolución), velocidad máxima (en m/s o rad/s) y esfuerzo máximo aplicado (en N o Nm) en la articulación.

Para los componentes estáticos que se encuentran anclados al suelo, se definió un enlace denominado "world" o "mundo". Este enlace especial sirvió como un marco de referencia absoluto o global para todo el modelo del robot. Estos componentes están conectados al mundo mediante

una articulación de tipo "fixed," donde el enlace padre es el mundo, tal como se observa en la figura 48. Esta articulación fija asegura que la base del robot se mantenga inmóvil y no pueda moverse. Sin embargo, durante la práctica se observó que al unir los componentes directamente al mundo, pueden ocurrir colisiones no deseadas. Por lo tanto, se tomó la precaución de colocar estos componentes a una distancia inferior a medio milímetro del mundo, con el fin de evitar posibles errores de colisión.

```
5   <link name="world"/>
6   <joint name="base_joint" type="fixed">
7     <parent link="world"/>
8     <child link="base_link"/>
9     <origin rpy="0 0 0" xyz="0.0 0.0 0.039"/>
10  </joint>
```

Figura 48. Código para anclar la base del robot al mundo en el URDF

Fuente: D'Agostini, J. y Nazar, S. (2023)

Posteriormente, se agregaron al URDF las transmisiones o "transmissions", los cuales son elementos utilizados para describir la relación entre las articulaciones (joints) y los actuadores en un robot. Las "transmissions" se emplean para modelar cómo la energía se transmite desde los actuadores (como motores eléctricos o neumáticos) a las articulaciones, lo que permite que estas últimas se muevan. Son particularmente útiles para especificar la relación entre las fuerzas o velocidades aplicadas por los actuadores y el movimiento resultante en las articulaciones. Esto es esencial en la simulación y el control del robot, ya que permite que el modelo URDF refleje de manera precisa la dinámica del sistema real. En la figura 49 se puede observar el código de la transmisión del componente brazo del robot Lab-Volt5250.

```
398 <transmission name="Brazo_Link_trans">
399   <type>transmission_interface/SimpleTransmission</type>
400   <joint name="Brazo_Joint">
401     <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
402     </joint>
403   <actuator name="Brazo_Link_motor">
404     <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
405     <mechanicalReduction>1</mechanicalReduction>
406   </actuator>
407 </transmission>
```

Figura 49. Código para las transmisiones en el URDF

Fuente: D'Agostini, J. y Nazar, S. (2023)

Al igual que los eslabones y las articulaciones, las transmisiones tiene una serie de parámetros que se deben definir:

<transmission name=" Nombre_de_la_transmision">: Identificación y asignación de nombre al mecanismo de transmisión.

<type>: Este atributo especifica el tipo de "transmission". Puede ser “Simple", "Screw", "Continuous", "Gearbox", entre otros. Cada tipo se usa para modelar una relación específica entre el actuador y la articulación. Todas las transmisiones utilizadas en el proyecto son de tipo “simple” (transmission_interface/SimpleTransmission), el cual es comúnmente utilizado cuando la relación entre la entrada y la salida es lineal y sencilla, esto significa que cuando el actuador se mueve o gira en una cierta cantidad, la articulación controlada se mueve o gira de manera proporcional y directa, sin complicaciones o cambios significactivos en la relación entre ambas.

<joint name="Nombre_Joint">: Permite definir la articulación que está siendo controlada por el mecanismo de transmisión.

<hardwareInterface>: Especifica la interfaz de hardware que se utiliza para controlar la articulación a través de comandos que permiten controlar el parámetro deseado. Pueden ser las siguientes:

- hardware_interface/PositionJointInterface: Se utiliza cuando se desea controlar la posición de una articulación.
- hardware_interface/VelocityJointInterface: Esta interfaz se usa para controlar la velocidad de una articulación.
- hardware_interface/EffortJointInterface: Se utiliza para controlar el esfuerzo (fuerza/torque) ejercido por una articulación.
- hardware_interface/PositionVelocityJointInterface: Esta interfaz combina el control de posición y velocidad.
- hardware_interface/PositionEffortJointInterface: Combina el control de posición y esfuerzo ejercido por una articulación.
- hardware_interface/JointStateInterface: Esta interfaz se utiliza para recibir información de estado de las articulaciones, como posición, velocidad y esfuerzo.

En el contexto de las transmisiones empleadas en el proyecto actual, se hizo uso de "hardware_interface/PositionJointInterface". Esto indica que la articulación en cuestión se

controla a través de la interfaz de posición. Esto implica la capacidad de establecer la posición deseada de la articulación con precisión.

<actuator name="Nombre_motor">: Define el actuador asociado a la articulación en estudio, el cual proporciona la entrada de control.

<mechanicalReduction>: Aquí se estableció la reducción mecánica del actuador. En el caso de la articulación del brazo del robot mostrado en la figura 49 y en todas las transmisiones del URDF, la reducción mecánica es 1, lo que significa que la entrada del actuador se traduce directamente en movimiento de la articulación sin una reducción significativa.

El enfoque específico aplicado al modelado de la banda transportadora representó un caso particularmente interesante en el contexto de ROS. Dado que el movimiento de una banda transportadora en ROS no se define de manera convencional, se desarrolló una estrategia para simular este componente crucial. Esta estrategia implica la creación de un eslabón al que se denominó Bandainv_link. Este eslabón se materializó mediante una lámina extremadamente delgada y se configura utilizando el comando <box size>. Es importante señalar que, debido a sus características específicas, este eslabón no es visible en la simulación; por lo tanto, se optó por dejar toda la sección de visualización en forma de comentario en el código.

Para posibilitar el movimiento de la banda transportadora, se introdujo una articulación también invisible, que se nombró como Bandainv_joint. Esta articulación se clasificó como prismática y juega un papel fundamental en la simulación del movimiento de la banda. El código correspondiente a esta parte no visible de la banda transportadora, que habilita su movimiento, se presenta a continuación:

```
      <link name="Bandainv_link">
          <inertial>
            <origin         xyz="-0.00504397395261535         0.739301055593877
0.0969045302044948" rpy="0 0 0" />
            <mass value="100" />
            <inertia ixx="0" ixy="0" ixz="0" iyy="0" iyz="0" izz="0" />
          </inertial>
          <!--
          <visual>
            <origin xyz="0 0 0" rpy="0 0 0" />
            <geometry>
```

```
              <box size="0.133 1.890 0.001"/>
            </geometry>
            <material name="">
              <color rgba="0.792156862745098 0.819607843137255 0.9333333333333
1" />
            </material>
          </visual> -->
          <collision>
            <origin xyz="0 0 0" rpy="0 0 0" />
            <geometry>
              <box size="0.133 1.890 0.001"/>
            </geometry>
          </collision>
        </link>

        <joint name="Bandainv_joint" type="prismatic">
          <origin xyz="0 0.825 0.130" rpy="0 0 0" />
          <parent link="Banda_link" />
          <child link="Bandainv_link" />
          <axis xyz="0 1 0" />
          <limit lower="0" upper="1" effort="1000" velocity="1000" />
        </joint>
```

Cuando se exportan los modelos tridimensionales desde SolidWorks a URDF, se hacen de manera individual, es decir, cada componente que constituye el entorno de simulación del robot Lab-Volt 5250 se convierte en su propio paquete URDF. No obstante, en el marco de este proyecto, se llevó a cabo una consolidación manual de todos los paquetes URDF en un único paquete. Esta estrategia facilitó la agrupación de todas las mallas en una sola carpeta, al tiempo que permitió la unificación de todos los archivos URDF en un solo archivo integral. La finalidad detrás de este proceso fue simplificar la gestión y administración de los componentes del robot y su representación en el entorno de simulación.

Finalmente, para garantizar la funcionalidad adecuada del archivo URDF generado y asegurarse de que todo estuviera en consonancia con las especificaciones previamente establecidas, se empleó una herramienta altamente práctica. Esta herramienta se presenta como un

"URDF viewer" y está disponible en la siguiente dirección web: https://gkjohnson.github.io/urdf-loaders/javascript/example/bundle/index.html. A través de esta aplicación, simplemente arrastrando la carpeta generada por SolidWorks junto con las respectivas modificaciones, se logra visualizar el modelo tridimensional como se muestra en la figura 50. Además, el "URDF viewer" ofrece controladores para cada articulación, lo que permite verificar el movimiento del modelo de manera eficiente.

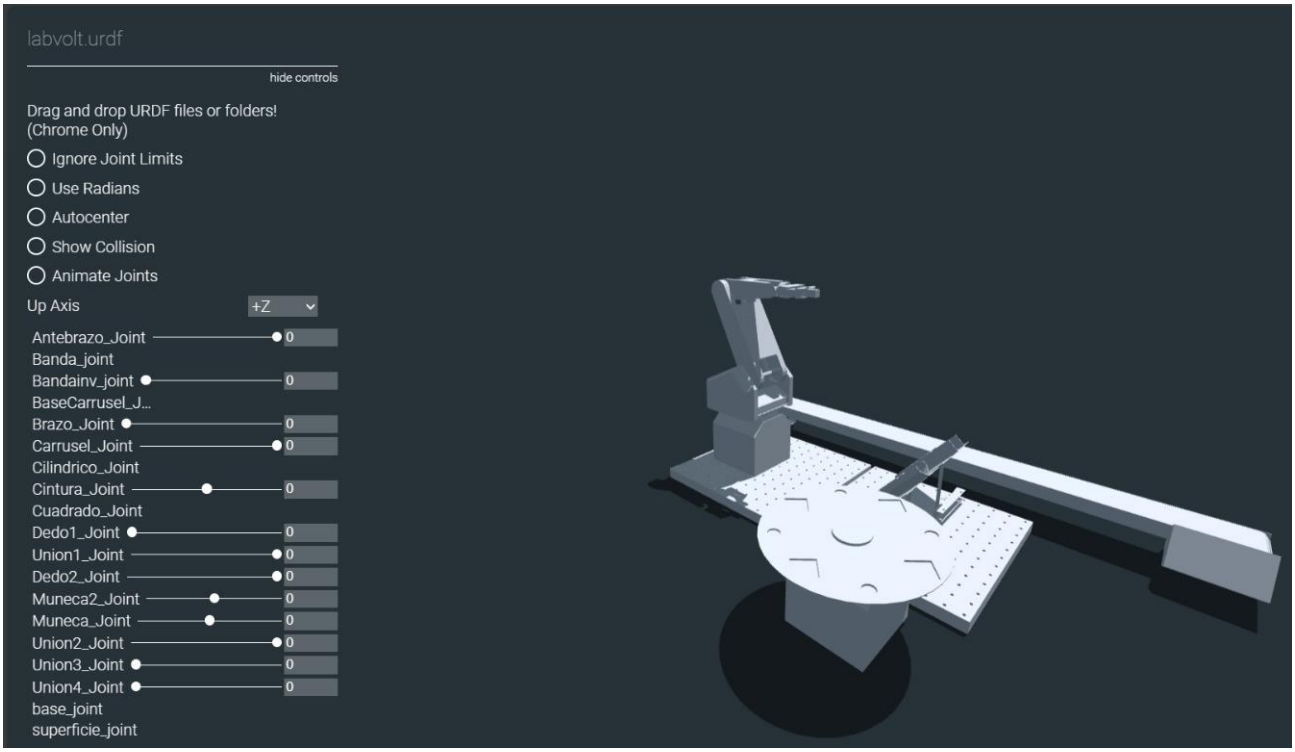


Figura 50. URDF Viewer

Fuente: D'Agostini, J. y Nazar, S. (2023)

Fase IV: Obtención del modelo cinemático del robot Lab-Volt 5250.

Esta etapa de la investigación, se centra en la obtención del modelo cinemático directo del robot Lab-Volt 5250. La función principal radica en determinar cómo las distintas articulaciones del robot se relacionan con la posición y orientación de su elemento terminal. Para lograr esta representación matemática precisa, se empleará una matriz de transformación homogénea, comúnmente denotada como T, que captura las rotaciones y traslaciones efectuadas por cada articulación. Este cálculo se basa en los parámetros de Denavit-Hartenberg (D-H), como se explicó anteriormente en las bases teóricas, son una convención estándar utilizada en robótica para

describir la disposición geométrica de las articulaciones de un robot y cómo están conectadas entre sí. Esto es esencial para calcular las relaciones espaciales entre las articulaciones del robot y su extremo, lo que permite determinar con precisión cómo las articulaciones deben moverse para alcanzar una posición o una orientación específica.

En esencia, partiendo de una posición inicial del efector final del robot, se buscó calcular su posición y orientación finales después de aplicar los movimientos de traslación y rotación, que dependen de la configuración particular de las articulaciones del robot. La obtención de este modelo cinemático no solo es esencial para lograr simulaciones realistas en el entorno virtual de ROS, sino que también es crítica para garantizar un control efectivo y seguro del robot en aplicaciones prácticas, como la automatización de tareas industriales o la investigación en robótica.

