UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE TELECOMUNICACIONES

Tema:

SISTEMA DE TELEOPERACIÓN CON REALIDAD VIRTUAL PARA LA MANIPULACIÓN DE UN ROBOT MÓVIL

Trabajo de titulación modalidad Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Telecomunicaciones

ÁREA: Física y Electrónica

LÍNEA DE INVESTIGACIÓN: Tecnología de la Información y Sistemas de Control

AUTOR: Bryan Raúl Galarza Toapaxi

TUTOR: Ing. Elizabeth Paulina Ayala Baño, Mg.

Ambato – Ecuador

agosto - 2023

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA DE TELEOPERACIÓN CON REALIDAD VIRTUAL PARA LA MANIPULACIÓN DE UN ROBOT MÓVIL, desarrollado bajo la modalidad Proyecto de Investigación por el señor Bryan Raúl Galarza Toapaxi, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, agosto 2023.

ELIZABETH PAULINA AYALA BANO

Firmado digitalmente porELIZABETH PAULINA AYALA BANO
DN: cn=ELIZABETH PAULINA AYALA BANO c=EC l=QUITO o=BANCO CENTRAL DEL ECUADOR ou=ENTIDAD DE CERTIFICACION DE INFORMACION-ECIBCE
Motivo:He revisado este documento
Ubicaci- n:
Fecha:2023-08-22 09:39-05:00

Ing. Elizabeth Paulina Ayala Baño, Mg.

TUTOR

AUTORÍA

El presente trabajo de titulación titulado: SISTEMA DE TELEOPERACIÓN CON REALIDAD VIRTUAL PARA LA MANIPULACIÓN DE UN ROBOT MÓVIL es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, agosto 2023

Bryan Raúl Galarza Toapaxi

CC. 1805287735

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, agosto 2023

Bryan Raúl Galarza Toapaxi

CC. 1805287735

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Bryan Raúl Galarza Toapaxi, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación titulado SISTEMA DE TELEOPERACIÓN CON REALIDAD VIRTUAL PARA LA MANIPULACIÓN DE UN ROBOT MÓVIL, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con la señora Presidente del Tribunal.

Firmado electrónicamente por:
ELSA PILAR URRUTIA URRUTIA

Ambato, agosto 2023.

Ing. Elsa Pilar Urrutia Urrutia, Mg.
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
VICTOR SANTIAGO MANZANO VILLAFUERTE

Ing. Santiago Manzano, Mg.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
MARCELO VLADIMIR GARCIA SANCHEZ

Dr. Marcelo García, Mg.
PROFESOR CALIFICADOR

DEDICATORIA

A Dios por siempre cuidarme y protegerme en cada paso que doy y quien con su amor eterno y gracia me ha llevado hasta aquí.

A mis padres, Elsa y Raúl, por ser mi guía y soporte en cada paso que doy, por siempre apoyarme y darme el aliento para seguir adelante, por siempre darme lo necesario y lo mejor que han podido, todo su esfuerzo está dando frutos y no ha sido en vano.

A mi hermana, Melanie, que siempre ha estado conmigo en las buenas y malas y a pesar de todos los problemas que nos ha dado la vida, siempre estás ahí para darme tu apoyo.

A toda mi familia que de alguna manera me apoyaron para llegar hasta donde estoy.

Este logro y los que vendrán siempre será gracias a ustedes.

Bryan Galarza

AGRADECIMIENTO

Agradezco a Dios por siempre cuidarme en todo momento y derramar milagros en mi vida y en la de mis padres, sin su gracia y bendición no fuera posible esto.

Agradezco principalmente a mis padres y a mi hermana por siempre estar ahí para mí, por su amor, paciencia y constante dedicación, por siempre creer y apoyarme en cada decisión y paso que doy.

A mis amigos quienes siempre han estado ahí durante toda la carrera universitaria, en los momentos de alegría, tristeza y melancolía. Gracias por su amistad y apoyo.

A mi docente tutor, Ing. Mg. Elizabeth Paulina Ayala Baño por su guía y apoyo durante el desarrollo de mi trabajo de investigación.

Bryan Galarza

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

En los últimos años la industria ha crecido significativamente hasta lo que hoy se conoce como Industria 4.0 y todo esto va de la mano con la automatización de robots. Las industrias empezaron a utilizar los robots móviles, para optimizar ciertas tareas de manufactura y se tenga el mejor resultado posible, evitando los errores humanos, además se asegura la posibilidad de llevar a cabo distintas tareas en áreas donde las personas no tienen acceso, ya que se trata de zonas de difícil ingreso o que resultan perjudiciales para la salud de las personas. El problema de esto es la falta de conocimiento sobre el funcionamiento y el correcto uso del robot.

Debido a estos factores, este proyecto de investigación presenta el desarrollo de un sistema de teleoperación con realidad virtual, empleando las gafas de realidad virtual HTC Vive Pro 2. Esto permitirá que la persona se sumerja en un entorno completamente virtual para familiarizarse con el funcionamiento y control de un robot, adaptándose a sus necesidades, en este caso se trata del Kuka Youbot el cual contiene ROS como sistema operativo.

El sistema de control del robot con realidad virtual se desarrolla utilizando el motor gráfico Unity 3D, permitiendo conectar con el sistema operativo ROS del robot y ejecutar órdenes de movimiento de acuerdo con las necesidades de quien lo esté manipulando. También, se lleva a cabo el control del robot mediante simulación en Gazebo, facilitando la comprensión de su funcionamiento y previniendo posibles daños en el robot real.

Este trabajo de investigación está vinculado al proyecto de investigación “Uso de técnicas de deep learning para optimización de trayectorias autónomas de robots móviles dentro de un proceso industrial” y beneficia a todos los estudiantes que estén interesados en ampliar sus conocimientos en el mundo de la robótica.

Palabras clave: Kuka Youbot, realidad virtual, unity 3D, ros, teleoperación.

ABSTRACT

During the last years the industry has grown significantly to what is now known as Industry 4.0 and all this goes hand in hand with robot automation. Industries began to use mobile robots to optimize certain manufacturing tasks and have the best possible result, avoiding human errors, in addition to ensuring the possibility of carrying out different tasks in areas where people do not have access, since these are areas that are difficult to enter or that are harmful to people's health. The problem with this is the lack of knowledge about the operation and correct use of the robot.

Due to these factors, this research project presents the development of a virtual reality teleoperation system, using the HTC Vive Pro 2 virtual reality glasses. This will allow the person to immerse themselves in a completely virtual environment to become familiar with the operation and control of a robot, adapting to their needs, in this case it is the Kuka Youbot which contains ROS as an operating system.

The control system of the robot with virtual reality is developed using the Unity 3D graphics engine, allowing it to connect with the robot's ROS operating system and execute movement orders according to the needs of whoever is manipulating it. Also, the control of the robot is carried out through simulation in Gazebo, facilitating the understanding of its operation and preventing possible damage to the real robot.

This research work is linked to the research project "Use of deep learning techniques for optimization of autonomous trajectories of mobile robots within an industrial process" and benefits all students who are interested in expanding their knowledge in the world of robotics.

Keywords: Kuka Youbot, virtual reality, unity 3D, ros, teleoperation.

CAPÍTULO I

MARCO TEÓRICO

1.1. Tema de investigación

SISTEMA DE TELEOPERACIÓN CON REALIDAD VIRTUAL PARA LA MANIPULACIÓN DE UN ROBOT MÓVIL

1.1.1. Planteamiento del problema

En los últimos años se ha desarrollado robots industriales, de los cuales los más esenciales y principales son los brazos articulados. Según con la descripción del "Robot Institute of America", un robot industrial se refiere a un dispositivo programable diseñado para manipular materiales, piezas y herramientas mediante movimientos preestablecidos para llevar a cabo diversas tareas. Anteriormente, el rango de aplicación de los robots era muy limitado por lo que surgió la necesidad de extender el campo de la robótica incrementando la autonomía de los robots y limitando la intervención humana. A partir de este momento se emigra hacia la robótica más avanzada, particularmente hacia los robots móviles, los cuales son destinados a trabajar fuera de las áreas de manufactura, principalmente en zonas de difícil acceso o que son consideradas zonas peligrosas que afectan la salud y la seguridad de las personas [1].

Según con la “Federación Internacional de Robótica IFR” había 42041 robots industriales instalados en América Latina a finales de 2017. En el año 2012 había 13100 robots industriales el cual representa un crecimiento del 26% con respecto al 2017, por lo tanto, América Latina está en la retaguardia de la llamada cuarta revolución industrial. Gran parte de los robots en América Latina trabaja para la industria manufacturera y el resto para la industria automotriz, mejorando de esta manera la exactitud y la velocidad en la producción [2].

Los robots móviles son dispositivos automatizados que emplean un software para realizar tareas de manera autónoma, siendo controlados por una computadora. La mayoría de estos robots incorporan diversos elementos como sensores, interfaces de hardware, simulación 3D y otras tecnologías que les permiten reconocer el entorno en el que operan, moverse con cierta habilidad e inteligencia y así tomar decisiones de acuerdo con el ambiente en el que se encuentran, la principal característica es que se mueven o desplazan en uno o más ejes de rotación o traslación. Uno de los principales problemas para la aplicación de los robots móviles es el desconocimiento en su uso y poca preparación [3].

La industria ecuatoriana teme involucrarse en esta área, debido a la falta de personal capacitado para su operación en forma adecuada. Según una "encuesta de la consultora estadounidense Deloitte" el 91% de los millenials (los nacidos entre 1982 y 1994) ecuatorianos considera que la tecnología beneficiará en gran medida la forma en que trabajan. Pero el 51% dice que la empresa en la que labora le está ayudando poco a prepararse para la denominada cuarta revolución industrial, definida por la conectividad con la inclusión de las tecnologías de la información y comunicación. En la ciudad de Quito según "Julio Aguirre" gerente de ingeniería de manufactura de la ensambladora de carros General Motors (GM) también ha comenzado a aplicar en su planta la industria 4.0, por el momento la empresa trabaja con realidad virtual, simulaciones, impresiones 3D, y además están entrenando a sus operadores con gafas virtuales para que reciban instrucciones de ensamblaje [4].

En la Universidad Técnica de Ambato se tiene por el momento el robot manipulador omnidireccional Kuka Youbot para el desarrollo e investigación de los estudiantes en el área de la automatización. La mayor parte de aplicaciones realizadas para el Kuka Youbot vienen de investigaciones con previa programación en el ordenador del robot, en la cual se cargan instrucciones para el movimiento y planificación de rutas desde la API y drivers del Kuka Youbot [5]. En este proyecto de investigación se pretende la manipulación de este con el uso de realidad virtual para que el usuario o la persona que lo esté controlando lo mueva de acuerdo con sus necesidades y de esta manera aprenda sobre su funcionamiento y control, evitando que el robot sufra algún daño.

Mediante el uso de las gafas de realidad virtual el usuario es capaz de sumergirse en un entorno totalmente virtual. Es por este motivo que resultaría útil este proyecto de investigación, que además generaría un gran aporte para los estudiantes que estén interesados en ampliar sus conocimientos en cuanto se refiere a la robótica y las últimas tecnologías emergentes para el aprendizaje como es la realidad virtual.

1.2. Antecedentes investigativos

La presente investigación realizada se ha tomado de distintas fuentes bibliográficas, principalmente de artículos académicos y trabajos investigativos con relación a la realidad virtual y la manipulación de robots móviles, los mismos se detallan a continuación:

En el año 2019 en el artículo de Lorenzo Peppoloni y Carlo Avizzano en su tema "Immersive ROS-integrated Framework for Robot Teleoperation", proponen una interfaz ROS (Robot Operating System) para realizar el control remoto que permite al usuario teleoperar un robot mediante el movimiento de las manos. El usuario tiene la posibilidad de ajustar la autonomía del robot en dos niveles: control directo y seguimiento de puntos de ruta. Para el control se utiliza las capacidades de seguimiento de manos y reconocimiento de gestos de un dispositivo Leap Motion. El usuario mediante un sensor Kinect recibe una retroalimentación visual aumentada en 3D en tiempo real y también con el uso de un HMD. El marco de referencia de las manos, gestos, se exponen en ROS mediante la interfaz Leap ROS, los gestos son utilizados ya sea por la máquina de flujo de estado de modalidad ROS como por el nodo de control ROS para elegir la modalidad de control. La realidad aumentada fue realizada utilizando un framework de alto rendimiento denominado Compact Components (CoCo). Como resultado se tiene una interfaz humano-robot que se basa en gestos para las tareas de manipulación del robot, el rendimiento está influenciado por el espacio de trabajo y la destreza del brazo, además durante las pruebas de control de seguimiento de puntos de ruta el sistema genera una trayectoria factible entre los puntos definidos [6].

En el año 2020 en el artículo de Carlos A. García, Gustavo Caiza y Pablo Vásconez en su tema "Augmented Reality for Robot Control in Low-cost Automation Context and IoT" se realiza una plataforma de realidad aumentada (AR) para controlar y manejar un manipulador robótico, en este caso se trata del Scorbot ER 4U, utilizando un microcontrolador de bajo costo. El primer paso que realiza es el análisis cinemático del brazo robótico para así definir el método de control del robot, el cual es diseñado en un ambiente AR utilizando el motor gráfico Unity 3D, en conjunto con las gafas Meta II para interactuar con el usuario, finalmente los usuarios pueden enviar mensajes de control al microcontrolador usando el protocolo MQTT, estos mensajes fueron validados comprobando que el movimiento del robot es el deseado por el usuario. El manipulador robótico interactúa con las gafas Meta II mediante una tarjeta RPI con los pines de entrada y salida GPIO. Como resultado de la investigación se tiene que la tarjeta embebida Raspberry Pi es un medio eficaz para el movimiento del robot, además que se adapta a varios protocolos de comunicación, mediante esta tarjeta se brinda seguridad y calidad en la transmisión del mensaje del entorno virtual al controlador, además que con el uso de AR es posible manipular cargas pesadas en ambientes peligrosos [7].

En el año 2020 en el artículo de Linh Kastner y Jens Lambrecht en el tema "Augmented-Reality-Based Visualization of Navigation Data of Mobile Robots on the Microsoft Hololens - Possibilities and Limitations" realizan una aplicación en la que se visualiza la navegación o el movimiento de un robot Kuka mobile Youbot dentro de un dispositivo 3D Microsoft Hololens para realidad aumentada. Los datos relevantes de navegación, escaneo láser, mapa del entorno, y los datos de planificación de rutas se visualizan en 3D dentro del dispositivo montada en la cabeza. El robot Kuka Youbot tiene como software ROS Hydro que se basa en un Linux 12.04. Para la parte de comunicación se utiliza Rosbridge para comunicación externa que sigue una notación JSON, ya que ROS y el dispositivo Hololens funcionan desde diferentes sistemas operativos. Para facilitar el trabajo también usan un proyecto de código abierto llamado ROSSharp el cual tiene un framework que permite la comunicación entre los dos sistemas mediante sockets web. El resultado de esta investigación permitió conocer que el Microsoft Hololens es capaz de mostrar grandes cantidades de datos de los sensores sin ninguna reducción de escala visual, para el caso del escaneo

láser se determinó que los datos en tiempo real cambian constantemente, es decir produce grandes cantidades de datos en paralelo y esto afecta el rendimiento de la aplicación ya que la definición de objetivos conlleva más tiempo en el modo de visualización de escaneo laser activado [8].

En el año 2020 en el artículo de Willian Montalvo, Carlos A. García *et.al* en el tema "Sistema de Tele-operación para Robots Móviles en la industria del Petróleo y Gas" implementan un sistema para la automatización en lo que se refiere en la industria del gas y petróleo, debido a que en estos lugares es complicado o difícil de realizar tareas cotidianas como es el mantenimiento o inspección de equipos. Dentro del protocolo de comunicación para el sistema de tele-operación se utiliza el protocolo MQTT para la integración de la información del proceso industrial. El lugar donde se implementa este sistema es en la empresa Petroamazonas EP en Ecuador para que el operario realice tareas de inspección, mantenimiento remoto, comprobación de tuberías, etc. Además, se implementa un entorno virtual desarrollado en Unity 3D con el fin de que el operario tenga cierta transparencia, experiencia y habilidad al momento de controlar el robot esclavo, en este caso se trata de un Kuka Youbot. Para el control del robot el operario utiliza un sensor Leap Motion para que detecte las ordenes de movimiento, tanto de posición como velocidad del brazo robótico y la plataforma omnidireccional, todo esto se ejecuta de acuerdo con las posiciones de manos del operario captados por el sensor Leap Motion. Como resultado se obtiene que la calidad del sistema de captación de movimiento del sensor Leap Motion se toma de acuerdo con órdenes enviadas y ejecutadas teniendo un 91.6% de eficiencia y la parte virtual ayuda a que el operario tenga una retrospectiva visual del entorno donde se encuentra el robot, además de que le permite escuchar el sonido del entorno real para que el usuario esté alerta de cualquier anormalidad sonora dentro del sistema [9].

En el año 2021 en el trabajo de investigación de Manuel Martínez en su tema "Interfaz inmersiva de realidad virtual para la teleoperación de robots móviles" se enfoca en el robot Turtlebot 2 con una interfaz de realidad virtual para la manipulación del robot. En este trabajo utiliza el robot Turtlebot 2 con sistema operativo ROS Kinetic mediante un ordenador con sistema operativo Linux, permitiendo así la lectura y escritura de datos mediante topics, para el caso de simulación se utiliza el simulador de robots

Gazebo7. El PC Linux se comunica con otro terminal con Windows 10 mediante un protocolo TCP-IP ejecutados desde el motor gráfico Unity 3D. Dentro del Unity se utilizan varios paquetes como ROS Rosbridge y la librería de RosSharp para Unity. Para el tratamiento del mundo virtual se utiliza las gafas de realidad virtual Oculus Quest 2, el cual se conecta a Unity mediante el plugin nativo Extended Reality (XR) Plug-In. Además, dentro de Unity se utilizó el modelo URDF del robot para controlar y comunicar el robot, todo esto se logra con el componente ROS Connector, el cual permite la comunicación mediante IP con ROS-Bridge en el lado de ROS y posibilita a los scripts de C# de Unity suscribirse o publicar tópicos en el robot. Como resultado se obtiene que el movimiento del robot en la realidad virtual se ve adelantado al movimiento del robot real, para corregir esto se sincronizó los tópicos y esto se logra con el paquete ROS message_filters, además de esto se configura las frecuencias de publicación de los tópicos, además que el operador dentro de la realidad virtual puede cambiar entre primera y tercera persona para que tenga distintos tipos de puntos de vista [10].

1.3. Fundamentación teórica

1.3.1. Introducción a la robótica

La robótica es un campo extenso que se enfoca en el diseño y construcción de sistemas robóticos capaces de realizar tareas humanas o que requieren cierto nivel de inteligencia para operar. Las disciplinas relacionadas incluyen el álgebra, los autómatas programables, las máquinas de estados, la mecánica, la física y la informática [11].

La automatización de procesos es un concepto que requiere mayor conocimiento que la robótica, debido a que, con la automatización, un proceso o determinadas partes de un proceso se realizan sin la intervención de una persona. El proceso funciona con aplicaciones informáticas predefinidas y maquinas eléctricas o mecánicas. Todas las aplicaciones que están predefinidas consisten en algoritmos desarrollados en los cuales las operaciones ya están determinadas y se ejecutan sin tener en cuenta los cambios que suceden en el entorno [12].

La robótica y la automatización están estrechamente relacionadas, ya que, en la mayoría de los casos, los robots forman parte de sistemas automatizados. Sin embargo, en ocasiones también se utilizan robots sin automatización o con un nivel mínimo de automatización, lo que se conoce como robótica no automatizada [12].

Robot móvil

Los sistemas robóticos que se desplazan en un entorno y poseen diversas capacidades para realizar tareas complejas pueden ser controlados de manera autónoma o por un operador humano. Estos robots móviles están equipados con sensores y actuadores que les permiten comprender el entorno y adaptarse a él. Existe una amplia clasificación de robots, pero los robots móviles son aquellos que llevan a cabo tareas que los robots manipuladores no son capaces de realizar debido a sus limitaciones físicas y su naturaleza estacionaria. En algunos casos, es posible encontrar manipuladores que se montan sobre plataformas móviles para ampliar sus capacidades [13].

Conceptos relacionados a los robots

Al hablar de los robots móviles hay que tener en cuenta los grados de libertad, es decir la cantidad de movimientos independientes que son capaces de realizar. Los conceptos que se relacionan a los robots o manipuladores móviles se describen en la Tabla 1.

Tabla 1. Conceptos relacionados a los robots [14].

Manipulador	Tipo de robot que funciona con componentes electromecánicos y es capaz de realizar interacciones con el entorno en el que se encuentra.
Efector final	Parte del manipulador donde se coloca una herramienta, por ejemplo, unas pinzas, con el propósito de recoger objetos.
Espacio de trabajo	Se refiere al espacio que puede llegar el efector final de un robot, se toma en cuenta la posición y orientación.
Posición	Ubicación del robot en un punto de referencia que se desea
Orientación	Se refiere a la ubicación rotacional o angular del robot

Articulaciones	Es la parte que se ubica entre dos eslabones del brazo robótico para permitir el movimiento.
Cinemática	Estudio del movimiento del robot sin tener en cuenta las fuerzas que actúan sobre él.
Actuador	Dispositivo que se usa con la finalidad de generar un efecto o movimiento.
Sensores	Obtienen datos o valores reales del movimiento del robot para el control de este.

1.3.2. Robot Kuka Youbot

El Kuka Youbot es un tipo de robot móvil manipulador diseñado principalmente para su uso en entornos educativos e investigativos, pero también es aplicable en investigaciones industriales. Este manipulador móvil se caracteriza por su integración y sincronización fluida con la movilidad, lo cual es una tecnología ampliamente utilizada en las industrias modernas [15].

La plataforma robótica omnidireccional Kuka Youbot consta de varias componentes notables, como un chasis, 4 ruedas mecanum, motores, alimentación y una placa Mini-ITX con un procesador Intel Atom Core D510. Una vez se instala el sistema operativo Ubuntu y el entorno de desarrollo ROS, el robot se transforma en una computadora autónoma que solo necesita una conexión básica a una pantalla, teclado y ratón. Además, los usuarios tienen la opción de ejecutar programas y controlar el robot de manera remota mediante una computadora[15].

El brazo robótico del Kuka Youbot está compuesto de 5 partes o articulaciones rotativas y también de una pinza que tiene dos dedos como efector final. La altura que alcanza aproximadamente es de 655 mm, también tiene un peso de 6.3 Kg y además puede transportar una carga útil de hasta 0.5 kg de peso. La máxima velocidad de rotación de cada articulación es de 90 grados y los dedos (efector final) se puede abrir

hasta 11.5 mm por dedo [15]. En la Figura 1 se visualiza el robot omnidireccional Kuka Youbot.

Figura 1. Robot Kuka Youbot [15].

El robot cuenta con un sistema operativo Ubuntu 12.04 con Ros Hydro, además cuenta con una API que está desarrollado en lenguaje de programación en C++, mediante esto el desarrollador o programador es capaz de controlar o manipular cada una de las partes del robot. Este software es de código abierto y cuenta con un subsistema funcional y desacoplado, donde el brazo robótico se representa como una cadena cinemática de 5 grados de libertad y la plataforma omnidireccional es tratada como un conjunto de juntas rotatorias [16].

La API de Youbot tiene 3 clases principales las cuales se detallan a continuación [17].

- **YoubotManipulator:** representa el brazo como una serie de articulaciones con una pinza.
- **YoubotBase:** representa la plataforma omnidireccional del Kuka Youbot.
- **YoubotJoint:** representa una articulación, puede ser el brazo o la plataforma.

1.3.3. Partes físicas del Kuka Youbot

El kuka Youbot tiene dos partes generales principales, el brazo robótico, el cual tiene como efector final dos pinzas y la plataforma omnidireccional. En la Figura 2 se visualiza de mejor manera la estructura que contiene el Kuka Youbot.

Figura 2. Partes del Kuka Youbot [18].

En la Tabla 2, se tiene una breve descripción de las partes que componen el Kuka Youbot.

Tabla 2. Descripción de las partes del Kuka Youbot [18].

Item	Descripción
1	Gripper: gripper_finger_joint_l, gripper_finger_joint_r
2	Youbot arm: arm_joint_1, arm_joint_2, arm_joint_3, arm_joint_4, arm_joint_5.
3	Youbot Platform (Plataforma)
4	Youbot loading (área de carga opcional)

Dentro de la movilidad del brazo se tiene un rango de giro de 338º, como se observa en la Figura 3.

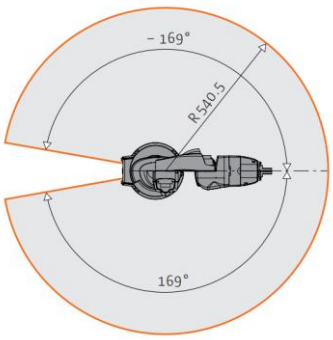

Figura 3. Rango de giro del brazo Kuka Youbot [19].

La plataforma móvil omnidireccional se encarga de movilizar el brazo con el fin de aumentar su rango de operabilidad, esta plataforma cuenta con llantas Mecanum para que se pueda tener un desplazamiento lateral sin necesidad de girar la parte frontal de la plataforma [19]. La Figura 4 muestra la plataforma omnidireccional.

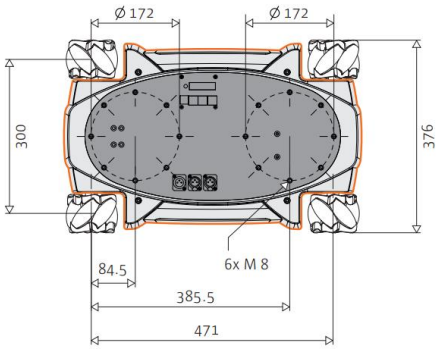

Figura 4. Plataforma omnidireccional del Kuka Youbot [19].

1.3.4. Sistema Operativo Robótico (ROS)

ROS es reconocido como un middleware robótico, es decir, una colección de marcos de trabajo para el desarrollo de software de código abierto destinado a robots. Este software proporciona una amplia gama de servicios, que abarcan desde la abstracción del hardware y el control de dispositivos de bajo nivel hasta la comunicación de mensajes entre procesos y la gestión de paquetes. Aunque ROS tiene una orientación principal hacia sistemas UNIX, como Ubuntu-Linux, también está siendo adaptado para ser compatible con otros sistemas operativos, como Fedora, macOS, Debian y Microsoft Windows. Además, ROS cuenta con herramientas y bibliotecas que facilitan la construcción, escritura y ejecución de código en múltiples computadoras [20].

Dentro de ROS se considera un nodo maestro, el cual permite que los demás nodos que existen establezcan comunicación, esto se hace de dos modos, ya sea por mensajes o por los llamados tópicos y mediante la comunicación servicio-cliente. Hay que tener en cuenta que la comunicación servicio-cliente es uno a uno y los tópicos es uno a varios. En todos los sistemas ROS existen una variedad de nodos los cuales por si solos no sirven de nada, a menos que se comuniquen entre si [15]. En la Figura 5 se tiene los medios para la comunicación en ROS.

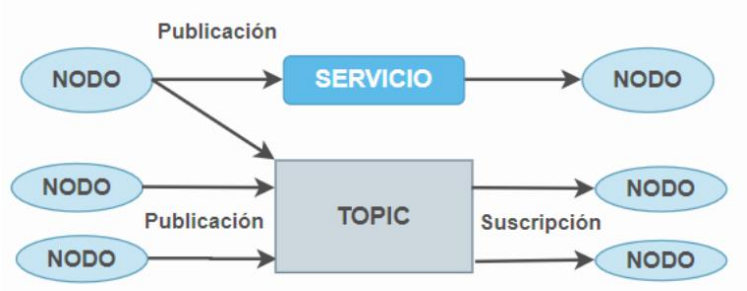

Figura 5. Medios de comunicación en ROS [15].

Comunicación en ROS

En el framework de ROS los nodos son procesos individuales que se llevan a cabo y se comunican entre sí para realizar tareas. Estos nodos se comunican mediante tópicos y mediante esto intercambian información en forma de mensajes, todo esto se explica a continuación.

Los nodos que se usan en ROS son prácticamente archivos ejecutables dentro de una aplicación de ROS, todos los nodos se comunican entre sí. Los nodos en otras palabras son los que se encargan de realizar una acción, servicio o publicar o suscribirse a alguna información. Dentro de esto, existen librerías internas que gestionan los nodos, como son rospy y roscpp. Rospy es para lenguaje Python y roscpp para lenguaje en c++ [21].

Los tópicos también conocidos como temas son la manera que tienen los nodos de comunicarse entre ellos. Mediante estos tópicos los nodos son capaces de pasarse mensajes. Estos mensajes tienen una estructura predeterminada o también pueden ser creados por el desarrollador. Cada tópico acepta solo un tipo de mensaje, es decir que solo aceptará los mensajes que tengan la estructura adecuada [21].

Por otro lado, los mensajes son solamente datos con una estructura determinada. Los mensajes estan compuestos por integers, floats, strings, booleano, arrays, etc [21]. En la Figura 6 se tiene un ejemplo de mensajes en ROS.

Figura 6. Ejemplo de mensajes en ROS [21].

Paquetes en ROS

Todos los paquetes que se crean dentro de ROS tienen la estructura básica y que es necesaria para crear los programas de ROS, dentro de cada paquete se tiene las siguientes carpetas [22]:

- **src:** ficheros ejecutables (Python, C++, java)
- **lib:** librerías que se generan al correr la compilación con rosmake
- **config:** parámetros de configuración del paquete
- **data:** información del usuario
- **msg:** definición de los mensajes que se usan
- **srv:** Son los servicios que se usan por el paquete
- **launch:** permiten lanzar el paquete y todos sus nodos al ingresar un solo comando

Espacio de trabajo (Catkin Workspace)

El espacio de trabajo es solamente una carpeta en el ordenador, desde esta carpeta se va a introducir los paquetes necesarios para compilar y ejecutar la aplicación desarrollada. La forma de agregar paquetes y la estructura viene definida por el sistema "catkin" (anteriormente rosbuild). En resumen, si se quiere realizar una aplicación robótica basado en ROS, tanto el workspace como los paquetes deberán estar hechos con el sistema catkin [23]. La estructura del "catkin workspace" se observa en la Figura 7.

Figura 7. Estructura del espacio de trabajo catkin [23].

Dentro del espacio de trabajo hay 3 carpetas, la que más importa es la carpeta src, dentro de esta carpeta se tiene la siguiente estructura, como se muestra en la Figura 8.

Figura 8. Estructura de la carpeta src [23].

En la Figura 8 se observa que se tiene las carpetas de los distintos paquetes y un archivo CMakeLists.txt, el cual es una plantilla que ayuda a entender la información si otro paquete quiere usar el workspace.

Por último, dentro de cada paquete se tiene una carpeta src que es donde se ubican los archivos de programación para el funcionamiento de la aplicación ROS y dos archivos más el CMakeLists.txt que tiene la misma función como se explicó antes y el package.xml que define el nombre del paquete, la versión de este, autores y dependencias [23]. La estructura del paquete se observa en la Figura 9.

Figura 9. Estructura de los paquetes [23].

1.3.5. Teleoperación

La teleoperación se refiere al conjunto de tecnologías que permiten controlar u operar un dispositivo de forma remota, siendo controlado por un ser humano. En otras palabras, el término "teleoperar" se utiliza para describir la acción de una persona al operar un dispositivo a distancia. Por otro lado, un sistema de teleoperación es aquel

que habilita la posibilidad de controlar cualquier dispositivo conocido como "dispositivo teleoperado" [24].

Elementos de un sistema de teleoperación

Los elementos que componen un sistema de teleoperación se observan en la Tabla 3.

Tabla 3. Elementos de un sistema de teleoperación [24].

Operador o teleoperador	Generalmente se trata de una persona el cual realiza a distancia el control de la operación que se está realizando.
Dispositivo teleoperado	Se trata de cualquier dispositivo, por ejemplo: un manipulador, un robot, un vehículo. Se trata del dispositivo remoto y que está siendo controlado.
Interfaz	Permite la interacción del operador con el sistema de teleoperación.
Control y canales de comunicación	Conjunto de dispositivos que se encargan de modular, transmitir y adaptar las señales que se envían desde la zona remota y la local.
Sensores	Conjunto de dispositivos que recogen información de la zona remota para ser utilizado por la interfaz.

