Escuela Superior Politécnica del Litoral

Facultad de Ingeniería en Mecánica y Ciencias de la Producción

Diseño de un sistema de teleoperación avanzado para el robot Hermes con integración de sensores LiDAR y cámaras de profundidad

INGE-2523

Proyecto Integrador

Previo la obtención del Título de:

Ingeniero en Mecatrónica

Presentado por:

Daniel Alberto Morán García

Giovanny William Mite Rodríguez

Guayaquil - Ecuador

Año: 2024

Dedicatoria

A Dios, a mis padres Luis y Marcela, mis hermanas, mis abuelitos, mis tíos y mis amigos.

A los ingenieros Nabih Pico, Christian Tutivén, y Carlos Saldarriaga, por su ayuda durante el desarrollo de este proyecto.

Daniel Moran García

Dedicatoria

El presente proyecto lo dedico en primer lugar a Dios por darme la fuerza y la sabiduría para poder afrontar con valentía mi vida académica.

A mi madre Pilar Rodríguez por haber sido mi apoyo fundamental y mi modelo a seguir a lo largo de mi carrera universitaria.

A mi padre Geovanny Mite por inculcarme desde pequeño la disciplina y el carácter para afrontar cualquier reto que se me cruce.

También dedico este trabajo a mi hermana Giovanna Mite por ser quien me ayudó a sentirme a gusto en mi etapa estudiantil y ser mi compañía en los primeros años de mi carrera.

Giovanny Mite

Agradecimientos

Quisiera agradecer primeramente a Dios por acompañarme y guiarme durante toda mi vida.

A mi papá Luis Morán por ser para mí un ejemplo de dedicación, esfuerzo y disciplina, y de quien supongo heredé mi gusto por los números.

A mi mamá Marcela García por apoyar mis sueños desde que era muy pequeño y amarme incondicionalmente.

A mis hermanas Mildred y Stefy por hacer la convivencia en casa más amena.

A Sonia, por alimentarme todos los días después de clases para que pueda estudiar.

A mis perros Kayla y Pulgarcito, y mis gatas Jeiza y Manchitas por quedarse despiertos conmigo mientras hacia este proyecto.

A mis abuelitos, tíos, y el resto de mi familia y amigos que han compartido conmigo a lo largo de los años.

Daniel Moran García

Agradecimientos

Mi más sincero agradecimiento a Dios por ser mi guía en este proyecto.

A mis padres, Geovanny Mite y Pilar Rodríguez, por su sacrificio y apoyo constante en cada proyecto y desafío que me ha tocado afrontar.

A mi perro Phoenix y a mis gatos Cachito y Manchita, por acompañarme en todas esas noches de desvelos y trabajo duro.

A mis amigos Luis Arroba, Luis Vanegas y Marcelo Sotomayor por siempre tener un momento para mi cuando más los he necesitado.

Finalmente, a todas mis amistades alrededor del mundo que me dejó Tennessee, que más que mis amigos, son mi familia.

Giovanny Mite

Declaración Expresa

Nosotros, Daniel Alberto Morán García y Giovanny William Mite Rodríguez acordamos y reconocemos que:

La titularidad de los derechos patrimoniales de autor (derechos de autor) del proyecto de graduación corresponderá al autor o autores, sin perjuicio de lo cual la ESPOL recibe en este acto una licencia gratuita de plazo indefinido para el uso no comercial y comercial de la obra con facultad de sublicenciar, incluyendo la autorización para su divulgación, así como para la creación y uso de obras derivadas. En el caso de usos comerciales se respetará el porcentaje de participación en beneficios que corresponda a favor del autor o autores.

La titularidad total y exclusiva sobre los derechos patrimoniales de patente de invención, modelo de utilidad, diseño industrial, secreto industrial, software o información no divulgada que corresponda o pueda corresponder respecto de cualquier investigación, desarrollo tecnológico o invención realizada por nosotros durante el desarrollo del proyecto de graduación, pertenecerán de forma total, exclusiva e indivisible a la ESPOL, sin perjuicio del porcentaje que nos corresponda de los beneficios económicos que la ESPOL reciba por la explotación de nuestra innovación, de ser el caso.