4.4.1 Identificación de Articulaciones y Grados de Libertad

Para obtener una comprensión completa del comportamiento cinemático del robot, resulta necesario identificar sus articulaciones y definir sus grados de libertad. A continuación, se detallan y categorizan cada una de estas articulaciones, asignándoles nombres específicos para una mejor comprensión (consulte el cuadro 2). Estos datos servirán como base para el análisis cinemático

Cuadro 2. Articulaciones del robot Lab-Volt 5250.

Articulación	Identificación	GDL
1	Cintura	1
2	Brazo	1
3	Antebrazo	1
4	Muñeca	2

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.4.2 Representación funcional del robot

Se creó un modelo funcional del robot Lab-Volt donde se incluyó la herramienta terminal, como se ilustra en la figura 51. En esta representación, se detallan cada una de las articulaciones del robot, se muestran sus conexiones y se especifican sus dimensiones. Posteriormente, se establecieron los sistemas de coordenadas, tal como se evidencia en la figura 52, lo que habilitó la descripción del movimiento relativo de los eslabones del robot mediante el uso de los parámetros de Denavit-Hartenberg (D-H). Cabe mencionar que las dimensiones mencionadas se obtuvieron a través de mediciones directas realizadas sobre el robot.

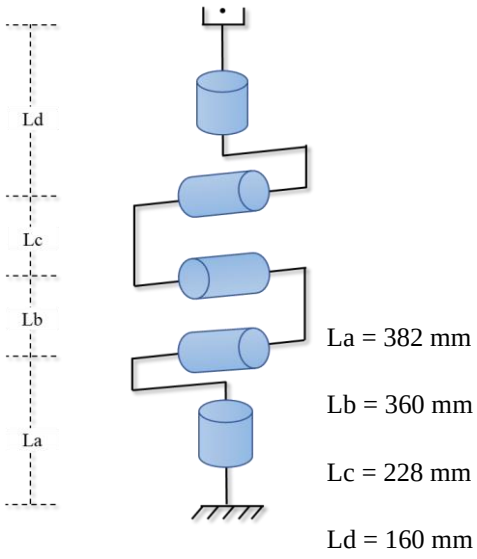


Figura 51. Modelo funcional del robot Lab-Volt 5250.

Fuente: D'Agostini, J. y Nazar, S. (2023)

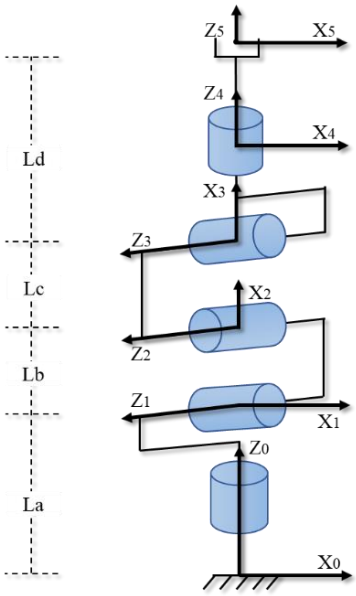


Figura 52. Sistema de coordenadas generalizada del robot Lab-Volt 5250.

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.4.3 Parámetros Denavit-Hartenberg del robot Lab-Volt 5250.

Tabla 2. Parámetros D-H del Lab-Volt 5250.

Articulación	θ	d	a	α	q
1	q_1	L_a	0	90 °	θ_1
2	q_2	0	L_b	0	θ_2

3	q_3	0	L_c	0	θ_3
4	q_4	0	0	-90 °	θ_4
5	q_5	L_d	0	0	θ_5

Fuente: D'Agostini, J. y Nazar, S. (2023)

La tabla 2 presenta los resultados derivados de la deducción de los parámetros de D-H a partir del modelo funcional del robot. En esta tabla, se identifica tanto el número de articulaciones como los valores correspondientes de los parámetros D-H. Cada articulación se caracteriza exhaustivamente mediante sus parámetros, que abarcan la distancia normal (a) entre ejes, el ángulo de torsión (α) entre los mismos, la distancia angular (θ) entre los ejes normales y la distancia lineal (d) entre ellos. Estos parámetros resultan sustanciales para una descripción precisa de la cinemática del robot, permitiendo comprender cómo cada articulación contribuye al movimiento y la posición del elemento terminal del robot en su conjunto.

4.4.4 Matrices de paso homogéneas del robot Lab-Volt 5250.

$$ {}^{i-1}A_i = \begin{bmatrix} \cos\theta_i & -\cos\alpha_i \sin\theta_i & \sin\alpha_i \sin\theta_i & a\cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cos\theta_i & -\sin\alpha_i \cos\theta_i & a\sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1} $$

Las matrices de paso homogéneas se obtienen mediante la sustitución de los parámetros derivados de la tabla 2 en la ecuación general (4.1). Este proceso condujo a la obtención de matrices particulares para cada articulación del robot:

$$ {}^{0}A_1 = \begin{bmatrix} \cos\theta_1 & 0 & \sin\theta_1 & 0 \\ \sin\theta_1 & 0 & -\cos\theta_1 & 0 \\ 0 & 1 & 0 & La \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2} $$

$$ {}^{1}A_2 = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 & Lb\cos\theta_2 \\ \sin\theta_2 & \cos\theta_2 & 0 & Lb\sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.3} $$

$$ {}^{2}A_3 = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 & 0 & Lc\cos\theta_3 \\ \sin\theta_3 & \cos\theta_3 & 0 & Lc\sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4} $$

$$ {}^{3}A_{4} = \begin{bmatrix} \cos\theta_4 & 0 & -\sin\theta_4 & 0 \\ \sin\theta_4 & 0 & \cos\theta_4 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5} $$

$$ {}^{4}A_{5} = \begin{bmatrix} \cos\theta_5 & -\sin\theta_5 & 0 & 0 \\ \sin\theta_5 & \cos\theta_5 & 0 & 0 \\ 0 & 0 & 1 & Ld \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6} $$

4.4.5 Matriz de transformación del robot Lab-Volt 5250.

La matriz de transformación T de robot se calcula mediante la pre-multiplicación sucesiva de las matrices particulares de paso homogéneo antes descritas, tal y como se ilustra en la ecuación (4.7). Debido a la naturaleza laboriosa de este proceso, se recurrió al uso del programa Octave para facilitar los cálculos y asegurar la precisión en la obtención de la matriz de transformación completa.

$$ T = {}^{0}A_{1} \cdot {}^{1}A_{2} \cdot {}^{2}A_{3} \cdot {}^{3}A_{4} \cdot {}^{4}A_{5} \tag{4.7} $$

A raíz de la considerable dimensión de la matriz T, se ha optado por presentarla en su forma general (ver ecuación 4.8), desglosando individualmente cada uno de sus elementos. Esto permitirá una comprensión más detallada y accesible de la matriz completa y sus componentes.

$$ T = \begin{bmatrix} nx & sx & ax & px \\ ny & sy & ay & py \\ nz & sz & az & pz \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8} $$

$$ \boldsymbol{nx} = c1 * c5 * \big(c4 * (c2 * c3 - s2 * s3) - s4 * (c2 * s3 + c3 * s2)\big) - s1 * s5 $$

$$ \boldsymbol{ny} = c1 * s5 + c5 * s1 * \big(c4 * (c2 * c3 - s2 * s3) - s4 * (c2 * s3 + c3 * s2)\big) $$

$$ \boldsymbol{nz} = c5 * \big(c4 * (c2 * s3 + c3 * s2) + s4 * (c2 * c3 - s2 * s3)\big) $$

$$ \boldsymbol{sx} = -c1 * s5 * \big(c4 * (c2 * c3 - s2 * s3) - s4 * (c2 * s3 + c3 * s2)\big) - c5 * s1 $$

$$ \boldsymbol{sy} = c1 * c5 - s1 * s5 * \big(c4 * (c2 * c3 - s2 * s3) - s4 * (c2 * s3 + c3 * s2)\big) $$

$$ \boldsymbol{sz} = -s5 * \big(c4 * (c2 * s3 + c3 * s2) + s4 * (c2 * c3 - s2 * s3)\big) $$

$$ \boldsymbol{ax} = c1 * \big(-c4 * (c2 * s3 + c3 * s2) - s4 * (c2 * c3 - s2 * s3)\big) $$

$$ \boldsymbol{ay} = s1 * \big(-c4 * (c2 * s3 + c3 * s2) - s4 * (c2 * c3 - s2 * s3)\big) $$

$$ \boldsymbol{az} = c4 * (c2 * c3 - s2 * s3) - s4 * (c2 * s3 + c3 * s2) $$

$$\boldsymbol{px} = c1 * \Big(Lb * c2 \ + Lc * c2 * c3 \ - \ Lc * s2 * s3 \ - \ Ld$$
$$* \big(c4 * (c2 * s3 \ + \ c3 * s2) + \ s4 * (c2 * c3 \ - \ s2 * s3)\big)\Big)$$
$$\boldsymbol{py} = s1 * \Big(Lb * c2 \ + \ Lc * c2 * c3 \ - \ Lc * s2 * s3 \ - \ Ld$$
$$* \big(c4 * (c2 * s3 \ + \ c3 * s2) + \ s4 * (c2 * c3 \ - \ s2 * s3)\big)\Big)$$
$$\boldsymbol{pz} = \mathrm{La} \ + \ \mathrm{Lb} * \mathrm{s2} \ + \ \mathrm{Lc} * \mathrm{c2} * \mathrm{s3} \ + \ \mathrm{Lc} * \mathrm{c3} * \mathrm{s2} \ + \ \mathrm{Ld} * (\mathrm{c4} * (\mathrm{c2} * \mathrm{c3} \ - \ \mathrm{s2} * \mathrm{s3}) \ - \ \mathrm{s4}$$
$$* (\mathrm{c2} * \mathrm{s3} \ + \ \mathrm{c3} * \mathrm{s2})$$

La importancia de contar con el modelo cinemático directo del robot, radica en comprobar las rotaciones o traslaciones efectuadas por cada articulación del robot para posteriormente llevar a cabo simulaciones lo más cercanas a la realidad en el entorno virtual de ROS, permitiendo una comprensión adecuada del comportamiento del robot en diversas situaciones.

Fase V: Demostración del manejo del robot virtual y sus accesorios en la interfaz gráfica de ROS.

4.5.1 Elección del sistema operativo

En el desarrollo del presente trabajo de investigación, se llevaron a cabo pruebas en diferentes sistemas operativos para identificar la plataforma más adecuada. Inicialmente, al explorar la opción de Windows, la instalación de ROS no presentó dificultades. Sin embargo, se descubrió que el simulador Gazebo, esencial para el proyecto, no era compatible con Windows, lo que llevó a descartar esta opción. En un intento posterior, se consideró el uso de VirtualBox, donde se implementó un sistema Ubuntu 20.04 para el desarrollo del proyecto. A pesar de no encontrar problemas en la configuración, se experimentó un rendimiento limitado por parte del hardware del ordenador.

Como solución, se optó por emplear WSL (Windows Subsystem for Linux), que demostró ser la elección óptima para instalar ROS. WSL proporciona un entorno Linux completo de forma nativa en una máquina con Windows, permitiendo a los usuarios aprovechar todas las ventajas de ROS, diseñado principalmente para sistemas Linux, sin la necesidad de configurar una máquina virtual o mantener un sistema dual. Esto resultó en una solución eficiente y sin complicaciones para el desarrollo de proyectos basados en ROS en un entorno Windows.

4.5.1.1 Instalación de WSL

Para la instalación de WSL se debe contar con Windows 10 o Windows 11. Es importante destacar que para la identificación de los comandos a introducir en la terminar se utilizó el símbolo "$", sin embargo, este no forma parte de la línea de comando.

1. Asegurarse de tener habilitado WSL en el sistema Windows, para ello, hay que dirigirse a "Panel de control", luego a "Programas y característica" y seleccionar "Activar o desactivar las características de Windows". Allí se marca la casilla "Subsistema de Linux para Windows" como se muestra en la figura 53 y dar clic en aceptar.

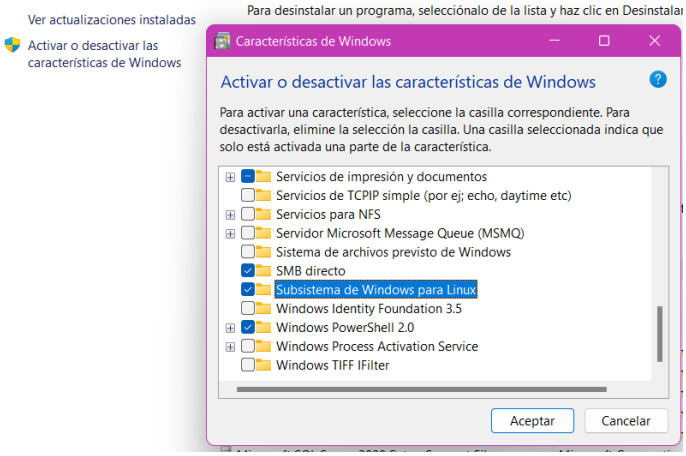


Figura 53. Activación de WSL en Windows

Fuente: D'Agostini, J. y Nazar, S. (2023)

2. Reiniciar la máquina.
3. Abrir PowerShell o el símbolo del sistema de Windows como administrador; para ello, se debe hacer clic con el botón derecho y seleccionar "Ejecutar como administrador".
4. Introducir el siguiente comando:

   $ wsl –install

   Este comando habilitará las características necesarias para ejecutar WSL e instalará la distribución Ubuntu de Linux.
5. Reiniciar la máquina.
6. Cambiar la distribución de Linux instalada.