1.3.6. Protocolos de comunicación orientados a la teleoperación de robots

Es fundamental tener conocimiento acerca de los canales o medios de comunicación disponibles para el intercambio de información entre Unity y el robot teleoperado. Dentro de esto la teleoperación de robots exige que el canal de comunicación debe presentar una baja latencia y que sea coherente con las acciones o necesidades que el robot va a realizar para evitar la inestabilidad del sistema [25].

Protocolo TCP IP

Este protocolo está diseñado para establecer una comunicación confiable de los datos, este se encarga solamente de enrutar los paquetes y no está orientado a la conexión. TCP es capaz de solventar varias debilidades como es la alta tasa de error y la confiabilidad. Además, TCP se puede emplear en conjunto con ROS garantizando que los mensajes enviados desde el robot lleguen a ROS ya que emplea nodos de comunicación. Por otro lado, TCP se asegura que los datos se entreguen en orden ya

que usa técnicas como la retransmisión de paquetes perdidos y la confirmación de recepción [25].

Protocolo RTP

Este protocolo funciona sobre UDP. Está orientado a la comunicación en tiempo real y se basa principalmente en el uso de buffers para compensar de cierta forma los cambios que se producen con el retardo y el reordenamiento de paquetes. RTP no garantiza la entrega confiable de los paquetes como lo hace TCP ya que en situaciones de congestión de red los paquetes pueden perderse, además de requerir un ancho de banda considerable especialmente si se transmite video o imágenes [25].

Protocolo MQTT

Este es un protocolo de transporte del tipo cliente-servidor que se basa en la mensajería del tipo publicación/suscripción y está diseñado principalmente para el IOT. Su principal característica es que es simple y fácil de implementarlo y resulta útil cuando se requiere un pequeño consumo de procesamiento de código para la comunicación y un bajo consumo ancho de banda. Un servidor central o bróker es el núcleo de esta arquitectura el cual se encarga de administrar las conexiones entre clientes MQTT, es decir que todos los clientes se conectan a este servidor y este se encarga de encaminar los mensajes. En cuanto se refiere a la implementación de MQTT con ROS Hydro que se basa en Linux 12.04 lamentablemente se encuentra bastante desactualizado y esto puedo dificultar la implementación con MQTT ya que varias bibliotecas y herramientas no son compatibles. MQTT no es un protocolo nativo de ROS Hydro ya que este usa principalmente ROS1 y su protocolo de comunicación ROS Topic/ROS Service se basa en TCP/IP [25].

1.3.7. Realidad Virtual

La realidad virtual es una tecnología informática que busca crear un entorno completamente simulado, ofreciendo a las personas una experiencia diferente a las interfaces tradicionales. En este espacio inmersivo, los usuarios interactuan con

objetos tridimensionales. Las limitaciones clave de esta tecnología se centran en la disponibilidad de contenido y las capacidades del ordenador empleado para su ejecución [26]. En la Figura 10 se presenta un ejemplo de realidad virtual.

Figura 10. Ejemplo de realidad virtual [26].

Clasificación de la realidad virtual

Existen diferentes tipos de realidad virtual y se deben conocer sus diferencias, pues cada uno de ellos tiene un uso más adecuado dependiendo de su aplicación. En la Tabla 4 se describe la clasificación de la realidad virtual.

Tabla 4. Clasificación de la realidad virtual [27].

RV Inmersiva	En este tipo de realidad la persona se sumerge completamente en el entorno virtual, en este caso se utilizan gafas de realidad virtual y software.
RV no Inmersiva	En este tipo no se sumerge del todo el usuario o la persona en el entorno virtual. Un ejemplo de esto son los videojuegos.
RV semi inmersiva	Esta representa una combinación de la realidad virtual inmersiva y no inmersiva, aquí generalmente se emplea elementos virtuales y físicos, un ejemplo son las cabinas de pilotaje de aviones.

Componentes de un sistema de realidad virtual

Para lograr la naturaleza que caracteriza a cualquier sistema de realidad virtual se hace énfasis en una serie de componentes que actúan para reproducir la realidad virtual [28].

- **Audio Inmersivo:** Dentro de un entorno virtual no solo es necesario la parte visual, el audio es importante para asegurar una experiencia total.

- **Conjunto de sensores:** Estos sistemas deben disponer de mecanismos que en los espacios tridimensionales se conoce como los 6 grados de libertad (arriba-abajo, derecha-izquierda, adelante-atrás).
- **Controladores de posición y seguimiento:** Los movimientos y las acciones quedan registrados y son expresado en el nuevo espacio simulado.
- **Lentes y pantallas:** Esto permite la creación del efecto estereoscópico dando lugar a la simulación 3D y generando un espacio en el cual se puede mover.
- **Cuadros por segundo:** Las gafas de realidad virtual deben ser capaces de reproducir las imágenes a una correcta velocidad para evitar dolores de cabeza, desorientación.

1.3.8. Hardware de Realidad Virtual

Existen distintas versiones de hardware con las cuales se consigue un mundo virtual.

Simuladores

Los simuladores por lo general están ubicados en salas específicas, no se trata de dispositivos para tener en la casa, es decir se encuentran en lugares donde hay una gran instalación. Un ejemplo claro es Zero Latency el cual es un espacio dedicado para realidad virtual en donde es posible moverse a través de distintas salas como si estuvieras dentro del juego, incluso es posible usar falsas armas [29]. En la Figura 11 se observa un ejemplo de este simulador.

Figura 11. Simulador Zero Latency [29].

Gafas para móviles

Dentro de esto existen las gafas Cardboard, las cuales son unas gafas de cartón en donde se introduce o se coloca el teléfono y mediante esto es posible disfrutar de esta

tecnología de RV. Son una opción barata para uso en la casa [29]. En la Figura 12 se muestra un ejemplo de gafas para móviles.

Figura 12. Gafas para móviles [29].

Gafas para gaming

Estos cascos son más profesionales, los cuales se conectan al computador. Un ejemplo de estas gafas de realidad virtual son las Oculus, también las HTC Vive. Estos dispositivos se caracterizan por sus grandes características como la resolución e inmersión total para realidad virtual, el precio de cada uno es variado, pero sumamente alto [29]. En la Figura 13 se presenta un ejemplo de gafas de realidad virtual.

Figura 13. Gafas para gaming [29].

1.3.9. Gafas HTC Vive Pro 2

Estas gafas de realidad virtual son una versión mejorada de los anteriores Vive Pro originales y además un gran competidor frente a Meta Quest 2. Una de las ventajas es su compatibilidad con los controladores de la versión anterior y que además tiene una de las mejores resoluciones 2488x2488 pixeles por cada ojo, llegando a tener una mejor resolución de 5K frente a Meta Quest 2 y su versión anterior de HTC [30].

En la Tabla 5 se presenta de forma más detallada las características de las HTC Vive Pro 2.

Tabla 5. Especificaciones de HTC Vive Pro 2 [30].

Especificaciones HTC Vive Pro 2	
Peso Total	850 gramos
Estructura del visor	Plástico resistente color negro
Cámaras	2 cámaras en la parte frontal
Panel frontal	Es posible alejar o acercar con un botón mejorando la comodidad
Sujeción del visor	Arnés acolchado de 3 puntos y ajustable
Controladores	Joystick, botón central, 2 botones secundarios, tecnología háptica
Estaciones base	2 estaciones para posicionarnos dentro del espacio.
Resolución	2488x2488 pixeles llegando a 5K
Pantalla	LCD RGB
Campo visión	120 grados
Frecuencia refresco	90 Hz, 120 Hz
Requisitos mínimos de PC	Intel Core i5-4590/ AMD ryzen 1500, GPU NVIDIA GTX 1060, 8 GB Ram, DisplayPort 1.2, USB 3.0

Componentes de las gafas HTC Vive Pro 2

Las gafas de realidad virtual están compuestas de varias partes que se necesitan para su funcionamiento. En la Figura 14 se muestra cada una de las partes que componen las gafas de realidad virtual.

Figura 14. Componentes de las gafas HTC Vive Pro 2.
Elaborado por: El Investigador basado en [30].

La descripción de cada una de las partes se presenta a continuación [30].

- **Estaciones Base:** Funcionan como sensores los cuales sirven para posicionar a la persona dentro del espacio virtual, de este modo el sistema es capaz de captar los movimientos en tiempo real y sincronizar con las gafas.
- **Gafas HTC:** Son las que permiten experimentar el mundo virtual desarrollado, en conjunto con el movimiento del robot.
- **Caja Link Box:** Permite conectar las gafas de realidad virtual a la computadora, de este modo se conecta la escena desarrollada en unity.
- **Controladores:** Cuenta con un joystick, botón central, y dos botones secundarios. Son los que se usan para el control del robot y la interacción con el mundo virtual.

1.3.10. Software para realidad virtual

Los principales programas que se utilizan para crear entornos y experiencias de realidad virtual son los denominados motores de videojuegos, los cuales requieren conocimientos de programación y también de diseño. Un motor de videojuego es indispensable para realizar una interfaz de VR. La mayor parte de motores gráficos son gratuitos y se integran con SDKs (kits de desarrollo de software). Hay SDKs para una gran variedad dependiendo de la plataforma (Oculus, HTC, PSVR) entre otros. Los motores gráficos más conocidos para el desarrollo de RV dentro de la industria son Unity, Unreal Engine y Godot Engine [31]. A continuación, se describe cada uno de ellos:

a) Unity 3D

Unity goza de un amplio reconocimiento y uso como uno de los motores de videojuegos más populares, especialmente en el campo de los juegos móviles. Una de las características destacadas de Unity es su función de vista previa inmersiva en realidad virtual, que permite una experiencia visual inmersiva a través de dispositivos de visualización montados en la cabeza (HMD), lo cual resulta altamente beneficioso para aumentar la eficiencia al diseñar entornos virtuales de realidad virtual. Unity es compatible con los principales HMD y se estima que aproximadamente el 60% de los contenidos de realidad aumentada y virtual se crean utilizando esta plataforma.

Además, Unity es un software de código abierto disponible para todas las plataformas [31]. Este software requiere de ciertos requisitos los cuales se describen a continuación:

Requisitos Mínimos

- Requiere de un sistema operativo Windows 7/8/10, MacOS X 10.8 o superior
- 4 GB de RAM como mínimo
- 15 GB de almacenamiento
- Tarjeta gráfica compatible con DX9 o DX11

Requisitos Recomendados

- 15 GB de almacenamiento
- 8 GB de memoria RAM disponible
- Sistema operativo Windows 7/8/10, MacOS X 10.8 o superior
- Tarjeta gráfica Intel Graphic 4000 o superior, Nvidia con 1GB VRAM dedicada.

A continuación, en la Figura 15 se muestra la interfaz de Unity 3D.

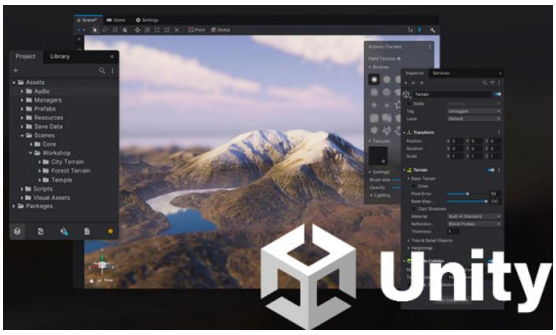

Figura 15. Interfaz de Unity 3D [31].

b) Unreal Engine

Es otro motor de videojuegos que proporciona un conjunto robusto de herramientas para el desarrollo de realidad virtual. Está especialmente diseñado para desarrolladores experimentados y se destaca en entornos empresariales donde varios programadores trabajan en colaboración, debido a su nivel de complejidad. Aunque la comunidad y la cantidad de tutoriales disponibles para Unreal Engine no son tan extensos como los de Unity, se considera una herramienta superior en términos de funcionalidades y capacidades. Además, permiten desarrollar juegos VR con altas velocidades de

renderizado, iluminación global, sombras, efectos visuales, etc. Una de las características principales es que tiene un sistema de blueprints lo cual facilita la programación ya que solo tienen que establecer la lógica y la interacción del juego sin necesidad de tener que programar [31]. Este software requiere de los siguientes requisitos:

Requisitos mínimos

- Se requiere de un sistema operativo Windows 10 de 64 bits
- Un procesador Quad-core o AMD de 2.5 GHz
- Memoria RAM de 8 GB
- Tarjeta gráfica Nvidia RTX-2000 o una AMD RX-6000 compatible con DirectX 11 o 12

Requisitos recomendados

- Se necesita de un SO Windows 10 de 64 bits versión 20H2
- Un procesador Six-Core Xeon 3.4 Ghz
- Se necesita 64 GB de memoria RAM
- Se necesita 256 GB en disco sólido
- Una tarjeta gráfica GeForce RTX 2080

A continuación, en la Figura 16 se muestra la interfaz de Unreal Engine.

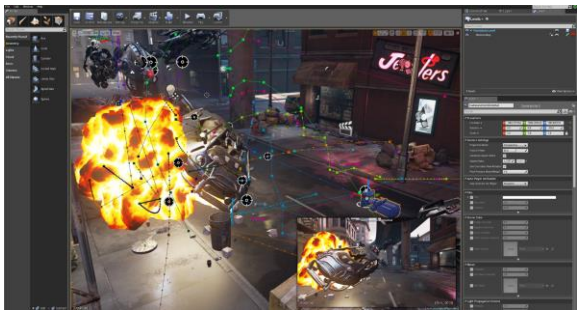

Figura 16. Interfaz de Unreal Engine [31].

c) Godot Engine

Es uno de los proyectos de desarrollo de videojuegos de código abierto más populares, es un software fácil de aprender e intuitivo, y ha experimentado un crecimiento significativo en su comunidad en los últimos años. Utiliza nodos y escenas como elementos fundamentales. Una escena en Godot es básicamente un árbol de nodos, y

cada nodo representa los diferentes elementos que componen el juego. Además, permite el desarrollo de varias aplicaciones en distintas plataformas como Windows, macOS, Linux, IOS. También incorpora un buen motor de física que permite crear escenarios complejos, además permite controlar las propiedades de los objetos como el tamaño, forma, color, comportamiento. Esta plataforma cuenta con una gran cantidad de desarrolladores como tutoriales, documentación y ejemplos de aprendizaje [32]. A continuación, se detalla los requisitos de este software.

Requisitos mínimos

- Sistema operativo Windows 7
- Procesador OpenGL 2.1
- Mínimo 2 GB de RAM

Requisitos recomendados

- Sistema operativo Windows 10
- Procesador con OpenGL 3.3
- Memoria de 4G de RAM

A continuación, en la Figura 17 se muestra la interfaz de Godot.

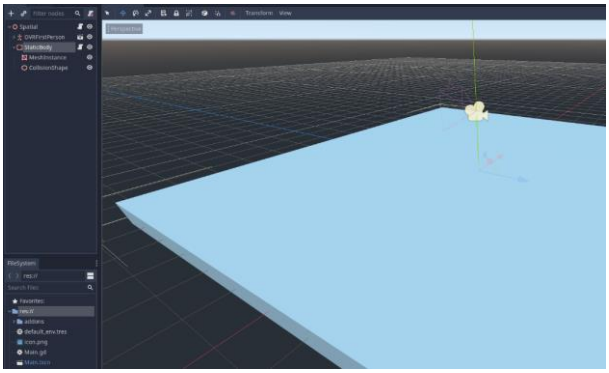

Figura 17. Interfaz de Godot Engine [32].

1.3.11. Software de simulación de robots

Los programas de simulación de robots son una herramienta importante ya que permite conocer el comportamiento del robot antes de su implementación en un entorno real, además que permiten realizar programas y algoritmos de programación en un entorno simulado [33]. De acuerdo con esto es importante seleccionar un software que permita simular el Kuka Youbot para evitar daños y no tener inconvenientes con el robot real.

A continuación, se presenta tres softwares que se usan para la simulación de robots con sus principales características.

a) Gazebo

Es posible realizar simulaciones realistas, incluido la física e interacciones de objetos, así como la dinámica y sensores de distintos tipos de robots, además de que es posible crear un propio robot a través del formato SDF, collada, y URDF. Este simulador utiliza el core de ROS, por lo que permite simular el comportamiento de cualquier robot al utilizar paquetes que son desarrollados para ROS. A continuación, se presenta algunas características [33], [34].

- Posibilidad de programar en Python, java, c++.
- Dispone de una gran biblioteca de sensores como cámaras, láseres, radares y GPS que se pueden simular.
- Se definen parámetros como la iluminación, interacción con objetos, creación de escenarios con múltiples robots

A continuación, en la Figura 18 se observa la interfaz de gazebo.

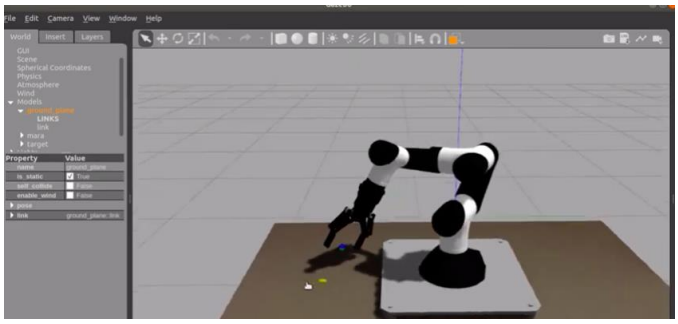

Figura 18. Entorno de simulación gazebo [33].

b) CoppeliaSim

Anteriormente conocido como V-REP, es un simulador para robótica que se usa para la parte de investigación y cuenta con un entorno de desarrollo integrado, es decir que proporciona herramientas integradas que facilita la creación de contenido en robótica. Además, se basa en el control distribuido, es decir que se necesita un script para cada objeto/modelo dentro de la simulación. Es posible programar en Python, Java, C++ o ROS. A continuación, se presenta algunas características [33], [34].

- Posibilidad de importar modelos CAD de robots.

- La instalación es compatible con versiones de Ubuntu a partir de la 18.04 en adelante.
- Tiene herramientas de compilación de código en un formato ejecutable
- Cuenta con un motor de física que permite simular las colisiones del robot, la dinámica y cinemática de los robots, así como la gravedad.
- Cuenta con herramientas de visualización y análisis en tiempo real para ver el comportamiento de los robots.

A continuación, en la Figura 19 se observa la interfaz de CoppeliaSim.

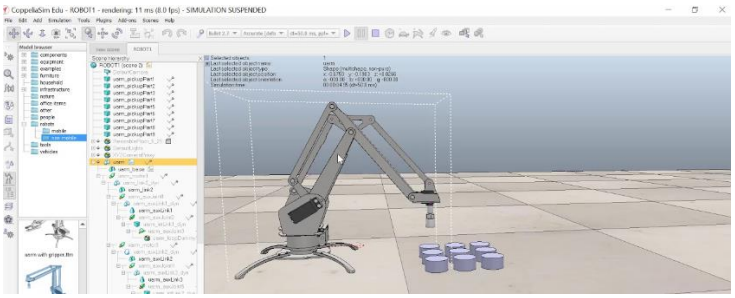

Figura 19. Entorno de simulación de CoppeliaSim [34].

c) Webots

Es un software de código abierto y multiplataforma que se usa para simular robots en entornos virtuales flexibles con distintos lenguajes de programación como son: C, C++, Python, Java, Matlab o también mediante ROS con una API simple. Se puede crear prototipos de robots, y también probar inteligencia artificial [33], [34]. A continuación, se muestra algunas características de este simulador.

- Una de sus principales características es que tiene modelos de brazos robóticos, drones y varios sensores como el sensor lidar.
- Permite ejecutar las simulaciones en tiempo real y también permite ejecutar la simulación en modo paso a paso lo que permite analizar el robot en cada parte de la simulación.
- Tiene herramientas para el análisis de datos, es decir cuando se usa los sensores se tiene herramientas que capturan estos datos.

A continuación, en la Figura 20 se observa la interfaz de webots.

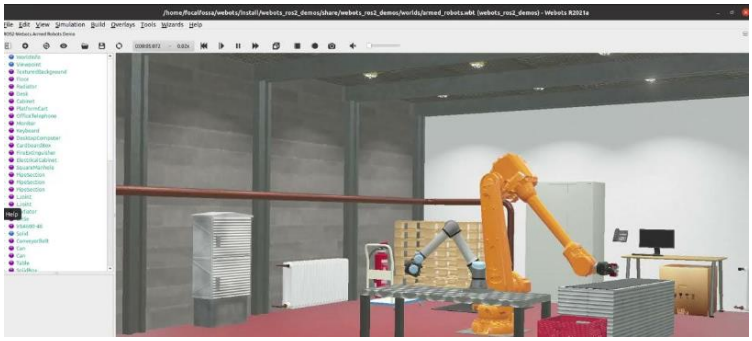

Figura 20. Entorno de simulación Webots [34].

1.4. Objetivos

1.4.1. Objetivo General

Desarrollar un sistema de teleoperación con realidad virtual para la manipulación de un robot móvil.

1.4.2. Objetivos Específicos

- Analizar la cinemática y el funcionamiento de la plataforma y el robot mediante la herramienta Gazebo para el control del robot móvil.
- Diseñar el entorno de realidad virtual mediante Unity 3D para la visualización y control del movimiento del robot móvil con las gafas de realidad virtual.
- Realizar las pruebas de funcionamiento y comunicación de la realidad virtual con el robot móvil mediante pruebas experimentales para la validación del sistema de teleoperación.

CAPÍTULO II

METODOLOGÍA

2.1. Materiales

Para el desarrollo del trabajo de investigación se hace un análisis de lo que se va a utilizar, con el fin de elegir lo que mejor se adapte a las necesidades requeridas y que también se ajuste al proyecto de investigación vinculado “Uso de técnicas de deep learning para optimización de trayectorias autónomas de robots móviles dentro de un proceso industrial”.

Con respecto al hardware se usa el robot Kuka Youbot debido a que se tiene en la facultad. Este robot fue desarrollado para la parte de educación e investigación y sus principales especificaciones se mencionan en el informe técnico de la cinemática del Anexo A. Para la parte de realidad virtual se utiliza las gafas HTC Vive Pro 2 ya que las especificaciones de este son compatibles con la PC que se va a utilizar, los requisitos mínimos para su funcionamiento se mencionaron en el marco teórico en la Tabla 5, además tiene mejores características en cuanto a resolución y comodidad, anteriormente no se las ha usado, ya que en otros proyectos de la Universidad se centran en el uso de las gafas Meta 2, por lo que se pretende usarlas para generar innovación y además que se tiene disponible en la facultad.

Con respecto al software de realidad virtual se utiliza Unity 3D, para esto se realizó una comparación con otros motores gráficos que se mencionan en la Tabla 6. Los 3 motores gráficos son usados para el desarrollo de videojuegos multiplataforma, y cuentan con características similares, sin embargo, la diferencia está en el contenido que permiten desarrollar. Unity esta más enfocado para el desarrollo de realidad virtual con una calidad gráfica media, a diferencia de Unreal Engine que es una calidad muy alta, por lo que se necesita una PC más potente, por otro lado, Godot tiene una calidad gráfica baja. Además, Unity permite realizar entornos con elementos de interfaz, interactuar con ellos, exportar paquetes de una manera sencilla. Otro aspecto importante es la programación en C# que resulta más fácil de aprender, ya que solo se

necesita conocimientos básicos, a diferencia de Unreal que usa programación en C++ y es un poco más complicado de aprender. Por último, Unity es compatible con las gafas HTC Vive Pro-2 ya que solo se necesita importar el plugin de SteamVR y comenzar a experimentar la realidad virtual.

Tabla 6. Comparación de softwares para realidad virtual [35].

Propiedades	Unity 3D	Unreal Engine	Godot
Plataformas de desarrollo	Unity se considera un motor grafico multiplataforma líder en la industria por la versatilidad	Unreal tiene muy poca optimización en los proyectos orientados para móviles.	Motor gráfico orientado a los juegos 2D y 3D multiplataforma
Curva de aprendizaje	Una persona que está empezando desde cero, trabaja más rápido con Unity, ya que el aprendizaje es más fácil, por la estructura de su editor, como por el lenguaje de programación.	Unreal puede tener una curva de aprendizaje alto al inicio, debido al editor más complejo y un lenguaje de programación difícil.	Fácil de aprender, su curva de aprendizaje es relativamente bajo, ya que existe una variedad de información y tutoriales para aprender.
Lenguaje de programación	Utiliza c# (C sharp) el cual es un lenguaje orientado a objetos y fácil de aprender.	Se basa en C++ el cual es un lenguaje un poco difícil de aprender y también con Blueprints los cuales se organizan en nodos.	Utiliza el lenguaje de programación GDScript el cual es un lenguaje especifico de Godot, también es posible en C++
Documentación	Tiene una gran cantidad de documentación disponible, ya que cuenta con una gran comunidad	No hay tanta documentación, además que no cuenta con una gran comunidad	Hay muchos tutoriales, videos de aprendizaje, ideal para el aprendizaje en videojuegos
Calidad de gráficos	Esta entre el rango de media-alta	Mayor nivel de calidad que Unity	Baja-Media
Costo	Es gratuito, también tiene versiones de pago	Es gratuito, sin embargo, Epic cobra el 5% en calidad de royalties cada trimestre.	Es gratuito, pero sus opciones son limitadas. Menor capacidad para objetos 2D y poca renderización 3D, texturas limitadas.

En cuanto se refiere al software de simulación de robots se utiliza Gazebo, este es un software que está muy apegado a ROS ya que tiene una integración nativa con ROS, en cambio CoppeliaSim y Webots necesitan de plugins y controladores adicionales. Por otro lado, Gazebo viene integrado en la instalación de ROS y los otros tienen ciertas restricciones. Además, Gazebo tiene una serie de modelos de robots incluido el Kuka Youbot, por lo que solo se necesita arrastrar el modelo hacia la simulación y empezar a programar, en cambio CoppeliaSim y Webots se necesita crear o importar un modelo desarrollado. Los detalles se muestran en la Tabla 7.

Tabla 7. Comparación de softwares para simulación de robots [36].

Propiedades	Gazebo	CoppeliaSim	Webots
Integración con ROS	Tiene una integración nativa con ROS, es decir que se puede utilizar directamente con las herramientas y paquetes de ROS, los modelos y sensores se publican y suscriben por nodos.	Ofrece soporte para ROS, es decir no viene por defecto, por lo que para la integración con ROS se necesita de un plugin "RosInterface". Este permite la comunicación entre CoppeliaSim y ROS.	De igual modo no tiene una integración directa por lo que se necesita de un controlador especifico llamado "ros_controller" el cual permite la comunicación mediante mensajes y servicios de ROS.
Soporte multiplataforma	Está disponible para sistemas operativos como: Linux, macOS Y Windows. Gazebo viene integrado con ROS en su instalación.	Disponible en plataformas como: Linux, macOS Y Windows, sin embargo, algunas características o plugins no están disponibles en todas las plataformas.	Ofrece soporte para las plataformas de Linux, macOS Y Windows, pero está enfocado para sistemas embebidos lo que significa que se ejecuta en hardware especifico.
Curva de aprendizaje	Su curva de aprendizaje es de moderada a alta debido a su amplio conjunto de características y su enfoque en la simulación de robots complejos	Tiene una curva de aprendizaje más avanzado ya que se requiere comprender conceptos de programación y uso de lenguajes de scripting.	Tiene una curva de aprendizaje moderada, es amigable para principiantes en robótica y ofrece una interfaz de configuración.

Documentación	La documentación oficial de gazebo es bastante completa y proporciona tutoriales y varios ejemplos.	La documentación de CoppeliaSim proporciona tutoriales y guías para comenzar a familiarizarse con el software.	También tiene una gran variedad de tutoriales y ejemplos para comenzar.
Modelos y entornos	Tiene una amplia variedad de modelos de robots y entornos predefinidos incluyendo el Kuka Youbot	Aunque tienen modelos de robots, las bibliotecas son muy limitadas.	También tiene varios modelos, pero la mayor parte de ellos se los tiene que crear con algún software de modelado 3D.

En cuanto se refiere al protocolo de comunicación se utiliza TCP/IP ya que en este se basa el framework de ROS, además existe un paquete en Unity que se llama ROSTCPConnector el cual solo es necesario descargarlo y comenzar a utilizarlo. En cuanto a MQTT no es posible usarlo debido a la versión de ROS 12.04 que tiene el robot real y no es compatible con las bibliotecas y paquetes necesarios para MQTT. En cuanto se refiere a RTP tampoco es factible ya que en este existe perdidas de paquetes y no garantiza la entrega de paquetes como lo hace TCP.

2.2. Métodos

2.2.1. Modalidad de investigación

A continuación, se describe los distintos tipos de investigación que se ha aplicado durante el desarrollo del siguiente proyecto de investigación.

En el proyecto se realizó una investigación aplicada, ya que se empleó los estudios existentes para solucionar el problema del robot móvil, debido a que este robot no cuenta con un sistema de teleoperación con realidad virtual y también se enfoca directamente a los problemas de la sociedad o el sector industrial.

En el proyecto se realizó una investigación bibliográfica, ya que se sustentó mediante la recolección de información en tesis, libros, artículos científicos, publicaciones en internet y en proyectos realizados anteriormente con la aplicación del robot Kuka Youbot.

En el proyecto se realizó una investigación de campo, ya que se trabajó directamente con el robot móvil Kuka Youbot y se realizaron pruebas de funcionalidad y control en la Universidad Técnica de Ambato, además que las gafas de realidad virtual se aplican directamente en el lugar para el control y visualización del robot móvil Kuka Youbot.

2.2.2. Población y muestra

Población

Se toma como población a los estudiantes de la carrera de Telecomunicaciones de la Facultad de Ingeniería en Sistemas Electrónica e Industrial, con un total de 500 estudiantes. Se escoge esta población ya que ellos son los que van a utilizar directamente el sistema con fines de aprendizaje y además están más familiarizados con los temas de comunicaciones y transmisión de datos. Dado que la interfaz de realidad virtual para manipular un robot tiene relación directa con estos aspectos, su conocimiento les permitirá comprender, evaluar la viabilidad y las implicaciones técnicas de la interfaz de realidad virtual. Además de que pueden ofrecer información más precisa y detallada respecto del tema.

Muestra

Para el cálculo de la muestra se toma en cuenta la fórmula para variables cualitativas de una población finita. Los parámetros de la muestra se tienen en la Tabla 8.

$$n = \frac{Z^2 * P * Q * N}{E^2(N-1) + Z^2 * P * Q}$$

Tabla 8. Datos de la muestra.

Parámetros	Datos
N: Tamaño de la población	500
Z: Nivel de confianza (95%)	1.96
P y Q: Proporción de éxito y fracaso (50%)	0.50
E: margen de error (15%)	0.15

Elaborado por: El Investigador

En donde el tamaño de la muestra es de:

$$n = \frac{1.96^2 * 0.5 * 0.5 * 500}{0.15^2(500 - 1) + 1.96^2 * 0.5 * 0.5}$$

$$n = 39.39 = 39$$

2.2.3. Recolección de información

Para el desarrollo del proyecto de investigación, la información se obtuvo de artículos científicos, libros, revistas científicas, fuentes online y proyectos desarrollados en relación con la realidad virtual y la teleoperación de robots móviles, por lo que se tomó en cuenta bases de datos confiables y repositorios universitarios para el desarrollo del proyecto de investigación. Además, se realiza una encuesta para presentar resultados.

2.2.4. Procesamiento y análisis de datos

Para el procesamiento y el análisis de los datos se realizó de la siguiente manera:

- Revisión de la información que tenga relación con el sistema de teleoperación con realidad virtual y el robot Kuka Youbot.
- Análisis de la información recopilada y características principales que debe tener la realidad virtual.
- Planteamiento de la solución mediante la aplicación de la realidad virtual.
- Control y verificación de los datos obtenidos de las pruebas de comunicación de la manipulación del robot móvil con la realidad virtual.