En los casos donde la Oficina de Transferencia de Resultados de Investigación (OTRI) de la ESPOL comunique a los autores que existe una innovación potencialmente patentable sobre los resultados del proyecto de graduación, no se realizará publicación o divulgación alguna, sin la autorización expresa y previa de la ESPOL.

Guayaquil, 21 de mayo del 2024.

Daniel Morán García

Giovanny Mite Rodríguez

Evaluadores

Firmado electrónicamente por:
CARLOS XAVIER
SALDARRIAGA MERO

Carlos Saldarriaga, Ph.D.

Profesor de Materia

Firmado electrónicamente por:
CHRISTIAN JAVIER
TUTIVEN GALVEZ

Christian Tutivén, Ph.D.

Tutor de proyecto

Resumen

En los últimos años, el uso de robot móviles ha incrementado tanto en la industria como en la realización de actividades cotidianas, aportando directamente al desarrollo sostenible. Sin embargo, las colisiones y atascos que el control autónomo de estos robots puede sufrir siguen siendo un desafío, lo cual vuelve indispensable el uso de un operador que permita el manejo correcto del robot en situaciones que puedan atentar contra la seguridad del robot. Por esta razón, en este proyecto se propuso el diseño de un sistema de teleoperación para un robot que se localiza en Corea del sur. Su objetivo es que, a través de la incorporación de diferentes sensores y cámaras de profundidad, se incorpore un sistema de alarmas sensoriales que permita al operador manejar al robot de manera más cómoda y priorizando la seguridad del robot. Para ello, se seleccionó un modelo de robot Hermes, el cual a través del software de desarrollo ROS, permita el manejo de las variables de interés del robot para el desarrollo de la interfaz gráfica que permitió al operador el correcto manejo del robot. Además, se implementó el diseño y construcción de un joystick que retroalimente sensorialmente la inestabilidad del robot. El desempeño de la síntesis entre el operador, el joystick y la interfaz gráfica nos permitió evidenciar un manejo eficiente de los tiempos de respuesta en un robot local con el robot de Corea del sur, obteniendo una diferencia de 2 segundos en tiempos de respuesta.

Palabras Clave: Teleoperación, Interfaz gráfica, ROS, Joystick.

Abstract

In recent years, the use of mobile robots has increased both in industry and in the performance of daily activities, directly contributing to sustainable development. However, the collisions and jams that the autonomous control of these robots can suffer continue to be a challenge, which makes the use of an operator essential to allow correct handling of the robot in situations that may threaten the safety of the robot. For this reason, in this project the design of a teleoperation system for a robot located in South Korea was proposed. Its objective is that, through the incorporation of different sensors and depth cameras, a sensory alarm system is incorporated that allows the operator to handle the robot in a more comfortable way and prioritizing the safety of the robot. For this, a Hermes robot model was selected, which through the ROS development software, allows the management of the variables of interest of the robot for the development of the graphical interface that allowed the operator to correctly manage the robot. In addition, the design and construction of a joystick that provides sensory feedback on the instability of the robot was implemented. The performance of the synthesis between the operator, the joystick and the graphical interface allowed us to demonstrate efficient management of response times in a local robot with the South Korean robot, obtaining a difference of 2 seconds in response times.

Keywords: Teleoperation, Graphical interface, ROS, Joystick.

Tabla de contenido

Abreviaturas

ESPOL	Escuela Superior Politécnica del Litoral
HID	Human Interface Devices
HTML	HyperText Markup Language
LiDAR	Light Detection and Ranging
PLA	Ácido poliláctico
ROS	Robot Operating System
SDK	Software Development Kit
SLAM	Simultaneous Localization And Mapping
ToF	Time of Flight

Simbología

m/s	Metros / Segundos
mm	Milímetros
N	Newton
RPM	Revoluciones Por Minuto
V	Voltios
w	Velocidad angular

Índice de Figuras

Índice de Tablas

Capítulo 1

1.1. Introducción

La robótica, como campo de estudio, ha evolucionado exponencialmente a partir de la segunda mitad del siglo XX, esta fue inicialmente concebida como una herramienta para automatizar tareas repetitivas en líneas de producción industriales, aunque con el pasar de los años, ha trascendido estas fronteras para abarcar un espectro increíblemente amplio de tecnologías y aplicaciones. Hoy, los robots no solo ensamblan automóviles o manejan productos en fábricas, sino que también exploran planetas, asisten en cirugías delicadas y facilitan la vida diaria mediante asistentes personales automatizados y robots de servicio.