   $ wsl --list –online

   Este comando permite observar una lista de las distribuciones de Linux disponibles para descargar.

   $ wsl --install -d <Nombre de la Distribución>

Para efectos de este trabajo, se utilizó la distribución Ubunto 20.04, ya que es la distribución compatible con la versión de ROS que se instaló.

7. Una vez que se haya instalado WSL, se deberá crear una cuenta de usuario y una contraseña para la distribución de Linux recién instalada

4.5.2 Instalación de ROS

La elección de ROS Noetic para este proyecto se basó en su posición actual como una versión de ROS respaldada por la comunidad, lo que significa que cuenta con un amplio apoyo y desarrollo continuo. Además, ROS Noetic es conocido por su robustez, estabilidad, lo que es fundamental para proyectos de robótica que requieren un funcionamiento confiable. Otra ventaja importante es su capacidad para ejecutarse en sistemas operativos modernos y ofrecer compatibilidad con hardware actual, permitiendo aprovechar las últimas tecnologías y realizar implementaciones de robótica más eficientes. Su capacidad para mantener la compatibilidad con versiones anteriores de ROS 1 también fue un factor importante, ya que permitió utilizar paquetes y recursos previamente desarrollados, lo que agilizó considerablemente el proceso de virtualización del robot Lab-Volt 5250.

4.5.2.1 Pasos para la instalación de de ROS Noetic en Ubuntu

1. Configuración del ordenador para que acepte software de packages.ros.org.

   $ sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu $(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-latest.list'

   Este comando se utiliza para configurar un repositorio (lugar donde se almacenan y gestionan los datos) específico de ROS en el sistema de gestión de paquetes APT (Advanced Package Tool) en Ubuntu. Esto es esencial para instalar y mantener paquetes relacionados con ROS al sistema Ubuntu.

2. Configuración de las llaves

   $ sudo apt install curl

   Permite la instalación de la herramienta curl (Clients for URL’S), la cual es utilizada para realizar transferencia de datos con URL.

   $ curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add –

Se utiliza para importar la clave GPG (GnuPG) del repositorio de ROS (Robot Operating System) en un sistema Linux. Esta clave GPG es necesaria para verificar la autenticidad e integridad de los paquetes de ROS durante la instalación o actualización.

3. Instalación

   $ sudo apt update

   Su propósito principal es actualizar la lista de paquetes disponibles en los repositorios de software configurados en el sistema.

   $ sudo apt install ros-noetic-desktop-full

   Este comando se utiliza para instalar la versión completa y de escritorio de ROS, incluyendo simuladores 2D/3D y paquetes de percepción 2D/3D.

4. Instalación de paquetes

   $ sudo apt install ros-noetic-PACKAGE

   Se utiliza para instalar un paquete específico, si no se conoce el paquete que se desea instalar, se puede utilizar el siguiente comando para mostrar una lista de paquetes relacionados con ROS dentro de los repositorios del sistema:

   $ apt search ros-noetic

5. Configuración del entorno

   $ source /opt/ros/noetic/setup.bash

   Se emplea para cargar y configurar el entorno de ROS en la terminal actual, es conveniente generar automáticamente este script cada vez que se inicie un nuevo Shell, esto se puede lograr a través de los siguientes comandos:

   $ echo "source /opt/ros/noetic/setup.bash" >> ~/.bashrc

   $ source ~/.bashrc

6. Dependencias para construir paquetes

   $ sudo apt install python3-rosdep python3-rosinstall python3-rosinstall-generator python3-wstool build-essential

   Se utiliza para instalar una serie de herramientas y dependencias necesarias para trabajar con ROS en sistemas Linux basados en Debian/Ubuntu. A continuación, se detalla el propósito de cada uno de los paquetes que se están instalando:

   - python3-rosdep: ROS utiliza rosdep para gestionar dependencias de paquetes. Este paquete permite instalar y gestionar las dependencias de ROS.

- python3-rosinstall: Proporciona la herramienta "rosinstall" para descargar e instalar paquetes ROS desde repositorios.
- python3-rosinstall-generator: Permite generar archivos de instalación para una serie de paquetes ROS. Esto facilita la instalación de múltiples paquetes a la vez.
- python3-wstool: Ofrece herramientas para gestionar varios aspectos de los paquetes de ROS, incluida la descarga y la administración de múltiples paquetes en un espacio de trabajo.
- build-essential: Es un conjunto de herramientas y paquetes esenciales para compilar y construir software en sistemas basados en Unix. Incluye compiladores y utilidades que son necesarios para compilar programas y bibliotecas.

7. Inicializar rosdep

   $ sudo rosdep init

   $ rosdep update

4.5.3 Creación y configuración del espacio de trabajo

En el contexto de ROS (Robot Operating System), un “workspace” o espacio de trabajo se refiere a un directorio o carpeta específica en la cual se organizan y se construyen los paquetes de software relacionados con un proyecto de robótica o una aplicación específico. Dentro de este entorno concreto, se despliega un proceso de desarrollo, compilación y gestión de los paquetes que configuran un sistema robótico o una aplicación robótica. Los espacios de trabajo desempeñan un papel crítico al mantener la estructura y el orden del proyecto, lo que, a su vez, habilita a los desarrolladores para desplegar sus tareas de manera sumamente eficiente. Para crear el espacio de trabajo se ejecutaron los siguientes comandos desde la terminal:

1. Actualización de la lista de paquetes disponibles en los repositorios de Ubuntu para asegurar que el sistema tenga información actualizada.

   $ sudo apt update

2. Desplazo al directorio de inicio o “home”.

   $ cd ~/

3. Creación de un nuevo directorio.

   $ mkdir catkin_ws

   El nombre asignado a la estructura central de este proyecto de investigación fue "catkin_ws", sin embargo, se le puede asignar cualquier nombre. Este directorio principal

constituye el núcleo del espacio de trabajo empleado para llevar a cabo la presente investigación en el marco de Robot Operating System (ROS).

4. Cambio del Directorio actual al nuevo espacio de trabajo que se creó.
   $ cd ~/catkin_ws
5. Creación de un subdirectorio “src” en el directorio del espacio de trabajo.
   $ mkdir src
6. Construcción y compilación del espacio de trabajo.
   $ catkin_make
7. Adición del Directorio devel/setup.bash al entorno actual.
   $ source devel/setup.bash
   Este comando configura al entorno para que ROS pueda encontrar y ejecutar los nodos y paquetes sin problemas.

Al ejecutar el comando catkin_make en el espacio de trabajo creado en ROS, se generan tres carpetas principales:

- build: La carpeta build es donde se almacenan los archivos temporales y se generan los ejecutables, bibliotecas y otros archivos binarios durante el proceso de compilación de los paquetes de ROS en el espacio de trabajo. En esta carpeta, se encuentra subcarpetas para cada paquete que estás construyendo. Por lo general, no se necesita acceder a esta carpeta directamente, ya que ROS se encarga de gestionarla de manera automática.
- devel (development – desarrollo): Contiene enlaces simbólicos a los archivos binarios generados en la carpeta build, lo que permite que los programas ROS encuentren y utilicen los paquetes y bibliotecas compilados sin necesidad de que los archivos binarios se ubiquen en rutas fijas en el sistema.
- src (source - fuente): Esta carpeta es donde normalmente se trabaja en el código fuente del paquete de ROS. Cada paquete se almacena en un subdirectorio dentro de src. Aquí se crean y editan los nodos, mensajes, servicios y otros componentes de software. Cuando se desea compilar el código, catkin_make buscará en esta carpeta para compilar los paquetes que contiene.

4.5.4 Introducción del paquete de ROS que contiene el URDF en el nuevo espacio de trabajo

Luego de la creación y configuración del espacio de trabajo, se procedió a realizar la transferencia del paquete de ROS previamente creado en SolidWorks al interior del espacio de trabajo de catkin. Este proceso de transferencia específicamente incluyó la ubicación del paquete en la carpeta "src" del espacio de trabajo catkin. Es importante destacar que el nombre de la carpeta generada por SolidWorks coincide con el nombre del paquete. Mantener inalterado este nombre resulta esencial, ya que dicho nombre se emplea en los scripts generados de manera automática dentro del paquete. Como ejemplo, el nombre del paquete utilizado en el presente trabajo fue "labvolt," y se copió en la siguiente ruta: "~/catkin_ws/src/labvolt." Esta secuencia de pasos garantiza que los scripts y recursos relacionados con el paquete se integren sin inconvenientes en el espacio de trabajo de ROS.

Por otro lado, el archivo CMakeLists.txt obtenido a través de SolidWorks es muy básico y no contiene otras dependencias importantes y necesarias realizar la simulación, por lo tanto, se procedió a editar el script. A continuación, se muestra el contenido del archivo CMakeLists.txt antes de ser editado:

```
cmake_minimum_required(VERSION 2.8.3)

project(labvolt)

find_package(catkin REQUIRED)

catkin_package()

find_package(roslaunch)

foreach(dir config launch meshes urdf)
      install(DIRECTORY ${dir}/
            DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}/${dir})
endforeach(dir)
```

Después de agregar las dependencias y componentes, la configuración se vio de la siguiente manera:

```
cmake_minimum_required(VERSION 2.8.3)

project(labvolt)

find_package(catkin REQUIRED COMPONENTS
      message_generation
      roscpp
      rospy
      std_msgs
      geometry_msgs
      urdf
      xacro
      message_generation
)

catkin_package(CATKIN_DEPENDS
      geometry_msgs
      roscpp
      rospy
      std_msgs
)

find_package(roslaunch)

foreach(dir config launch meshes urdf)
      install(DIRECTORY ${dir}/
            DESTINATION ${CATKIN_PACKAGE_SHARE_DESTINATION}/${dir})
endforeach(dir)
```

Es importante asegurarse de usar TAB para las identaciones y no espacios.

Componentes agregados:

- message_generation: Este componente se utiliza para generar mensajes personalizados que pueden ser utilizados para la comunicación entre nodos en ROS.
- roscpp: Proporciona las bibliotecas y herramientas para programar en C++ en el entorno ROS.

- rospy: Proporciona bibliotecas y herramientas similares a roscpp, pero para programación en Python.
- std_msgs: Contiene definiciones de mensajes estándar utilizados en ROS para intercambiar información básica entre nodos.
- geometry_msgs: Define mensajes para representar información geométrica, como puntos y transformaciones.
- urdf: Se utiliza para trabajar con modelos URDF que describen la estructura y cinemática de robots.
- xacro: Es una herramienta que permite la generación dinámica, más eficiente y mantenible de modelos URDF a partir de descripciones más compactas en formato XML.

Por otro lado, en el archivo predeterminado package.xml del paquete, se observó que existen muy pocas dependencias incluidas en su configuración inicial, la cual se presentó de la siguiente manera:

```
<package format="2">
  <name>labvolt</name>
  <version>1.0.0</version>
  <description>
    <p>URDF Description package for labvolt</p>
    <p>This package contains configuration data, 3D models and launch files
for labvolt robot</p>
  </description>
  <author>TODO</author>
  <maintainer email="TODO@email.com" />
  <license>BSD</license>
  <buildtool_depend>catkin</buildtool_depend>
  <depend>roslaunch</depend>
  <depend>robot_state_publisher</depend>
  <depend>rviz</depend>
  <depend>joint_state_publisher</depend>
  <depend>gazebo</depend>
  <export>
    <architecture_independent />
  </export>
```

```
    </package>
```

Esto planteó un desafío en el contexto del objetivo principal de realizar simulaciones completas y precisas. Para lograr un entorno de simulación efectivo, fue crucial agregar una serie de dependencias adicionales que ampliaron la funcionalidad y las capacidades del sistema. Asimismo, se incorporó la identificación del autor junto con su dirección de correo electrónico. Este ajuste resulta especialmente relevante, ya que, al compartir el proyecto, facilita la comunicación y colaboración con quienes lo visualicen.

```
    <package format="2">
      <name>labvolt</name>
      <version>1.0.0</version>
      <description>
        <p>URDF Description package for labvolt</p>
        <p>This package contains configuration data, 3D models and launch files
for labvolt robot</p>
      </description>
      <author>SahiraNazarandJanDAgostini</author>
      <maintainer email="correo@gmail.com" />
      <license>BSD</license>
      <buildtool_depend>catkin</buildtool_depend>

      <build_depend>message_generation</build_depend>
      <build_depend>roscpp</build_depend>
      <build_depend>rospy</build_depend>
      <build_depend>std_msgs</build_depend>
      <build_depend>geometry_msgs</build_depend>
      <build_depend>urdf</build_depend>
      <build_depend>xacro</build_depend>
      <build_depend>std_msgs</build_depend>
      <build_depend>message_generation</build_depend>

      <depend>roslaunch</depend>
      <depend>robot_state_publisher</depend>
      <depend>rviz</depend>
```

```
  <depend>joint_state_publisher</depend>
  <depend>joint_state_publisher_gui</depend>
  <depend>gazebo</depend>
  <depend>moveit_simple_controller_manager</depend>

  <build_export_depend>roscpp</build_export_depend>
  <build_export_depend>rospy</build_export_depend>
  <build_export_depend>std_msgs</build_export_depend>
  <build_export_depend>geometry_msgs</build_export_depend>
  <build_export_depend>urdf</build_export_depend>
  <build_export_depend>xacro</build_export_depend>

  <exec_depend>roscpp</exec_depend>
  <exec_depend>rospy</exec_depend>
  <exec_depend>std_msgs</exec_depend>
  <exec_depend>geometry_msgs</exec_depend>
  <exec_depend>urdf</exec_depend>
  <exec_depend>xacro</exec_depend>
  <exec_depend>message_runtime</exec_depend>

  <export>
    <architecture_independent />
  </export>
</package>
```

Al igual que en el archivo anterior, se usaron TABS para las identaciones y no espacios.