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1. Análisis y discusión de los resultados

El desarrollo de un sistema de teleoperación con realidad virtual representa una innovadora estrategia para la manipulación de robots, permitiendo a cualquier persona sumergirse en un mundo totalmente virtual y aprender eficazmente sobre el funcionamiento y control de un robot para luego aplicarlo en zonas donde sean inaccesibles o sean perjudiciales para la salud de las personas. Además, mediante la realidad virtual la persona realiza distintas pruebas para saber cuándo colisiona el robot con sus propias articulaciones o con los objetos que lo rodean. Todo esto se logra en conjunto con el análisis de la cinemática ya que permite entender el movimiento de cada parte del robot y con esto se conoce la posición del efector final. Los resultados obtenidos mediante encuesta son positivos ya que se aplicó el método SUS el cual sirve para medir la usabilidad de la realidad virtual. Además, mediante pruebas de comunicación se concluye que mediante TCP es posible controlar un robot sin ningún problema, sin presentar mucha latencia entre cada paquete enviado y recibido.

3.2. Desarrollo de la propuesta

El desarrollo del proyecto se realizó en base al análisis de distintos autores de artículos científicos y proyectos de investigación, y mediante esto se desarrolló un sistema de teleoperación con realidad virtual para la manipulación de un robot móvil. El control del robot se lo hace desde Unity mediante la creación de una escena para realidad virtual usando las gafas HTC Vive Pro-2 y aquí la persona elige que parte del robot desea controlar o manipular. Además, este trabajo es de gran aporte hacia el proyecto de investigación “Uso de técnicas de deep learning para optimización de trayectorias autónomas de robots móviles dentro de un proceso industrial”.

3.2.1. Requerimientos del sistema de teleoperación con realidad virtual

Para controlar el Kuka Youbot con una interfaz de realidad virtual, es necesario disponer de los objetos y modelos que se importen a Unity para crear la escena.

Asimismo, se requiere el modelo del robot para permitir que el usuario observe con las gafas de realidad virtual el movimiento de este. El desarrollo de la interfaz de realidad virtual se lo hace teniendo en cuenta las directrices y normas que se aplican a la usabilidad de la realidad virtual.

Principios y directrices de la usabilidad de la realidad virtual

La usabilidad no es un término homogéneo es decir que sea igual para todo. De hecho, hace referencia a varias definiciones como son el rendimiento, el tiempo en que se ejecuta, la satisfacción del usuario y la dificultad de aprendizaje [37].

La usabilidad de la realidad virtual es importante para garantizar una experiencia satisfactoria y efectiva para los usuarios. Sin embargo, no existen normas específicas para la usabilidad de entornos virtuales, pero si hay principios y directrices que se aplican al diseño de experiencias en realidad virtual las cuales se describen en la Tabla 9.

Tabla 9. Principios y directrices de la usabilidad de la realidad virtual [37].

Comodidad física	Esto implica minimizar la fatiga ocular, el mareo y la incomodidad física. Evitar movimientos bruscos, velocidades excesivas en la perspectiva de la cámara que puedan causar malestar en los usuarios.
Intuitividad	La interfaz y la interacción deben ser fáciles de entender, es decir que los usuarios entiendan rápido como navegar por el entorno virtual.
Retroalimentación clara	Implica utilizar señales visuales, auditivas o táctiles para indicar acciones realizadas correctamente.
Facilidad de navegación	Proporcionar a los usuarios opciones para moverse dentro del entorno virtual como por ejemplo el desplazamiento mediante teletransportación.
Legibilidad y tamaño del texto	El texto debe ser legible y de un tamaño adecuado. Considerar la resolución de los dispositivos de RV.
Interacción física	Aprovechar capacidades de seguimiento como: gestos de las manos, movimiento de la cabeza, controladores hápticos.

Normativas aplicables a la usabilidad de la realidad virtual

La ISO tiene normas las cuales se relacionan con la usabilidad y la experiencia del usuario las cuales se aplican al diseño y el desarrollo de experiencias en realidad

virtual, además otros autores también mencionan el concepto de usabilidad y se describen en la Tabla 10.

Tabla 10. Definiciones de usabilidad aplicables según normas ISO [38].

Definiciones de la usabilidad	Estándar
Trata sobre la calidad en el software y hace referencia a la usabilidad como la capacidad que tiene un software o programa desarrollado para ser comprendido, teniendo en cuenta la facilidad de aprendizaje de uso y que sea atractivo para el usuario en ciertas condiciones específicas de uso.	ISO/IEC 9126-1
Hace referencia a la usabilidad como el nivel al cual puede llegar un software para ser utilizado por ciertos usuarios específicos en un contexto específico para completar una meta concreta con eficacia, eficiencia y un grado de satisfacción.	ISO 9241-11
Describe el proceso de diseño de un software involucrando al usuario y entendiendo los requerimientos de este, además incluye directrices como: asignación adecuada de funciones al sistema, procesos iterativos, retroalimentación al usuario. Este se aplica para involucrar a los usuarios en la evaluación y mejora de experiencias de realidad virtual.	ISO 13407
Proporciona directrices para diseñar interfaces de usuario accesibles. Esta es relevante para garantizar que las experiencias de realidad virtual sean accesibles para personas que tienen discapacidades.	ISO 20282
Describe la usabilidad para asegurar que las herramientas desarrolladas sean fáciles de usar, eficientes, eficaces y satisfactorias permitiendo al usuario alcanzar los objetivos deseados.	Nielsen y Molich
Menciona que la usabilidad es importante para el aprendizaje y que sea atractivo para el usuario apoyándose en la robustez y flexibilidad tanto en las funciones, mensajes o texto diseñados para una mejor experiencia del usuario.	Jacko y Sears

3.2.2. Esquema general del sistema de teleoperación

El esquema general donde se presenta el control del robot con el sistema de realidad virtual se muestra en la Figura 21. A continuación se detalla cada una de las partes del esquema.

Sistema Maestro: Se compone de la interfaz de realidad virtual desarrollado en Unity, en donde se importa el paquete de SteamVR para que de este modo Unity reconozca las gafas de realidad virtual HTC Vive Pro 2. Además, se importan los paquetes "URDF Importer" para tener el archivo URDF del robot el cual contiene todas las mallas, texturas y archivos necesarios que se usan para el control del robot, el paquete

“ROS TCP Connector” en donde se coloca la dirección IP del robot y el puerto 10000 de ROS para establecer comunicación y enviar ordenes de movimiento desde Unity a ROS. Los scripts tanto de control del robot como de la comunicación con ROS se desarrollan mediante programación en C# mediante el editor de código Visual Studio.

Protocolo de comunicación: El protocolo para comunicar Unity con ROS es TCP IP, el cual es el encargado de conectar “ROS TCP Connector” de Unity con el nodo “ROS TCP Endpoint” que se encuentra en el lado de ROS del robot. Hay que destacar que para el nodo ROS TCP Endpoint hay que crear un espacio de trabajo en ROS con comandos de ROS y colocar los archivos dentro de esta carpeta. Para establecer comunicación con Unity hay que inicializar el nodo del driver del robot el cual es el encargado de iniciar los motores correspondientes de cada articulación del brazo y además el nodo de comunicación para que se conecten los publicadores de Unity.

Sistema Esclavo: Esta parte se compone de ROS, es decir el sistema operativo del robot, en el cual se encuentra el nodo que recibe los mensajes de Unity y también el nodo del driver del Kuka youbot, el cual es el encargado de inicializar los tópicos para que reciba los mensajes desde Unity a ROS y los motores de las articulaciones del robot. Hay que recalcar que la programación del lado de ROS se la puede realizar en C++, Python y Java. Para este caso se realiza un script de programación en Python para que una vez ejecutado desde la terminal se quede en modo de escucha hasta que reciba los datos enviados desde Unity.

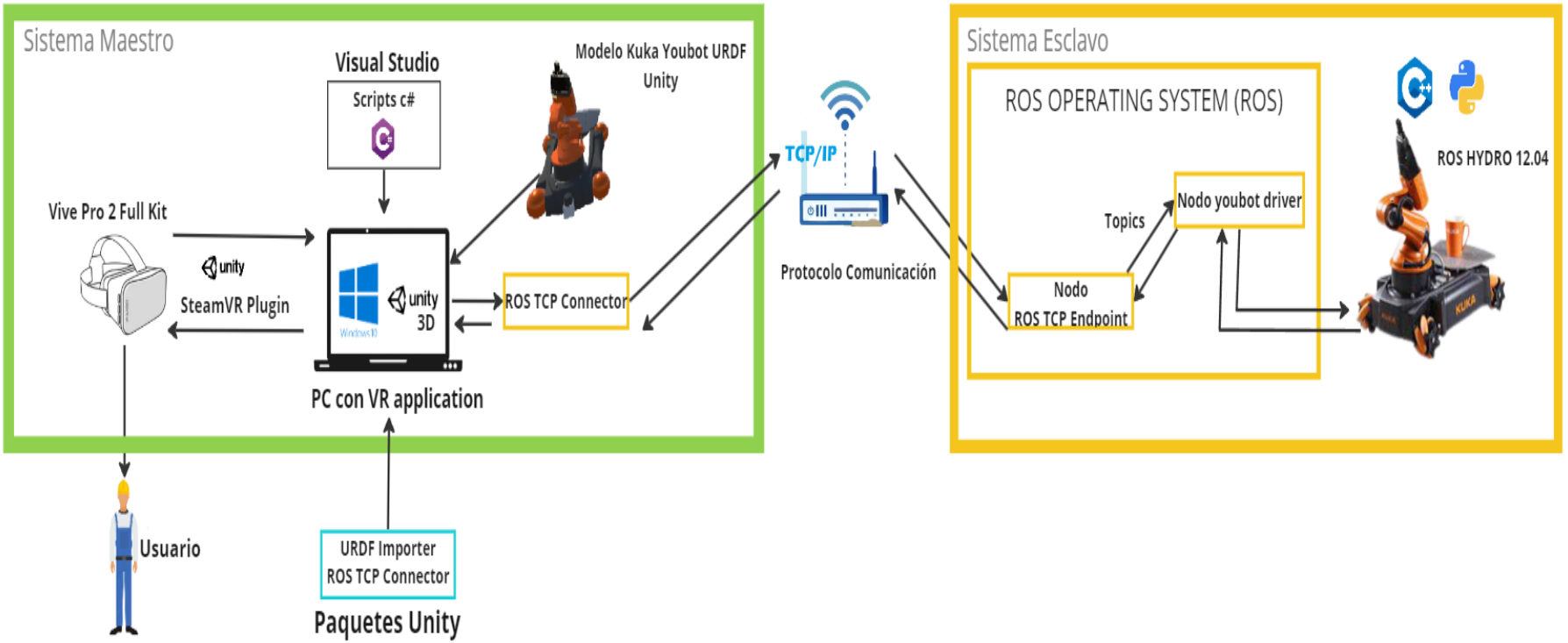

Figura 21. Esquema general del sistema de teleoperación con realidad virtual.
Elaborado por: El Investigador

3.2.3. Análisis de la cinemática de un robot manipulador

Se comienza el proyecto investigando la cinemática del robot, es decir el movimiento que realiza este en base a un sistema de referencia sin tener en cuenta las fuerzas que lo provocan, para de este modo conocer cómo se mueve y orienta el extremo final del brazo robótico en base a las posiciones de los eslabones. Este estudio permitirá más adelante entender cómo funciona el robot, conocer cada una de las partes que lo compone y los limites rotacionales inferiores y superiores de cada junta del brazo, para luego hacer funcionar el robot tanto en simulación como en la práctica real. El análisis de la cinemática del kuka youbot se presenta en el informe técnico que se detalla en el Anexo A.

3.2.4. Simulación en Gazebo con el modelo del robot

El control del robot Kuka Youbot se realiza en Gazebo, y con esto se aprende las partes que lo componen y además el funcionamiento del robot.

Para usar el simulador Gazebo se necesita del sistema operativo ROS, el cual permite crear nodos ROS que a su vez se comunican con otros nodos mediante tópicos, los cuales se suscriben y publican en cada una de las partes del robot Kuka Youbot. Una de las ventajas de ROS es que permite la fácil integración de Gazebo y mediante tópicos se controla el robot simulado dentro de Gazebo.

Integración del kuka youbot en Gazebo

Para la simulación se utiliza una máquina virtual de Linux versión 16.04 basado en ROS Kinetic. Uno de los motivos por el cual se utiliza esta versión es que el driver del robot trabaja sobre esta versión y los paquetes necesarios son solo para ROS Kinetic. El proceso de instalación de ROS y los paquetes necesarios del robot se indica en el Anexo B, una vez instalado solo es necesario configurar el entorno ROS y ejecutar gazebo con el modelo del Kuka Youbot.

Esquema de integración del kuka yobot con ROS y gazebo

En la Figura 22 se presenta el esquema en donde se define el modelo del Kuka Youbot dentro de la simulación en Gazebo y el espacio de trabajo ROS.

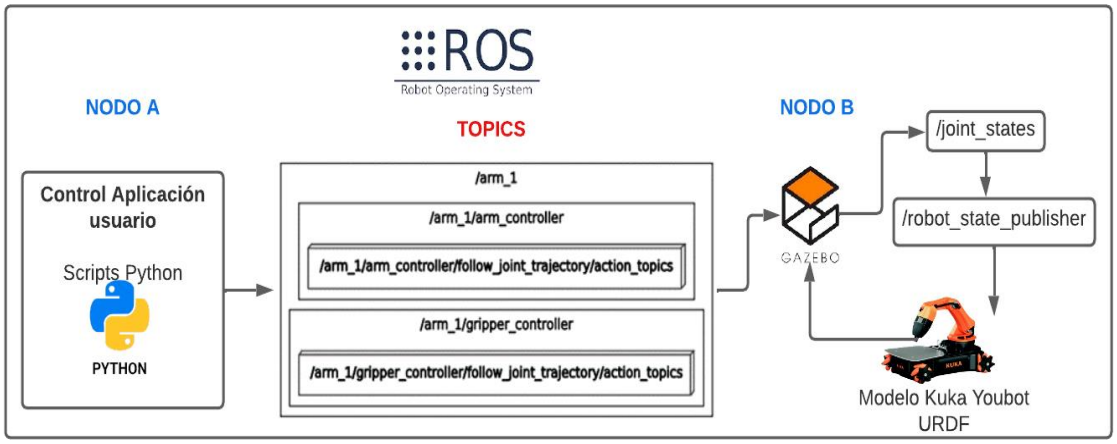

Figura 22. Esquema de gazebo con ROS.
Elaborado por: El Investigador

- **Control de Aplicación de usuario:** El control del robot se realiza mediante scripts de Python para cada una de las partes del robot. Dentro del script se importa la librería rospy la cual permite utilizar los parámetros correspondientes de los tópicos para el control del robot en la simulación. Dentro de esto también se inicializa y se define el nodo A.
- **Tópics:** Son conocidos como temas los cuales se encargan de transportar la información. Estos tópicos se crean una vez descargados los drivers necesarios del kuka youbot. Cuando se inicializa el driver también se inicia todos los tópicos, existe un tópico diferente para las articulaciones del robot, para la pinza (gripper) y otro para las ruedas o la plataforma móvil.
- **Gazebo:** El simulador de robots en este caso se presenta como un Nodo B, ya que en este se va a recibir las ordenes que fueron ejecutadas en los scripts de Python. Este nodo actúa como esclavo ya que solo contiene el modelo del robot y solo cuando se ejecute el script, se mueve el robot.
- **Modelo URDF:** El modelo URDF se trata de un archivo en formato XML el cual describe un robot, es decir se presentan los enlaces o eslabones del robot, así como la parte cinemática y dinámica del robot. Este archivo XML es importante ya que

en este se establece los limites rotacionales de cada articulación, así como la velocidad, la fuerza, la fricción con la que va a actuar el robot dentro de la simulación.

Creación y compilación del entorno de simulación Gazebo con el modelo del robot

Dentro de ROS se crea una carpeta en donde se van a almacenar los archivos necesarios de compilación del entorno de trabajo de ROS, así como los paquetes necesarios del robot. En la Figura 23 se presenta el diagrama de proceso para la creación y compilación del entorno de simulación.

El diagrama de proceso empieza con la creación de la carpeta del espacio de trabajo, luego se crea un fichero ejecutable llamado “src” el cual es el encargado de almacenar todos los archivos necesarios para la ejecución de gazebo, una vez realizado esto se inicializa el espacio de trabajo con “catkin_init_workspace” el cual es un comando de ROS. Para compilar el espacio de trabajo se ingresa “catkin_make” el cual es otro comando de ROS y si todo compila bien se crean los archivos devel y build los cuales son necesarios ya que contienen los archivos de configuración e inicialización del espacio de trabajo de ROS, en caso de que no compile hay que inicializar el espacio de trabajo correctamente. Una vez realizado esto se define la fuente de ROS para que de este modo gazebo funcione y finalmente se importan los paquetes del kuka youbot dentro del fichero ejecutable. Una vez realizado todo este proceso la simulación de gazebo debería funcionar correctamente.

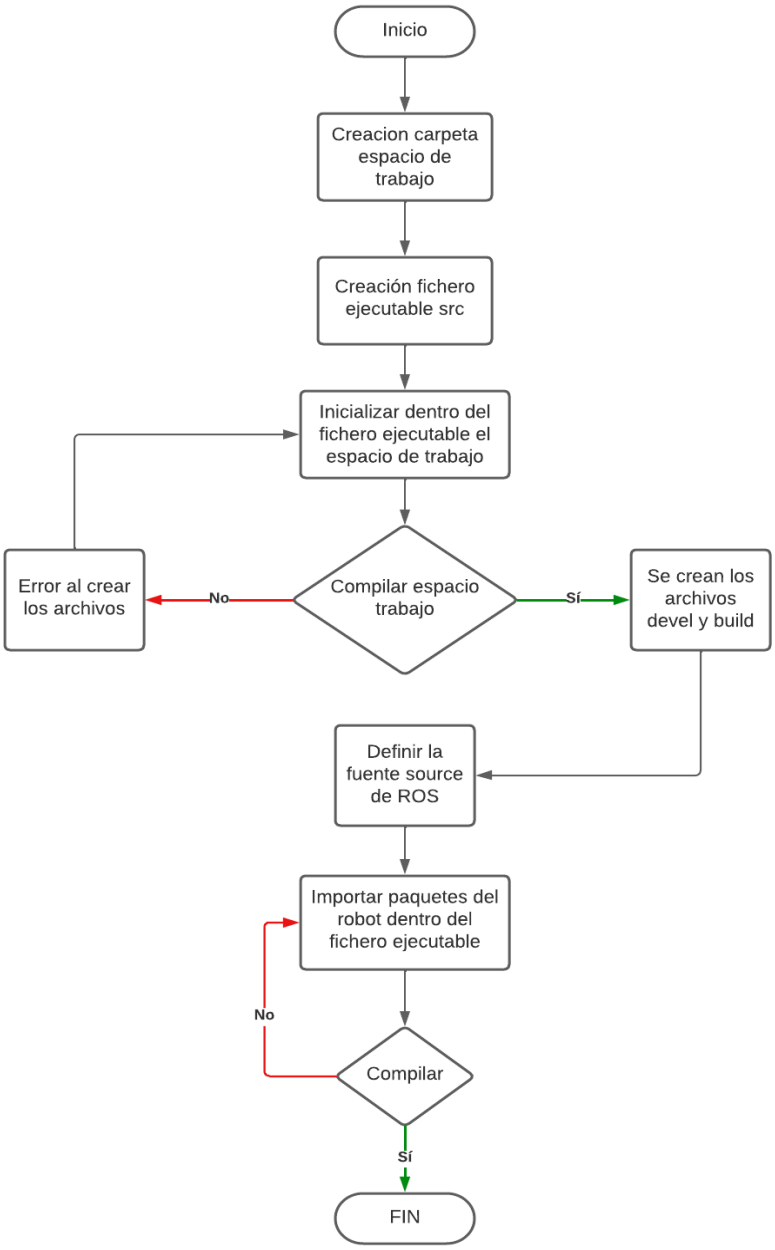

Figura 23. Flujograma del proceso de creación y compilación de ROS.
Elaborado por: El Investigador

3.2.5. Control del brazo manipulador y la plataforma móvil en Gazebo

Control del brazo Manipulador

En el control del brazo se utiliza la librería rospy, el cual permite a los desarrolladores en Python interactuar con los temas, servicios y todos los parámetros de ROS, además de las siguientes librerías:

- **from trajectory_msgs.msg import JointTrajectory:** Proporciona una estructura de mensajes para la representación de trayectorias de movimiento de las articulaciones del robot, además contiene información como el tiempo de inicio de la trayectoria, el tiempo de duración y una lista de puntos de trayectoria [39].
- **from trajectory_msgs.msg import JointTrajectoryPoint:**. Permite almacenar información de una posición especifica de las articulaciones en una trayectoria de movimiento. Contiene información sobre la posiciones, velocidades, aceleraciones y esfuerzos de las articulaciones en un punto de la trayectoria [39].
- **std_mesgs.msg:** Proporciona mensajes estándar para el intercambio de datos, es decir permite comunicar de manera estándar con otros nodos en el sistema ROS. Contiene mensajes como: “String”, “Bool”, “Int8”, “Float32”, “Header”, entre otros [39].

En la Figura 24 se observa el diagrama de clase UML para el código desarrollado en Python.

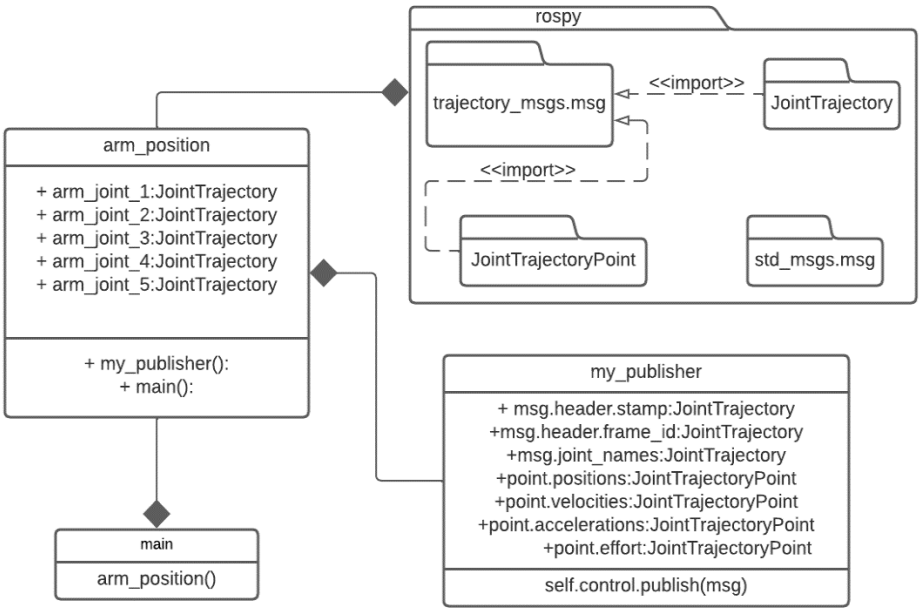

Figura 24. Diagrama de clase para el movimiento del brazo manipulador.
Elaborado por: El Investigador.

En la Figura 24 se muestra la clase principal (arm_position) para el movimiento del brazo manipulador, la cual contiene sus atributos para cada articulación y el tipo de mensaje es JointTrajectory, además estos atributos son visibles (+) para todos los elementos de la clase, a su vez este contiene dos métodos:

- **my_publisher():** Realiza el movimiento de las articulaciones del brazo a la posición indicada dentro de un ciclo while, el cual se encarga de publicar las posiciones de las articulaciones indefinidamente. Contiene un método self que se usa para hacer referencia al objeto que se está manipulando. Se define la inicialización del nodo y del topic en el cual se va a publicar la posición de las articulaciones. Dentro de este método se define la aceleración, velocidad y esfuerzos de las articulaciones.
- **main():** Llama a la clase principal para ejecutar las órdenes recibidas de posición por el método "my_publisher". Dentro de este método se define una función la cual mantiene la secuencia de comandos ejecutándose hasta que recibe una señal de apagado.

La clase principal depende de la librería rospy, el cual es necesario para el funcionamiento de los métodos de cada clase y también depende de la clase "main" la cual se encarga de llamar a la clase principal. Existe una relación de composición la cual indica que la clase principal predomina sobre las demás y estas no pueden funcionar de manera independiente sin la clase principal.

Dentro del método "my_publisher" para establecer los giros de cada una de las partes del brazo manipulador hay que tener en cuenta los limites rotacionales superiores e inferiores de cada una de las partes del brazo manipulador. Estos límites se presentan en la Tabla 11.

Tabla 11. Limites inferior y superior del brazo manipulador.

Eslabón	Nombre	Límite Inferior	Limite Superior
1	arm_joint_1	0.0100692	5.84014
2	arm_joint_2	0.0100692	2.61799
3	arm_joint_3	-5.02655	-0.015708
4	arm_joint_4	0.0221239	3.4292
5	arm_joint_5	0.110619	5.64159

Elaborado por: El Investigador

En la Figura 25 se tiene el flujograma de proceso para la ejecución del script desarrollado en conjunto con la ejecución de los nodos que son necesarios para el movimiento del brazo en el simulador Gazebo. Dentro del script se define un nodo publicador con el nombre "brazo" para que se encargue de enviar los datos hacia el nodo suscriptor, en este caso gazebo. Una vez definido esto se inicia la función "publisher" la cual se encarga de ejecutar un ciclo infinito while mientras rospy se está

ejecutando para posteriormente llamar a las articulaciones del brazo, se establece las posiciones adecuadas como se mostró anteriormente en la Tabla 11, la velocidad y aceleración. A continuación, se publica los datos en el nodo suscriptor mediante el tópico del brazo e ingresa a una condición en la cual si todo está bien se ejecuta el movimiento del brazo, caso contrario regresa a las posiciones de las articulaciones, ya que no deben estar dentro del rango correspondiente para el movimiento.

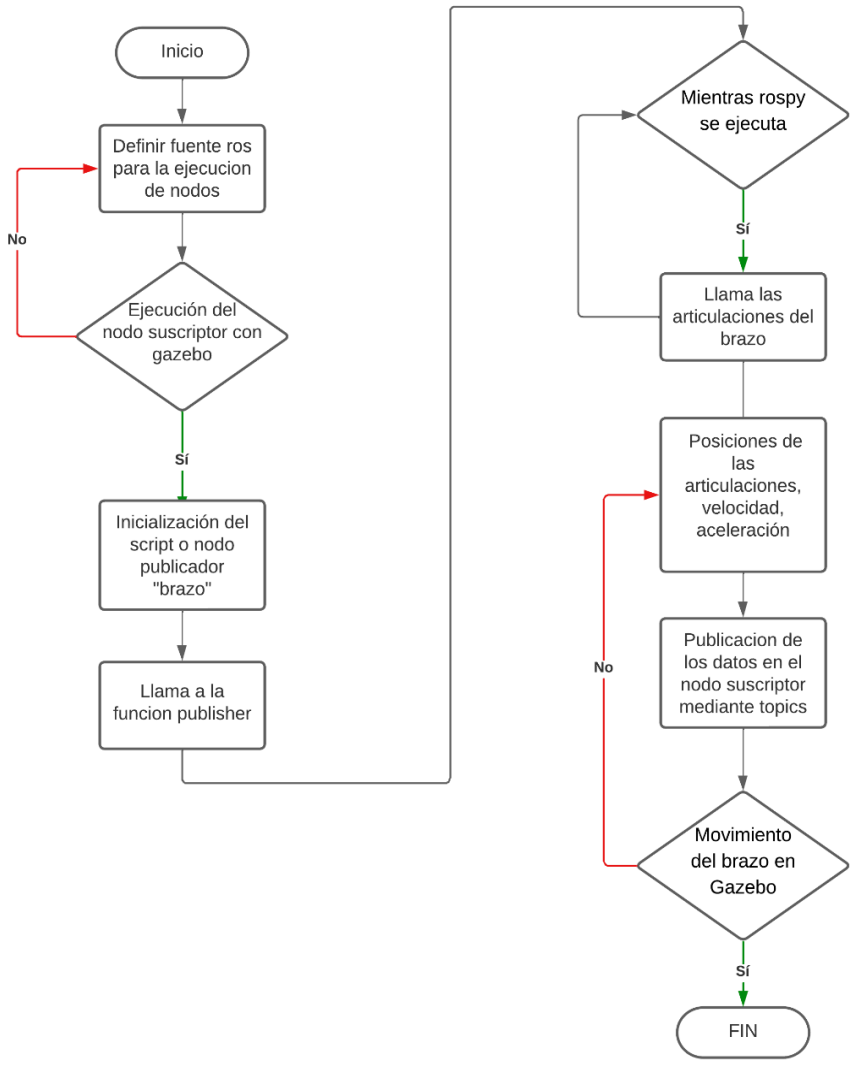

Figura 25. Flujograma de proceso para el movimiento del brazo en Gazebo.
Elaborado por: El Investigador

En la Figura 26 se muestra la posición inicial del robot al momento de ejecutar el nodo B de Gazebo o también conocido como suscriptor. En la Figura 27 una vez ejecutado el código desarrollado en Python conocido como Nodo A o nodo publicador, el robot

comienza a recibir las ordenes de posición del brazo manipulador mediante los tópicos que se han inicializado en la ejecución de gazebo (ver Figura 22).

Figura 26. Posición inicial del brazo manipulador.
Elaborado por: El Investigador

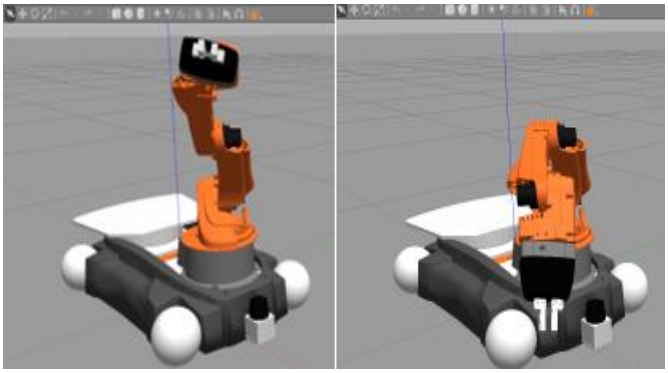

Figura 27. Posición final del brazo manipulador.
Elaborado por: El Investigador

Mediante la herramienta "rqt_graph" se ve una representación visual de los nodos y las conexiones que se tienen. Esta herramienta resulta útil para comprender la arquitectura de un sistema ROS ya que permite ver como se comunican los diferentes nodos mediante los temas de ROS. Una vez ejecutado esta herramienta se muestra un diagrama con círculos que representan los nodos y las flechas indican las conexiones entre los nodos. Cada nodo se representa por su nombre y los temas se muestran como flechas dirigidas desde los nodos publicadores hacia los nodos suscriptores. Esta herramienta se va actualizando automáticamente mientras se crean o se eliminan nodos y conexiones en ROS. En la Figura 28 se muestra los nodos iniciales de robot, en este caso se tiene a gazebo como publicador el cual se conecta con el estado de las articulaciones (/joint_states) y este a su vez se conecta con el nodo del robot (/robot_state_publisher). Los demás nodos sirven para el control de distintas partes del robot.