Dentro de esta vasta gama de aplicaciones para la robótica, existen ciertas tareas, como la exploración, o los sistemas de riego por drones, que requieren controlar el robot a través de largas distancias, ya que no es factible que un operador humano esté presente en el entorno de trabajo, esto supone un reto más complejo. La teleoperación de robots surgió como la solución al problema, combinando la robótica y las telecomunicaciones de forma integral.

La teleoperación es el proceso mediante el cual se puede ejercer un control remoto de robots, comúnmente en entornos inaccesibles o peligrosos para los seres humanos. Esta se ha desarrollado como una solución fundamental para superar los límites físicos y los riesgos asociados al trabajo humano directo [1]. Esta ha jugado un papel muy importante en el desarrollo de la historia humana reciente, siendo partícipes de acontecimientos mundiales (**Figura 1.1**), como fue el marco de la pandemia global del 2020, donde el uso de robots teleoperados en el ámbito médico para chequeos de rutina y cirugías creció significativamente, debido al confinamiento inevitable [2]. Otro ejemplo son los Rovers marcianos, que han sido
instrumentales en el ámbito de la exploración espacial permitiendo a los científicos recopilar datos y analizar el terreno de Marte con una rapidez y eficiencia sin precedentes.

Figura 1.1

(a) Robot cirujano Da Vinci, utilizado en telecirugías

Nota. Imagen obtenida de [3]

(b) Robot de exploración espacial Rover en la superficie de Marte

Nota. Imagen obtenida de [4]

El robot Hermes, es una plataforma robótica móvil de código abierto (**Figura 1.2**), fabricado por la empresa china SLAMTEC, el cual destaca por su gran versatilidad de aplicaciones en tareas cotidianas. Fue diseñado principalmente para tareas de distribución, inspección y desinfección, por lo que en países de alto nivel de desarrollo su uso es común en bodegas retail y restaurantes inteligentes, entre otros [5]. Su diseño robusto le permite adaptarse a variados escenarios y evitar obstáculos de manera eficiente, asegurando operaciones seguras y eficaces. Al ser una plataforma de código abierto, posee la capacidad ser bastante personalizable, permitiendo la integración de diversos tipos de sensores como LiDAR, cámaras de profundidad y cámaras ToF, para mejorar la percepción del entorno según la aplicación lo requiera.

Figura 1.2.

Plataforma robótica Hermes

Nota. Imagen obtenida de [5]

Los sensores LiDAR y las cámaras de profundidad, son periféricos de gran impacto en la teleoperación de robots, ya que son especialmente útiles para generar mapas del entorno del robot [6]. Estos y otros tipos de sensores, como los sensores de luz, temperatura, ópticos, de velocidad y aceleración, han sido de gran importancia en los sistemas de teleoperación, ya que, por medio de ellos, el robot percibe e interactúa con el ambiente que le rodea. Los sensores LiDAR y las cámaras de profundidad, también han experimentado avances notables en los últimos años, que han permitido obtener datos cada vez más precisos y fieles mundo real, a una velocidad mayor. Esto se puede observar en la **Figura 1.3**, donde se ilustra la evolución que ha tenido la tecnología de sensores para localización y visión artificial.

Figura 1.3.

Evolución en el uso de sensores de proximidad y visión artificial

Nota. Imagen obtenida de [7]

En el marco de los objetivos de desarrollo sostenible (ODS) de las Naciones Unidas establecidos en la Agenda 2030, los sistemas de teleoperación avanzados se destacan por su contribución a la innovación industrial (ODS 9) y la promoción de trabajos decentes y crecimiento económico (ODS 8). A nivel macro, la adopción de robots puede contribuir significativamente al crecimiento económico y la competitividad de las diversas industrias, incluyendo la manufactura y la logística, debido a la capacidad de las maquinas aumentar la productividad y reducir los costos laborales [8]. Según proyecciones del Foro Económico Mundial, se estima que alrededor de 85 millones de empleos serán desplazados por la robótica y la automatización para el año 2025, pero en contraparte, 97 millones de empleos serán generados en el sector de robótica e IA, dando un balance positivo en el mercado laboral [9][10]. De esta forma, el desarrollo de sistemas de

teleoperación avanzados no solo garantiza la eficiencia operativa, sino que también impulsa el progreso tecnológico y el crecimiento económico.