En esta configuración están presentes las siguientes etiquetas:

<build_depend>: Estas etiquetas enumeran las dependencias de compilación necesarias para construir el paquete.

<depend>: Especifican los paquetes necesarios para que el paquete del proyecto funcione correctamente en tiempo de ejecución. Las dependencias que se incluyeron son:

- roslaunch es una herramienta en ROS que se utiliza para iniciar nodos y lanzar archivos de configuración (paquetes).

- robot_state_publisher es un nodo que se utiliza para publicar la cinemática inversa del robot, lo que permite representar la ubicación de las articulaciones del robot en ROS. Este nodo es crucial para la visualización y simulación precisa de un robot.
- RViz se utiliza para visualizar el robot, datos de sensores y planificación de movimientos. Es una parte integral del desarrollo y prueba de aplicaciones de robótica.
- joint_state_publisher es un nodo que permite al usuario interactuar con las articulaciones del robot y publicar estados de las mismas. Es útil para modificar manualmente la posición de las articulaciones.
- joint_state_publisher_gui es una interfaz gráfica de usuario (GUI) para la edición de estados de las articulaciones del robot. Proporciona una forma conveniente de ajustar y observar las posiciones de las articulaciones.
- Gazebo permite simular robots y entornos, lo que es esencial para el desarrollo y prueba de aplicaciones de robótica.
- MoveIt permite la planificación de movimientos en robots. "moveit_simple_controller_manager" es un componente que permite controlar los movimientos del robot simulado.

<build_export_depend>: Indican las dependencias de exportación de compilación. Estas dependencias se utilizan cuando otros paquetes dependen del paquete principal (paquete del proyecto en curso) y necesitan acceder a las bibliotecas y recursos de construcción de este paquete.

<exec_depend>: Estas etiquetas definen las dependencias de ejecución que otros paquetes necesitan para ejecutar correctamente un nodo o componente proporcionado por el paquete del proyecto en curso.

<export>: Esta sección se usa para exportar información sobre el paquete. En este caso, se especifica que el paquete es independiente de la arquitectura, lo que significa que debería funcionar en diferentes plataformas.

4.5.5 Adición de plugins al URDF

Los plugins son componentes cruciales en la creación y simulación de modelos URDF en ROS. Los plugins se utilizan para extender las capacidades y funcionalidades de los modelos URDF, permitiendo la incorporación de lógica personalizada, dinámicas, y comportamientos específicos. Al agregar plugins a un URDF, se posibilita la representación de sensores, actuadores

y controladores en el modelo, lo que es esencial para la simulación precisa del robot y la interacción con su entorno.

El plugin "gazebo_ros_control" fue uno de los plugins utilizados ya que es una parte fundamental en la configuración de robots simulados en el entorno de simulación Gazebo dentro del framework ROS. Su función crítica radica en proporcionar una interfaz robusta y altamente especializada, destinada a la administración y control del comportamiento del robot simulado en Gazebo. El plugin actúa como un componente crucial para habilitar la coherencia en la ejecución de operaciones de control y supervisión, lo que es esencial para llevar a cabo pruebas exhaustivas, desarrollo iterativo y experimentación avanzada dentro de un entorno simulado.

```
    <gazebo>
       <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
           <robotNamespace>/</robotNamespace>
           <robotSimType>gazebo_ros_control/DefaultRobotHWSim</robotSimType>
           <legacyModeNS>true</legacyModeNS>
       </plugin>
    </gazebo>
```

El plugin "conveyor_belt_plugin" desempeñó un papel esencial en la simulación del movimiento de la banda transportadora. Este componente se integra en el marco de ROS para controlar el desplazamiento de la lámina invisible ubicada sobre la banda transportadora, la cual fue previamente definida en el modelo URDF. La acción principal de este plugin es permitir que la lámina se mueva a una velocidad específica utilizando una articulación prismática, lo que significa que su movimiento es paralelo a la superficie de la banda transportadora. Es importante destacar que el movimiento de la lámina es cíclico, es decir, al llegar a su posición final, esta desaparece momentáneamente y vuelve a aparecer en su posición inicial. Sin embargo, esta transición ocurre de manera tan rápida que resulta imperceptible para el ojo humano. En conjunto, este proceso crea la ilusión de un movimiento continuo y realista de la banda transportadora en la simulación.

```
    <gazebo>
        <plugin                                          name="conveyor_belt_plugin"
ilename="libROSConveyorBeltPlugin.so">
            <robot_namespace>/</robot_namespace>
```

```
        <link>Banda_link_invisible</link>
        <update_rate>10</update_rate>
        <max_velocity>0.2</max_velocity>
    </plugin>
</gazebo>
```

4.5.6 Creación de archivo .yaml

Para garantizar el funcionamiento adecuado del sistema robótico en el contexto de ROS, se requirió la creación de un archivo en formato YAML, el cual se alojó en la carpeta "config" dentro del paquete principal labvolt. En este archivo, se definieron y configuraron controladores conjuntos para todos los elementos del brazo robótico y sus accesorios, como las articulaciones de los brazos robóticos, las articulaciones del efector final, articulaciones del carrusel, y en general, cualquier otro controlador necesario para su aplicación particular. Los pasos para la creación del archivo .yaml fueron los siguientes:

1. Abrir la terminal.
2. Ir a la carpeta config del paquete del robot.

   $ cd ~/catkin_ws/src/labvolt/config

   Donde catkin_ws representa el nombre del espacio de trabajo y labvolt representa el nombre del paquete URDF extraído de SolidWorks.
3. Crear un archivo “joint_trajectory_controller.yaml” en la carpeta config del paquete URDF a través de la terminal.

   $ touch joint_trajectory_controller.yaml

   Este archivo puede tener cualquier nombre, sin embargo, se debe usar el mismo nombre de archivo al cargar los controladores en el archivo de launch.
4. Abrir el archivo “joint_trajectory_controller.yaml”. Puede ser de forma manual dirigiéndose directamente a la carpeta o a través del comando:

   $ gedit joint_trajectory_controller.yaml
5. En el archivo se registró el conjunto de controladores creados, especificando su tipo y las articulaciones que gestionan, usando TABS para las identaciones y no espacios ya que no son aceptados por los archivos .yaml.

```
      #Controller to control robot arm joints
      robot_arm_controller:
        type: "position_controllers/JointTrajectoryController"
        joints:    [Cintura_Joint,    Brazo_Joint,    Antebrazo_Joint,
Muneca_Joint, Muneca2_Joint]
      #Controller to control end effector joints
      hand_ee_controller:
        type: "position_controllers/JointTrajectoryController"
        joints: [Union1_Joint, Union2_Joint, Union3_Joint, Union4_Joint,
Dedo1_Joint, Dedo2_Joint]
      #Controller to control carrusel
      carrusel_controller:
        type: "position_controllers/JointTrajectoryController"
        joints: [Carrusel_Joint]
      #Controller to control banda transportadora
      banda_controller:
        type: "position_controllers/JointTrajectoryController"
        joints: [Bandainv_joint]
      #Controller to continuously publish joint states/positions
      joint_state_controller:
        type: "joint_state_controller/JointStateController"
  publish_rate: 50
```

La última sección (joint_state_controller_) tiene un propósito diferente. Este controlador se encarga de publicar continuamente el estado y la posición de las articulaciones del robot. Utiliza el tipo "joint_state_controller/JointStateController" y tiene una tasa de publicación especificada en "publish_rate," que en este caso es de 50 Hz.

4.5.7 Creación de archivo .launch

El archivo launch se utilizó para ejecutar múltiples archivos, scripts, comandos o nodos desde un solo archivo. En este archivo launch, se pueden especificar los nodos que deben ejecutarse, sus argumentos, los parámetros que deben configurarse y las conexiones entre los nodos. Esta flexibilidad es esencial para la configuración de sistemas robóticos, donde múltiples nodos trabajan juntos para lograr una tarea específica. Los archivos .launch facilitan la

administración y el lanzamiento de estos nodos como un conjunto coherente. Los pasos para la creación del archivo .launch son los siguientes:

1. Abrir la terminal.
2. Ir a la carpeta launch del paquete principal.

   $ cd ~/catkin_ws/src/labvolt/launch

   Donde catkin_ws es el nombre del espacio de trabajo y labvolt el nombre del paquete URDF.
3. Crear un archivo “arm_urdf.launch” en la carpeta launch del paquete URDF a través de la terminal.

   $ touch arm_urdf.launch

   Este archivo puede recibir cualquier otro nombre.
4. Abrir el archivo “arm_urdf.launch”. Puede ser de forma manual dirigiéndose directamente a la carpeta o a través del comando:

   $ gedit arm_urdf.launch
5. Dentro de este archivo se configuraron varios aspectos para simular un robot en Gazebo y controlarlo.

```
      <launch>
          <arg name="arg_x" default="0.00" />
          <arg name="arg_y" default="0.00" />
          <arg name="arg_z" default="0.00" />
          <arg name="arg_R" default="0.00" />
          <arg name="arg_P" default="0.00" />
          <arg name="arg_Y" default="0.00" />

          <!--Urdf file path-->
          <param       name="robot_description"       textfile="$(find
labvolt)/urdf/labvolt.urdf"/>

          <!--spawn a empty gazebo world-->
          <include  file="$(find  gazebo_ros)/launch/empty_world.launch"
/>
```

```
        <node name="tf_footprint_base" pkg="tf"
type="static_transform_publisher" args="0 0 0 0 0 0 base_link
base_footprint 40" />

        <!--spawn model-->
        <node name="spawn_urdf" pkg="gazebo_ros" type="spawn_model"
args="-x $(arg arg_x) -y $(arg arg_y) -z $(arg arg_z) -Y $(arg arg_Y) -
param robot_description -urdf -model labvolt
        -J Cintura_Joint 0.0 -J Brazo_Joint 0.0 -J Antebrazo_Joint 0.0
-J Muneca_Joint 0.0 -J Muneca2_Joint 0.0 -J Union1_Joint 0.0 -J
Union2_Joint 0.0 -J Dedo1_Joint 0.0 -J Union3_Joint 0.0 -J Union4_Joint
0.0 -J Dedo2_Joint 0.0 -J Carrusel_Joint 0.0 -J Bandainv_joint 0.0" />


        <!-- Spawn a robot into Gazebo
        <node name="spawn_urdf1" pkg="gazebo_ros" type="spawn_model"
args="-file $(find carrusel)/urdf/carrusel.urdf -urdf -z 0 -model
Carrusel" />"    /> -->


        <!--Load and launch the joint trajectory controller-->
        <rosparam file ="$(find
labvolt)/config/joint_trajectory_controller.yaml" command="load"/>
        <node name= "controller_spawner" pkg= "controller_manager"
type="spawner" respawn="false" output="screen"
args="joint_state_controller robot_arm_controller hand_ee_controller
carrusel_controller"/>

        <!-- Robot State Publisher for TF of each joint: publishes all
the current states of the joint, then RViz can visualize -->
    <node   name="robot_state_publisher"   pkg="robot_state_publisher"
type="robot_state_publisher" respawn="false" output="screen"/>
```

A continuación, se desglosa el código paso a paso:

- Definición de Argumentos: Se definieron argumentos (variables) que se utilizan en todo el archivo. Estos argumentos se inicializan con valores predeterminados.
- Definición de parámetro "robot_description": Se definió un parámetro llamado "robot_description" que se utilizó para cargar la descripción URDF del robot. Esta descripción se tomó de un archivo URDF en la carpeta del paquete llamada "labvolt.urdf".
- Creación del Mundo Gazebo: Se incluyó el archivo "empty_world.launch" del paquete "gazebo_ros" para crear un mundo vacío en Gazebo.
- Transformación Estática: Se creó una transformación estática entre los marcos de referencia base_link y base_footprint utilizando el nodo static_transform_publisher del paquete "tf". Esto se utilizó para definir una relación fija entre estos dos marcos de referencia.
- Cargar el Modelo URDF: Se cargó el modelo URDF del robot en Gazebo utilizando el nodo "spawn_model" del paquete "gazebo_ros". Se especificaron los valores iniciales de posición y orientación para el robot, así como las articulaciones del robot y sus posiciones iniciales (todas en 0).
- Cargar Controladores: Se cargó y lanzó un conjunto de controladores definidos en un archivo YAML llamado "joint_trajectory_controller.yaml" ubicado en la carpeta de config del paquete "labvolt". Estos controladores son necesarios para controlar las articulaciones del robot. Además, se lanzó un nodo llamado "controller_spawner" del paquete "controller_manager" para administrar estos controladores.
- Publicador de Estado del Robot (TF): Finalmente, se lanzó un nodo llamado "robot_state_publisher" del paquete "robot_state_publisher". Este nodo se encarga de publicar todos los estados actuales de las articulaciones del robot, lo que permite a RViz visualizar el estado del robot.