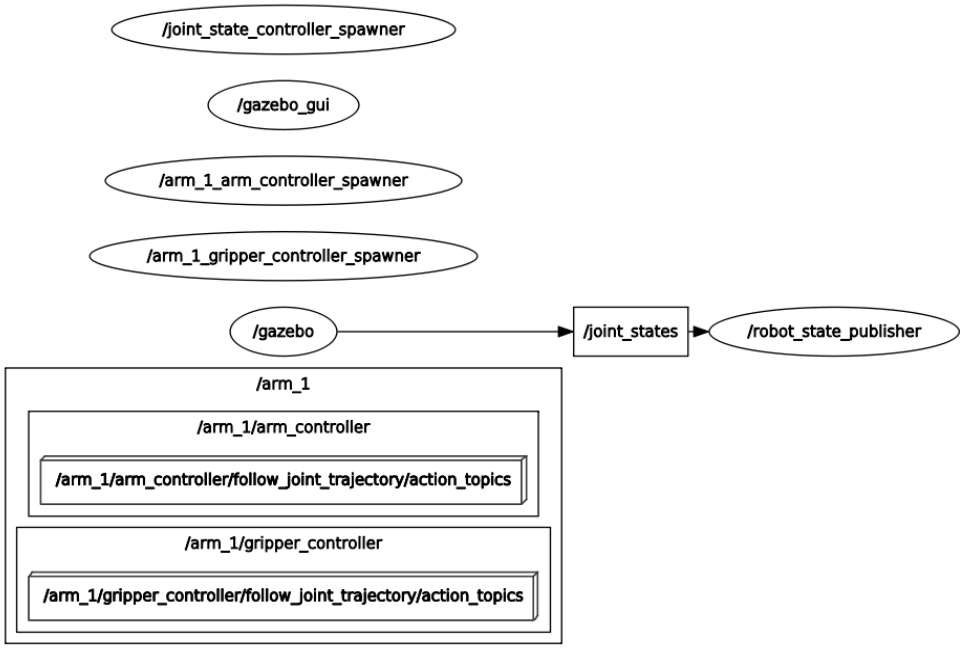

Figura 28. Nodos iniciales de Gazebo.
Elaborado por: El Investigador

Una vez que se ejecuta el script de control del brazo se publican los datos desde el nodo “brazo” creado en el código el cual se muestra de color rojo mediante el tópico o tema correspondiente, en este caso “/arm_1/arm_controller/command” que se muestra de color verde hacia el nodo en Gazebo. Después, este nodo de Gazebo se encarga de publicar los datos hacia el nodo del estado de las articulaciones (/joint_states) y finalmente llega hacia el robot para realizar el movimiento. Todo esto se observa en la Figura 29.

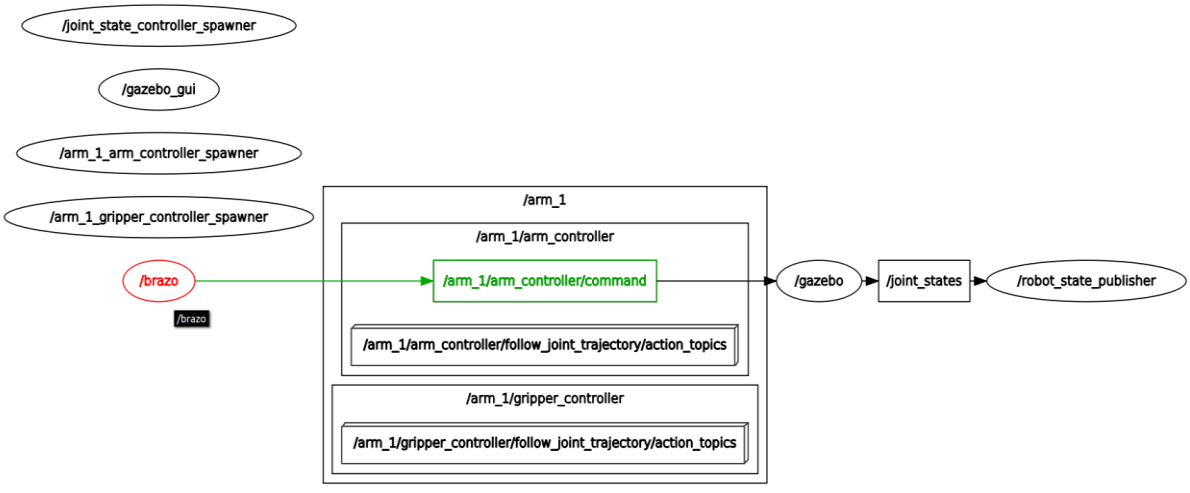

Figura 29. Diagrama de nodos del brazo manipulador en gazebo.
Elaborado por: El Investigador

Control de la plataforma Móvil

Para el funcionamiento y control de la plataforma robótica móvil también se toma en cuenta la librería rospy ya que esta permite integrar Python con ROS. Además de las siguientes librerías:

- **from geometry_msgs.msg import Twist:** Esta librería proporciona mensajes estándar para representar información geométrica en entornos robóticos, el módulo Twist permite expresar la velocidad de la plataforma móvil en términos de velocidad lineal y velocidad angular.

En la Figura 30 se tiene el diagrama de clase UML para el control de la plataforma robótica móvil desarrollado en Python en la cual se muestra la clase principal, en este caso se la denomina "ControlKuka", la cual contiene los atributos correspondientes para el movimiento de las ruedas de la plataforma móvil, hay que considerar que la plataforma es omnidireccional por lo que se mueve tanto de forma lineal como angular, es por este motivo que los distintos atributos tienen su movimiento tanto en (x,y,z) y son del tipo float. Dentro de los métodos se tiene los siguientes:

- **Constructor:** Este método permite inicializar la clase principal. Aquí se define el nodo que se va a crear y el tópico o tema en el cual se va a publicar las velocidades lineales y angulares de las ruedas, en este caso se trata del tema (/cmd_vel). Dentro de esto también se define un tiempo para cada mensaje que se envía y un ciclo while para enviar los datos de velocidad indefinidamente.
- **Shutdown():** Este método sirve para detener el movimiento de la plataforma móvil, es decir una vez que se ejecute el script la plataforma comienza a moverse hasta que el usuario decida por consola detener el movimiento ejecutado, para esto se crea una función la cual envía una señal al robot para que se detenga.
- **Main():** Llama a la clase principal para que el script comience a funcionar y cuando se detiene muestre por consola un mensaje de que se ha detenido el robot.

La clase principal tiene una relación de composición con la librería rospy, es decir esta clase no funciona independientemente sin la librería que se encarga de comunicar ROS con la aplicación desarrollada por el usuario. Dentro de esta librería está contenido otro paquete de la clase "geometry_msg.msg", explicado anteriormente el cual es el encargado de los movimientos lineales y angulares.

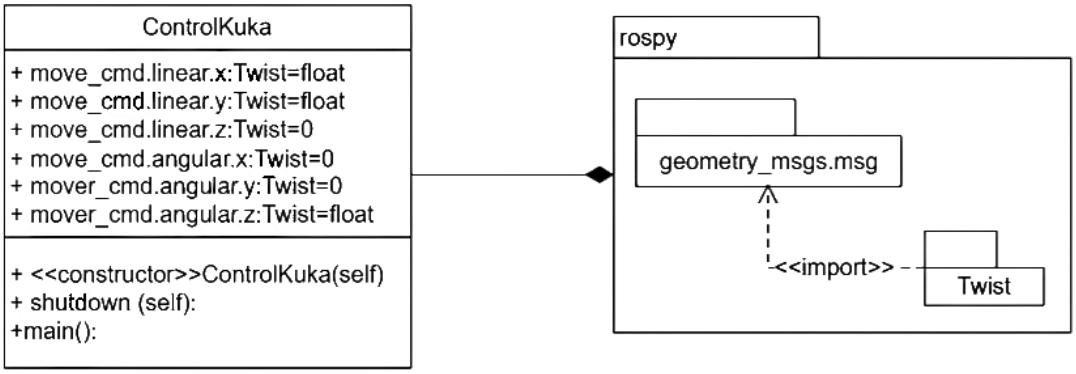

Figura 30. Diagrama de clase UML para el control de la plataforma móvil.
Elaborado por: El Investigador

En la Figura 31 se observa el flujograma de procesos para el script desarrollado del movimiento de la plataforma, en el cual primero se inicia el nodo publicador denominado (ControlKuka). Se asigna un tiempo para cada uno de los datos que se envían hacia el robot y se definen valores para el movimiento lineal o angular. Posteriormente, se establece una condición para determinar si el valor ingresado es lineal, asignándolo a la variable "x". Si el valor es positivo o mayor que cero, se ejecuta el movimiento hacia adelante, caso contrario determina si el valor ingresado en “x” es negativo o menor que cero realiza el movimiento de la plataforma hacia atrás. En caso de que el valor en “x” sea igual a cero la plataforma no se mueve. Una vez determinado estas condiciones publica los datos en el tema (/cmd_vel) para el movimiento de la plataforma. Por otro lado, si el valor ingresado es lineal se asigna a la variable “y”. Si el valor es positivo o mayor que cero realiza el movimiento de la plataforma hacia la izquierda, caso contrario si el valor es menor a cero o negativo la plataforma se mueve hacia la derecha, en caso de que el valor lineal en “y” sea igual a cero la plataforma tampoco se mueve. La velocidad lineal en “z” no se coloca ya que el robot no realiza ese movimiento. Para la parte de la velocidad angular el único movimiento capaz de realizar la plataforma es en “z” por lo tanto se evalúa si el valor ingresado en “z” es mayor a cero la plataforma realiza una rotación a la derecha caso contrario rota a la izquierda. Para la ejecución de este script se debe primero ejecutar el nodo de gazebo en el cual se encuentra el modelo del robot para que este se suscriba a los datos que está enviando el nodo publicador del script realizado.

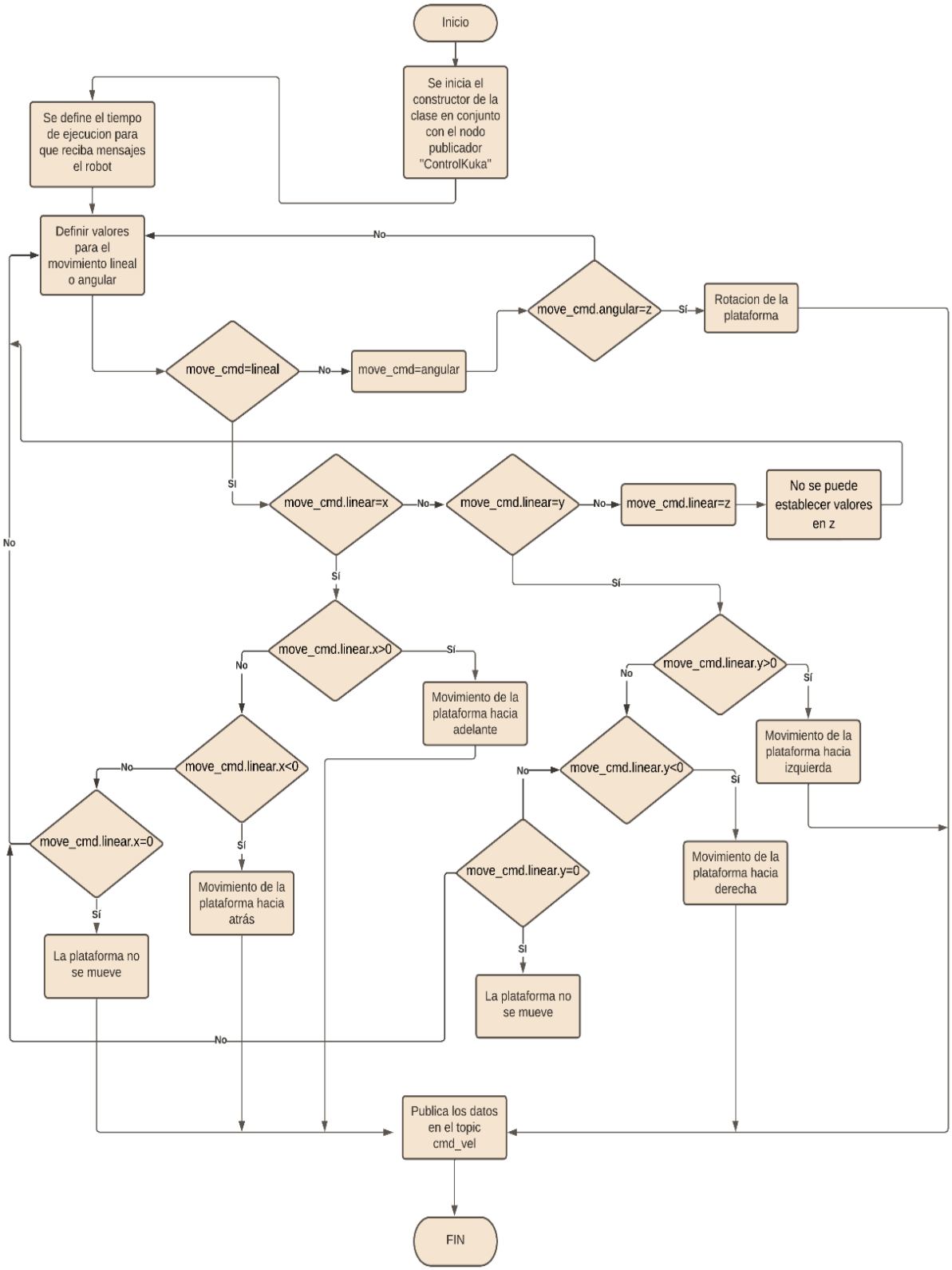

Figura 31. Flujograma de procesos para el movimiento de la plataforma móvil.
Elaborado por: El Investigador.

Movimiento lineal hacia adelante y hacia atrás de la plataforma móvil

Para el movimiento de la plataforma hacia adelante y hacia atrás se establece en el script un movimiento lineal utilizando la librería "geometry_msgs.msg" con el módulo Twist el cual tiene 3 componentes (x,y,z) los cuales representan las velocidades lineales como se explicó en el flujograma de la Figura 31. En este caso para mover la plataforma para adelante se asigna un valor positivo en la componente en "x" o en caso de que se quiera mover para atrás se asigna en la componente "x" un valor negativo. La posición inicial con los respectivos ejes de la plataforma móvil se ve en la Figura 32.

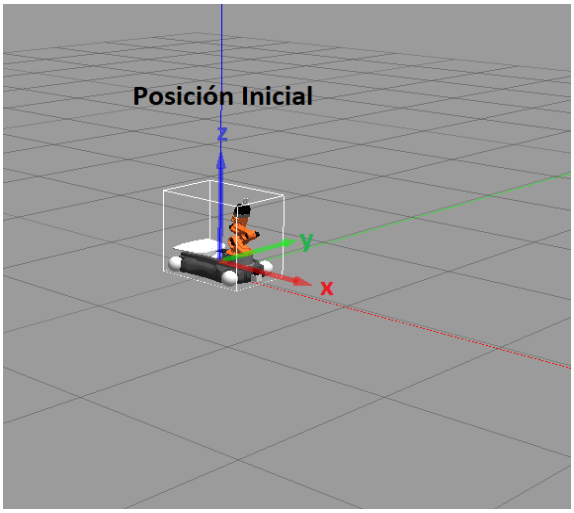

Figura 32. Posición Inicial de la plataforma móvil.
Elaborado por: El Investigador

Una vez ejecutado el script se observa en la Figura 33 el movimiento hacia adelante que realiza la plataforma móvil.

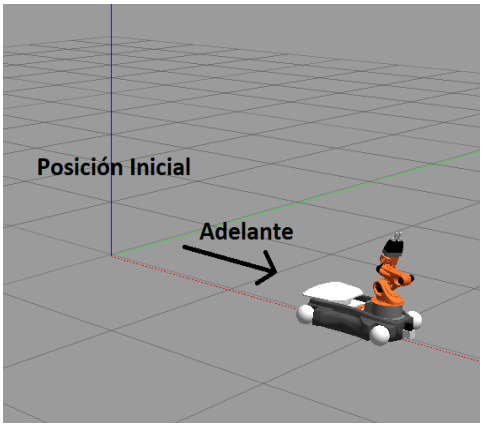

Figura 33. Movimiento hacia adelante de la plataforma móvil.
Elaborado por: El Investigador

En la Figura 34 se observa el movimiento hacia atrás de la plataforma móvil.

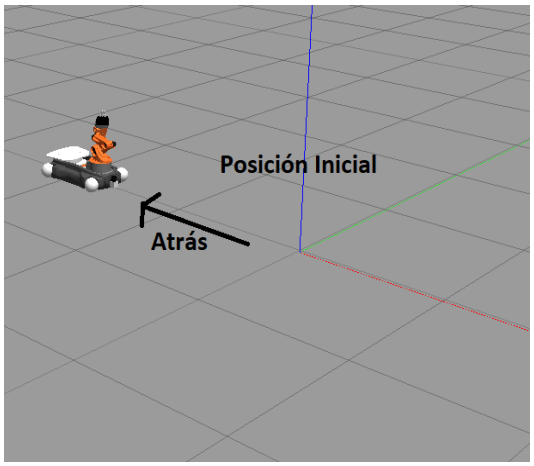

Figura 34. Movimiento hacia atrás de la plataforma móvil.
Elaborado por: El Investigador

Movimiento lineal hacia la izquierda y derecha de la plataforma Móvil

Para el movimiento hacia a la izquierda o derecha lo único que se tiene que establecer es una velocidad lineal en la componente “y”. Para el movimiento hacia la derecha se establece un valor negativo y para la izquierda un valor positivo. En la Figura 35 se observa el movimiento hacia la izquierda y en la Figura 36 se observa el movimiento hacia la derecha.

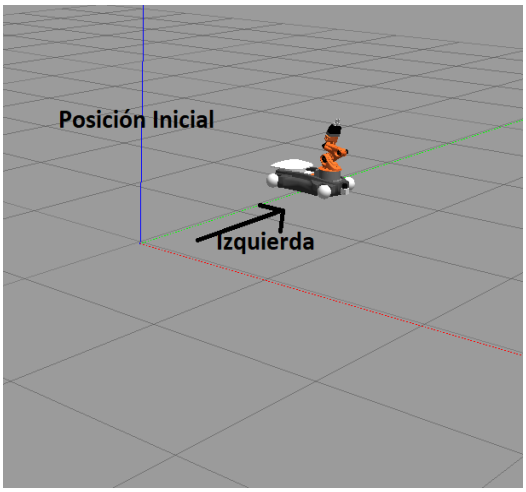

Figura 35. Movimiento hacia la izquierda de la plataforma móvil.
Elaborado por: El Investigador

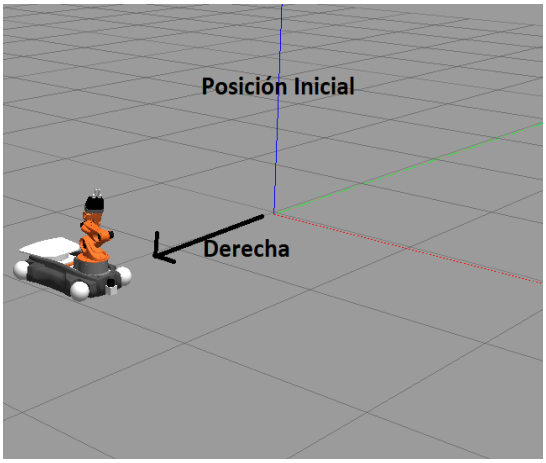

Figura 36. Movimiento hacia la derecha de la plataforma móvil.
Elaborado por: El Investigador

Movimiento rotacional de la plataforma móvil

Para el movimiento de rotación de igual manera se hace uso de la librería Twist el cual así mismo tiene 3 componentes (x,y,z). Para este caso el movimiento rotacional en la componente en (x,y) no se realiza ya que la plataforma no puede realizar ese movimiento, por lo tanto solo se establece un valor en la componente en "z". La rotación depende de igual forma del valor positivo o negativo que se le asigne. En la Figura 37 se observa la plataforma con los distintos ejes de rotación.

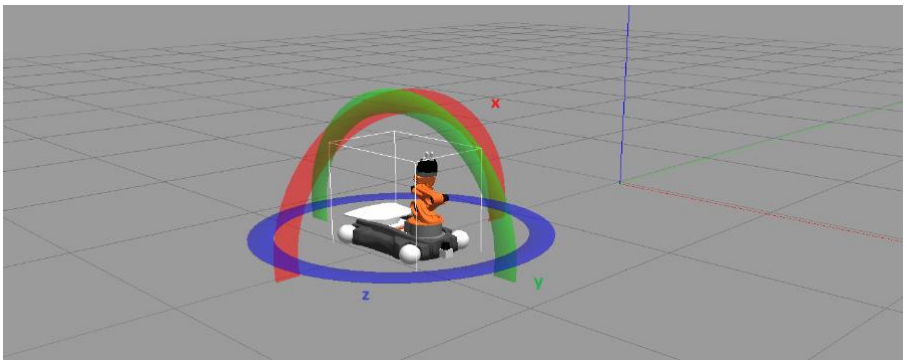

Figura 37. Ejes de rotación de la plataforma móvil.
Elaborado por: El Investigador

En la Figura 38 se observa el movimiento de rotación de la plataforma móvil, teniendo en cuenta que se establece una velocidad de rotación en el eje z.

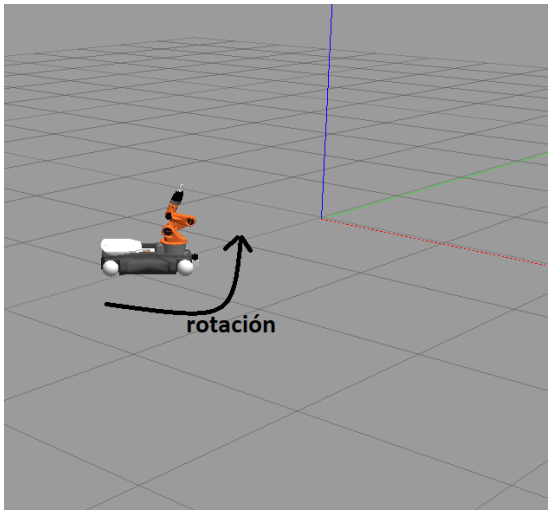

Figura 38. Movimiento de rotación de la plataforma móvil.
Elaborado por: El Investigador

De la misma manera que en el movimiento del brazo, se usa la herramienta rqt_graph para observar los nodos y los tópicos o temas que se están ejecutando al momento de realizar el control de la plataforma móvil. En la Figura 39 se observa el nodo creado en el script denominado “ControlKuka” el cual se muestra de color rojo y este se conecta al tema “cmd_vel” que se muestra de color verde, luego este nodo se conecta con gazebo para finalmente enviar los datos del estado de las articulaciones (/joint_states) hacia el robot simulado en gazebo.

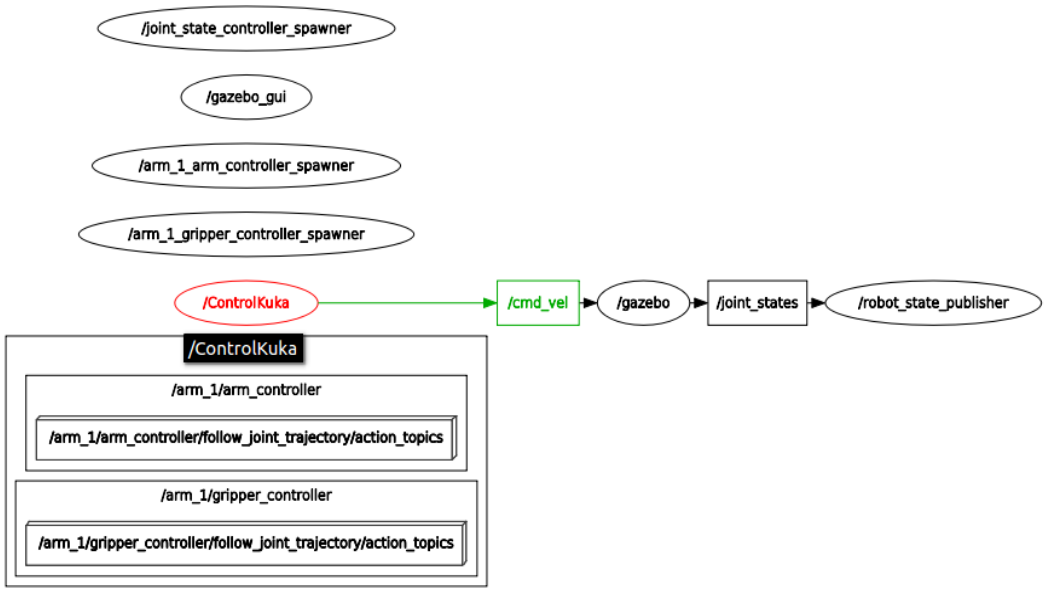

Figura 39. Diagrama de nodos del control de la plataforma móvil en gazebo.
Elaborado por: El Investigador

3.2.6. Diseño de la realidad virtual en Unity

La realidad virtual generalmente requiere que la persona lleve puesto un casco o gafas de realidad virtual que permita sumergirse en una simulación de 360 grados, haciendo notar que el usuario se sienta que está colocado o inmerso en un entorno virtual. Para el diseño de la realidad virtual se tienen dos niveles las cuales se presentan a continuación:

Nivel de desarrollo

Para el nivel de desarrollo se tiene dos niveles, el primero consta del desarrollo en Visual Studio el cual contiene el código o las instrucciones en lenguaje C# para el control y movimiento del modelo del robot Kuka Youbot, además de los paquetes y librerías que son necesarios para la comunicación con el robot real bajo el protocolo ROS TCP, también se agregan las librerías que son necesarias para la interfaz de usuario, control de objetos y demás componentes de la realidad virtual.

El segundo nivel consta de la programación en Python en el robot real para recibir los datos que son enviados desde Unity para el control y movimiento. En este nivel se reciben los datos mediante un tópico que se publica desde Unity para él envió de datos y en Python se define los suscriptores para que se asocien a los datos recibidos y publique en los tópicos correspondientes del robot para su movimiento.

Nivel de interfaz

Para el desarrollo de la interfaz de realidad virtual se tomó en cuenta las directrices y normas ISO aplicables a la usabilidad de la realidad virtual descritas en la Tabla 9 y 10 para que de esta manera el usuario o la persona que este inmerso en la realidad virtual lo use fácilmente sin tener ningún inconveniente. Todas las normas y directrices se enfocan en 4 cosas principales: la facilidad de uso, la facilidad de aprendizaje, que sea atractivo para el usuario y el grado de satisfacción que obtuvieron de acuerdo con los requerimientos para el cual fue desarrollado la aplicación o software.

Considerando estas directrices y normas se diseña la interfaz de realidad virtual con las siguientes características:

- **Facilidad de uso:** Para tener una facilidad de uso en la realidad virtual se coloca paneles con texto para que el usuario sepa cuales son las funciones de cada botón del control y de esta manera manipule el robot de la mejor manera posible. Además, se colocó una función de traslación para que el usuario se desplace fácilmente por todo el escenario sin tener que moverse físicamente, de esta manera se facilita la observación del movimiento del robot si el usuario se encuentra alejado.
- **Facilidad de aprendizaje:** Para esto se colocó información y datos relevantes de posición de las partes del robot para que la persona controle de manera correcta el robot. Además, se tiene una tarea pick and place (recoger un objeto) en el cual se observa el movimiento del robot para alcanzar un objetivo.
- **Atractivo para el usuario:** Para lograr esto se creó una función que permita ver al usuario que parte del robot va a controlar, es decir el usuario va a observar en el robot un color sombreado en la parte o articulación elegida que va a controlar. Además de que se colocó objetos como sillas, mesas, cuadros para que sea atractivo la realidad virtual.
- **Grado de satisfacción:** Esto tiene que ver con el usuario, es decir que grado de satisfacción obtuvo al interactuar con la realidad virtual, si al momento de utilizar las gafas no tuvo incomodidad física, por ejemplo: que no se produzcan velocidades excesivas en la perspectiva de la cámara que causen malestar en los usuarios, fatiga ocular o mareo.

Diagrama de componentes

El diagrama de componentes para el entorno virtual muestra todos los objetos que se utilizan dentro de la escena para el desarrollo de la interfaz de realidad virtual. El diagrama de componentes es importante ya que representa la estructura y la relación entre los componentes (gameobjects de Unity). Estos componentes encapsulan la funcionalidad del sistema. Dentro de esto se tiene relaciones de dependencia las cuales indican que un componente depende de otro para funcionar correctamente.

La Figura 40 muestra el diagrama de componentes que se usan en la realidad virtual, para esto se tienen conectores de interfaz los cuales quieren decir que cada componente se conecta a la interfaz de la escena que va a observar la persona al colocarse las gafas de realidad virtual.

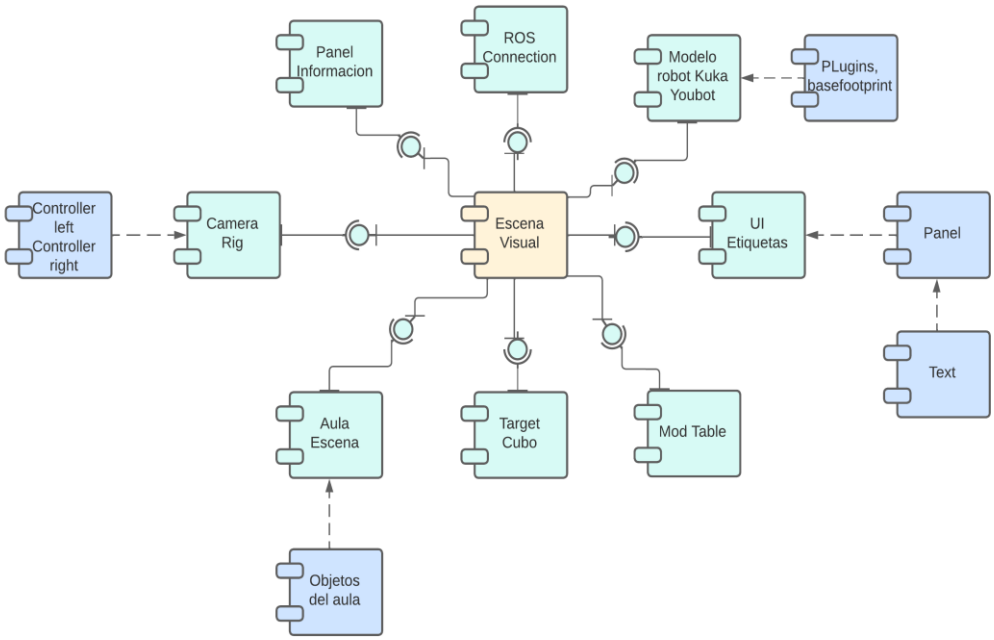

Figura 40. Diagrama de componentes de la realidad virtual.
Elaborado por: El Investigador

- **Camera Rig:** Es el componente de la cámara principal el cual contiene los controles izquierdo y derecho de las gafas HTC Vive Pro 2 y tienen una relación de dependencia con la cámara.
- **Aula Escena:** Es el componente el cual contiene todos los objetos necesarios que componen la realidad virtual como sillas, mesas, laptos, etc.
- **Target Cubo:** Es el componente que representa el cubo, con el cual la persona interactúa en conjunto con el robot para la tarea pick and place.
- **Mod Table:** Representa la tabla sobre la cual está el cubo.
- **UI Etiquetas:** Representa todas las etiquetas que se utilizan en la realidad virtual para que la persona tenga facilidad de uso y se guie dentro de la realidad virtual.
- **Modelo Robot Kuka Youbot:** Es el componente que contiene el modelo URDF del robot exportado a Unity y tiene una relación de dependencia con varios plugins que son necesarios para el movimiento de cada articulación.

- **ROS Connection:** Es el componente que se encarga de la comunicación con el robot real, contiene datos como la dirección IP y el puerto con el cual se comunica.
- **Panel Información:** Es el componente que muestra la información necesaria del robot dentro de la realidad virtual.

En la Figura 41 se muestra la escena o interfaz desarrollada de realidad virtual en base a las normas y directrices aplicables a la usabilidad explicado anteriormente en el nivel de interfaz, además también se muestra todos los objetos que componen la realidad virtual en base al diagrama de componentes de la Figura 40.

Figura 41. Escena de realidad virtual.
Elaborado por: El Investigador

En la Figura 42 se muestra un panel con una breve información de los controles en donde el usuario debe presionar el botón de "menú" para tener una breve retrospectiva del funcionamiento de los botones de cada control y mediante esto controle el robot y además sea capaz de interactuar dentro de la realidad virtual.