En el presente trabajo, se plantea diseñar un sistema de teleoperación avanzado para el control de un robot Hermes, ubicado en Seúl, Corea del Sur desde Guayaquil, Ecuador. Se incluirá una fase de integración con sensores LiDAR y cámaras de profundidad. Este enfoque le permitirá al sistema proporcionar un reconocimiento preciso y detallado del entorno del robot, y le servirá al operador para tomar decisiones en cuanto al manejo del mismo. Además, para enriquecer aún más la experiencia del operador, se diseñará un Joystick con sistema de retroalimentación háptica, que mejorara la interacción del operador con el entorno, traduciendo los datos sensoriales emitidos por el robot, en respuestas que pueden ser captadas por el operador a través de algún sentido.

1.2. Descripción del problema

El uso de robots teleoperados está ganando popularidad globalmente, ofreciendo la capacidad de ejecutar tareas precisas y, a veces, peligrosas, a través de grandes distancias. Sin embargo, estos sistemas enfrentan desafíos significativos que comprometen su efectividad y seguridad. Uno principales problemas para el modelo del robot Hermes es la limitada visibilidad que ofrecen las cámaras estándar, cuyo de rango de visibilidad no cubre completamente el entorno operativo del robot, generando puntos ciegos que pueden resultar en colisiones o atascos.

A su vez, los sensores utilizados por el robot para percibir su entorno pueden ser propensos a errores o mal funcionamiento, lo que puede llevar a decisiones incorrectas o a una operación ineficiente.

Por ello, el desarrollo de un sistema de teleoperación para el robot Hermes, incluyendo sensores LiDAR y cámaras de profundidad, se presenta como una alternativa para atacar esta

problemática. La integración de sensores con el robot permitirá obtener datos acerca del entorno de este y afrontar la problemática de los puntos ciegos que poseen las cámaras, esto le permitiría al operador tener un control más eficiente y seguro, al tener información constante del entorno de trabajo del robot. Para ello se utilizará el framework de desarrollo Robot Operating System (ROS), una plataforma de software flexible orientada a robótica, que ofrece un conjunto de herramientas y librerías que nos ayudaran a integrar de manera robusta los componentes de hardware y software del sistema de teleoperación[11]. De la misma forma, se utilizará el kit de desarrollo de software open-source SLAMWARE SDK, creado por la compañía manufacturera del robot, que permite trabajar con funcionalidades de navegación y mapeo, y es compatible con ROS [12].

Además, para mejorar la experiencia del usuario, se diseñará un joystick con retroalimentación sensorial, que permitirá al operador conocer con mayor precisión el estado en su entorno, mediante estímulos sensoriales transmitidos por el joystick, según los datos de estabilidad y navegación del robot Hermes.

1.3. Justificación del problema

La teleoperación de robots es esencial en numerosos campos donde el trabajo humano directo es riesgoso o no viable. Esta versatilidad de aplicaciones los convierte en herramientas valiosas en múltiples sectores productivos, incluyendo la industria petrolera, la minería, y la gestión logística, o tareas hogareñas de servicios [13]. Por lo que mejorar su desempeño al abarcar un componente tan importante como la seguridad del robot en la navegación a través del uso de sensores, resulta muy beneficioso. La elaboración de este proyecto tendrá un impacto significativo en la industria de la robótica al abordar uno de los desafíos más críticos en la teleoperación de robots a largas distancias: la eliminación de puntos ciegos y la mejora de la experiencia de control

del usuario. Esto es un avance fundamental para aumentar la seguridad, precisión y eficiencia de las tareas de un robot teleoperado en terrenos complejos. Al abordar las limitaciones del rango de visión de los sensores, minimizar ángulos muertos y mejorar la fusión de datos sensoriales, es posible reducir significativamente el riesgo de choques, garantizando la seguridad tanto del robot como de su entorno, optimizando la operatividad de los robots teleoperados, además de proveer una visión a futuro de como el sistema propuesto influenciará positivamente en la teleoperación, y en aplicaciones que envuelvan a la industria a largo plazo.