4.5.8 Cargar el modelo en Gazebo

Luego de las modificaciones realizadas, es necesario construir y compilar los cambios realizados en los paquetes para asegurarse de que estén listos para ser utilizados en otros programas o sistemas dentro del entorno ROS, siguiendo la secuencia para construir y compilar:

8. Abrir la terminal y ejecutar
9. $ cd ~/catkin_ws
10. $ source devel/setup.bash
11. $ catkin_make
12. $ source devel/setup.bash

Una vez completados los pasos previos, se pudo cargar el robot en Gazebo que permitió observar el ambiente como se muestra en la figura 53, mediante el siguiente comando:

roslaunch labvolt arm_urdf.launch

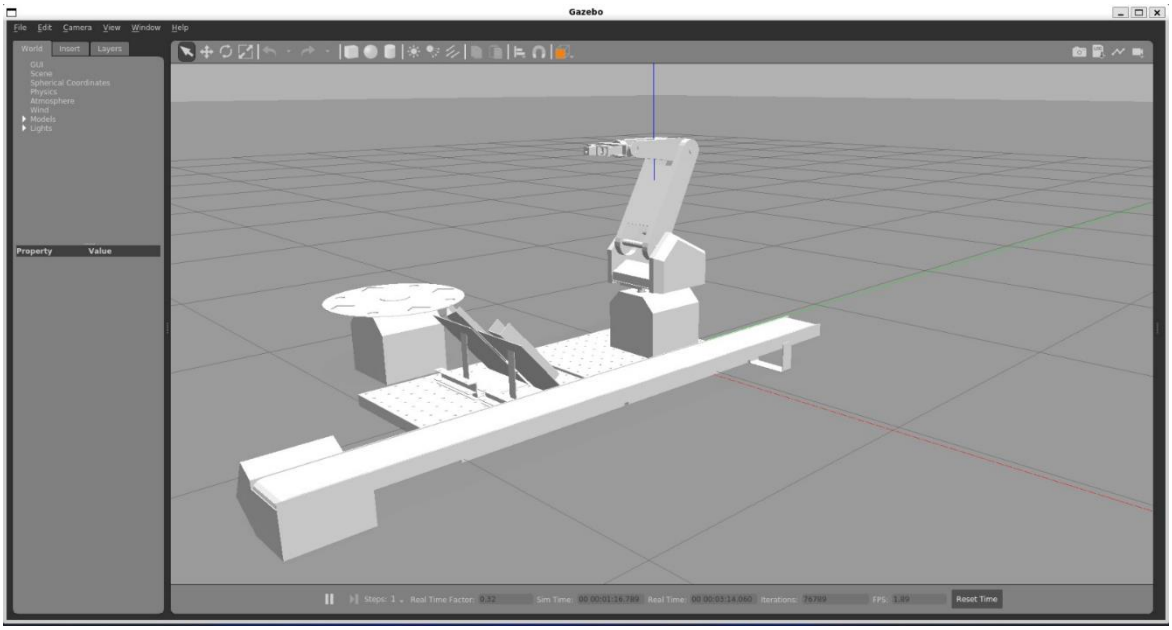


Figura 54. Simulación del robot Lab-Volt 5250 en Gazebo

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.5.9 Creación de nuevo paquete para controlar los modelos URDF con MoveIt

El siguiente paso de suma importancia involucra la implementación de un paquete especializado. Este paquete desempeña un papel fundamental en el control del brazo robótico, haciendo uso de la cinemática inversa para calcular las posiciones de las articulaciones. Además, se encarga de la planificación de trayectorias, lo que es esencial para el movimiento preciso del

robot. Para habilitar estas capacidades, se procedió a la instalación de MoveIt, una herramienta ampliamente respetada en el mundo de la robótica. Los comandos necesarios para instalar MoveIt 1 en el entorno de trabajo son los siguientes:

$ sudo apt install ros-noetic-moveit

Install ROS Controllers:

$ sudo apt-get install ros- melodic -ros-control ros- melodic -ros-controllers

4.5.9.1 Configuración del URDF en MoveIt Assistant

Tras la instalación de MoveIt, la configuración y planificación de trayectorias se realizó eficazmente utilizando MoveIt Assistant, la cual es una herramienta fundamental. Para acceder a esta utilidad a través de la terminal, se procedió a compilar el espacio de trabajo previamente configurado. Luego, se introdujo el comando específico que permitió abrir MoveIt Assistant, lo que habilitó la configuración y planificación precisas de las trayectorias del robot. Esta herramienta se convirtió en una parte esencial del proceso de desarrollo y simulación del proyecto, facilitando la interacción con modelos de robots y agilizando el control de movimiento en el entorno Gazebo. Para abrir MoveIt Assistant:

1. Ir al directorio del espacio de trabajo.

   $ cd ~/moveit_ws
2. Construir el espacio de trabajo

   $ catkin_make

   $ source devel/setup.bash
3. Abrir Moveit Setup Assistant

   $ roslaunch moveit_setup_assistant setup_assistant.launch

Al abrirse MoveIt Assistant, se cargó en archivo URDF. Si este paso se realiza correctamente, los elementos cargados y presentes en el URDF pueden visualizarse en un recuadro a la derecha del programa como se visualiza en la figura 55.

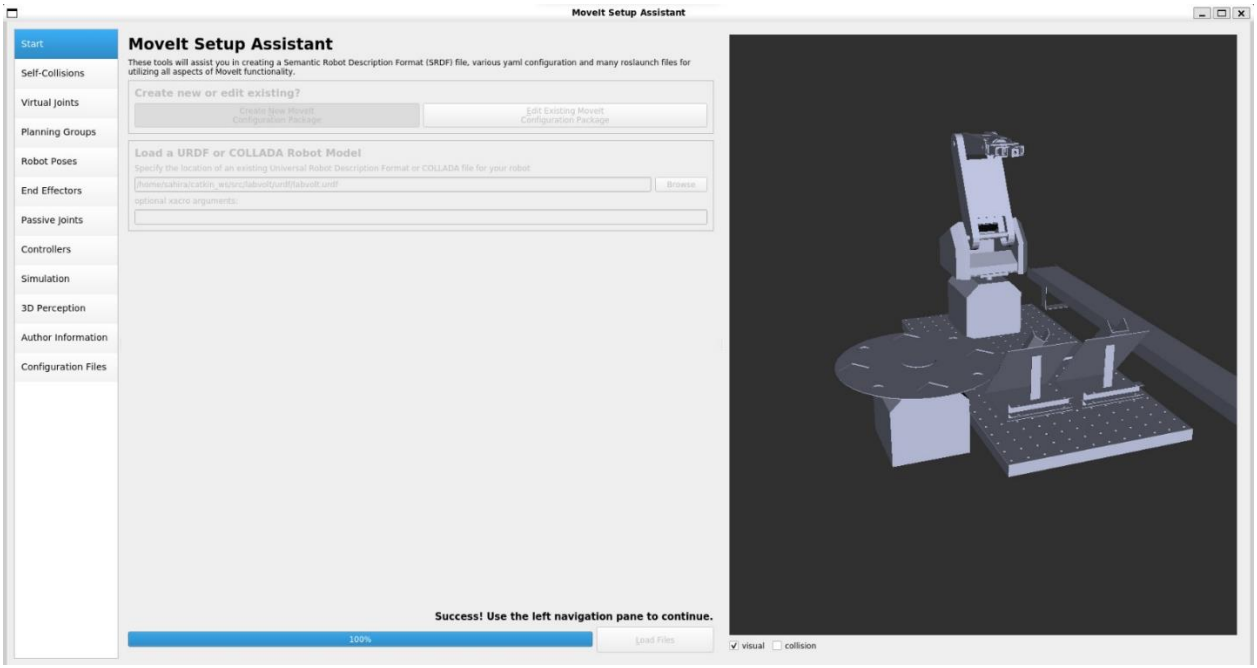


Figura 55. Pantalla principal de MoveIt Assistant con URDF cargado

Fuente: D'Agostini, J. y Nazar, S. (2023)

En la interfaz principal del asistente de MoveIt, en el lado izquierdo, se presenta una serie de pestañas destinadas a realizar configuraciones específicas en relación al modelo cargado. La primera pestaña, denominada "Self-Collisions", permite realizar una evaluación exhaustiva de las posibles auto-colisiones del robot. Esto se logra mediante la generación de una matriz de colisión que identifica pares de enlaces del robot y categoriza su comportamiento en función de colisiones potenciales. Esta categorización incluye la detección de enlaces que siempre están en colisión, aquellos que nunca colisionarán, aquellos que pueden entrar en colisión en posiciones específicas y los que pueden estar en colisión cuando los eslabones son adyacentes en la cadena cinemática. Además, se ofrece la posibilidad de ajustar la densidad de muestreo, que determina cuántas posiciones aleatorias del robot se verificarán para detectar colisiones. En este contexto, se recomienda aumentar la densidad de muestreo al máximo para lograr una evaluación precisa.

En la pestaña "Planning Groups", se definieron los grupos de planificación, cada uno con configuraciones y controladores específicos. La adición de nuevos grupos de planificación se realiza a través del botón "Add group". Cada grupo recibe un nombre y se le asigna un solucionador cinemático para abordar problemas relacionados con la cinemática del robot. En este escenario, se empleó el "KDL Kinematics Plugin" como solucionador cinemático. Además, se seleccionó un algoritmo de planificación para el grupo, en este caso, se utilizó el algoritmo "RRT (Rapidly-

Exploring Random Tree)" para la planificación de trayectorias, y por último, se agregaron las articulaciones pertenecientes al grupo que se está definiendo a través del botón “Add Joints”. Al terminar de configurar cada grupo, la estructura debe quedar definida como se muestra en la figura 55.

El KDL Kinematics Plugin se refiere al Kinematics and Dynamics Library (KDL), una biblioteca de código abierto desarrollada en el marco de la comunidad de código abierto de ROS para el cálculo de cinemática y dinámica en robots. Este plugin es una parte esencial de ROS y permite el cálculo de cinemática inversa, que es fundamental para determinar las posiciones de las articulaciones del robot dadas ciertas posiciones finales del efector final. También se utiliza en el cálculo de la cinemática directa para determinar la posición final del efector final del robot en función de las posiciones de las articulaciones.

Un Group Default Planner RRT (Rapidly-Exploring Random Tree) es un algoritmo de planificación de movimiento utilizado en robótica para resolver problemas de planificación de trayectorias. Este tipo de algoritmo se enfoca en encontrar una ruta de movimiento segura y eficiente para un robot manipulador desde una posición inicial a una posición objetivo en un espacio de configuración. El RRT es un algoritmo que construye un árbol de configuración mediante la exploración aleatoria y sistemática del espacio de configuración del robot. A medida que se generan nuevos nodos en el árbol, se busca una conexión entre la posición actual y un punto aleatorio en el espacio. Esto permitió una búsqueda eficiente de soluciones de planificación.

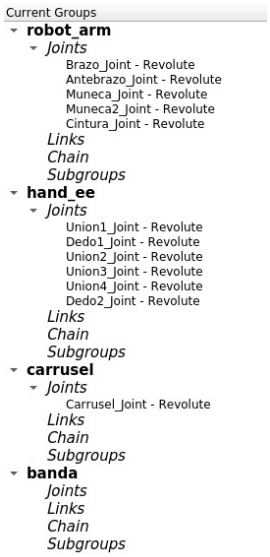


Figura 56. Definición de grupos de planificación en MoveIt Assistant

Fuente: D’Agostini, J. y Nazar, S. (2023)

Posteriormente, en la pestaña "Robot Poses", se procedió a definir exhaustivamente todas las posiciones o puntos específicos de interés en los modelos, los cuales se gestionan mediante los controladores de las articulaciones, tal como se ilustra en la Figura 56. La configuración de estas poses se llevó a cabo en varios pasos: en primer lugar, se seleccionó el grupo que se deseaba mover; a continuación, se asignó un nombre distintivo a la pose en cuestión. Finalmente, se utilizó el conjunto de controladores de articulaciones para posicionar el modelo de forma precisa en la pose deseada. Este proceso se replicó para cada pose necesaria en el desarrollo del proyecto, garantizando un completo y preciso control del modelo en diferentes configuraciones.