Figura 42. Panel de información dentro de la realidad virtual.
Elaborado por: El Investigador

Diagrama de dependencia de ensamblaje

La Figura 43 muestra el diagrama de dependencias de ensamblaje el cual sirve para explicar las librerías y la dependencia que tienen entre ellos las cuales se utilizan en el lenguaje C#. Un ensamblaje prácticamente contiene uno o más espacios de nombres (namespaces) y tipos (types) como clases, interfaces, estructuras, etc. Se representa los ensamblajes como cajas y las dependencias entre ellos como flechas que indican la dirección de la dependencia. Para el control del robot y los demás componentes de la escena de realidad virtual se utilizan de manera general en todos los scripts desarrollados las siguientes librerías que se detallan a continuación:

- **UnityEngine:** Este es el ensamblaje principal el cual tiene dependencias con "System.Collections y System.Collections.Generic" para el manejo de colecciones y estructuras de datos. También tiene dependencia con "Unity.Robotics.UrdfImporter" para importar modelos URDF en Unity.
- **Unity.Robotics.UrdfImporter:** Esta depende de "Unity.Robotics.UrdfImporter.Control" para el control de los modelos URDF importados, en este caso el modelo del robot.
- **UnityEngine.UI:** Esta librería sirve para la interfaz de usuario en Unity.
- **Valve.VR:** Es utilizado para la integración de la realidad virtual en Unity
- **TMPro:** Es una librería usada para el manejo de texto en Unity.
- **System:** Esta librería se incluye automáticamente al momento de crear un nuevo script.
- **Unity.Robotics.ROSTCPConnector:** Es la librería encargada de establecer comunicación TCP/IP con el sistema operativo ROS en Unity.
- **RosMessageTypes.UnityRoboticsDemo:** Es una librería que contiene los tipos de mensajes específicos de ROS.

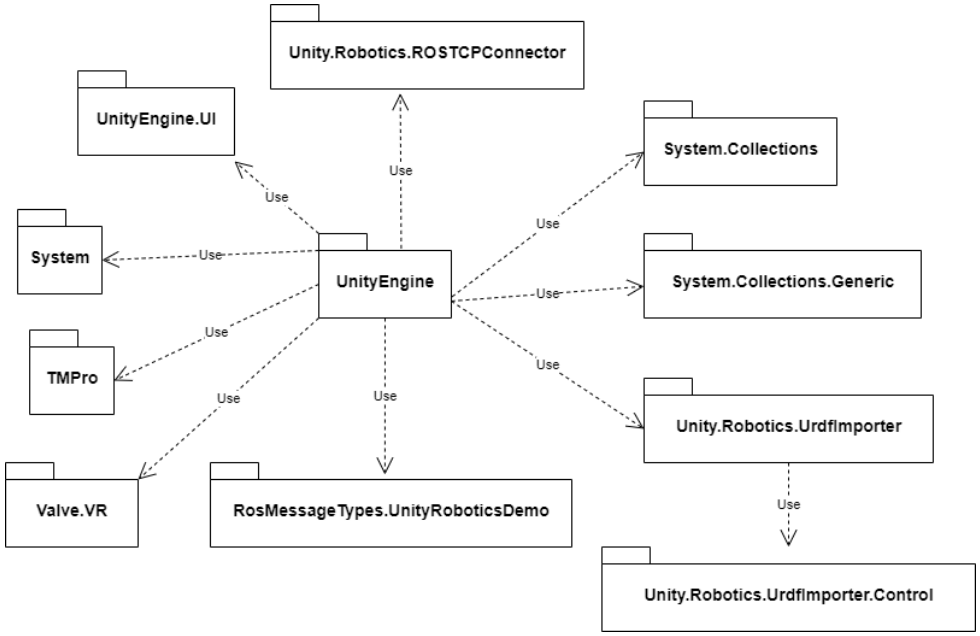

Figura 43. Diagrama de dependencia de ensamblaje de librerías en Unity.
Elaborado por: El Investigador

Diagramas de clase

Se realizan para representar los atributos y métodos que contiene el script de programación para el control y movimiento del robot en la realidad virtual. Estos diagramas de clase se lo realizan en base al código de cada control de las gafas de realidad virtual, es decir para el control derecho e izquierdo y un script global que se encarga de almacenar datos.

- **Movimiento del robot y posición inicial**

El movimiento y control de cada articulación del robot se lo realiza con los botones del control derecho, de igual manera el control de la plataforma móvil se lo realiza con el giroscopio de este control teniendo en cuenta la posición del brazo para el movimiento, además cuando ya se ha realizado los movimientos de cada articulación se puede regresar a su posición inicial con el botón de atrás (trigger) del mismo control. En la Figura 44 se muestra el diagrama de clase del movimiento del robot el cual contiene una sola clase principal con varios métodos los cuales son necesarios para la ejecución del programa, además la clase principal tiene una relación de composición con los otros métodos ya que esta clase no funciona de manera independiente y los

demás métodos no pueden iniciarse sin esta clase, cada uno de los elementos se detallan en la Tabla 12.

Tabla 12. Descripción de la clase del movimiento del robot.

Controlar	Esta es la clase principal en donde se definen e inicializan todas las variables que se van a utilizar en el programa, además se debe establecer el tipo de variable. Tiene distintos atributos los cuales se usan a lo largo del programa por ejemplo dentro de esta clase se define el objeto del robot que se va a controlar, las variables de las articulaciones en las cuales se van a ir extrayendo las partes del modelo URDF del robot, los targetPosition van almacenando los valores de cambio de posición tanto de las articulaciones como de las ruedas de la plataforma móvil, se define el selectedJointindex el cual contiene el nombre de cada articulación. Además, se define las acciones que se van a asignar a los botones del control, para ello se utiliza la librería de SteamVR la cual indica el tipo de acción que debe tener cada botón. Finalmente se define el topicname el cual es un string que define el nombre del tema y mediante esto se envían los datos desde Unity a ROS. Esta clase tiene varios métodos los cuales son necesarios para la ejecución del programa.
Awake	Sirve para obtener el componente "SteamVR_Behaviour_Pose" del objeto en el que se encuentra el script. El método "GetComponent<T>()" se usa para obtener una referencia al componente especificado por su tipo. Esta función se ejecuta cuando el objeto que contiene el script se inicia en la realidad virtual.
Inicio	Dentro de este se define la variable "articulationChain=robot.GetComponentsInChildren<ArticulationBody>()" el cual obtiene todos los componentes que se encuentran en los hijos del objeto robot y asigna a la variable "articulationChain". También se define la instancia de la conexión de ros utilizando la clase "ROSConnection" y se registra un publicador en la instancia de conexión de ROS.
SelectJoint	Dentro de este se almacena los valores actuales de "selectedJoin" en la variable "previousIndex". De igual manera sucede con el valor de "selectedJoin" el cual se asigna a la variable "selectJointIndex". En resumen, esta función selecciona diferentes componentes del objeto robot y asigna los enlaces a las variables específicas, así como los nombres de los enlaces.
Actualizar	Se definen los atributos para el control de la posición, así como también se obtienen las variables globales de las articulaciones para controlar el robot. Dentro de esto se establece el panel correspondiente que indica una etiqueta en el control y también se obtienen los datos de posición y rotación para el control de la plataforma móvil con el giroscopio. De forma general aquí se realiza todas las condiciones y funciones para mover las articulaciones, además se establece los limites rotacionales de cada una de las articulaciones y también se realiza cálculos para el uso del giroscopio.

Posición Original	Se realiza varias acciones para devolver los componentes "ArticulationBody", es decir los componentes del objeto robot a sus posiciones originales. Se define una variable "controlPosition" la cual se usa para controlar el estado de movimiento de los componentes.
selectPoint	Aquí se evalúa las variables de las articulaciones para llamar a la función sendROS el cual es el encargado de mandar los datos a ROS. Se utiliza la variable targetRound para redondear el targetPosition al entero más cercano y enviar ese valor.
sendROS	Aquí se realizan varias condiciones con las variables "selectedJointName" y "targetPosition" para enviar los datos mediante un tópico utilizando la conexión ROS.

Elaborado por: El Investigador

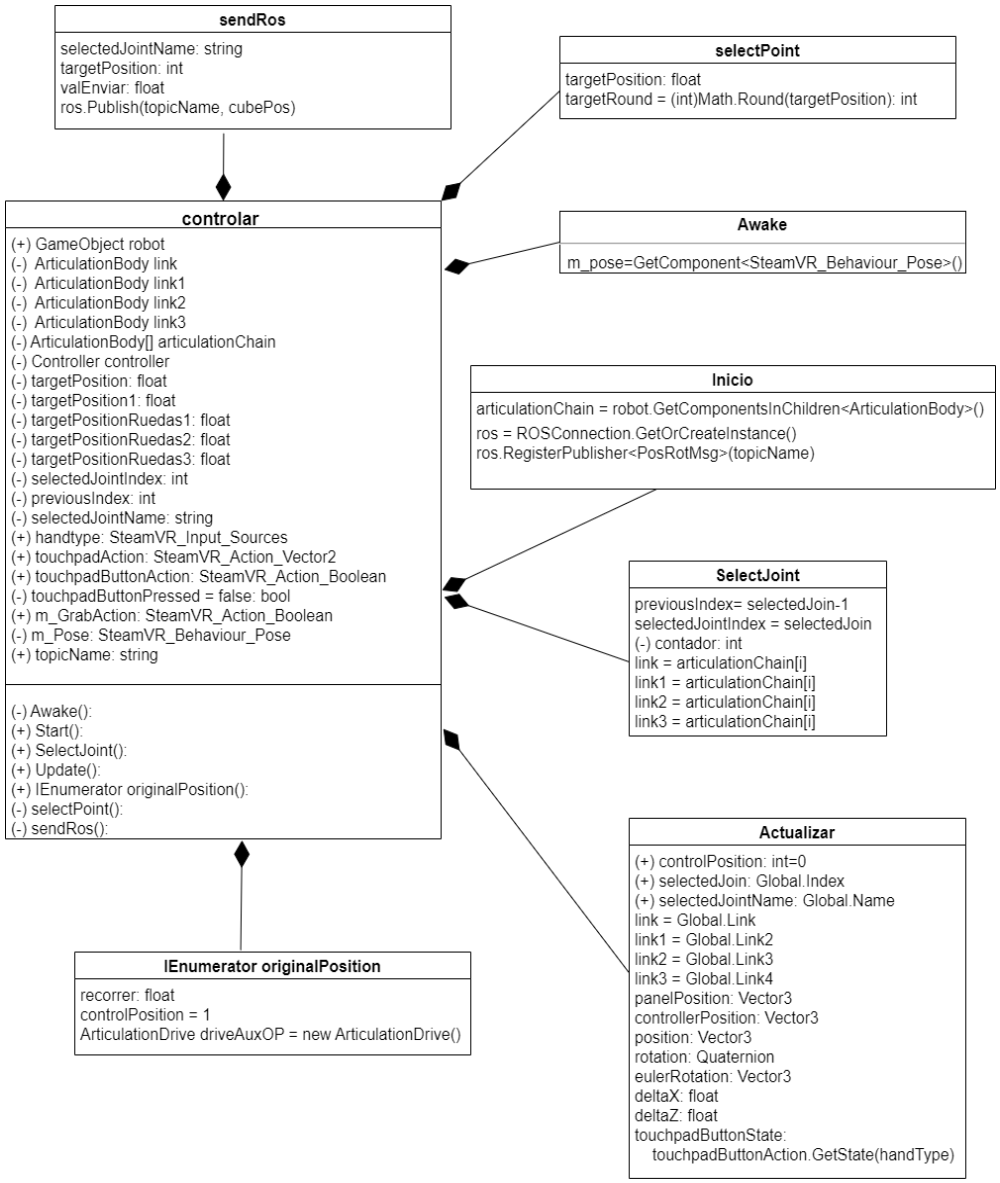

Figura 44. Diagrama de clase del movimiento del robot.
Elaborado por: El Investigador

- **Selección de las articulaciones del robot y traslación**

La selección de cada una de las articulaciones del robot, así como el movimiento de traslación para que la persona pueda movilizarse, acercarse o alejarse del robot dentro de la realidad virtual se lo hace en el control izquierdo. A continuación, se presenta el diagrama de clase en la Figura 45 y los detalles de cada parte en la Tabla 13.

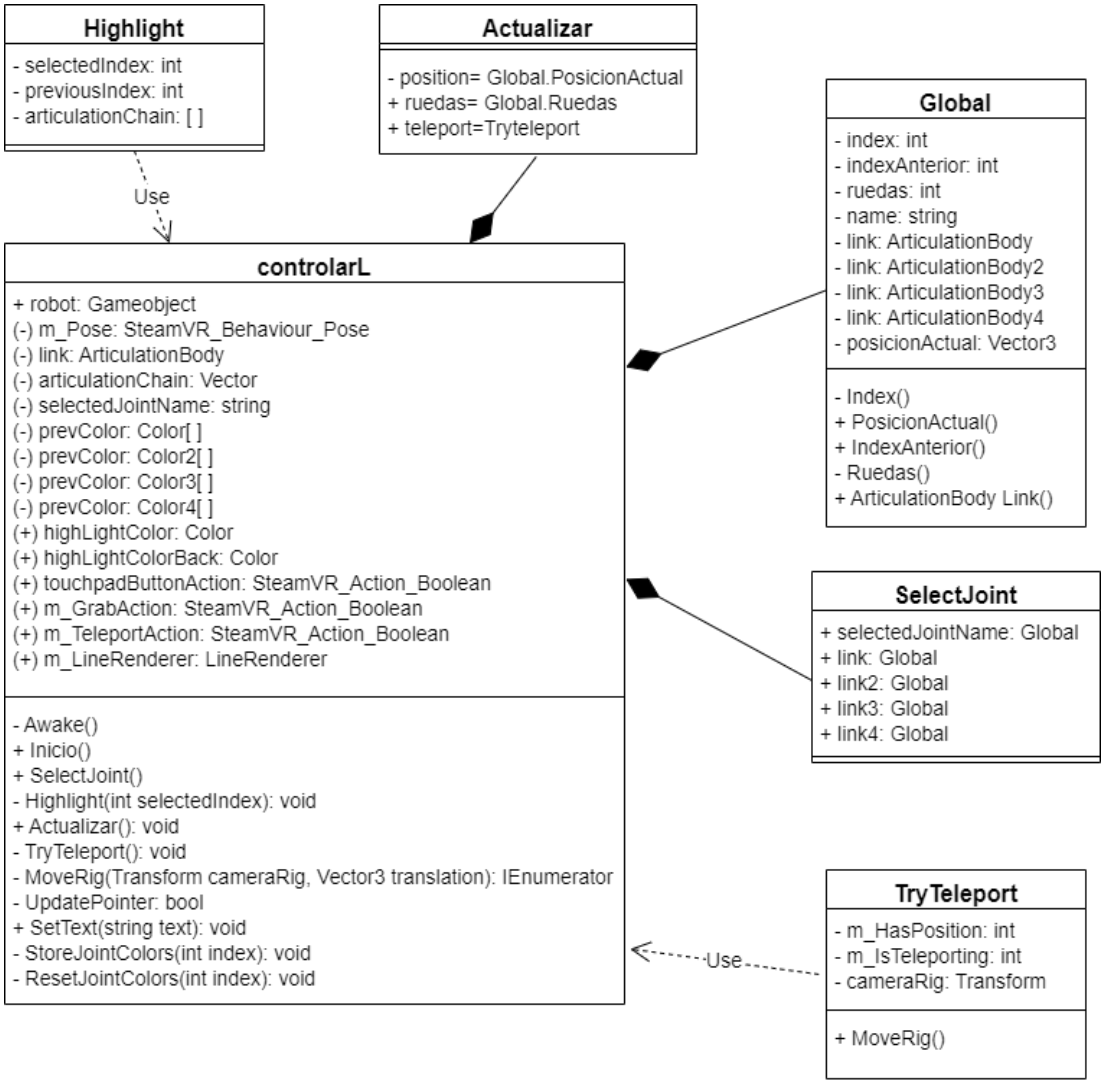

Figura 45. Diagrama de clase de la selección de las articulaciones y traslación.
Elaborado por: El Investigador

Tabla 13. Descripción de la clase selección de las articulaciones y traslación.

ControlarL	Esta es la clase principal la cual predomina sobre las demás, es decir tiene una relación de composición con las otras partes en donde si se elimina esta las demás también. Dentro de esta clase se tiene varios atributos entre las cuales se tienen: el gameobject del robot, las acciones que se asignan al control izquierdo, también se tiene los vectores los cuales se usan para seleccionar las articulaciones del robot en base a un color, es decir mientras la persona selecciona una articulación en la realidad virtual se puede observar con un color sombreado la parte seleccionada. Dentro de los métodos se tiene la teletransportación o traslación dentro de la escena, el movimiento de la cámara, entre otros.
Highlight	Este tiene una relación de dependencia, es decir un vínculo menos fuerte que la relación de composición ya que si se elimina esta no afecta a la clase principal. Dentro de esto se tiene los atributos de las articulaciones y del index que se va seleccionando. En resumen, esto sirve para seleccionar un elemento con un color en este caso la articulación seleccionada del robot.
Actualizar	Aquí se selecciona las articulaciones y se almacenan en una variable global, lo mismo sucede con las ruedas. Esto sirve para enviar estos datos al control derecho y realizar el movimiento y control de las articulaciones del robot. Además, se utiliza una variable la cual sirve para utilizar el efecto de traslación dentro de la escena.
Global	Es una clase estática la cual contiene varias propiedades estáticas y campos utilizados para almacenar valores globales dentro del contexto de la aplicación del control del robot. Esta se utiliza para almacenar los valores del control izquierdo y utilizarlas en el control derecho. Su funcionamiento se base en acceder a valores en diferentes partes del código o en distintas clases.
SelectJoint	Permite acceder a los nombres y valores de los enlaces para almacenar estos valores en una variable y posteriormente guardarla en una variable global, la cual será utilizada por otra clase, en este caso la clase del control derecho. Dentro de esto se realiza varias funciones para acceder a los enlaces del robot que se encuentra en una lista.
TryTeleport	Se verifica si se tiene una posición válida “m_Hasposition” y si no se está trasladando “m_Isteleporting”, luego se obtiene el objeto de la cámara “cameraRig” el cual representa el sistema de cámaras de la realidad virtual para obtener la posición de la cabeza de la persona que usa las gafas. En resumen, lo que hace es calcular la posición a la cual se quiere trasladar la persona y luego inicia el proceso de movimiento de la cámara.

Elaborado por: El Investigador

A continuación, se presenta la selección de cada una de las articulaciones que componen el robot kuka youbot en donde la articulación seleccionada se resalta con

un color para que de esta manera sea más interactivo la realidad virtual y también se conozca que parte del robot se va a controlar o manipular.

En la Figura 46 se presenta la selección de las articulaciones dentro de la realidad virtual y en la Figura 47 se presenta la selección de las ruedas para el movimiento de la plataforma móvil.

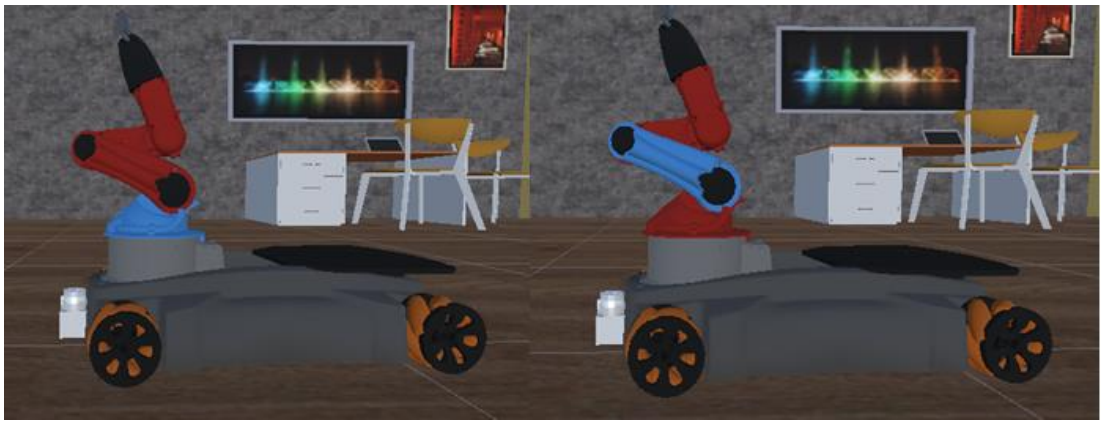

Figura 46. Selección de las articulaciones del robot en la realidad virtual.
Elaborado por: El Investigador

Figura 47. Selección de las ruedas para el movimiento de la plataforma móvil.
Elaborado por: El Investigador

En la Figura 48 se tiene el movimiento del brazo robótico una vez seleccionado cada una de las articulaciones, de esta manera la persona es capaz de ir manipulando y controlando el brazo hasta la posición que desee.

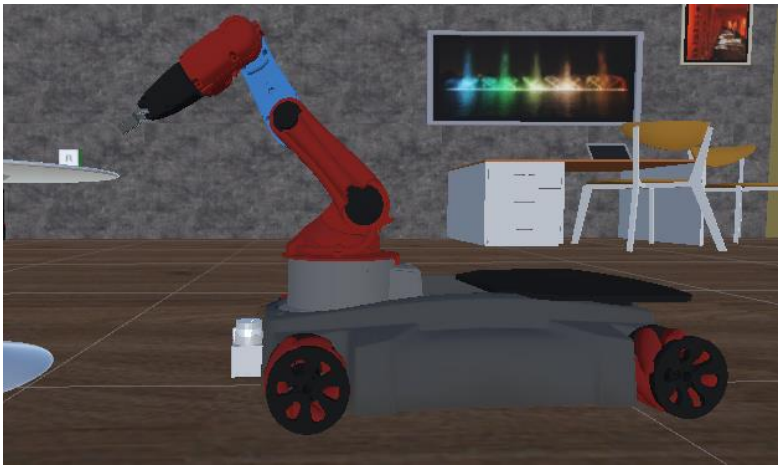

Figura 48. Movimiento del brazo en la realidad virtual.
Elaborado por: El Investigador

Procesamiento de datos de Unity a ROS

Para la manipulación del robot real es necesario recibir los datos desde Unity de cada articulación, así como los datos de las ruedas para el movimiento de la plataforma móvil. Para ello se necesita realizar un script en Python que permita suscribirse al tema que está publicando los datos desde Unity y luego estos datos sean publicados a cada uno de los temas respectivos del robot real en ROS.

Para que ROS sea capaz de conectarse con Unity se realiza un script el cual permite la conexión con Unity e inicializa el nodo “unity_endpoint” el cual va a actuar como si se tratara de un bróker el cual recibe comunicaciones de un lado (Unity) y se las envía a otros (ROS). En la Figura 49 se muestra el diagrama de proceso para el código desarrollado, en la cual se tienen que importar las librerías de “rospy” y “ros_tcp_endpoint” necesarios para la comunicación, se define una función principal con argumentos vacíos, este a su vez se encarga de inicializar el nodo, crear una instancia de la clase TcpServer utilizando el nombre del nodo. Una vez que se ejecuta el servidor TCP si la instanciación está bien se ejecuta el bucle principal de ROS que se va a encargar de recibir las conexiones entrantes desde Unity.

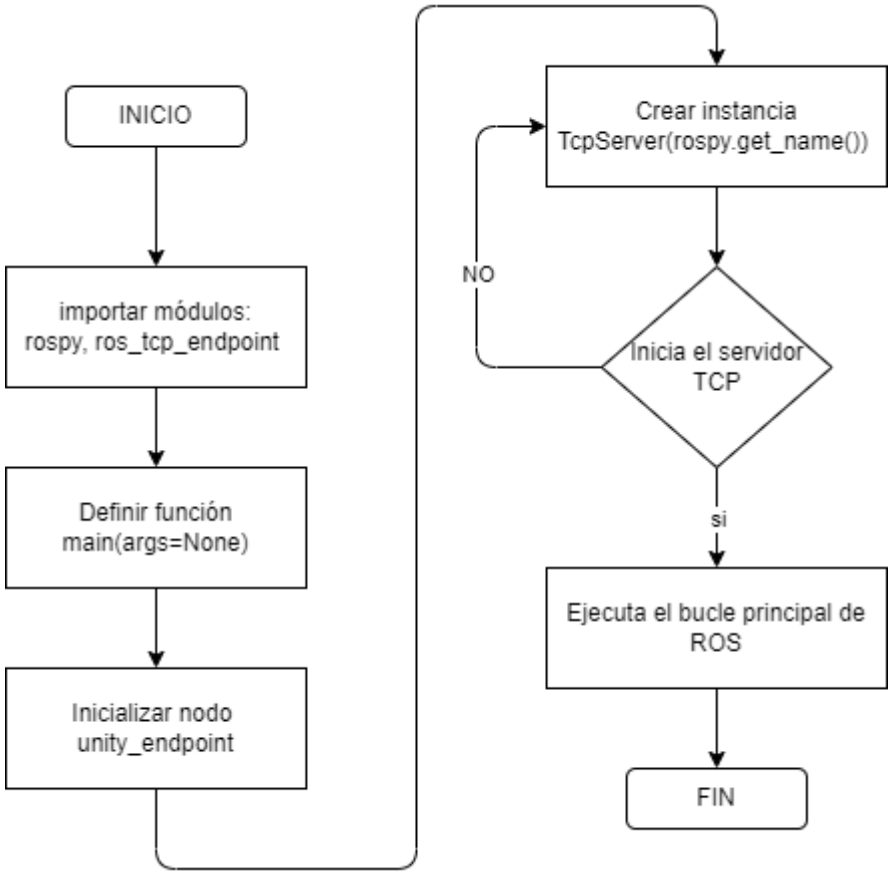

Figura 49. Flujograma de procesos de conexión Unity ROS
Elaborado por: El Investigador

Una vez que la conexión desde Unity a ROS se ha establecido es necesario procesar los datos recibidos en el nodo "unity_endpoint" para que se comunique con el nodo del robot real "youbot_driver_ros_interface" y a su vez se inicialicen los temas correspondientes para el movimiento tanto de las articulaciones como de la plataforma móvil.

En la Figura 50 se muestra el diagrama de procesos para el procesamiento de datos hacia el robot real en el cual se tiene que importar los módulos y paquetes necesarios para el tratamiento y envió de los datos. Dentro del programa se inicializa el nodo ROS denominado "listener" el cual permite suscribirse a los mensajes o datos que se encuentran en el tema "/pos_rot" enviados desde Unity. Una vez que se ha recibido los datos se llama a la función callback(data) en el cual se definen e inicializan las variables, se realiza un procesamiento de datos, es decir se convierten los datos recibidos de cada articulación en datos que se encuentren dentro del rango válido de

los limites rotacionales superiores e inferiores para que no se produzca daños en los motores de las articulaciones. Este método callback una vez que publica los comandos de control espera a recibir el siguiente mensaje hasta que se detenga el nodo ROS. Una vez realizado el procesamiento de datos entra a una serie de condiciones donde se evalúa la articulación, es decir si el valor recibido es 6 se realiza el movimiento de la plataforma en el eje "x" y publica los datos en el tema "/cmd_vel", de la misma manera sucede para el valor 8 el cual realiza un movimiento lineal en el eje "y" y el valor 9 realiza un movimiento rotacional en el eje z. En caso de tener el valor 7 se produce el movimiento del gripper en su tema correspondiente y en caso de que estos valores no sean ni 6,7,8,9 se produce el movimiento de las articulaciones del brazo.

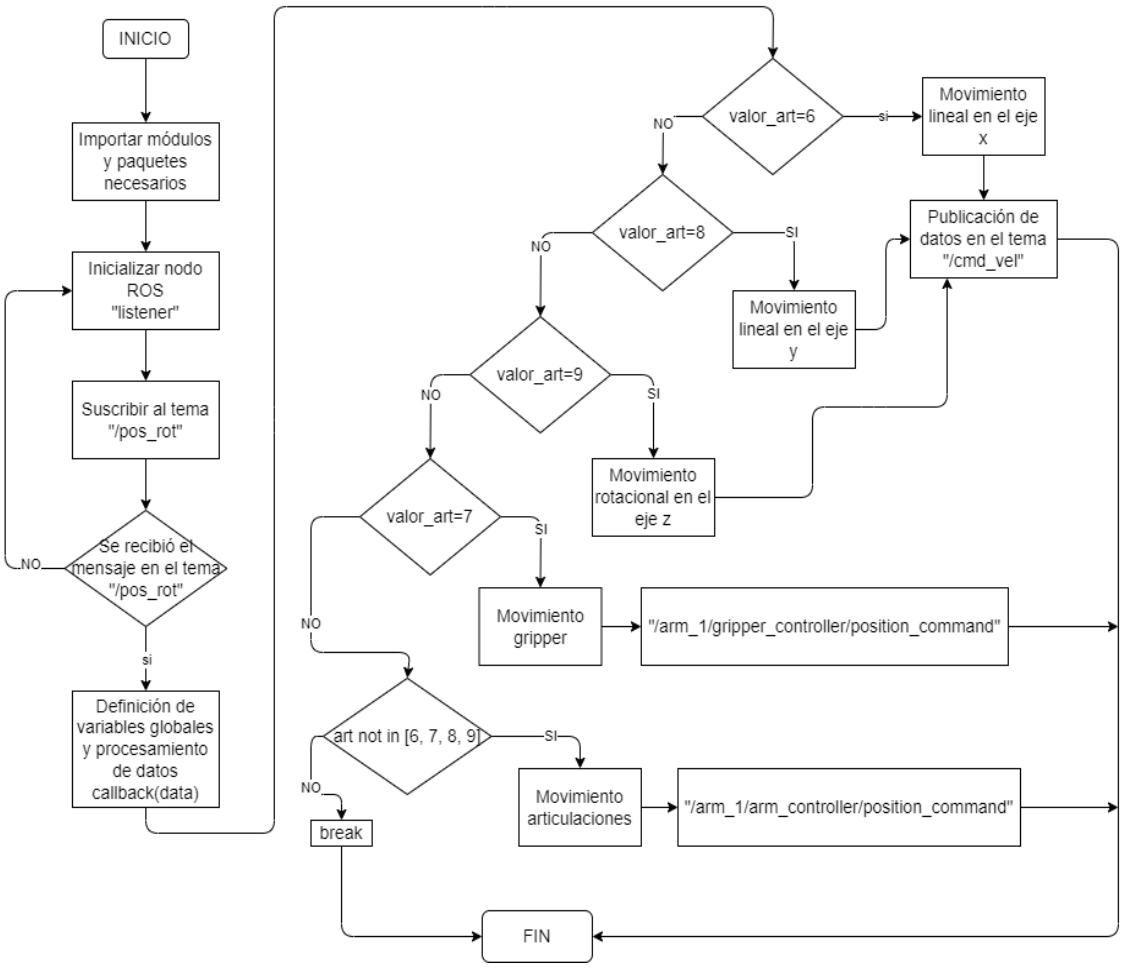

Figura 50. Flujograma de procesos del procesamiento de datos
Elaborado por: El Investigador

En la Figura 51 se observa el diagrama de nodos en donde se reciben los datos desde Unity mediante el nodo "/unity_endpoint" el cual se suscribe al tema "/pos_rot" que se encuentra definido en Unity. El nodo que se encarga de recibir los datos se

denomina “/listener”, este se encarga de procesar los datos recibidos y transformarlos al rango dentro de los limites rotacionales superiores e inferiores del robot real. Una vez que los datos han sido procesados se publican en sus temas respectivos, para el caso del control del brazo los datos se publican en “/arm_1/arm_controller/position_command”, para el gripper en “/arm_1/gripper_controller/position_command” y para las ruedas en “/cmd_vel”. Todos estos datos son enviados al nodo “/youbot_driver” el cual realiza los movimientos tanto del brazo como de la plataforma móvil.

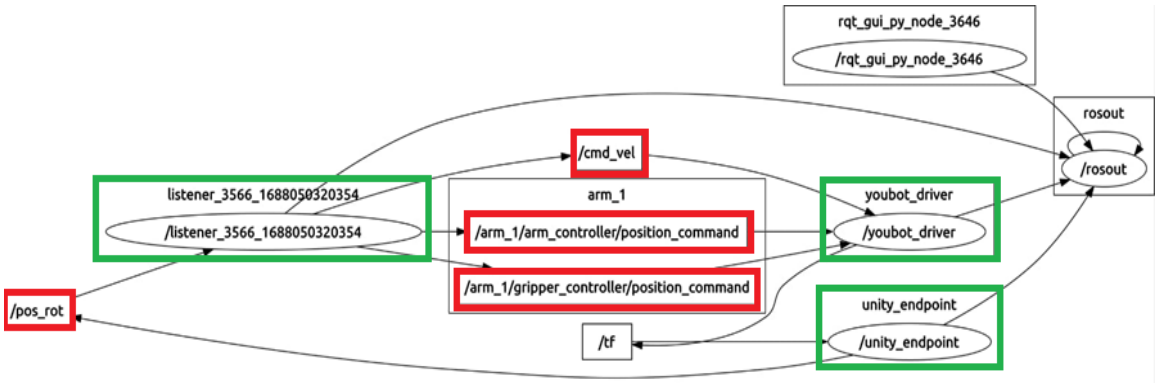

Figura 51. Diagrama de nodos del procesamiento de los datos recibidos desde Unity
Elaborado por: El Investigador

En la Figura 52 y 53 se observa el movimiento del robot en la escena de realidad virtual y el movimiento que el robot real realiza.