De esta forma, el desarrollo de un sistema de teleoperación para el robot Hermes, con alarmas de proximidad y retroalimentación háptica, propone mejoras capaces de afrontar esta problemática, que no solo protegen la integridad del robot y su entorno, sino que también amplían las capacidades operativas en diversas aplicaciones industriales y de servicio.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar un sistema de teleoperación para el robot Hermes que incorpore sensores LiDAR y cámaras de profundidad, con el fin de mejorar el acceso a entornos difíciles y mejorar la ergonomía del operador.

1.4.2. Objetivos específicos

1. Analizar las condiciones específicas del entorno en el que operará el robot Hermes, con el fin de establecer los criterios técnicos y funcionales necesarios para la integración efectiva de sensores LiDAR y cámaras de profundidad.

2. Crear una interfaz de usuario para operar al robot Hermes de forma remota, posibilitando la observación y evaluación instantánea de los datos provenientes de los sensores LiDAR y cámaras de profundidad.

3. Implementar un sistema de alertas retroaliment2adas a partir de un sensor LiDAR para evitar posibles choques o daños en el robot.

4. Diseñar un joystick capaz de controlar al robot y retroalimentar las sensaciones superficiales al usuario mediante vibraciones que emule su entorno, asegurando una experiencia efectiva.

1.5. Marco teórico

1.5.1. Historia de la teleoperación y su evolución

La teleoperación de robots, se refiere a la capacidad un usuario para controlar un robot o máquina herramienta de forma remota, normalmente a través de largas distancias. Debido a esta característica, hoy es utilizada en diversos campos en la industria. La telerobótica, tuvo sus inicios en el año 1898, cuando el inventor serbio Nikola Tesla, dio los primeros destellos de progreso en este campo, al inventar un pequeño bote teleoperado controlado por ondas de radio, que fue presentando en la exposición de ingeniería eléctrica celebrada en Nueva York en dicho año[14]. A pesar del interés y los halagos de los presentes, no fue hasta medio siglo después que se popularizaron las ideas y esfuerzos por controlar maquinas sin utilizar cables. A mediados del siglo XX, en el año 1945 el ingeniero Raymond C. Goertz, desarrollo el primer manipulador maestro-esclavo, con la finalidad de maniobrar materiales radioactivos con mayor seguridad [15].

Figura 1.4

Raymond C. Goertz utilizando manipuladores electromecánicos

Nota. Imagen obtenida de [15]

A partir de este momento, múltiples aportes se han realizado en la rama de la telerobótica, permitiendo ejercer un control sobre los robots, de manera más eficiente a distancias cada vez más largas. En las últimas décadas, se han incorporado múltiples sensores para captar el estado del entorno del robot, se desarrolló la tecnología SLAM (Localización y mapeo simultáneos), y recientemente se han desarrollo algoritmos de control con inteligencia artificial, entre otros avances.

1.5.2. Elementos básicos de la teleoperación

Un sistema de teleoperación, se describe básicamente como el control que ejerce un operador humano, sobre un robot un robot esclavo de forma remota, este tipo de sistemas involucran de los siguientes elementos básicos:

Operador: También llamado teleoperador, es la persona que controla los movimientos del robot de manera remota, toma constantemente decisiones en cuando al accionar del robot según la retroalimentación que reciba de este.

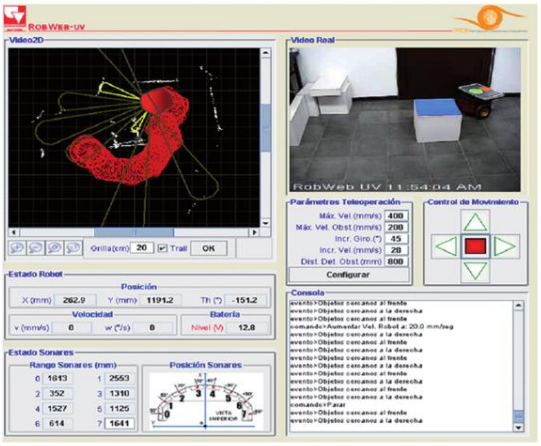
Robot teleoperado: También conocimiento como dispositivo esclavo, es el robot o manipulador robótico que se encuentra físicamente presente en el entorno de trabajo y refleja las acciones del operador para lograr algún objetivo determinado [16]**.**

Estación de teleoperación: Se define como el sistema que utiliza el operador para controlar al robot esclavo, normalmente consta de una computadora, a la cual se le conectan periféricos de entrada y salida para interactuar con la misma. [17].