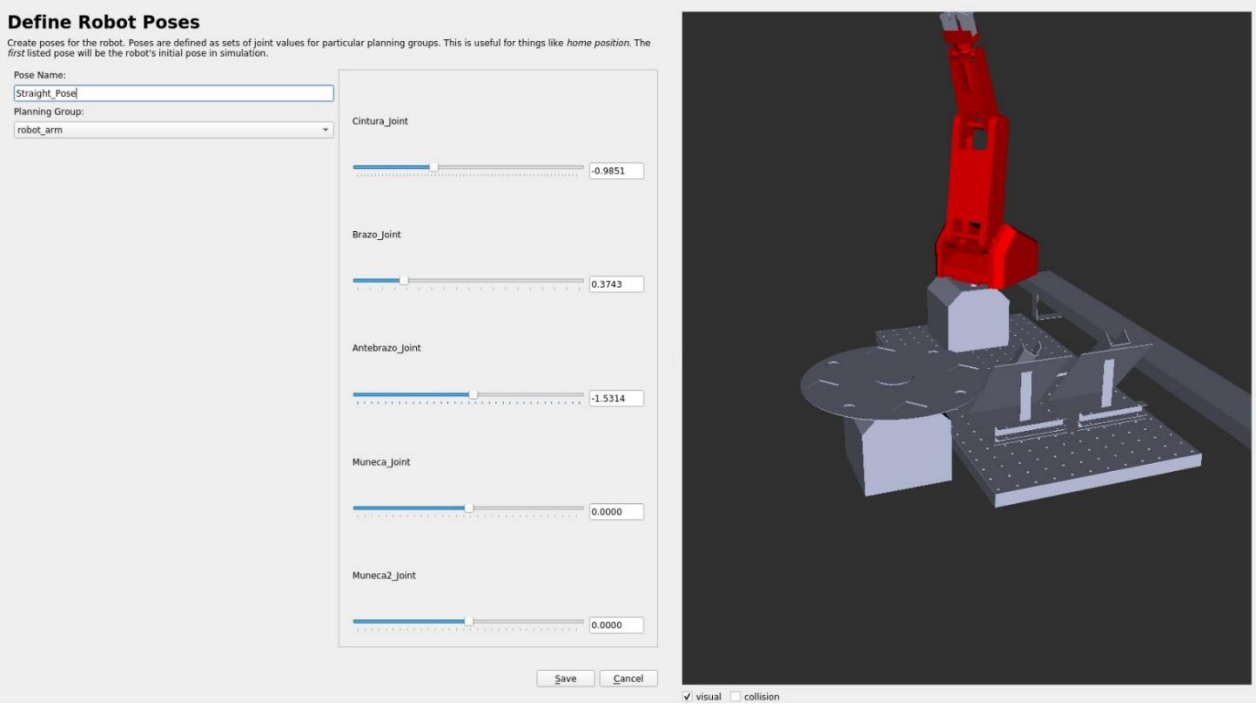


Figura 57. Definición de pose del brazo robótico en MoveIt Assistant

Fuente: D'Agostini, J. y Nazar, S. (2023)

A continuación, en la pestaña "Define End Effector," se procedió a configurar el efector final, conocido como "gripper." Para ello, se asignó el nombre "hand_ee" al efector y se vinculó con el grupo de planificación del gripper, previamente establecido en la opción "End Effector Group." Asimismo, se definió el eslabón "Muneca2_Link" como el eslabón padre, ya que este es el enlace en contacto directo con el gripper. En la opción "Parent Group," se dejó en blanco.

En cuanto a la pestaña "Controllers," no se realizaron modificaciones, ya que los controladores se configuraron de manera manual, lo cual se explicará detalladamente en la siguiente sección. Por otro lado, en la pestaña "Author Information," se completaron los campos

con los datos de los autores del presente trabajo de investigación. Finalmente, en la pestaña "Configuration Files," se especificó el directorio deseado para el paquete de configuración de MoveIt que se generará. En este caso, se estableció el directorio como "/home/nombre_usuario/catkin_ws/src/labvolt_moveit," donde "nombre_usuario" corresponde al nombre del usuario en el sistema operativo Ubuntu, y "labvolt_moveit" es el nombre asignado al paquete generado por MoveIt. Al resto de las pestañas no se les realizó ninguna modificación.

Una vez generado el paquete de MoveIt, este paquete almacena todos los datos esenciales para el control del robot. Sin embargo, es fundamental verificar si se generó correctamente, dado que el archivo predeterminado "ros_controller.yaml" suele crearse de manera incompleta, lo que impide su utilización para el control del robot. Además, MoveIt Assistant genera algunos archivos iniciales para la primera demostración, los cuales se pueden visualizar en la figura 57 y se describen a continuación:

- demo.launch: Abre el robot únicamente en RViz.
- gazebo.launch: Abre el modelo solo en Gazebo.
- demo_gazebo.launch: Abre el robot tanto en RViz como en Gazebo, pero las simulaciones no se presentas conectadas entre sí ya que el archivo ros_controllers.yaml se generó incompleta.

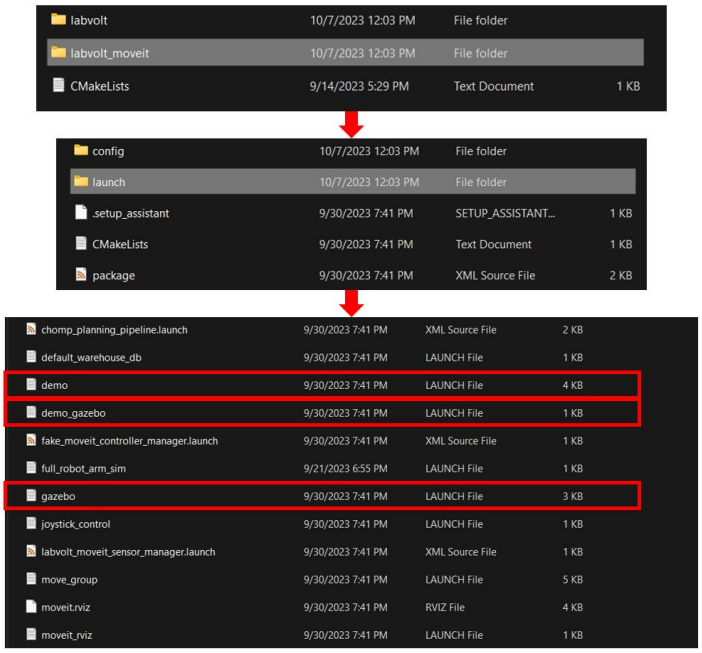


Figura 58. Carpetas de lanzamiento de demostración incial en RViz y Gazebo

Fuente: D'Agostini, J. y Nazar, S. (2023)

Para probar el paquete generado se utilizaron los siguientes comandos:

1. Abrir la terminal.
2. Ir al espacio de trabajo.

   $ cd ~/catkin_ws
3. Reconstruir el espacio de trabajo.

   $ source devel/setup.bash

   $ catkin_make

   $ source devel/setup.bash
4. Lanzar el archivo demo_gazebo.launch

   $ roslauch labvolt_moveit demo_gazebo.launch

Luego de ejecutar el último comando, se espera un tiempo razonable para que los simuladores RViz y Gazebo se carguen. Una vez que se han abierto ambos simuladores, se intentó planear un movimiento para el brazo robótico. La posición inicial del robot se encontraba en posición vertical, mientras que la posición final se definió en el "Goal State". Se seleccionó la posición "home", la cual se configuró previamente utilizando MoveIt. Sin embargo, al intentar ejecutar el movimiento a través de la opción "Execute", se generó un error ("failed"), y el movimiento no se ejecutó en el simulador Gazebo. En contraste, en RViz, se logró planificar el movimiento, como se puede observar en la figura 58. La razón de esta discrepancia se relaciona con el problema mencionado anteriormente, la generación incompleta del archivo "ros_controllers.yaml".

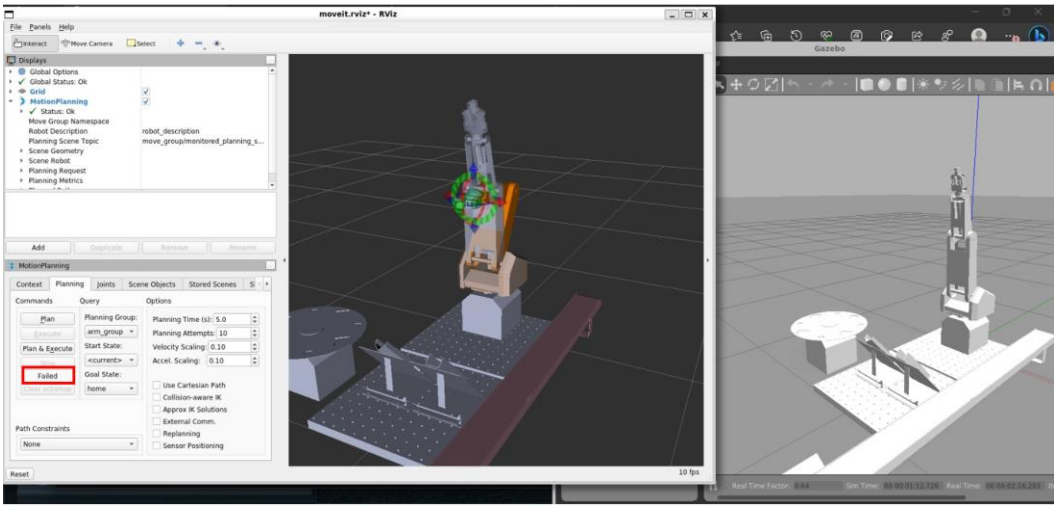


Figura 59. Falla de planificación y ejecución de movimientos en RViz y Gazebo

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.5.10 Arreglos al paquete creado por Moveit

Una vez obtenido el paquete generado por el asistente de configuración de MoveIt, denominado "labvolt_moveit", se encuentra que este paquete es fundamental para la simulación del robot. Contiene una amplia variedad de archivos, como configuraciones, descripciones de robots, poses y controladores, necesarios para el funcionamiento de la simulación en ROS. No obstante, es común que este paquete pueda contener información incompleta o que no se ajuste perfectamente a las necesidades del robot específico.

Para garantizar que la simulación funcione sin problemas y se adapte a los requisitos específicos del usuario, fue esencial abordar cualquier deficiencia en el paquete "labvolt_moveit". En esta parte de la fase, se describen las acciones necesarias para identificar y solucionar los problemas en la configuración del paquete, asegurando así que el robot pueda ser simulado con precisión y eficacia tanto en Rviz como en Gazebo.

4.5.10.1 Crear un nuevo controlador (ROS controller)

Fue esencial en este punto crear un nuevo archivo dentro del paquete con el propósito de establecer la conexión entre el controlador joint_trajectory_controller (previamente creado) y el manipulador ROS. Esta conexión permitió un control eficiente y preciso del robot durante la simulación en Rviz y Gazebo.

Para llevar a cabo este nuevo archivo, es necesario seguir los siguientes pasos:

1. Abrir una nueva terminal.
2. Luego, dirigirse a la carpeta de configuración (config) del paquete MoveIt que se ha creado. Para hacer esto hacemos uso del siguiente comando:
   $ cd ~/catkin_ws/src/labvoolt_moveit /config
3. Una vez ubicados en la carpeta confing, se debe ingresar el siguiente comando para crear un archivo con extensión ".yaml" con el nombre "new_ros_controllers.yaml" y luego presionar la tecla "Enter":
   $ touch new_ros_controllers.yaml
4. El nuevo archivo ha sido creado.
5. Para abrir el archivo recién creado, el usuario puede utilizar el explorador de archivos predeterminado de Windows y colocar el siguiente código:

```
#This is a movit contoller connecting follow_joint_trajectory
controller with JointTrajectoryController
```

```
controller_list:
  - name: robot_arm_controller
    action_ns: follow_joint_trajectory
    type: FollowJointTrajectory
    default: true
    joints:
      - Cintura_Joint
      - Brazo_Joint
      - Antebrazo_Joint
      - Muneca_Joint
      - Muneca2_Joint

  - name: hand_ee_controller
    action_ns: follow_joint_trajectory
    type: FollowJointTrajectory
    joints:
      - Union1_Joint
      - Union2_Joint
      - Union3_Joint
      - Union4_Joint
      - Dedo1_Joint
      - Dedo2_Joint

  - name: carrusel_controller
    action_ns: follow_joint_trajectory
    type: FollowJointTrajectory
    joints:
      - Carrusel_Joint
```

Asegurarse de seguir la indentación correctamente. Use dos espacios en lugar de TAB, ya que el uso de TAB no es compatible en un archivo YAML.

6. Una vez que haya finalizado, guarde y cierre el archivo.

A continuación, se explica el propósito de cada uno de estas líneas de comandos:

- controller_list: Este es el encabezado de la lista de controladores. Indica que se enumeraron varios controladores bajo esta sección.

- name: robot_arm_controller: Aquí se define un controlador específico con el nombre "robot_arm_controller". Este nombre es una etiqueta que se utiliza para identificar y referirse a este controlador en otros lugares del sistema ROS.

- action_ns: follow_joint_trajectory: Esto especifica el espacio de nombres de la acción para el controlador. En este caso, el controlador se utiliza para seguir una trayectoria conjunta (follow_joint_trajectory). Las acciones son una forma de comunicación en tiempo real en ROS, y esta línea indica qué tipo de acción se espera que realice el controlador.