Figura 52. Primer movimiento del kuka youbot real
Elaborado por: El Investigador

Figura 53. Segundo movimiento del kuka youbot real
Elaborado por: El Investigador.

3.2.7. Usabilidad del sistema

La usabilidad del sistema con realidad virtual se evalúa mediante el método SUS (System Usability Scale) desarrollado por John Brooke. Este método consta de 10 enunciados predefinidos, los cuales sirven para evaluar la usabilidad de cualquier sistema. Las respuestas de cada enunciado siguen la escala de Likert de modo que existe 5 opciones de respuesta para cada pregunta. La encuesta aplicada a los estudiantes se detalla en el Anexo C.

La encuesta se aplica a una muestra de 39 estudiantes de la carrera de Telecomunicaciones. Estos estudiantes voluntariamente accedieron a utilizar y evaluar la interfaz. Antes de que los estudiantes comenzaran a evaluar el sistema se proporcionó una breve explicación sobre la interfaz de realidad virtual. Los resultados obtenidos se observan en la Figura 54.

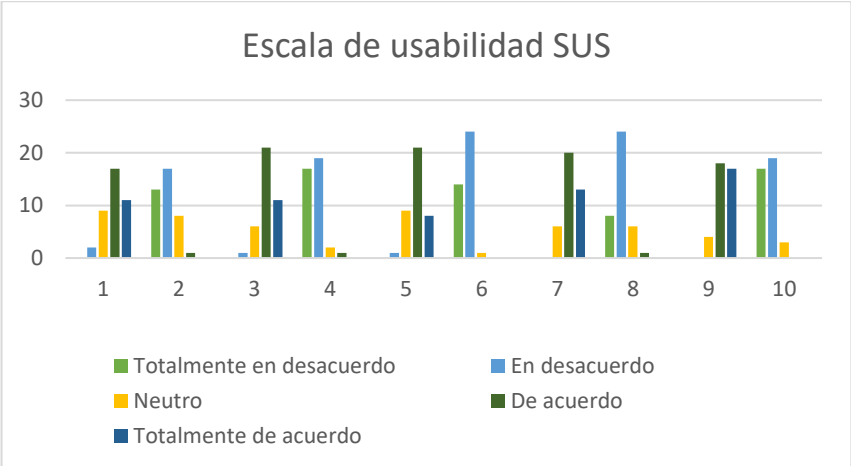

Figura 54. Resultados de la usabilidad del sistema.
Elaborado por: El Investigador.

Una vez realizado la encuesta se calcula con los datos obtenidos la puntuación individual para cada pregunta, hay que recalcar que las preguntas 1,3,5,7,9 son positivas y las preguntas 2,4,6,8,10 son negativas ya que el método SUS establece las preguntas de esa manera para evitar distorsión en las respuestas. En la Tabla 14 se muestra los resultados de la encuesta de usabilidad.

Tabla 14. Resultados encuesta de usabilidad.

Nº	Escala de usabilidad	Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo
1	¿Le gustaría utilizar este sistema con frecuencia?	0	2	9	17	11
2	¿Encuentra el sistema innecesariamente complejo?	13	17	8	1	0
3	¿El sistema es fácil de usar?	0	1	6	21	11
4	¿Necesitaría el apoyo de un técnico para poder utilizar este sistema?	17	19	2	1	0
5	¿Las diversas funciones de este sistema estaban bien integradas?	0	1	9	21	8
6	¿Había demasiada inconsistencia en este sistema?	14	24	1	0	0
7	¿La mayoría de los estudiantes aprendería a utilizar este sistema con rapidez?	0	0	6	20	13
8	¿Encuentra el sistema muy complicado de usar?	8	24	6	1	0
9	¿Se sintió muy seguro usando el sistema?	0	0	4	18	17
10	¿Se necesita aprender muchas cosas antes de empezar a utilizar el sistema?	17	19	3	0	0

Elaborado por: El Investigador.

Las respuestas de cada una de las preguntas se las evalúa en la escala de Likert del 1 al 5, en donde el mayor peso asignado corresponde al valor 5, lo que indica su mayor importancia y relevancia en la evaluación de las respuestas. En la Tabla 15 se muestra la encuesta en una escala de Likert.

Tabla 15. Escala de Likert para la encuesta de usabilidad.

Nº	Escala de usabilidad	Totalmente en desacuerdo	En desacuerdo	Neutro	De acuerdo	Totalmente de acuerdo
1	¿Le gustaría utilizar este sistema con frecuencia?	1	2	3	4	5
2	¿Encuentra el sistema innecesariamente complejo?	1	2	3	4	5
3	¿El sistema es fácil de usar?	1	2	3	4	5
4	¿Necesitaría el apoyo de un técnico para poder utilizar este sistema?	1	2	3	4	5
5	¿Las diversas funciones de este sistema estaban bien integradas?	1	2	3	4	5
6	¿Había demasiada inconsistencia en este sistema?	1	2	3	4	5
7	¿La mayoría de los estudiantes aprendería a utilizar este sistema con rapidez?	1	2	3	4	5
8	¿Encuentra el sistema muy complicado de usar?	1	2	3	4	5
9	¿Se sintió muy seguro usando el sistema?	1	2	3	4	5
10	¿Se necesita aprender muchas cosas antes de empezar a utilizar el sistema?	1	2	3	4	5

Elaborado por: El Investigador.

Los valores de la escala de Likert se deben de sumar de acuerdo con lo siguiente para obtener el puntaje de usabilidad [40]:

- Suma las respuestas de los enunciados impares y después resta 5

$$(4 + 4 + 4 + 4 + 4) = 20 - 5 = 15$$

- Suma las respuestas de los enunciados pares y resta ese total a 25

$$(2 + 2 + 2 + 2 + 2) = 25 - 10 = 15$$

- Suma ambos resultados y multiplícalo por 2,5.

$$(15 + 15) * 2.5 = 75$$

De acuerdo con la Figura 55 el puntaje obtenido de SUS con un valor de 75 se concluye que el puntaje de usabilidad del sistema desarrollado con realidad virtual está en el grado **B**, un adjetivo de calificación **Buena**, la aceptabilidad es **Aceptable**, y la puntuación neta del promotor (NPS) corresponde a la categoría de **Pasivo,** esto sirve

para ver la lealtad del usuario hacia al sistema, es decir que están satisfechos con la experiencia de realidad virtual, pero se podría realizar mejoras.

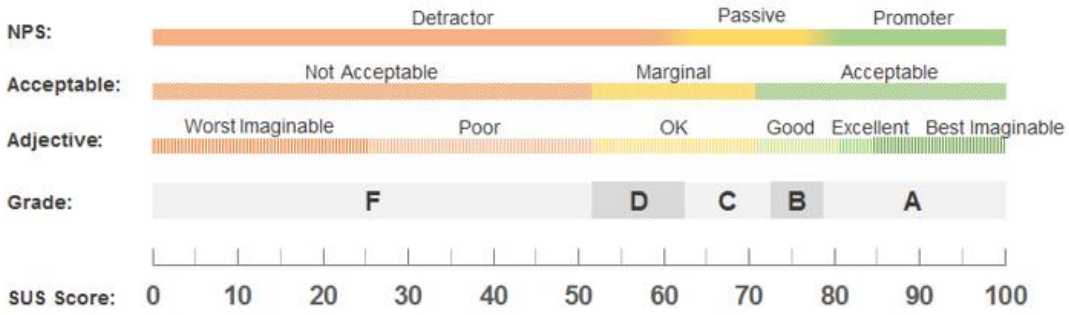

Figura 55. Rango de aceptabilidad de la escala de usabilidad SUS [40].

En la Figura 56 se observa la gráfica que muestra la distribución del puntaje SUS calculado con el percentil correspondiente a 73%, el percentil es útil para comprender la usabilidad del sistema, si la puntuación SUS está en un percentil alto, indica que la mayoría de los usuarios están satisfechos con la usabilidad del sistema, mientras que un percentil bajo indica que se debe mejorar varios aspectos de usabilidad.

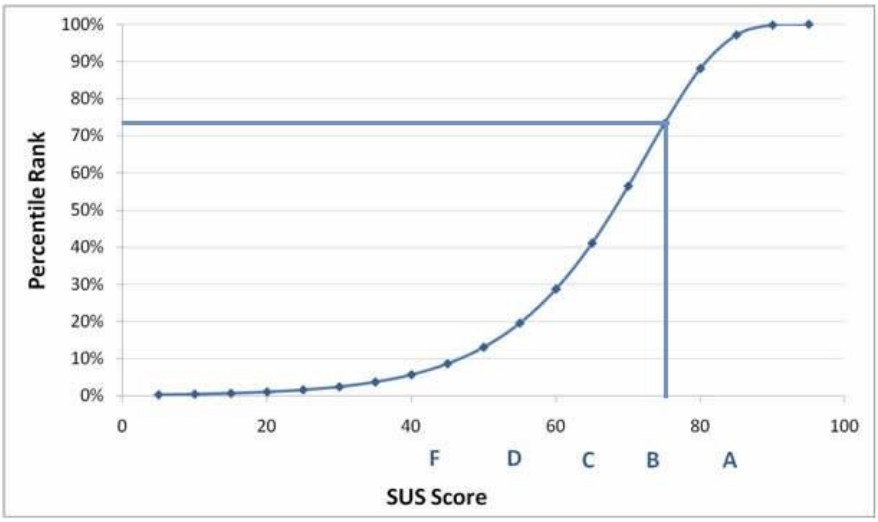

Figura 56. Distribución de puntaje del SUS.
Elaborado por: El Investigador.

3.2.8. Pruebas de funcionamiento y comunicación

Para el funcionamiento y comunicación del sistema se realiza distintas pruebas con el objetivo de verificar que todos los procesos funcionen correctamente. A continuación, se presenta todas las pruebas realizadas:

Prueba de manipulación del robot

Esta prueba consiste en que los estudiantes utilicen la interfaz de realidad virtual mientras controlan el robot móvil. Para ello se les explicó brevemente como es el funcionamiento del robot y el objetivo de esta prueba es ir controlando libremente cada una de las partes del robot.

En esta prueba se tiene 5 niveles de desempeño de control y se asigna puntuaciones a cada nivel para obtener el control alcanzado que tienen al manipular el robot, esto se realiza de la siguiente manera:

- **Excelente:** 5 puntos (estudiantes que demuestran un dominio completo y preciso sobre todas las partes del robot. Pueden mover el robot de manera fluida con alta exactitud y destreza).
- **Bueno:** 4 puntos (estudiantes que muestran un buen dominio en la manipulación del robot. Aunque pueden cometer algunos errores menores, en general, pueden manipular el robot con precisión y eficacia).
- **Aceptable:** 3 puntos (estudiantes que pueden controlar el robot de cierta forma, pero pueden cometer errores significativos. Pueden requerir una guía y apoyo de forma parcial).
- **Limitado:** 2 puntos (pueden enfrentar dificultades para manipularlo correctamente y realizar tareas con precisión. Requieren una supervisión y asistencia constante para lograr los objetivos).
- **Bajo:** 1 punto (tienen un control nulo sobre el robot, tienen dificultades significativas para comprender el control del robot mostrando un desempeño muy por debajo del nivel esperado).

En la Figura 57, 58 y 59 se observa a los estudiantes controlando el robot móvil y lo que están visualizando en la realidad virtual.

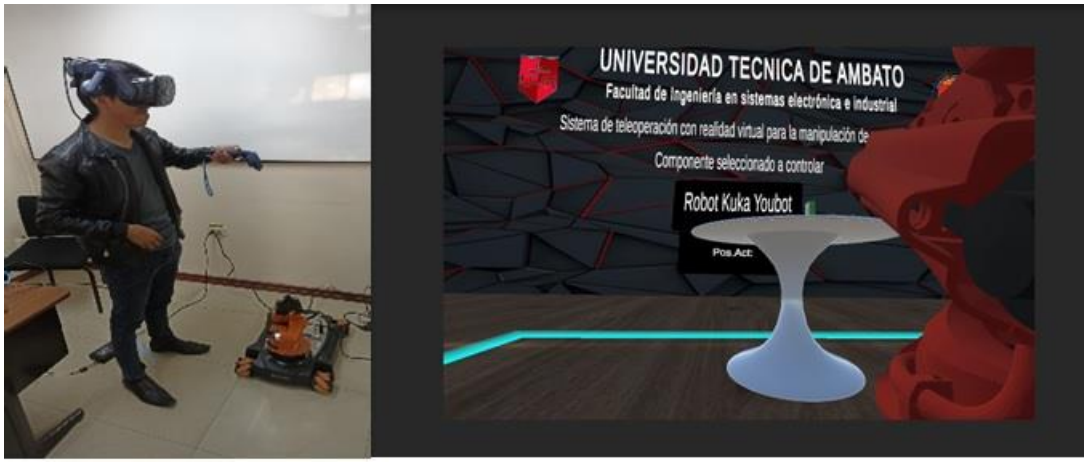

Figura 57. Primera prueba de manipulación del robot.
Elaborado por: El Investigador.

Figura 58. Segunda prueba de manipulación del robot.
Elaborado por: El Investigador.

Figura 59. Tercera prueba de manipulación del robot.
Elaborado por: El Investigador.

En la Tabla 16 se muestra los resultados por estudiante durante la manipulación del robot, se asigna la puntuación para cada nivel y se calcula el control alcanzado del robot. Como resultado de la media del control alcanzado se tiene que los 39 estudiantes

tienen una puntuación promedio de 4.36 el cual está ubicado en el nivel de “Bueno” lo que representa el 87.18% de dominio al controlar el robot. Es decir, los estudiantes pueden controlar el robot con un buen dominio, pero pueden cometer ciertos errores menores lo cual no afecta la manipulación del robot.

Tabla 16. Resultados de la prueba de manipulación.

Nº	**Nivel de desempeño**	**Puntuación**	**Control alcanzado con éxito (%)**
1	Excelente	5	100%
2	Bueno	4	80%
3	Bueno	4	80%
4	Bueno	4	80%
5	Excelente	5	100%
6	Aceptable	3	60%
7	Excelente	5	100%
8	Bueno	4	80%
9	Bueno	4	80%
10	Bueno	4	80%
11	Excelente	5	100%
12	Excelente	5	100%
13	Bueno	4	80%
14	Bueno	4	80%
15	Bueno	4	80%
16	Limitado	2	40%
17	Excelente	5	100%
18	Bueno	4	80%
19	Bueno	4	80%
20	Excelente	5	100%
21	Excelente	5	100%
22	Excelente	5	100%
23	Aceptable	4	80%
24	Excelente	5	100%
25	Limitado	2	40%
26	Excelente	5	100%
27	Excelente	5	100%
28	Excelente	5	100%
29	Bueno	4	80%
30	Bueno	4	80%
31	Excelente	5	100%

32	Excelente	5	100%
33	Bueno	4	80%
34	Excelente	5	100%
35	Excelente	5	100%
36	Bueno	4	80%
37	Excelente	5	100%
38	Bueno	4	80%
39	Excelente	5	100%
Promedio puntuación			4,36
Promedio control alcanzado			87,18%

Elaborado por: El Investigador.

En la Figura 60 se indica el control alcanzado del robot de los diferentes estudiantes en la prueba de manipulación del robot. Se tiene que 2 estudiantes tienen un desempeño de control del 40%, es decir que tienen dificultades para manipular el robot y necesitan una supervisión constante, 1 estudiante tiene un desempeño de control del 60%, es decir puede controlar el robot de cierta forma pero comete errores significativos, necesita una ayuda parcial, 17 estudiantes tienen un desempeño de control del 80% es decir que muestran un buen dominio del robot cometiendo errores menores que no afectan la manipulación del robot. Por último 19 estudiantes tienen un desempeño del 100% es decir que pueden controlar el robot sin ningún problema mostrando una gran destreza y fluidez al controlar el robot.

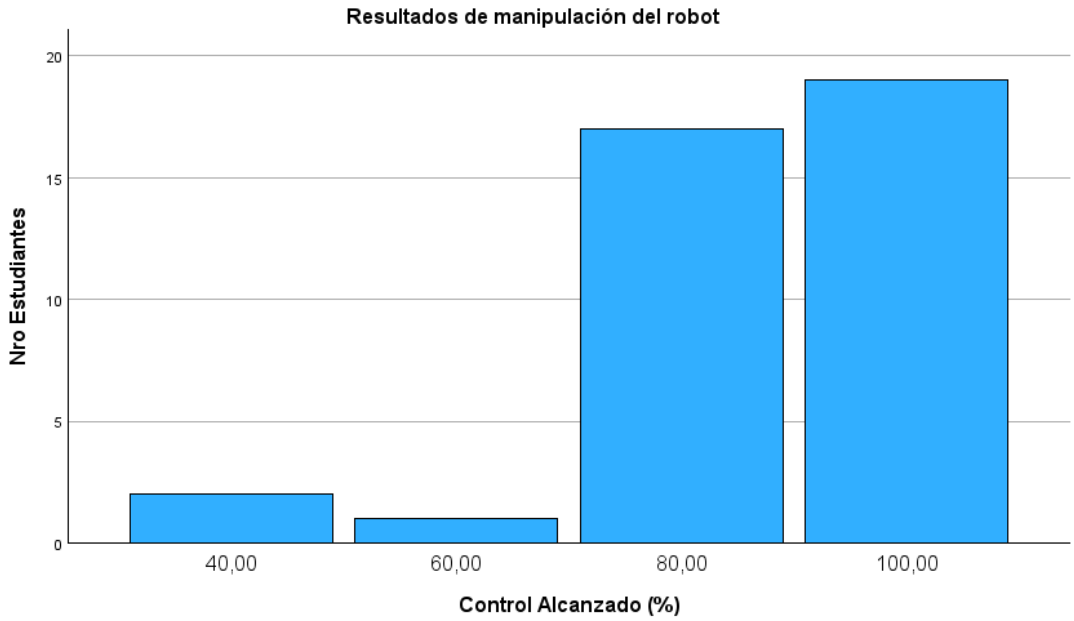

Figura 60. Resultados de manipulación del robot.
Elaborado por: El Investigador.

Prueba de posición inicial de las articulaciones del brazo

Esta prueba consiste en regresar a la posición inicial las articulaciones del brazo, esto con el fin de simplificar el proceso de manipulación si el usuario desea reiniciar una tarea o empezar de nuevo, también con esto se proporciona una manera rápida de reestablecer el brazo si el usuario realiza movimientos o cambios de posición que lleven a situaciones inseguras o incómodas.

Para esto se mueve dos articulaciones las cuales corresponden a la articulación 1 y 2 del robot hasta la posición en la cual se indica en la Figura 61.

Figura 61. Posiciones de las articulaciones 1 y 2.
Elaborado por: El Investigador.

Para comprobar que las articulaciones regresan a su posición inicial se puede comprobar en la realidad virtual en la Figura 62 que la posición actual es 0 es decir que se encuentra en su estado de reposo.

Figura 62. Posición inicial del brazo robótico.
Elaborado por: El Investigador.

Prueba de tiempos de ejecución para los movimientos del robot

Se procede a realizar una adquisición de datos, en este caso se toma en cuenta el tiempo de ejecución desde que se manda a ejecutar el proceso en Unity hasta que se ejecuta el movimiento en el robot. Para tomar cada uno de estos datos se aprovecha el número de personas que evaluaron la interfaz de realidad virtual y controlaron el robot móvil.

Además, en el artículo de Rassolkin A, *et.al* [41] indica que para tener un resultado estadísticamente significativo se debe seguir la regla estadística que establece que se debe usar un tamaño de muestra superior a 30. Esta regla se basa en el teorema del límite central en la cual se dice que cuando el tamaño de muestra es mayor a 30 la distribución de las medias muestrales se aproxima a una distribución normal independientemente de la forma de la distribución original de los datos.

En la Tabla 17 se muestra el tiempo de ejecución para las partes del robot móvil. Hay que destacar que estos tiempos se obtuvieron cuando se ejecutó el movimiento del robot una vez procesado los datos.

Tabla 17. Tiempo de ejecución para cada una de las pruebas realizadas.

Número ejecución	Parte	Tiempo ejecución (s)	Número ejecución	Parte	Tiempo ejecución (s)
1	arm_link_1	0.0085	**22**	arm_link_1	0.0078
2	arm_link_2	0.0094	**23**	arm_link_2	0.0102
3	arm_link_3	0.0127	**24**	arm_link_3	0.0105
4	arm_link_4	0.0080	**25**	arm_link_4	0.0124
5	arm_link_5	0.0090	**26**	arm_link_5	0.0116
6	gripper	0.0098	**27**	gripper	0.0106
7	ruedas	0.0070	**28**	ruedas	0.0110
8	arm_link_1	0.0142	**29**	arm_link_1	0.0060
9	arm_link_2	0.0157	**30**	arm_link_2	0.0062
10	arm_link_3	0.0099	**31**	arm_link_3	0.0142
11	arm_link_4	0.0120	**32**	arm_link_4	0.0114
12	arm_link_5	0.0068	**33**	arm_link_5	0.0170
13	gripper	0.0074	**34**	gripper	0.0109
14	ruedas	0.0076	**35**	ruedas	0.0059
15	arm_link_1	0.0104	**36**	arm_link_1	0.0057
16	arm_link_2	0.0134	**37**	arm_link_2	0.0097
17	arm_link_3	0.0122	**38**	arm_link_3	0.0136

18	arm_link_4	0.0185	**39**	arm_link_4	0.0089
19	arm_link_5	0.0086	**40**	arm_link_5	0.0099
20	gripper	0.0101	**41**	gripper	0.0111
21	ruedas	0.0082	**42**	ruedas	0.0112

Elaborado por: El Investigador.

- **Media Aritmética**

Se calcula la media aritmética para saber el tiempo de ejecución estimado que le toma al robot ejecutar los movimientos de acuerdo con las pruebas realizadas. Para este caso se tiene que la media es de:

$$Media(x) = 0{,}0104 \text{ seg}$$

El tiempo promedio que le toma al robot ejecutar el movimiento desde que el dato es enviado desde Unity hasta su procesamiento en ROS es de 0.0104 segundos o lo que es lo mismo 10.4 ms. Hay que destacar que este tiempo es afectado por varios factores como la latencia de red, el tráfico de red, la capacidad de procesamiento de los datos. Para tener una mejor visualización de los datos se muestra en la Figura 63 el tiempo de ejecución para cada uno de los movimientos, así como el tiempo promedio. La gráfica muestra que el mayor tiempo de ejecución sucede en los tiempos 0.0170 y 0.0185 segundos o lo que es lo mismo 17 ms y 18.5 ms. En estos tiempos es cuando se tiene mayor latencia al procesar los datos hacia el robot.

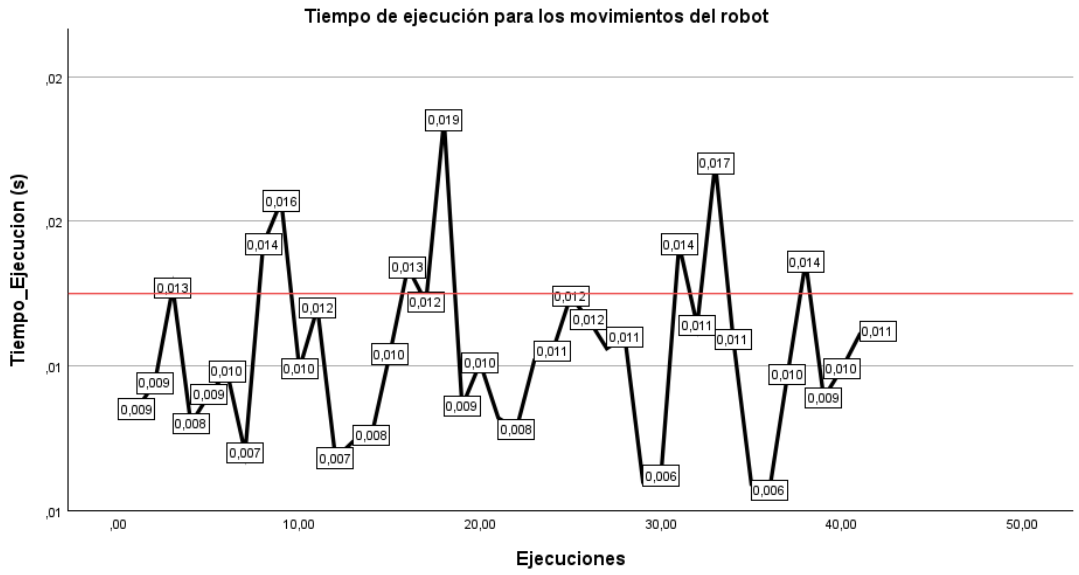

Figura 63. Tiempo de ejecución para los movimientos del robot.
Elaborado por: El Investigador.

- **Desviación estándar `**

Se calcula la desviación estándar para saber la dispersión de los datos, es decir cuanto varían los tiempos de ejecución en relación con la media. El resultado obtenido de la desviación estándar es:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(Xi - x)^2}{N}}$$

$$\sigma = 0.00295$$

El valor de la desviación estándar de 0.00295 indica la dispersión de los valores medidos de tiempo de ejecución en relación con la media, en este caso el valor calculado indica que los valores tienden a estar cerca de la media y tienen una variabilidad relativamente baja. Esto indica que los tiempos de ejecución del movimiento del robot son bastante consistentes y no varían mucho en relación con la media.

- **Distribución normal**

Tomando en cuenta que la muestra es mayor a 30 datos se procede a realizar una distribución normal una vez calculado los datos de la media y la desviación estándar.

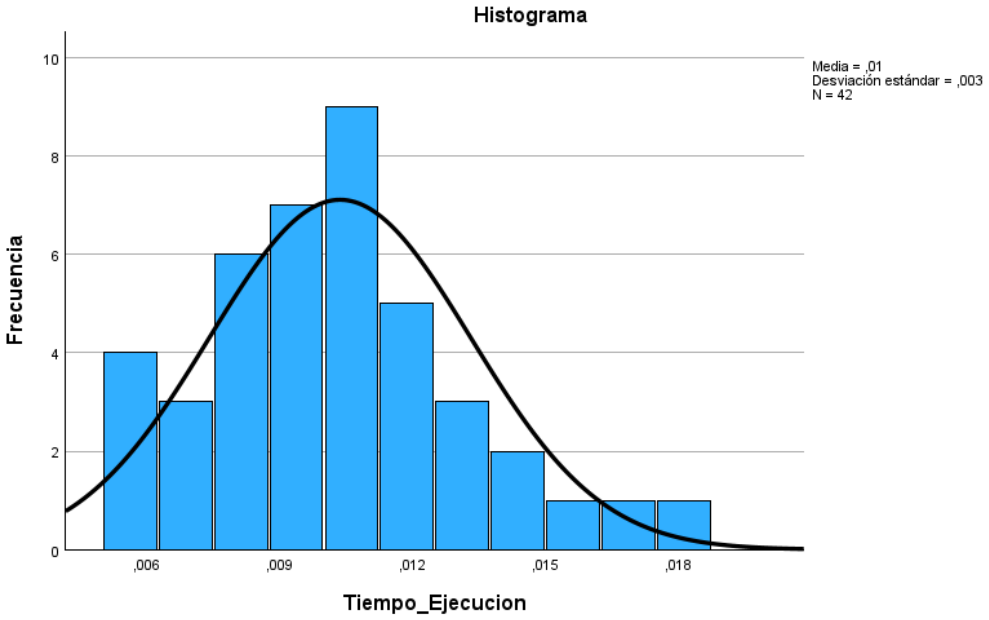

Figura 64. Distribución normal de los tiempos de ejecución.
Elaborado por: El Investigador.

En la Figura 64 se observa que la media es de 0.0104 seg y este representa el punto central de los datos, entonces se dice que existe una tendencia hacia una distribución simétrica de los tiempos de ejecución, además la desviación estándar indica que los tiempos tienen a estar agrupados cerca de la media. Con estos datos se asume que los tiempos de ejecución se aproximan a una distribución normal ya que los datos se asemejan a una campana de gauss. Esta distribución normal sirve para predecir y estimar valores futuros con respecto a los tiempos de ejecución o el rendimiento del sistema.

- **Prueba de Normalidad**

Para conocer de una manera más precisa si los datos obtenidos tienden a una distribución normal se realiza la aplicación de una prueba estadística de normalidad conocida como Shapiro-Wilk para la respectiva validación, ya que los datos de la muestra son menores a 50. El proceso de cálculo se lo realiza con el software SPSS y se detalla en el Anexo D. Los resultados obtenidos de la prueba son los que se detallan en la Tabla 18.

Tabla 18. Prueba de normalidad Shapiro-Wilk

Tiempo ejecución	Estadístico	gl	p
	0.965	42	0.227

Elaborado por: El Investigador.

El valor obtenido es de 0.227 entonces se concluye que los datos obtenidos del tiempo de ejecución para los movimientos del robot siguen una distribución normal ya que la prueba estadística indica que si el valor de p es mayor a 0.05 los datos tienden a una distribución normal y esto indica que los datos tienen una buena consistencia y se acercan a la media. Esto garantiza que estos datos son confiables para una buena comunicación entre Unity y ROS al no tener mucha diferencia o discrepancia con la media calculada.

Otra forma de evaluar si los datos se aproximan a una distribución normal es con el grafico Q-Q normal de la Figura 65 el cual también se muestra cuando se hace el cálculo en SPSS. Este gráfico muestra una línea diagonal que representa la distribución

normal esperada. Si los puntos se acercan a esta línea recta indica una buena aproximación hacia una distribución normal.

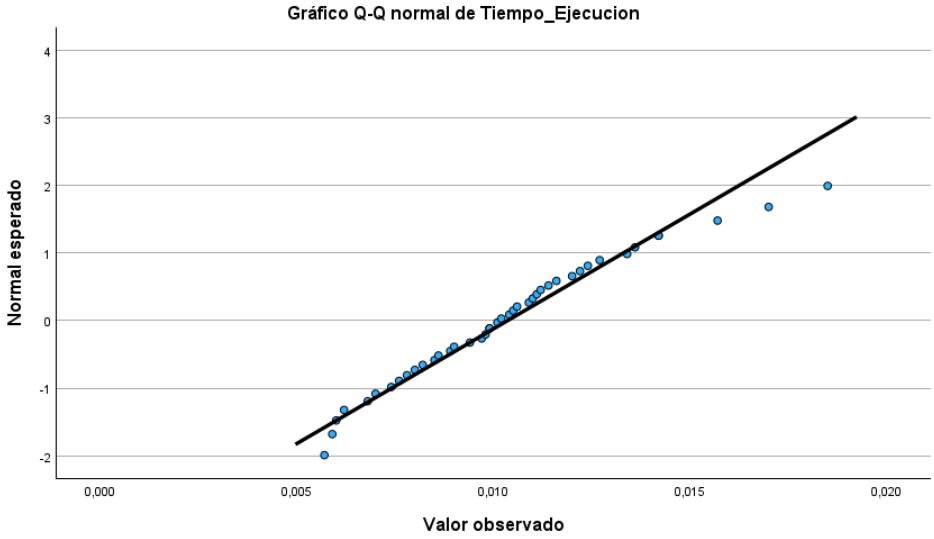

Figura 65. Gráfico Q-Q de normalidad.
Elaborado por: El Investigador.

Prueba de análisis de trafico de paquetes

Para el análisis del tráfico de paquetes se utiliza el software de código abierto Wireshark el cual permite obtener el número de paquetes procesados para el movimiento del robot, así como el tiempo entre cada uno de los paquetes. Para capturar los paquetes de interés se realiza un filtrado solo del puerto TCP 10000, ya que por este puerto se comunica Unity con ROS. En la Figura 66 se muestra el total de paquetes capturados, así como el tiempo transcurrido en cada paquete. En este caso se tomó el último paquete para ver los tiempos, y se tiene que este paquete fue capturado en 80.35 segundos y el tiempo que transcurrió desde el anterior paquete capturado fue de 0.000020459 segundos o lo que es lo mismo 0.0204 ms. A continuación, en la Tabla 19 se muestra un resumen del análisis de paquetes.

Tabla 19. Resumen de los paquetes de Unity-ROS.

Nodo Publicador Unity	192.168.1.16
Nodo Suscriptor ROS	192.168.1.17
Paquetes totales enviados desde Unity	715
Paquetes totales recibidos en ROS	693
Paquetes perdidos	22

Elaborado por: El Investigador.

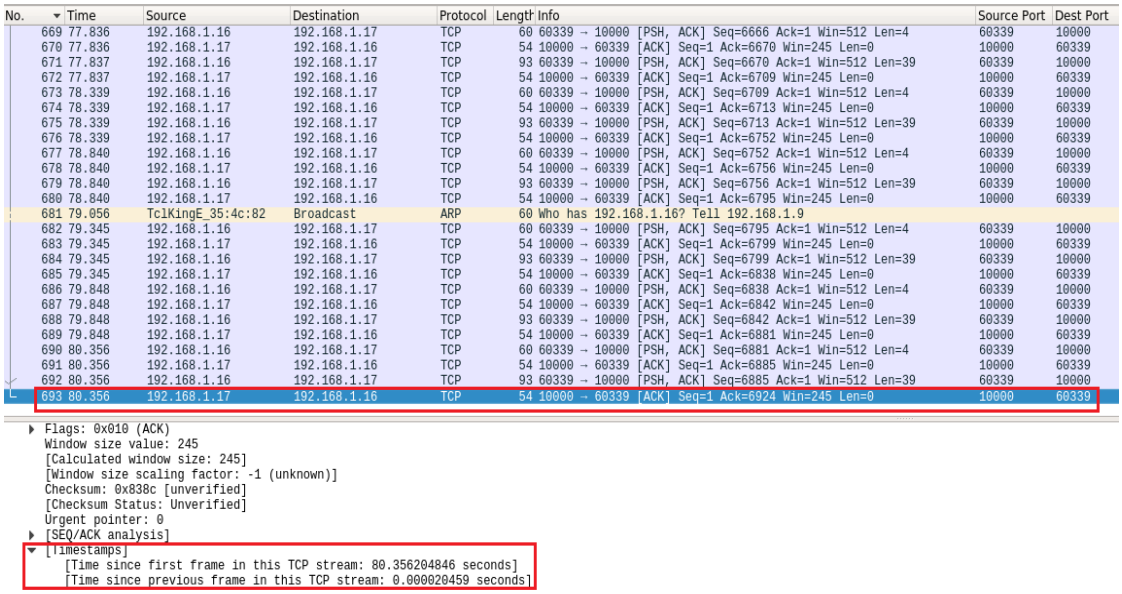

Figura 66. Paquetes capturados en Wireshark.
Elaborado por: El Investigador.

En la Figura 67 se muestra la latencia de los datos, en este caso los paquetes se capturaron cada 100ms, en esta gráfica se tiene en el eje horizontal el tiempo en el que fue capturado cada paquete, siendo el último paquete capturado en un tiempo de 80.35 segundos. En el eje vertical se muestra el tiempo que transcurrió desde el último paquete capturado, es decir el tiempo de ida desde el origen hasta el destino.

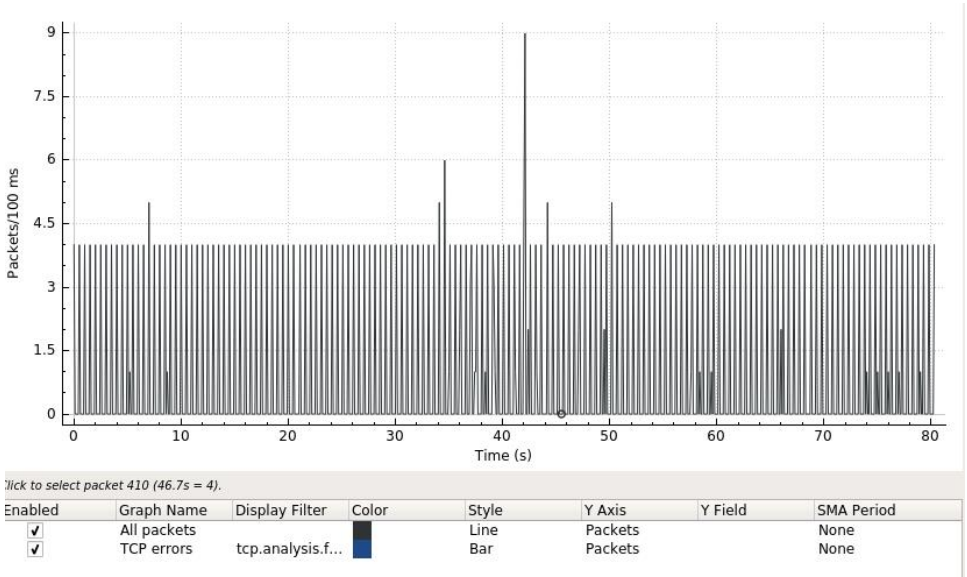

Figura 67. Latencia de los datos.
Elaborado por: El Investigador.

Prueba de retardo de los datos (Round Trip Time)

En la Figura 68 se muestra el retardo de red durante la ejecución de los movimientos del robot, en esta gráfica se observa la variación del tiempo de ida y vuelta entre los paquetes enviados y recibidos. En el eje horizontal se muestra el tiempo transcurrido entre paquetes y en el eje vertical se muestra el valor de RTT (retardo) en ms de cada paquete. La media del retardo de los datos es de 0.115 ms y además se tiene un valor pico máximo de 1.65 ms lo que significa que, en algunas ocasiones, los paquetes pueden experimentar latencias más altas que la media siendo este el peor de los casos y un valor pico mínimo de 0.032 ms lo que significa que en algunas ocasiones los paquetes pueden experimentar retardos muy bajos. A partir de esto se tiene una varianza de los datos de 1.180557 que indican cuánto se alejan los valores de la media.

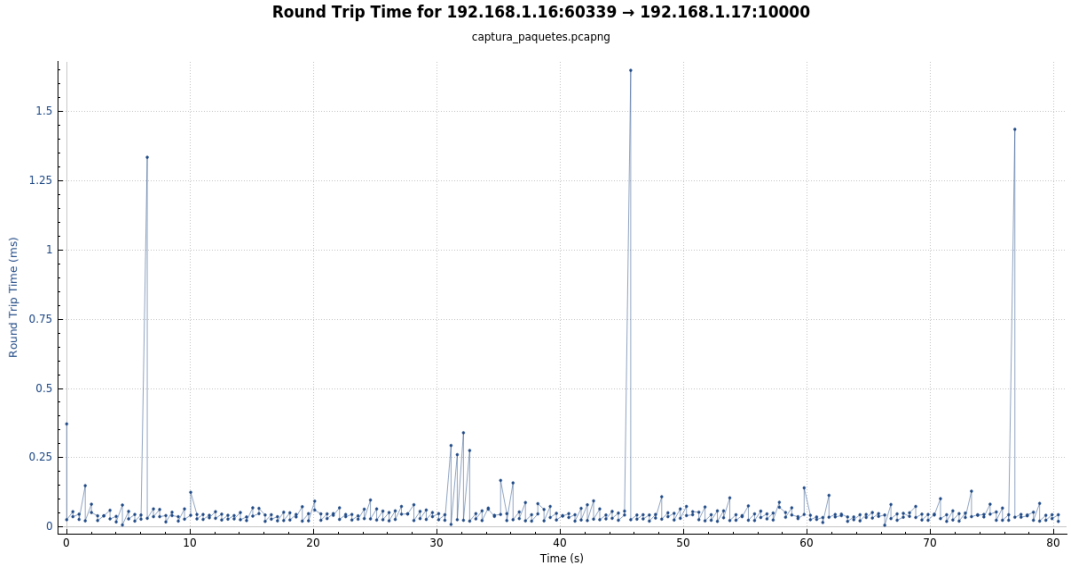

Figura 68. Retardo de los datos.
Elaborado por: El Investigador.

3.2.9. Eficacia del sistema

La eficacia del sistema se realiza con los datos obtenidos del análisis de paquetes de la Tabla 19 mediante la siguiente fórmula.

$$Eficacia = \frac{Paquetes\ recibidos\ correctamente}{Total\ de\ paquetes\ entregados} * 100$$

$$Eficacia = \frac{693}{715} * 100$$

$$Eficacia = 96.92\%$$

Se tiene como resultado un 96.92% que, de acuerdo con el nivel de eficacia de un sistema de la Figura 69 se encuentra en el rango de “Muy eficaz”. Esto quiere decir que el sistema desarrollado es efectivo ya que no existe tanta pérdida de paquetes al momento de enviar datos desde Unity hacia el robot real.

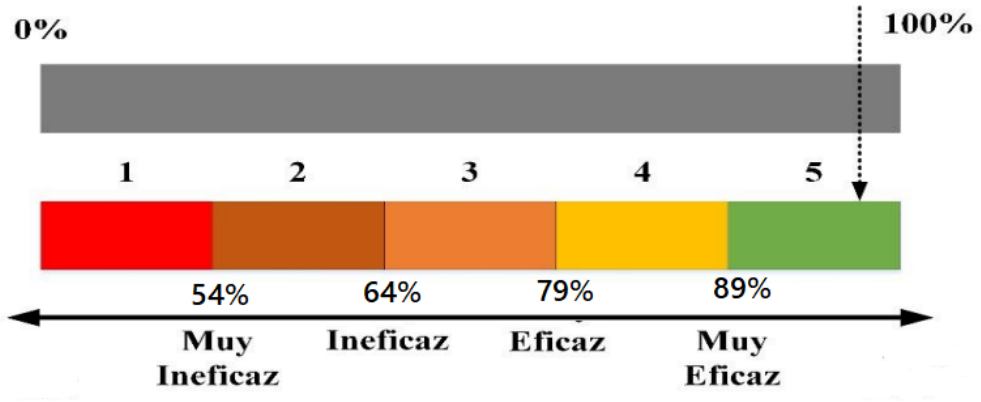

Figura 69. Nivel de eficacia de un sistema. [15].

3.2.10. Presupuesto

En el presupuesto, es necesario señalar que el robot kuka youbot y las gafas de realidad virtual HTC Vive Pro 2 no se consideran, ya que estos elementos fueron suministrados por la Universidad. Lo que se incluye en el presupuesto son las horas empleadas para el desarrollo de la investigación, además de ciertos recursos que se detallan en la Tabla 20.

Para las horas empleadas hay que considerar el salario de un Ingeniero en Telecomunicaciones en Ecuador de acuerdo con el Art. 7 de la LOTAIP (Ley Orgánica de Transparencia y Acceso a la Información Pública) que corresponde a $700 por mes [42].

Se calcula el pago diario tomando en cuenta 22 días laborables del mes sin tomar en cuenta los fines de semana que son 8 días al mes.

$$Pago_{dia} = \frac{700}{22} = \$31.81$$

Para el desarrollo de la investigación se trabajó un promedio de 6 horas al día.

$$Pago_{hora} = \frac{31.81}{6} = \$5.30$$

Finalmente se calcula el presupuesto de las horas empleadas multiplicando el pago por hora al día por el número total de horas empleadas, en este caso se trabajó por 4 meses lo que representa un total de 528 horas. Entonces el presupuesto de las horas empleadas para la investigación es de $2798.40

Tabla 20. Presupuesto del proyecto de investigación.

N°	Detalle	Cantidad	Precio Unitario	Precio total
1	Curso de Unity	1	$40	$40
2	Impresiones	-	$20	$20
3	Transporte	-	$80	$80
4	Horas de investigación	528	$5.30	$2798.40
			Total	$2938.40

Elaborado por: El Investigador.

CAPÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- El análisis de la cinemática del robot es importante ya que permite comprender los movimientos que el robot realiza, es decir mediante esto se logra un funcionamiento correcto y preciso de los robots en diversas aplicaciones industriales ya que al conocer las posiciones de cada articulación y del efector final se puede lograr movimientos eficientes y ver cuando el robot va a colisionar con sus articulaciones o con el entorno que lo rodea, además mediante la simulación en gazebo se comprendió el funcionamiento del robot ya que permitió realizar pruebas, experimentos y controlarlo de distintas maneras para así ver cómo responde el robot a distintas condiciones. Dentro de gazebo se establece la cinemática del robot y mediante esto se facilita la comprensión de los conceptos fundamentales de la cinemática.
- La metodología utilizada para entornos virtuales destinado a la manipulación de robots demostró ser efectivo ya que se combina tecnologías como la realidad virtual y la teleoperación. Se concluye que la interfaz inmersiva de realidad virtual permite a las personas controlar un robot de manera intuitiva y precisa, también mediante la aplicación del método SUS se midió la usabilidad del sistema en el cual se obtuvo un valor de 75 el cual según este método está dentro del rango aceptable. Además, esta tecnología es escalable y adaptable ya que permite su aplicación en diferentes tipos de robots y entornos abriendo la posibilidad de aplicar en varios campos como la industria, la medicina, educación, agricultura, entre otros.
- Los datos obtenidos mediante TCP presentan buena consistencia en cuanto a los tiempos de ejecución del robot ya que no presenta mucha latencia, el tiempo promedio de ejecución dio como resultado 10.4 ms lo cual no sobrepasa los 250 ms que son considerados como no aptos para un sistema de teleoperación. Estos tiempos de ejecución fueron evaluados mediante la prueba estadística Shapiro-Wilk y se determinó que los datos tienden a una distribución normal lo cual indica

que para posteriores mediciones los datos no van a estar tan alejados de su media, garantizando una estabilidad en los tiempos de ejecución.

- La implementación de un sistema de teleoperación con realidad virtual permite a la persona adquirir habilidades de aprendizaje y conocimientos mediante una simulación y entrenamiento en entornos virtuales antes de controlar y operar un robot en situaciones reales, de esta manera se garantiza que las personas se familiaricen con los controles, aprendan sobre el robot, y sobre todo reconozcan los posibles escenarios peligrosos y así no tengan que involucrarse directamente y correr riesgos físicos o daños irreversibles en su salud, además mediante la realidad virtual se evalúa posibles riesgos y se planifica estrategias adecuadas antes de posicionar al robot en la ubicación real.

4.2. Recomendaciones

- Para optimizar el rendimiento del sistema de teleoperación con realidad virtual en Unity, es recomendable implementar técnicas de optimización y gestión eficiente de recursos, por ejemplo, para optimizar la simulación del robot móvil en Unity es recomendable usar técnicas como LOD (Level of Detail) para renderizar modelos detallados solo cuando sea necesario y aprovechar sistemas de física eficientes para simular interacciones realistas, especialmente si se utilizan modelos 3D complejos. Además, se pueden aplicar técnicas de culling y frustum para evitar renderizar objetos que no sean visibles para el usuario en la escena virtual.
- Antes de comenzar el desarrollo del control en Unity, es fundamental contar con un modelo preciso y exhaustivo del robot youBot, que englobe su geometría, articulaciones y cinemática. Esto asegura una representación visual y de movimiento fiel al robot real. Asimismo, es de gran importancia obtener los parámetros cinemáticos del robot, ya que proporcionan datos esenciales sobre las medidas, estructura y conexiones espaciales entre las articulaciones del robot.
- Antes de utilizar las gafas HTC Vive Pro 2, se recomienda llevar a cabo una calibración minuciosa. Esto implica establecer con precisión la altura y posición del usuario en el entorno virtual, así como garantizar una correcta alineación entre el mundo virtual y el mundo real donde el robot opera. Una calibración exacta garantizará una correspondencia más precisa entre los movimientos de la cabeza,

las manos del usuario y los movimientos del robot en el entorno físico. Además, es crucial garantizar disponer de un espacio adecuado que permita moverse con libertad y realizar los movimientos necesarios para controlar el robot de manera segura y cómoda.

BIBLIOGRAFÍA

[1] A. Baturone, *Robótica: Manipuladores y robots móviles*. Barcelona España: Escuela Superior de Ingenieros Universidad de Sevilla, 2001.

[2] R. Suescun and Shi eun Lee, "Robots en América Latina: ¿cuántos son, dónde están y cuánto tributan? - Gestión fiscal," Mar. 21, 2019. https://blogs.iadb.org/gestion-fiscal/es/robots-en-america-latina-cuantos-son-donde-estan-y-cuanto-tributan/ (accessed Apr. 14, 2023).

[3] H. Cubides, L. Cuvi, J. L. Cuzco, and E. Ordoñez, "Diseño construcción e implementación de una plataforma robótica multifuncional con propósitos didácticos DINGO," Mar. 2012, Accessed: Apr. 14, 2023. [Online]. Available: https://dspace.ups.edu.ec/bitstream/123456789/8408/1/Dise%c3%b1o%20construcci%c3%b3n%20e%20implementaci%c3%b3n%20de%20una%20plataforma%20rob%c3%b3tica%20multifuncional%20con%20prop%c3%b3sitos%20did%c3%a1cticos%20DINGO.pdf

[4] El Universo, "Era del robot se instala a paso lento en Ecuador," *El Universo*, Quito, Aug. 19, 2018.

[5] A. Leguisamo, "Sistema de control para la teleoperación del robot kuka youbot mediante comunicaciones inalámbricas y el uso de tarjetas embebidas," Ambato, Sep. 2021.

[6] L. Peppoloni, F. Brizzi, C. A. Avizzano, and E. Ruffaldi, "Immersive ROS-integrated framework for robot teleoperation," *2015 IEEE Symposium on 3D User Interfaces, 3DUI 2015 - Proceedings*, pp. 177–178, Jun. 2015, doi: 10.1109/3DUI.2015.7131758.

[7] G. Caiza, P. Bonilla-Vasconez, C. A. Garcia, and M. V. Garcia, "Augmented Reality for Robot Control in Low-cost Automation Context and IoT," *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, vol. 2020-September, pp. 1461–1464, Sep. 2020, doi: 10.1109/ETFA46521.2020.9212056.

[8] L. Kästner and J. Lambrecht, "Augmented-Reality-Based Visualization of Navigation Data of Mobile Robots on the Microsoft Hololens-Possibilities and Limitations," 2020.

[9] W. Montalvo, C. A. Garcia, J. E. Naranjo, A. Ortiz, and M. V Garcia, “Sistema de Tele-operación para Robots Móviles en la industria del Petróleo y Gas Tele-operation system for Mobile Robots using in Oil & Gas industry,” 2020.

[10] M. Tovar Martínez, “Interfaz inmersiva de realidad virtual para la teleoperación de robots móviles,” Jul. 2021, Accessed: Apr. 12, 2023. [Online]. Available: http://rua.ua.es/dspace/handle/10045/116749

[11] M. Gonzáles, “Robótica Conceptos Básicos.” pp. 1–3, Jan. 02, 2023.

[12] “Introducción a la robótica y la automatización,” *MinnaLearn logo*, Feb. 10, 2019.

[13] A. Pérez, “Robótica móvil: Qué es y sus aplicaciones,” *Open Webinars*, Jul. 06, 2022.

[14] P. Vásconez, “Diseño de sistemas de control industrial de robots,” Apr. 2020, Accessed: Apr. 25, 2023. [Online]. Available: https://repositorio.uta.edu.ec/bitstream/123456789/29952/1/Tesis_1601id.pdf

[15] F. Montenegro, “Monitorización de parámetros eléctricos de un manipulador móvil,” Aug. 2021.

[16] F. Mirelez-Delgado, A. Morales-Díaz, R. Ríos-Cabrera, and H. Pérez-Villeda, “Control Servovisual de un Kuka youBot para la manipulación y traslado de objetos ⋆,” 2015.

[17] Kuka, “KUKA youBot User Manual,” Dec. 2013. Accessed: Apr. 14, 2023. [Online]. Available: https://usermanual.wiki/Document/KUKAyouBotUserManual.1109421140/view

[18] KUKA Laboratories, “KUKA youBot Operating and Assembly Instructions,” 2012.

[19] KUKA, “KUKA youBot Research & Application Development in Mobile Robotics”, Accessed: Apr. 14, 2023. [Online]. Available: https://www.generationrobots.com/img/Kuka-YouBot-Technical-Specs.pdf

[20] D. Delgado, “Qué es ROS (Robot Operating System),” *https://openwebinars.net/blog/que-es-ros/*, Sep. 21, 2017.

[21] ROS Tutorial, “Nodos, Topics y mensajes en ROS,” *http://rostutorial.com/4-nodos-topics-y-mensajes-turtlesim/*, 2018.

[22] I. Mendoza, “Introducción a ROS,” *https://www.cs.us.es/cursos/mcva-2022/docs/introduccion_ros_1.pdf*. Universidad de Sevilla, Sevilla.

[23] “Roscore y workspace – Fundamentos de ROS,” 2018. http://rostutorial.com/3-roscore-y-workspace/ (accessed Apr. 15, 2023).

[24] E. N. Ortega and L. Basañez Villaluenga, “Teleoperación: técnicas, aplicaciones, entorno sensorial y teleoperación inteligente,” 2004.

[25] L. Burbano, “Estudio e implementación en matlab de un entorno de comunicación basado en protocolos del internet de las cosas para clientes de teleoperación en robótica,” Escuela Politécnica Nacional, Quito, 2017.

[26] M. Gonzáles, “Qué es la Realidad virtual?,” Aug. 04, 2022. https://filmora.wondershare.es/video-editing-tips/what-is-vr.html?gclid=CjwKCAjwue6hBhBVEiwA9YTx8EYHInJ-0IDm9dMVhduehW0gFE8MHwSi1tjWfsxPV4_tqMYY_k4HdBoCKc8QAvD_BwE (accessed Apr. 15, 2023).

[27] LUDUS, “Aplicaciones de la realidad virtual,” 2018. https://www.ludusglobal.com/blog/aplicaciones-de-la-realidad-virtual (accessed Apr. 15, 2023).

[28] Obicex, “Sistemas de Realidad Virtual: ¿Cómo funcionan?,” 2020. https://www.obicex.es/blog/aprende-con-obicex/componentes-de-los-sistemas-de-realidad-virtual (accessed Apr. 15, 2023).

[29] Rocio GR, “Realidad virtual, la tecnología que ya está cambiando nuestras vidas,” *https://www.adslzone.net/reportajes/tecnologia/realidad-virtual-rv/*, Jan. 31, 2023.

[30] Hector R, “HTC Vive Pro 2 (Las gafas VR de gama alta) – Revisión completa,” *https://metaversoflow.com/htc-vive-pro-2/*, 2022.

[31] V. Rodríguez, “Programas para crear realidad virtual,” *https://i-amvr.com/programas-para-crear-realidad-virtual/,* Feb. 01, 2022.

[32] J. Alcántara, “Primeros pasos en Godot con realidad virtual,” *https://blog.kaleidos.net/Primeros-pasos-en-Realidad-Virtual-con-Godot/#:~:text=Godot%20es%20actualmente%20uno%20de,y%20con%20una%20documentaci%C3%B3n%20estupenda.*, Mar. 11, 2020.

[33] D. Pascual, “Simuladores para tecnología y robótica ,” *https://diocesanos.es/blogs/equipotic/2021/02/08/simuladores-para-tecnologia-y-robotica/,* Feb. 08, 2021.

[34] Gazebo, “Gazebo,” *https://rych.dcc.uchile.cl/doku.php?id=documentacion:gazebo,* 2017.

[35] J. Bologna, “Sistema de realidad aumentada para el entrenamiento de estudiantes en el manejo de instrumentación hart de la facultad de tecnologías de la información, telecomunicaciones e industrial de la universidad técnica de ambato,” Jan. 2020, Accessed: Apr. 28, 2023. [Online]. Available: https://repositorio.uta.edu.ec/handle/123456789/30736

[36] A. Farley, J. Wang, and J. A. Marshall, “How to pick a mobile robot simulator: A quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion,” *Simul Model Pract Theory*, vol. 120, p. 102629, Nov. 2022, doi: 10.1016/J.SIMPAT.2022.102629.

[37] M. de la C. Soler, “Interfaz de realidad virtual para el manejo de robots submarinos,” Universidad Jaume I, 2020.

[38] R. Muñoz, “Usabilidad en Mundos Virtuales,” Pontificia Universidad Católica de Valparaíso, Chile, 2011.

[39] I. Pérez, “Desarrollo de una plataforma para experimentación con Interface Cerebro Ordenador (BCI),” Universidad Politécnica de Cartagena, Cartagena, 2017.

[40] J. Brooke, “A quick and dirty usability scale,” 1986.

[41] A. Rassolkin, V. Rjabtsikov, T. Vaimann, and B. Asad, “Interface Development for Digital Twin of an Electric Motor Based on Empirical Performance Model.,” *IEEE Access*, vol. 10, pp. 7–8, Jan. 2022.

[42] LOTAIP, “Ley Orgánica de Transparencia y Acceso a la Información Publica,” *https://www.telecomunicaciones.gob.ec/wp-content/uploads/downloads/2016/04/literal-c-Remuneracion-mensual-por-puesto.-MARZO.pdf*, 2021.

[43] J. Legarreta and R. Martinez, “Modelado Geometrico y cinematico del robot,” *https://ocw.ehu.eus/pluginfile.php/50445/mod_resource/content/8/T5%20CINEMATICA%20OCW_Revision.pdf*.

[44] Ariadne’s Clew, “Cinematica del robot ,” pp. 15–18.

[45] W. Montalvo, C. A. Garcia, J. E. Naranjo, A. Ortiz, and M. V Garcia, “Sistema de Tele-operación para Robots Móviles en la industria del Petróleo y Gas Tele-operation system for Mobile Robots using in Oil & Gas industry.”

[46] E. Barahona, “Navegación autónoma basada en maniobras bajo estimación de posturas humanas para un robot omnidireccional Kuka Youbot,” Universidad Técnica de Ambato, 2019.

[47] K. Nagatani, S. Tachibana, M. Sofue, and Y. Tanaka, “Improvement of odometry for omnidirectional vehicle using optical flow information,” *IEEE International Conference on Intelligent Robots and Systems*, vol. 1, pp. 468–473, 2000, doi: 10.1109/IROS.2000.894648.

[48] E. Cardoso, A. Fernández, Sergio A. Marrero-Osorio, and Pablo F. Guardado, “Modelos cinemático y dinámico de un robot de cuatro grados de libertad ,” *Revista de Ingeniería Electrónica, Automática y Comunicaciones*, vol. 3, Sep. 2017.

ANEXOS

Anexo A: Informe técnico de la cinemática

UNIVERSIDAD TECNICA DE AMBATO

FACULTAD DE INGENIERIA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERIA EN TELECOMUNICACIONES

INFORME TÉCNICO DE LA CINEMÁTICA DEL KUKA YOUBOT

Tema: "SISTEMA DE TELEOPERACIÓN CON REALIDAD VIRTUAL PARA LA MANIPULACIÓN DE UN ROBOT MÓVIL"

Objetivo: Analizar la cinemática del kuka youbot para la comprensión del movimiento del robot y su aplicación dentro de una interfaz virtual.

1. Robot Kuka Youbot

El robot kuka youbot es un manipulador móvil de 5 grados de libertad el cual fue desarrollado por el fabricante Kuka principalmente para el desarrollo e investigación en el área de la educación, sin embargo, también sirve para la investigación en la industria [15].

El robot consta de las siguientes partes: una plataforma omnidireccional el cual sirve para que el robot se movilice hacia cualquier lugar ya que tiene unas ruedas tipo mecanum y también consta de un brazo manipulador y una pinza el cual sirve para recoger pequeños objetos y trasladarlos hacia otro lugar, este brazo manipulador tiene 5 eslabones los cuales están unidos por una serie de juntas llamadas articulaciones [15].

1.1.Especificaciones del kuka youbot

Las especificaciones del robot para el brazo manipulador y el gripper se mencionan en la Tabla A.1 y en la Tabla A.2 se mencionan las especificaciones para la plataforma móvil omnidireccional.

Tabla A. 1. Especificaciones del brazo y gripper del Kuka Youbot [18].

Especificaciones del brazo Kuka Youbot		
Numero de articulaciones	5	
Material	Yeso de magnesio	
Altura	655 mm	
Peso	5.690 kg (sin el gripper)	
Datos de las articulaciones	Rango	Máxima Velocidad
Articulación 1	$\pm 168^{\circ}$	90 º/s
Articulación 2	$+89°/-64°$	90 º/s
Articulación 3	$+145°/-150°$	90 º/s
Articulación 4	$\pm 102^{\circ}$	90 º/s
Articulación 5	$\pm 166^{\circ}$	90 º/s
Carga Útil	0.5 Kg	
Transmisión de datos	EtherCat	
Consumo de energía	240 W (24 V/10ª)	
Especificaciones del Gripper		
Diseño	2 pinzas	
Peso	0.245 Kg	
Altura	20 mm	
Máxima apertura	70 mm	
Máxima Carga	0.5 Kg	

Tabla A. 2. Especificaciones de la plataforma móvil Kuka Youbot. [18].

Especificaciones de la plataforma móvil Kuka Youbot	
Material	acero
Longitud	531 mm
Ancho	380 mm
Altura	140 mm
Diámetro ruedas	100 mm
Peso	20 Kg sin batería
Carga útil	20 Kg
Transmisión de datos	EtherCat
Youbot PC	Mini ITX board Intel Atom dual core 2 GB Ram 32 GB de disco solido SSD Ethernet, USB, COM
Temperatura ambiente	$5° - 35°$

2. Modelo cinemático de un robot manipulador

Se refiere al movimiento del robot en relación con un sistema de referencia en la cual se describe la forma analítica del movimiento espacial del robot como una función del

tiempo, sin tener en cuenta las fuerzas que lo provocan. La mecánica del robot se representa como una cadena cinemática en la cual los eslabones están conectados por articulaciones. La cinemática se divide en cinemática directa e inversa [14]. En la Figura A.1 se observa el modelo cinemático de un robot.

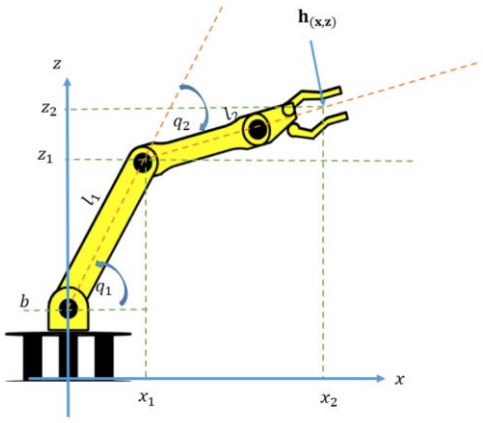

Figura A. 1. Cinemática de un robot manipulador [14]

3. Análisis de la cinemática de un robot manipulador

Para el análisis de la cinemática del robot manipulador kuka youbot es necesario empezar con los conceptos básicos de la cinemática. Dentro de la cinemática existe la cinemática directa y la cinemática inversa.

3.1.Cinemática Directa

Implica la identificación de la posición y orientación del extremo del robot en relación con un sistema de coordenadas de referencia. Esto requiere el conocimiento de los valores de las articulaciones y los parámetros geométricos de los componentes del robot. Los valores obtenidos al resolver esto permite conocer en qué punto se encuentra el extremo final del robot cuando se aplica un movimiento en las articulaciones lo que ayudará a determinar si ya se ha alcanzado la meta [43].