- type: FollowJointTrajectory: Aquí se especifica el tipo del controlador. El tipo "FollowJointTrajectory" indica que este controlador está diseñado para seguir trayectorias conjuntas, lo que significa que es capaz de mover las articulaciones de un robot siguiendo una serie de posiciones y tiempos específicos.

- default: true: Esta línea indica que este controlador es el controlador predeterminado. Cuando se planifican movimientos en el sistema, este controlador se utiliza como el controlador por defecto para ejecutar las trayectorias.

4.5.10.2 Editar el archivo administrador de controladores

Dado que se ha creado un nuevo controlador de ROS, fue necesario actualizar el archivo de configuración del administrador de controladores (controller manager) que se encuentra en la carpeta "launch" de este mismo paquete. El nombre de este archivo de configuración sigue la sintaxis "simple_moveit_controller_manager.launch.xml". Este archivo se utiliza para iniciar el controlador predeterminado generado por el asistente de configuración de MoveIt. Sin embargo, en este paso, fue necesario reemplazarlo con el nuevo controlador creado.

A continuación, se describen los pasos a seguir:

1. Utilizar el administrador de archivos para navegar hasta la carpeta "launch" de su paquete MoveIt. La ruta en este caso es “~/catkin_ws/src/labvolt_moveit /launch".
2. Encontrar el archivo con el nombre "simple_moveit_controller_manager.launch.xml ".
3. Abrir el archivo con el editor de texto de su preferencia.
4. Buscar la línea que hace referencia al archivo "ros_controllers.yaml" y reemplácela por "new_ros_controller.yaml" según se indica a continuación. Asegúrese de guardar los cambios una vez se haya modificado el archivo.

```
<launch>
  <!-- Define the MoveIt controller manager plugin to use for trajectory
execution -->
  <param                                name="moveit_controller_manager"
value="moveit_simple_controller_manager/MoveItSimpleControllerManager"
/>

  <!-- Load controller list to the parameter server -->
  <rosparam                                                    file="$(find
labvolt_moveit)/config/new_ros_controllers.yaml" />
</launch>
```

A continuación, se explica el propósito de cada componente de este código:

- <launch>: Esto indica que se está definiendo un bloque de lanzamiento que contiene las configuraciones y acciones necesarias.
- <param name="moveit_controller_manager" value="moveit_simple_controller_manager/MoveItSimpleControllerManager" />: Esta línea establece un parámetro llamado "moveit_controller_manager" con un valor asociado. En este caso, el valor es "moveit_simple_controller_manager/MoveItSimpleControllerManager". Lo que hace es definir el plugin del administrador de controladores de MoveIt que se utilizará para la ejecución de trayectorias. El "MoveItSimpleControllerManager" es un administrador de controladores incluido en MoveIt que gestiona el control de robots y ejecuta las trayectorias planificadas por MoveIt.
- <rosparam file="$(find labvolt_moveit)/config/new_ros_controllers.yaml" />: Esta línea carga una lista de controladores desde un archivo YAML y la almacena en el parámetro del servidor ROS. El archivo YAML se encuentra en la ruta "$(find labvolt_moveit)/config/new_ros_controllers.yaml".

Estos pasos permitió actualizar el archivo de configuración del administrador de controladores para que use el nuevo controlador de ROS en lugar del controlador predeterminado.

4.5.10.3 Crear un nuevo archivo de lanzamiento para iniciar la simulación de MoveIt en Rviz y Gazebo

En este paso, se requirió crear un nuevo archivo de lanzamiento para cargar y configurar la simulación de MoveIt en Rviz y Gazebo. El nuevo archivo de lanzamiento deberá cargar los siguientes componentes:

- El archivo de lanzamiento "robot_arm.launch" creado en el paquete "robot_arm_urdf" (según se describe en el capítulo 4.5.7 de esta investigación).
- Iniciar el nodo del grupo MoveIt "move_group.launch" creado en el paquete de MoveIt por el asistente de configuración de MoveIt.
- Cargar la configuración Rviz "moveit.rviz" para la visualización en Rviz.

Los pasos a continuación mostrados, fueron los utilizados para crear el archivo de lanzamiento que cargará el nodo del grupo de MoveIt, Rviz y la simulación de Gazebo para el brazo robótico Lab-Volt 5250 y sus accesorios:

1. Abrir una terminal.
2. Dirigirse a la carpeta de lanzamiento (launch) del paquete de MoveIt del brazo robótico. La ruta en este caso es
   $ cd ~/catkin_ws/src/labvolt_moveit /launch
3. Crear un archivo llamado "full_robot_arm_sim.launch" en esta carpeta.
   $ touch full_robot_arm_sim.launch
5. Abrir el archivo creado con el editor de texto de su preferencia.
6. Copiar y pegar el siguiente código.

```
<launch>

    <!-- Launch Your robot arms launch file which loads the robot
in Gazebo and spawns the controllers -->
    <include file = "$(find labvolt)/launch/arm_urdf.launch" />

    <!-- Launch Moveit Move Group Node -->
    <include             file             =             "$(find
labvolt_moveit)/launch/move_group.launch" />
```

```
    <!-- Run Rviz and load the default configuration to see the
state of the move_group node -->
    <arg name="use_rviz" default="true" />

    <include                                          file="$(find
labvolt_moveit)/launch/moveit_rviz.launch" if="$(arg use_rviz)">
        <arg           name="rviz_config"           value="$(find
labvolt_moveit)/launch/moveit.rviz"/>
    </include>

</launch>
```

7. Guardar y cerrar el archivo.

Este nuevo archivo de lanzamiento permitirá cargar y configurar los componentes necesarios para la simulación de MoveIt en Rviz y Gazebo.

4.5.10.4 Compilar el espacio de trabajo

Es fundamental tener en cuenta que, tras llevar a cabo las configuraciones y ediciones previas, resulta primordial compilar el espacio de trabajo de catkin. Este procedimiento garantiza la correcta implementación de todas las modificaciones y ajustes, permitiendo que los paquetes estén completamente preparados para su empleo en el entorno de ROS. A continuación, se detallan los pasos necesarios para llevar a cabo esta compilación:

1. Abrir una nueva terminal.
2. Navegar a la carpeta del espacio de trabajo de catkin, utilizando el siguiente comando:
   $ cd ~/catkin_ws
3. Ejecutar el archivo “setup.bash” para cargar las configuraciones y variables de entorno necesarias, con:
   $ source devel/setup.bash
4. Compilar el espacio de trabajo de catkin utilizando el comando “catkin_make”. Este comando construirá todos los paquetes en su espacio de trabajo:
   $ catkin_make
5. Nuevamente, ejecutar el archivo “setup.bash” con explicado en el paso 3.

Este proceso de compilación es crucial para garantizar que todas las configuraciones y cambios realizados en el espacio de trabajo de estén funcionando de manera efectiva y que el sistema esté listo para iniciar la simulación.

4.5.11 Inicializar la simulación completa del robot

Este paso representa el punto culminante de la configuración y preparación de la simulación del robot en el entorno de ROS. Tras haber realizado todas las configuraciones necesarias, definido controladores, compilado el espacio de trabajo y ajustados parámetros fundamentales, es hora de poner en marcha la simulación. En este paso final, se cargarán los componentes necesarios para ejecutar la simulación de MoveIt con Rviz y Gazebo.

A continuación, se detallan los pasos necesarios:

1. Abrir una nueva terminal y ejecutar el comando:

$ roscore

2. Abrir otra terminal y ejecutar el siguiente comando de lanzamiento

$ roslaunch movit_robot_arm_sim full_robot_arm_sim.launch

Este procedimiento permite observar el robot en acción, ejecutar trayectorias planificadas y evaluar su comportamiento en un entorno de simulación tridimensional (ver figura 56). Al concluir este paso, se habrá establecido una simulación completa y funcional del robot en ROS. Esto facilitará el desarrollo, la depuración y las pruebas de aplicaciones y algoritmos de control antes de llevarlos al mundo real.

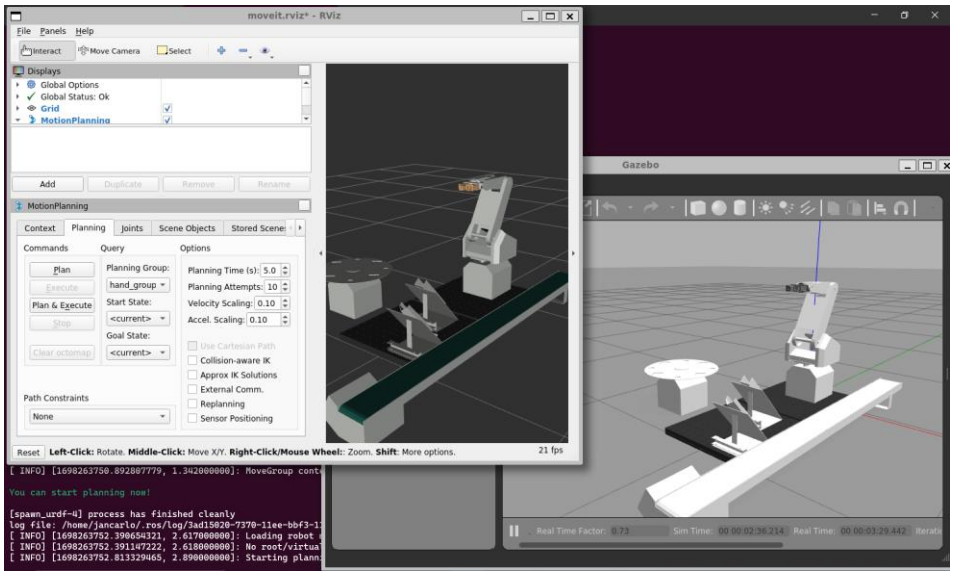


Figura 60. Simulación del robot Lab-Volt 5250 en ROS

Fuente: D'Agostini, J. y Nazar, S. (2023)

4.5.11.1 Crear una secuencia de movimientos

En este paso, se define y configura una secuencia de movimientos para el robot dentro del entorno de simulación de ROS. Esta secuencia de movimientos implica la especificación de comandos que indican cómo el robot debe realizar movimientos en sus articulaciones y en su efector final en el espacio. La creación de esta secuencia de movimientos es de suma importancia, ya que permite la prueba y verificación del comportamiento del robot en diversos escenarios, así como la ejecución de tareas específicas.

Para lograr este objetivo, se aprovechar las herramientas y paquetes de MoveIt, que facilitan la planificación y ejecución de trayectorias. Asimismo, es posible cargar trayectorias predefinidas o crear nuevas trayectorias personalizadas de acuerdo a las necesidades del movimiento. Es importante destacar que este código se implementa en Python para el control y la ejecución de la secuencia de movimientos del robot.

El archivo.py debe estar ubicado en la carpeta "src" que se encuentra dentro del paquete "labvolt". Una vez en esta ubicación, se puede implementar un código similar al siguiente para llevar a cabo la secuencia de movimientos deseada:

```
#!/usr/bin/env python3

## chmod +x /home/user/catkin_ws/src/labvolt/src/<file>.py
## rosrun <package>  <file>.py

## Import librerys
import sys,copy,rospy,moveit_commander,moveit_msgs.msg,
geometry_msgs.msg
from math import pi
from std_msgs.msg import String
from moveit_commander.conversions import pose_to_list

def move_group_python_interface_tutorial():

    moveit_commander.roscpp_initialize(sys.argv)
    rospy.init_node('move_group_python_interface_tutorial',
anonymous=True)
```

```
    robot = moveit_commander.RobotCommander()
    scene = moveit_commander.PlanningSceneInterface()
    display_trajectory_publisher                                    =
rospy.Publisher('/move_group/display_planned_path',

moveit_msgs.msg.DisplayTrajectory,

queue_size=20)
    rospy.sleep(3)

 # Open the Gripper
 group = moveit_commander.MoveGroupCommander("hand_group")
 group.set_named_target("hand_open")
 plan1 = group.plan()
 group.go(wait=True)
 group.clear_pose_targets()

 # Approach the object
 group = moveit_commander.MoveGroupCommander("arm_group")
 group.set_named_target("approach_2")
 plan1 = group.plan()
 group.go(wait=True)
 group.clear_pose_targets()

 # Grasp the object
 group = moveit_commander.MoveGroupCommander("hand_group")
 group.set_named_target("grasp")
 plan1 = group.plan()
 group.go(wait=True)
 group.clear_pose_targets()

 # Move the Robot away
 group = moveit_commander.MoveGroupCommander("arm_group")
 group.set_named_target("approach_6")
 plan1 = group.plan()
```

```
    group.go(wait=True)                                                    4
    group.clear_pose_targets()

    moveit_commander.roscpp_shutdown()
if __name__=='__main__':
  try:
    move_group_python_interface_tutorial()
  except rospy.ROSInterruptException:
    pass
```

El código comienza importando las bibliotecas necesarias y configurando un nodo ROS. A continuación, se define una función llamada move_group_python_interface_tutorial. Esta función contiene toda la lógica para controlar el robot en la simulación. Primero, se inicializa un nodo ROS y se crea un objeto RobotCommander, que permite interactuar con el modelo del robot en la simulación. El código define objetos de grupo de movimiento para diferentes partes del robot, como el brazo y el gripper. Estos grupos de movimiento se utilizan para planificar y ejecutar movimientos específicos del robot. Luego, se establecen objetivos de posición predefinidos para el robot, como "approach_1", "hand_open", "grasp", entre otros. Estos objetivos representan posiciones y estados que el robot debe alcanzar durante la secuencia de movimientos.