Para resolver este problema existen las siguientes maneras:

1. **Métodos Geométricos:** Para esto es necesario encontrar una relación que permita mediante la construcción de una o varias relaciones geométricas obtener los valores deseados de posición y orientación [43]. Se observa en la Figura A.2 un ejemplo de este método y las fórmulas que describen la posición y orientación se describen en las ecuaciones 1 y 2 respectivamente.

$$x = I_1 cosq_1 + I_2 \cos(q_1 + q_2) \quad (1)$$

$$y = I_1 senq_1 + I_2 \text{sen}(q_1 + q_2) \quad (2)$$

$$z = 0 \quad (3)$$

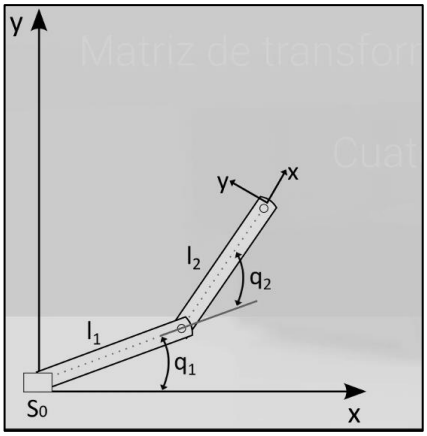

Figura A. 2. Método Geométricos [43].

El problema de este método se tiene cuando el número de grados de libertad es mayor a 3 ya que encontrar una relación geométrica que involucre todos los elementos del brazo es muy complejo, lo que no lo hace práctico para robots industriales.

2. **Matrices de transformación homogénea:** Se basa en el algebra vectorial y matricial. Esto consiste en que cada uno de los elementos que componen el brazo robótico es una cadena cinemática en el que cada eslabón se encuentra unido por una articulación. A la hora de utilizar una matriz de transformación se debe utilizar un sistema de referencia que relacione cada elemento. Lo habitual es utilizar la representación de Denavit Hartenberg conocido como DH.

   Esta representación define 4 transformaciones básicas para cada eslabón como se muestra a continuación y calcula la posición del último elemento en base a los elementos anteriores [43].

   - Rotación alrededor del eje z_{i-1}un ángulo θ_i
   - Traslación a lo largo de z_{i-1}una distancia d_i: vector $d_i(0,0,d_i)$
   - Traslación a lo largo de x_i una distancia a_i: vector $a_i(a_i,0,0)$
   - Rotación alrededor del eje x_i un ángulo α_i

3. **Cuaternios:** Un cuaternio es una representación de 4 valores o más bien un escalar y un vector como se muestra en la ecuación 4. Calcula el punto final mediante traslaciones y rotaciones [43].

$$(q_0, q_1, q_2, q_3)$$

S representa un escalar

V representa un vector

$$Q = [q_0, q_1, q_2, q_3] = [s, v] \qquad (4)$$

3.2. Cinemática Inversa

Intenta resolver como configurar la cadena de eslabones del robot para alcanzar una posición y orientación en el extremo conocido. En otras palabras, debe calcular la posición y orientación de cada una de las articulaciones del robot para que el extremo se situé en donde se desee.

El problema de la cinemática inversa es más complicado ya que es posible que existan varias o múltiples soluciones para una misma posición y orientación del elemento terminal como se muestra en la Figura A.3. Muchas de las veces para un robot manipulador, las soluciones pueden no estar dentro del rango de configuraciones de su elemento terminal y de la estructura de sus brazos [44], como se muestra en la Figura A.4.

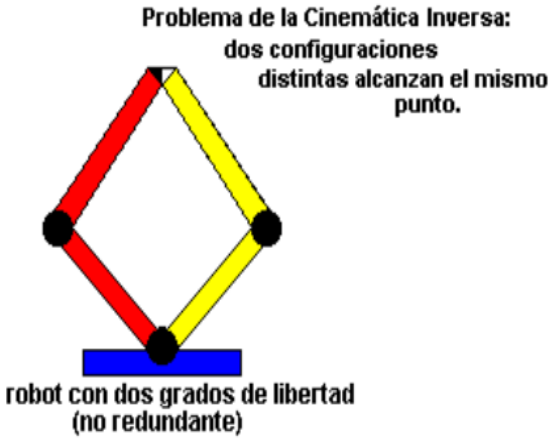

Figura A. 3. Problema de la cinemática inversa para un robot con 2 grados de libertad [44].

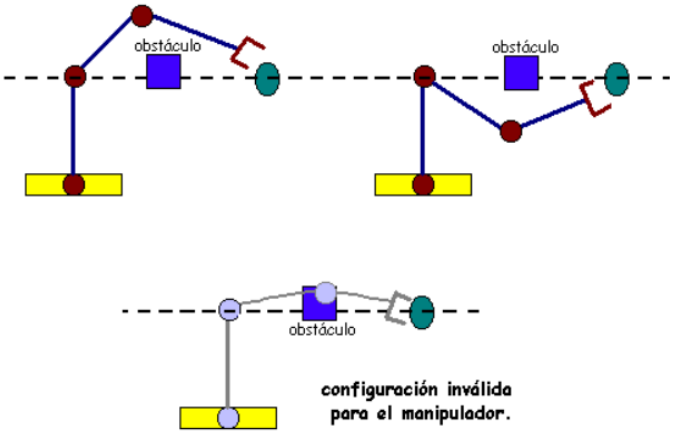

Figura A. 4. Problema cinemático para un robot de 3 grados de libertad [44].

Hay que tener en cuenta que el aumento de los grados de libertad también aumenta la complejidad computacional de la cinemática inversa, la planificación de trayectorias y la detección de colisiones.

4. Modelo Cinemático del Kuka Youbot

Dentro del modelo cinemático del robot hay dos modelos que toca analizar, los cuales corresponden al modelo cinemático de la plataforma omnidireccional y del brazo robótico.

4.1. Modelo cinemático de la plataforma omnidireccional

La plataforma robótica omnidireccional está conformada por 4 ruedas tipo Mecanum, las cuales permiten un desplazamiento libre en cualquier dirección, sin necesidad de que la parte frontal de la plataforma gire, sin embargo, hay que considerar en las ruedas la disposición de los rodillos para tener un modelo cinemático adecuado [45]. Las medidas de la plataforma se expresan en milímetros de acuerdo con la Figura A.5.

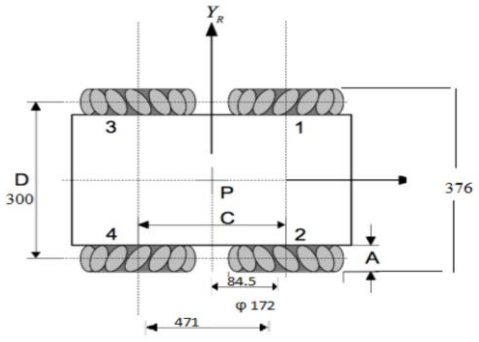

Figura A. 5. Medidas de la plataforma móvil [46].

De acuerdo con William Montalvo *et.al* [45] y Keiji Nagatani *et.al* [47] el modelo cinemático de la plataforma omnidireccional se representa por las ecuaciones (5), (6) y (7), en donde ẋ y ẏ representan las velocidades en los ejes X y Y, además θ_b representa la velocidad angular de la plataforma omnidireccional. El parámetro d_i se refiere a la velocidad lineal de cada una de las ruedas, por otro lado α_b y β dependen del ángulo en que se encuentren los rodillos dentro de cada una de las ruedas, entonces el movimiento dentro del plano dependerá de las combinaciones de las velocidades lineales de cada rueda. Los movimientos tanto de traslación como de rotación en el cual la plataforma difuminada indica hacia donde se moverá de acuerdo con las velocidades lineales se indica en la Figura A.6.

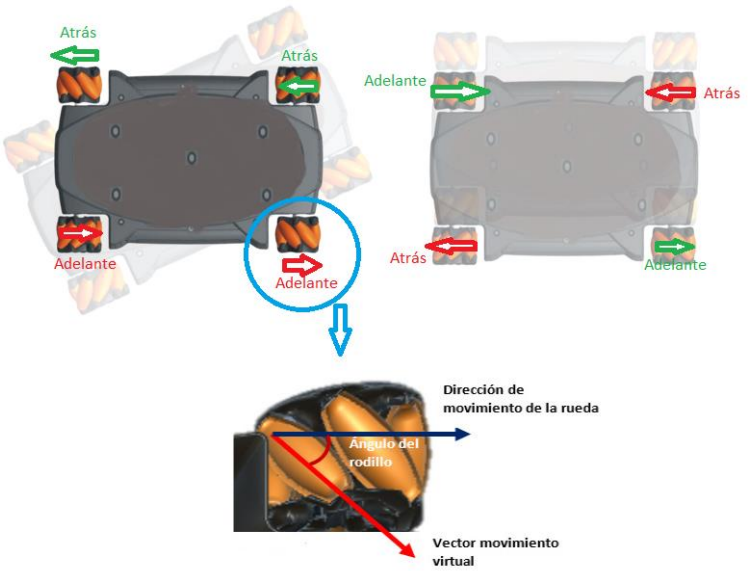

Figura A. 6. Movimiento de la plataforma móvil.
Elaborado por: El Investigador

$$\dot{x} = \frac{1}{4}(d1 + d2 + d3 + d4) \quad (5)$$

$$\dot{y} = \frac{1}{4}(-d1 + d2 + d3 - d4)\tan(\alpha_b) \quad (6)$$

$$\theta_b = \frac{1}{4}(d1 + d2 + d3 + d4)\beta \quad (7)$$

4.2. Modelo cinemático del brazo robótico

Para realizar el modelo cinemático del brazo se toma en cuenta el análisis realizado anteriormente de la cinemática directa e inversa, en donde se definió que la cinemática directa se divide en 3 métodos, en el cual el método geométrico tiene un problema

cuando el número de grados de libertad es mayor a 3 ya que encontrar una relación geométrica que involucre todos los elementos del brazo es muy complejo, lo que no lo hace práctico para robots industriales. El método de cuaternios se basa en una ampliación de los números complejos y se utiliza para calcular la orientación de un objeto en un espacio tridimensional. Por otro lado, la cinemática inversa resulta más complicado cuando se tiene varios grados de libertad y además puede existir múltiples soluciones para una misma posición y orientación del elemento terminal.

De acuerdo con Eileen Cardoso *et.al* [48] para el análisis de manipuladores robóticos resulta más práctico cuando se asignan marcos de referencia a cada eslabón para formar una cadena cinemática y el método fundamental para la asignación de marcos de referencia es la de Denavit Hartenberg (DH) el cual describe el robot en función de los parámetros estáticos de los eslabones y las variables de cada articulación, además que utiliza las matrices de transformación homogénea.

El brazo manipulador tiene 5 grados de libertad, por otro lado, la pinza de dos dedos es idóneo para la sujeción de pequeños elementos [45]. En la Figura A.7 se muestra el posicionamiento de los marcos de referencia para cada articulación.

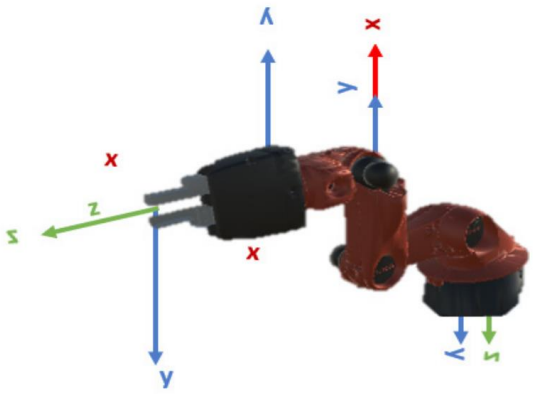

Figura A. 7. Marcos de referencia para las articulaciones [45].

En la Figura A.8 se observa las medidas del brazo robótico.

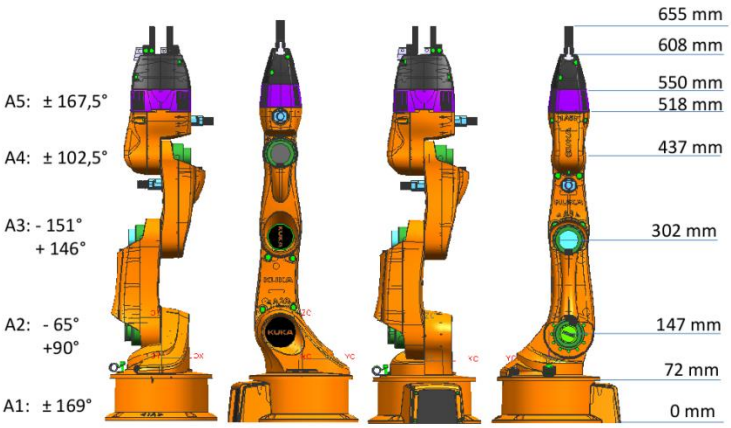

Figura A. 8. Medidas del brazo manipulador [45].

Para tener el modelo cinemático del brazo robótico se tiene en cuenta el algoritmo de Denavit-Hartenberg, teniendo en cuenta los marcos de referencia de cada articulación de la Figura A.7. Los parámetros de Denavit-Hartenberg se presentan en la Tabla A.3.

Tabla A. 3. Parámetros DH para el manipulador Kuka Youbot.

Eslabón	$\boldsymbol{\theta}$	**d**	**a**	$\boldsymbol{\alpha}$
1	2.94	0.147	0.0330	$\frac{\pi}{2}$
2	1.12	0	0.1550	0
3	-2.54	0	0.1350	0
4	3.35	0	0	$\frac{\pi}{2}$
5	2.92	0.2175	0	0

Elaborado por: El Investigador basado en [16]

En la Tabla A.4 se presenta las distancias y ángulos entre los marcos de referencia para el brazo Kuka Youbot.

Tabla A. 4. Cadena Cinemática para el brazo Kuka Youbot.

Articulación	**Marco Anterior**	**Traslación (cm)**			**Rotación (grados)**		
		x	**y**	**z**	**x**	**y**	**z**
Articulación 1	Base	2.4	0	11.5	180 °	0 °	0 °
Articulación 2	Articulación 1	3.3	0	0	90 °	0 °	−90 °
Articulación 3	Articulación 2	15.5	0	0	0 °	0 °	−90 °
Articulación 4	Articulación 3	0	13.5	0	0 °	0 °	0 °

Articulación 5	Articulación 4	0	11.36	0	−90 °	0 °	0 °
Pinza	Articulación 5	0	0	5.716	90 °	0 °	180 °

Elaborado por: El Investigador basado en [16]

De acuerdo con los parámetros de las Tablas A.3 y A.4 se saca las matrices de transformación homogénea, teniendo en cuenta la matriz de transformación homogénea de la ecuación 8.

$$A_i^{i-1} = \begin{pmatrix} C\theta_i & -S\theta_i C\alpha_i & S\theta_i C\alpha_i & a_i C\theta_i \\ S\theta_i & C\theta_i C\alpha_i & -C\theta_i S\alpha_i & a_i S\theta_i \\ 0 & -S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (8)$$

Los parámetros S y C corresponden a las funciones sin() y cos(). Finalmente, para obtener la matriz de transformación total se multiplica todas las matrices homogéneas como se muestra en la ecuación (9).

$$T_n^o = A_1^O \ldots\ldots A_n^{n-1} \qquad (9)$$

Para el cálculo de la matriz de transformación total se realiza un código en python que permite ingresar los parámetros de Denavit-Hartenberg que se mencionaron en la Tabla A.3, como se muestra en la Figura A.9

```python
import math
PI = math.pi

class YouBotDHParameters:
    """ Parametros de Denavit-Hartenberg del brazo kuka Youbot"""
    DH_A = (0.033, 0.155, 0.135, 0, 0)
    DH_ALPHA = (PI/2, 0, 0, PI/2, 0)
    DH_D = (0.147, 0, 0, 0, 0.218)
    DH_THETA = (PI*169.0/180.0, PI*65.0/180.0+PI/2, -PI*146.0/180.0, PI*102.5/180.0+PI/2, PI*167.5/180.0)

    def __init__(self):
        # TODO init from params?
        print('Error')
```

Figura A. 9. Parámetros de Denavit-Hartenberg para el cálculo de la cinemática directa.

Elaborado por: El Investigador

El diagrama de clase UML de la Figura A.10 muestra las clases del código desarrollado y está compuesto por dos clases “YouBotDHParameters” y “Kinematics”, las cuales son llamadas mediante la clase principal “Fkinematics”, es decir esta clase principal tiene una relación de composición que predomina sobre las demás clases y estas clases secundarias no funcionan sin la clase principal. La clase “Kinematics” también depende de la clase “YouBotDHParameters” ya que recoge los parámetros de Denavit-

Hartenberg mencionados en la Figura A.9, además esta clase está compuesta por los atributos de las matrices de transformación las cuales son del tipo traslación y rotación y los métodos que utiliza esta clase contienen el constructor de la clase, la función para establecer los parámetros DH y la función "forward" la cual es el encargado de calcular la matriz de transformación total. La clase "YouBotDHParameters" está compuesto por un solo método que es el constructor para la inicialización de la clase y tiene los atributos DH del tipo vector ya que estos valores se ingresan de esta manera para luego ser convertidos a matrices. La clase principal "Fkinematics" tiene los mismos atributos de la clase "Kinematics" ya que extrae los valores de este para imprimirlos por consola, además depende del paquete rospy las cuales definen las librerías correspondientes para él envió de datos al brazo robótico. Este paquete viene definido cuando se instala todos los paquetes de ROS.

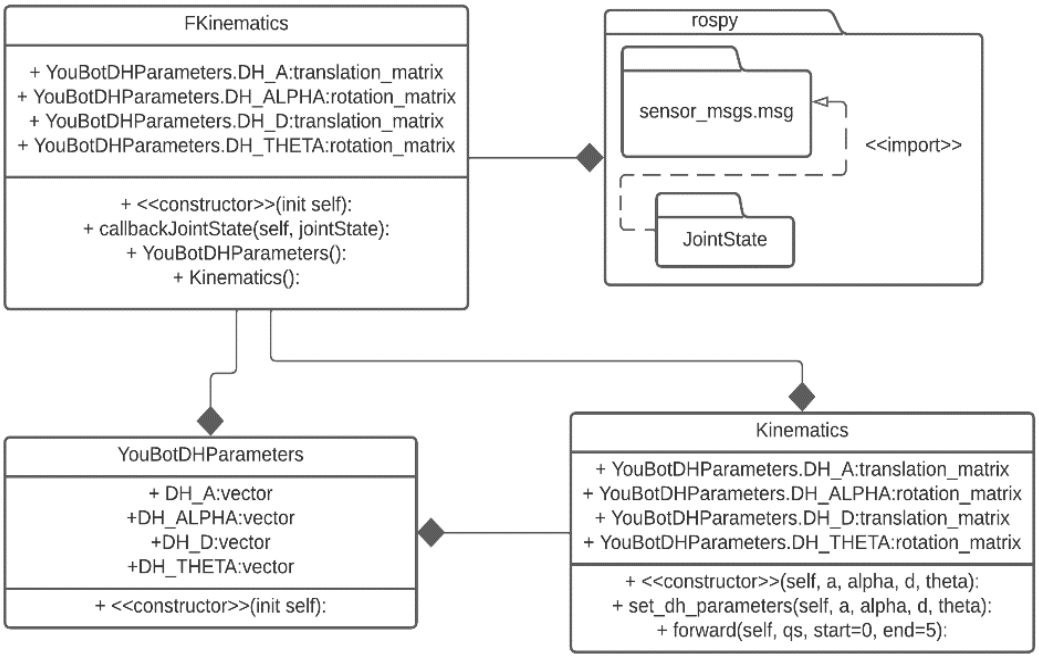

Figura A. 10. Diagrama de clase para el código de la cinemática directa del brazo robótico.
Elaborado por: El Investigador

Una vez ejecutado el código se muestra una interfaz en donde se encuentra el brazo robótico, en la cual se va cambiando los valores de posición de cada una de las articulaciones. Hay que tener en cuenta que los parámetros de la cinemática directa calculan la posición y orientación del extremo final del brazo robótico en base a un sistema de referencia que vendría siendo la plataforma y también depende de las

posiciones de los eslabones, es decir si es que las posiciones cambian, también cambia la posición y orientación del extremo final.

En la Figura A.11 se muestra el brazo robótico con la interfaz que permite ir cambiando los valores de posición de cada una de las articulaciones, además se muestra una etiqueta señalando la posición y orientación del extremo final calculado.

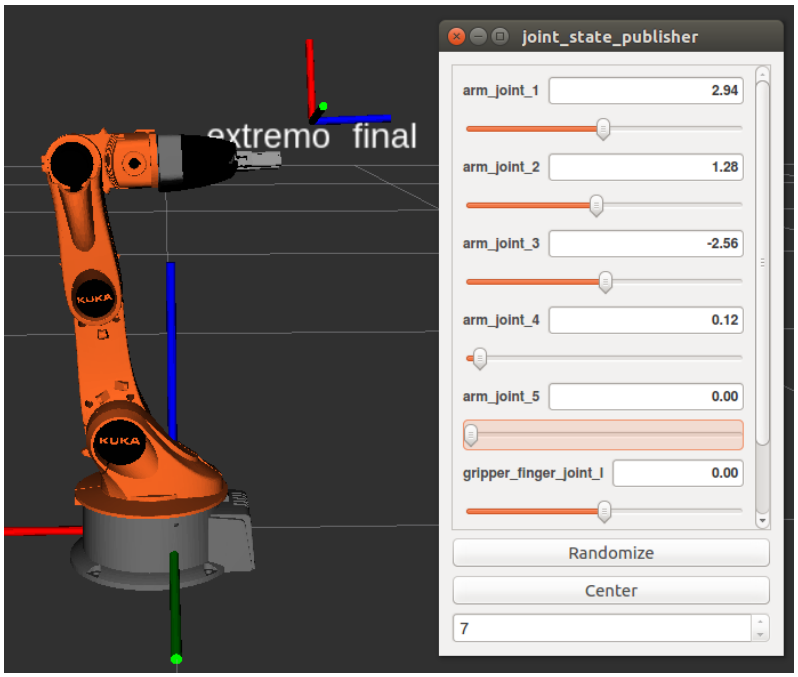

Figura A. 11. Interfaz del brazo robótico.
Elaborado por: El Investigador

En la ecuación 10 se muestra los resultados obtenidos de la cinemática directa en donde se muestra la matriz de transformación homogénea total calculada, y en la Tabla A.5 los datos de posición y orientación del extremo final del brazo robótico de acuerdo con las posiciones establecidas de la Figura A.11

$$T_n^o = \begin{pmatrix} 0.03 & -0.0041 & -0.99 & -0.14 \\ -0.21 & 0.97 & -0.01 & -0.001 \\ 0.97 & 0.21 & 0.036 & 0.44 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (10)$$

Tabla A. 5. Resultados obtenidos de la cinemática directa.

Posiciones eslabones	arm_joint_1	2.9372
	arm_joint_2	1.2772
	arm_joint_3	-2.5586
	arm_joint_4	0.1226
	arm_joint_5	0

Coordenadas posición extremo final	x	-0.1449
	y	-0.0017
	z	0.4422
Orientación extremo final	x	1.4023
	y	-1.3495
	z	-1.3939

Elaborado por: El Investigador

5. Análisis de resultados

Los resultados obtenidos del cálculo de la cinemática directa muestran que la posición y orientación del extremo final del brazo robótico van a depender de las posiciones en las que se encuentran cada uno de los eslabones del brazo considerando que se tiene una base (plataforma) la cual se usa como sistema de referencia para calcular los marcos de referencia de cada uno de los eslabones. Además, para obtener las matrices de transformación homogénea se toma en cuenta estos parámetros de los eslabones y las posiciones de las articulaciones teniendo en cuenta que estas posiciones están definidas en radianes ya que el robot trabaja en estas unidades. Los datos obtenidos de la cinemática de la posición indican que el extremo final se encuentra sobre un sistema de referencia tridimensional (x,y,z) y la orientación se define en ángulos de Euler o una representación alrededor de los ejes (x,y,z).

6. Conclusiones

- Los datos de posición y orientación del extremo final de la cinemática directa son importantes en varios aspectos relacionados con la operación y control del brazo robótico ya que varias aplicaciones dependen de esto como es el caso de la planificación de trayectorias en donde los datos de posición y orientación sirven para mover de manera precisa hacia una ubicación en especifica y los algoritmos de planificación de movimiento utilizan estos datos para calcular la secuencia de movimientos de las articulaciones y alcanzar el objetivo deseado. Además, hay que considerar que estos datos también son importantes para el caso de la teleoperación en zonas de difícil acceso en donde la persona usa estos datos de posición y orientación para garantizar que el brazo robótico se mueva dentro de los límites permitidos y evitar posibles riesgos.
- Los datos de la cinemática directa son indispensables para la verificación de colisiones, es decir con esto se verifica la presencia de posibles choques entre el brazo robótico y el entorno en donde se encuentra. Al conocer la posición y

orientación del extremo final se puede simular o evaluar si el brazo chocará con objetos cercanos o con su propio cuerpo durante un movimiento planificado.

Anexo B: Instalación de ROS

Proceso de Instalación de ROS

El proceso de instalación se lo hace de la siguiente manera:

1. Configurar el pc para aceptar la descarga de paquetes de software del tipo ros.org mediante el siguiente comando

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

2. Configurar las claves

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

3. Actualizar los paquetes de Debian: sudo apt-get update
4. Se recomienda hacer una instalación completa con el siguiente comando:

```
sudo apt-get install ros-kinetic-desktop-full
```

5. Configurar el ambiente de ROS mediante el bash

```
echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
source ~/.bashrc
```

6. Se instalan las dependencias necesarias de ROS con el comando:

```
sudo apt install python-rosdep python-rosinstall python-rosinstall-generator python-wstool build-essential
```

7. Antes de usar ROS hay que instalar varias herramientas las cuales son necesarias para correr varios componentes en ROS

```
sudo apt install python-rosdep
```

8. Una vez instalado se inicializa rosdep

```
sudo rosdep init
rosdep update
```

Proceso de instalación de los paquetes del Kuka Youbot

El proceso de instalación de los drivers y paquetes necesarios del robot, así como la compilación con ROS se lo realiza mediante los siguientes pasos:

1	Descargar los paquetes necesarios (Drivers Kuka)	sudo apt-get install ros-kinetic-youbot-driver ros-kinetic-pr2-msgs ros-kinetic-brics-actuator ros-kinetic-moveit git ros-kinetic-ros-control ros-kinetic-ros-controllers ros-kinetic-gazebo-ros-control ros-kinetic-joy
2	Creación del espacio de trabajo catkin	mkdir -p /catkin_ws/src cd catkin_ws catkin_make
3	Definir la fuente ROS	source /catkin_ws/devel/setup.bash
4	Clonar el repositorio de youbot	cd catkin_ws/src git clone -b kinetic –recursive https://github.com/ut-ims-robotics/youbot.git
5	Compilar el espacio de trabajo	cd catkin_ws catkin_make

Anexo C: Encuesta aplicada

UNIVERSIDAD TECNICA DE AMBATO

FACULTAD DE INGENIERIA EN SISTEMAS ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGENIERIA EN TELECOMUNICACIONES

Esta encuesta como parte de la investigación del proyecto de titulación "SISTEMA DE TELEOPERACIÓN CON REALIDAD VIRTUAL PARA LA MANIPULACIÓN DE UN ROBOT MÓVIL" está dirigido a los estudiantes de la carrera de Telecomunicaciones que estén dispuestos a utilizar unas gafas de realidad virtual para entender el funcionamiento y controlar un robot móvil.

Objetivo: Mediante esta encuesta se pretende evaluar el nivel de usabilidad de una interfaz en realidad virtual para la manipulación de un robot móvil, en la cual la persona va a sumergirse en una interfaz virtual y entender el funcionamiento y control de un robot.

Por favor seleccione su edad y género correspondiente.

Edad

- 18-25
- 26-35
- 36 en adelante

Genero

- Masculino
- Femenino

Cuestionario de preguntas

1. **¿Le gustaría utilizar este sistema con frecuencia?**
    - Totalmente en desacuerdo
    - En desacuerdo
    - Neutro
    - De acuerdo
    - Totalmente de acuerdo

2. **¿Encuentra el sistema innecesariamente complejo?**
    - Totalmente en desacuerdo
    - En desacuerdo
    - Neutro
    - De acuerdo
    - Totalmente de acuerdo
3. **¿El sistema es fácil de usar?**
    - Totalmente en desacuerdo
    - En desacuerdo
    - Neutro
    - De acuerdo
    - Totalmente de acuerdo
4. **¿Necesitaría el apoyo de un técnico para poder utilizar este sistema?**
    - Totalmente en desacuerdo
    - En desacuerdo
    - Neutro
    - De acuerdo
    - Totalmente de acuerdo
5. **¿Las diversas funciones de este sistema estaban bien integradas?**
    - Totalmente en desacuerdo
    - En desacuerdo
    - Neutro
    - De acuerdo
    - Totalmente de acuerdo
6. **¿Había demasiada inconsistencia en este sistema?**
    - Totalmente en desacuerdo
    - En desacuerdo
    - Neutro
    - De acuerdo
    - Totalmente de acuerdo

7. **¿La mayoría de los estudiantes aprendería a utilizar este sistema con rapidez?**
    - Totalmente en desacuerdo
    - En desacuerdo
    - Neutro
    - De acuerdo
    - Totalmente de acuerdo
8. **¿Encuentra el sistema muy complicado de usar?**
    - Totalmente en desacuerdo
    - En desacuerdo
    - Neutro
    - De acuerdo
    - Totalmente de acuerdo
9. **¿Se sintió muy seguro usando el sistema?**
    - Totalmente en desacuerdo
    - En desacuerdo
    - Neutro
    - De acuerdo
    - Totalmente de acuerdo
10. **¿Se necesita aprender muchas cosas antes de empezar a utilizar el sistema?**
    - Totalmente en desacuerdo
    - En desacuerdo
    - Neutro
    - De acuerdo
    - Totalmente de acuerdo

Anexo D: Prueba de normalidad

Dentro de esta prueba de normalidad existen la prueba de Kolmogorov-Smirnov y la prueba de Shapiro-Wilk. Ambas pruebas determinan si los datos se ajustan a una distribución normal, la diferencia está en el tamaño de las muestras. Shapiro-Wilk sirve para muestras menores o iguales a 50 y Kolmogorov-Smirnov para muestras mayores a 50.

Para realizar esto se tiene la hipótesis nula y la hipótesis alterna en donde se dice que:

- **Hipótesis nula:** Los datos tienen una distribución normal
- **Hipótesis alterna:** Los datos no tienen una distribución normal

Si $p<0.05$ se rechaza la hipótesis nula y se acepta la hipótesis alterna

Si $p>0.05$ se acepta la hipótesis nula y se rechaza la hipótesis alterna.

Para realizar el cálculo se determinó que la muestra de los datos es de 42 aprovechando el número de encuestados. Entonces como es menor a 50 se aplica la prueba Shapiro-Wilk.

Dentro de esto se plantea los siguientes datos:

- Nivel de confianza es 95%
- Nivel de significancia (alfa) 5

Proceso de cálculo

Dentro del software SPSS se tabula los datos de tiempo de ejecución obtenidos y en estadística descriptiva se procede al respectivo cálculo. Como primera instancia se obtiene un resumen del procesamiento de los casos.

Resumen de procesamiento de casos

	Casos					
	Válido		Perdidos		Total	
	N	Porcentaje	N	Porcentaje	N	Porcentaje
Tiempo_Ejecucion	42	100,0%	0	0,0%	42	100,0%

A continuación, se muestra una tabla con el cálculo de la media, la mediana, la varianza, la desviación estándar, entre otras cosas las cuales son necesarias para el cálculo.

Descriptivos

			Estadístico	Error estándar
Tiempo_Ejecucion	Media		,0104	,00045
	95% de intervalo de confianza para la media	Límite inferior	,0094	
		Límite superior	,0113	
	Media recortada al 5%		,0102	
	Mediana		,0102	
	Varianza		,000	
	Desv. estándar		,00295	
	Mínimo		,01	
	Máximo		,02	
	Rango		,01	
	Rango intercuartil		,00	
	Asimetría		,660	,365
	Curtosis		,478	,717

Finalmente se obtiene los resultados para la prueba en donde se indica que p=0.227 por lo tanto es >0.05 por lo que se acepta la hipótesis nula y se rechaza la hipótesis alterna, es decir que los datos tienden a una distribución normal.

Pruebas de normalidad

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Tiempo_Ejecucion	,079	42	,200*	,965	42	,227

*. Esto es un límite inferior de la significación verdadera.

a. Corrección de significación de Lilliefors