Cada grupo de movimiento planifica y ejecuta movimientos específicos para alcanzar los objetivos. Esto incluye acciones como acercarse a un objeto, abrir o cerrar la garra, moverse a una posición específica, liberar un objeto, entre otros. La secuencia de movimientos se compone de una serie de pasos que simulan una tarea específica que el robot podría llevar a cabo en una aplicación real. Al final del código, el nodo ROS se cierra adecuadamente usando el comando `moveit_commander.roscpp_shutdown()`.

CONCLUSIONES

Concluyendo el presente trabajo de tesis, centrado en la virtualización del robot LabVolt 5250 de la Universidad José Antonio Páez utilizando el entorno de ROS (Robot Operating System), se han obtenido valiosas perspectivas y resultados que destacan la importancia de esta investigación. A lo largo de este estudio, se han identificado y analizado numerosos aspectos que subrayan la relevancia y la ventaja de virtualizar un robot en un entorno de código abierto como ROS. El uso de ROS en este proyecto ha demostrado ser altamente beneficioso. ROS proporciona una infraestructura sólida y modular que simplifica el desarrollo, la implementación y el control de robots.

La simulación en ROS proporciona un entorno de pruebas seguro y controlado que permite experimentar y validar el comportamiento del robot sin riesgos para el hardware físico. Esto reduce la posibilidad de daños en el robot real durante el desarrollo y la depuración, al tiempo que acelera el proceso de desarrollo y prueba. Por otro lado, una de las conclusiones fundamentales de este trabajo es la importancia de la virtualización en el ámbito de la robótica. La virtualización del robot LabVolt 5250 a través de ROS permite mantener el robot actualizado y relevante en un entorno tecnológico en constante evolución. Sin esta adaptación a un entorno de código abierto, existe el riesgo de que el robot quede obsoleto y no pueda beneficiarse de las últimas tecnologías y avances en robótica.

En lo que respecta a la transferencia de conocimientos, esta tesis emerge como un gran aporte destinado a orientar y enriquecer futuros proyectos en el campo de la robótica y la virtualización de robots en el entorno de ROS. Los resultados obtenidos y las metodologías desarrolladas en este estudio no solo aportan claridad a la complejidad de la virtualización de robots, sino que también establecen un sólido punto de partida para investigaciones y desarrollos subsiguientes. Su valor radica en proporcionar una guía sustancial y una fuente de referencia para científicos, ingenieros y entusiastas de la robótica que deseen explorar y expandir los horizontes de la simulación y el control robótico en entornos ROS. Además, es una contribución a la comunidad activa y colaborativa que rodea a ROS, la cual también ha sido un recurso invaluable para el desarrollo de este proyecto. La colaboración y el intercambio de información son piezas clave para el dominio de ROS.

RECOMENDACIONES

- Es importante tomar en cuenta los requisitos para poder instalar y utilizar ROS, entre ellos están:
  - Procesador (CPU): Debe contar con al menos cuatro núcleos.
  - Memoria RAM: Se requieren mínimo 8GB, preferiblemente más.
  - Tarjeta gráfica (GPU): Se recomienda una tarjeta gráfica dedicada con soporte para OpenGL.
  - Almacenamiento: Alrededor 20GB libres. Un disco duro sólido (SSD) es preferible debido a su velocidad y rendimiento.
- Para proyectos futuros, se aconseja llevar a cabo un análisis dinámico que permita determinar la matriz de inercia, lo que proporcionará resultados más precisos en la simulación del movimiento del robot virtual en comparación con el robot físico. Además, este enfoque facilitará la obtención de información sobre los materiales exactos utilizados en el modelo, aprovechando la capacidad de SolidWorks para calcular la matriz de inercia.
- Para llevar a cabo proyectos en ROS con éxito, es imprescindible contar con una sólida base de conocimientos en programación. Esto abarca aspectos esenciales de la programación, como el paradigma de programación orientada a objetos (POO), en la que se enfatiza la modularidad y la organización del código. Los lenguajes de programación clave para ROS, como Python, C++, y JavaScript, se convierten en herramientas esenciales para la implementación de algoritmos, controladores y nodos que interactúan en el entorno de ROS.
- Se sugiere la consideración de ROS 2 para futuros proyectos, ya que esta versión tiene un futuro prometedor. Muchos profesionales que trabajan con ROS están migrando sus proyectos a ROS 2, lo que indica que ROS 1 podría ir quedando en desuso gradualmente.
- Establecer una conexión con la comunidad de ROS es esencial. Se recomienda registrarse en foros como ROS Answers y ROS Discourse, así como unirse a grupos de ROS en plataformas de redes sociales como Discord y Telegram. La comunidad de ROS es conocida por su amabilidad y disposición para ayudar a los principiantes. Es importante tener en cuenta las diferencias de horarios, ya que gran parte de la comunidad está

compuesta por miembros de diferentes países, lo que puede afectar los tiempos de respuesta.

- Utilizar Linux como sistema operativo, especialmente Ubuntu, ya que ROS se integra de manera más efectiva en esta plataforma. Además, es esencial familiarizarse con el uso de la línea de comandos de Linux, ya que gran parte de la gestión de ROS se realiza mediante comandos en la terminal.
- Optar por versiones de ROS LTS (Soporte a Largo Plazo) en proyectos importantes. Estas versiones ofrecen un respaldo continuo y actualizaciones de seguridad durante un período prolongado, lo que garantiza una base sólida y confiable para el desarrollo de proyectos a largo plazo. Se recomienda tener muy claro y comprender el funcionamiento de ROS, la comunicación entre nodos, los espacios de trabajo, paquetes, cada tipo de archivo y su función.
- Si experimenta problemas en el simulador Gazebo, como la falta de estabilidad en el modelo que intenta simular o inercias demasiado pequeñas, se recomienda considerar un ajuste en las inercias. Gazebo a veces puede comportarse inadecuadamente cuando se encuentran inercias extremadamente bajas en los componentes. Aumentar un poco las inercias podría resolver estos problemas.
- Es importante prestar una atención especial a las identaciones al escribir código, ya que algunos archivos pueden no ser compatibles con el uso de espacios, y en su lugar, deben utilizarse TABS. Asegurarse de que la codificación esté correctamente formateada con identaciones adecuadas puede evitar problemas y errores en el código.
- En caso de enfrentarse a problemas como la no visualización de los modelos en MoveIt o un rendimiento deficiente en Gazebo, la causa podría estar relacionada con la tarjeta gráfica del computador. En tales casos, es posible que desee considerar la opción de forzar la carga de gráficos a través del procesador en lugar de la tarjeta gráfica. Esto se puede lograr mediante la ejecución de los siguientes comandos en la terminal:

  ```
  $ export LIBGL_ALWAYS_SOFTWARE=1
  $ export LIBGL_ALWAYS_INDIRECT=0
  ```

REFERENCIAS

Arias, F. (2012). **El Proyecto de Investigación. Introducción a la Metodología Científica.** 6ta Edición. Editorial Episteme. Caracas, Venezuela.

Barrientos, Peñin, Balaguer, Aracil. (2007). **Fundamentos de Robótica.** 2da Edición. Editorial McGraw-Hill. Madrid, España.

D'Addario, M. (2016). **Manual de Robótica Industrial. Fundamentos, usos y aplicaciones.** 1ra Edición. Editorial SafeCreative. Madrid, España.

Escobar, J, (2019). **Diseño de sistemas de control industrial de robots basados en la industria 4.0.** Trabajo de Grado. Universidad Técnica de Ambato.

Festo Didactic Inc. (2023). LabVolt. Recuperado de https://labvolt.festo.com

FESTO. (2011). **Automatización y robótica. Manual de Usuario del sistema Lab-Volt series 5250.**

Figueredo, Jiménez, González, Martínez, Moreno, Weffer. (2020). **Manual para la elaboración y presentación de los anteproyectos, proyectos de trabajos de grado, trabajos de grado, tesis doctoral e informe de pasantía y extramuros de la Universidad José Antonio Páez.** Valencia, Venezuela.

Granda, A. (2022). **Propuesta de diseño de un brazo robótico industrial para proceso de soldadura SMAW.** Trabajo de Grado. Universidad José Antonio Páez.

International Federation of Robotics. (2023). Industrial Robots. Recuperado de https://ifr.org/industrial-robots

ISO. (2021). ISO 8373: **Robots y dispositivos robóticos.** Vocabulario (ISO 8373: Robots and robotic devices – Vocabuly) (3ra ed., Versión 1) [Página de inicio]. Recuperado de https://www.iso.org/obp/ui/es/#iso:std:iso:8373:ed-3:v1:en

Labrador, J., Romero, D. (2020). **Simulación de un robot hexápodo con ROS y Gazebo.** Trabajo de Grado. Universidad de Bogotá Jorge Tadeo Lozano.

Morales, K., Hoyos C., García, J. (2019). **Diseño y optimización de la estructura mecánica de un brazo robótico antropomórfico desarrollado con fines educativos.** Trabajo de Grado. Universidad Nacional Experimental del Táchira.

Palella, Martins. (2006). **Metodología de la investigación cuantitativa.** 2da Edición. Editorial FEDUPEL. Caracas, Venezuela.

Peña, A. (2021). **Control de brazo robótico mediante ROS en plataformas de bajo coste.** Trabajo Final de Máster. Universidad de Alicante.

ROS Answers. **Official ROS question and answer forum.** Recuperado de https://answers.ros.org/questions/

ROS Discourse. **The ROS discusión forum.** Recuperado de https://discourse.ros.org/

ROS GitHub Issues. **Core ROS packages.** Recuperado de https://github.com/ros/ros/issues

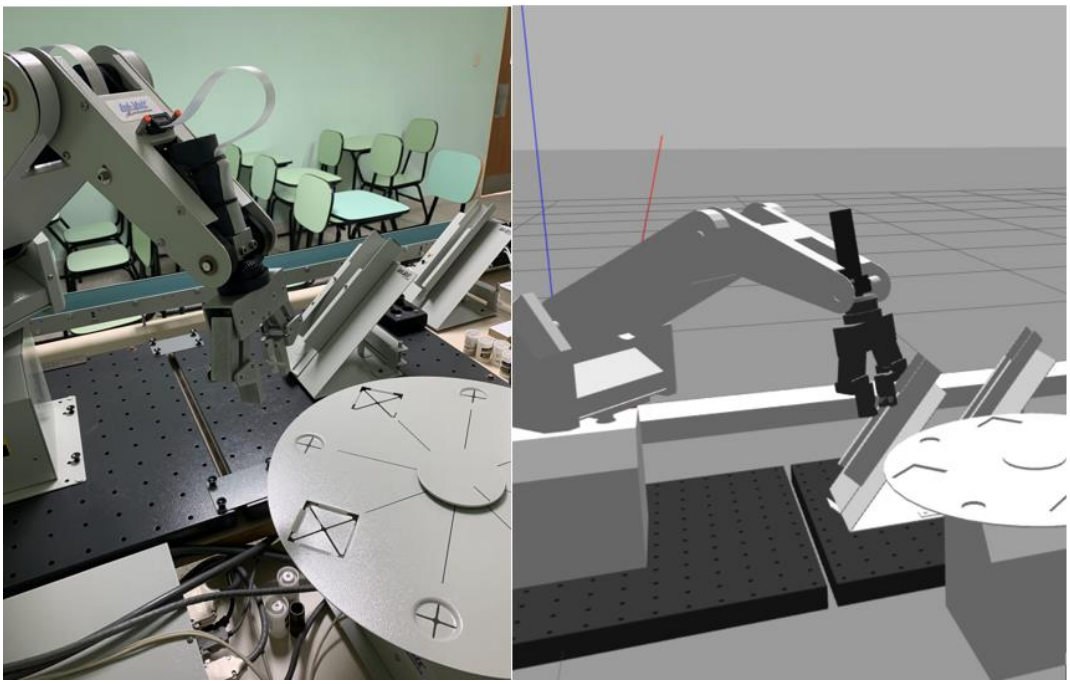
Sampieri, R. (2018). **Metodología de la Investigación.** 6ta Edición. Editorial McGraw Hill. México D.F.

Sanz, W. (2013). **Cinemática de Robots Industriales.** 2da Edición. Dirección de Medios y Publicaciones. Universidad de Carabobo. Valencia, Venezuela.

ANEXOS

A continuación, se muestran imágenes de comparación entre el modelo físico del robot y su correspondiente simulación en ROS.

Anexo N°1 Imagen comparativo del robot real y su simulación (Posición 1)



Anexo N°2 Imagen comparativo del robot real y su simulación (Posición 2)