Interfaz gráfica: La interfaz es el programa en el cual el operador puede observar de manera visual, la información relevante transmitida por el robot.

Figura 1.5.

Interfaz gráfica para un robot teleoperado



Nota. Imagen obtenida de [18]

Dispositivo de control: Es el dispositivo que interactúa directamente con el operador y transmite las acciones de este a la computadora maestra para ser enviadas al robot, puede ser un joystick, teclado, o directamente una pantalla táctil.

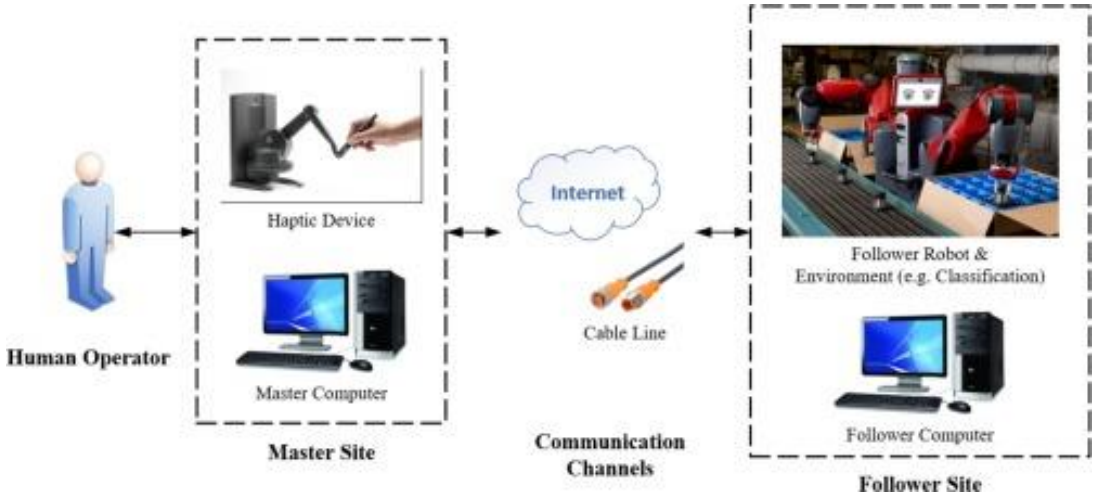
Canales de comunicación: Es el grupo de dispositivo que procesan y adaptan las señales transmitidas entre la estación de teleoperación y robot esclavo.

Computadora remota: También conocida con computadora esclava, es una unidad computacional adjunta al robot, recibe las órdenes del operador transmitidas por la estación de teleoperación y las ejecuta sobre los actuadores del robot

Sensores: Los sensores son dispositivos instalados en el robot capaces de medir variables físicas y de convertirlas a algún otro tipo de señal entendible por el robot, de esta forma le permiten al robot tener una percepción del entorno físico en el que se ubica [19].

Figura 1.6.

Elementos básicos de sistema de teleoperación



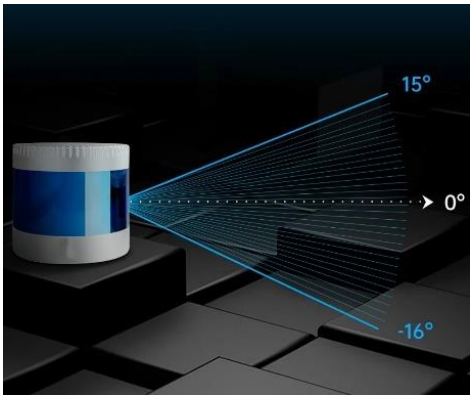
Nota. Imagen obtenida de [20]

1.5.3. Sensor LiDAR

Un sensor LiDAR (Light Detection and Ranging) es un dispositivo utilizado para medir distancias y generar mapas tridimensionales del entorno mediante el uso de pulsos láser. Funciona emitiendo pulsos de luz láser hacia un objeto o superficie y midiendo el tiempo que tarda en reflejarse la luz de vuelta al sensor. Utilizando esta información de tiempo de vuelo, el sensor puede calcular la distancia entre el sensor y el objeto.

Figura 1.7.

Funcionamiento básico del sensor LiDAR



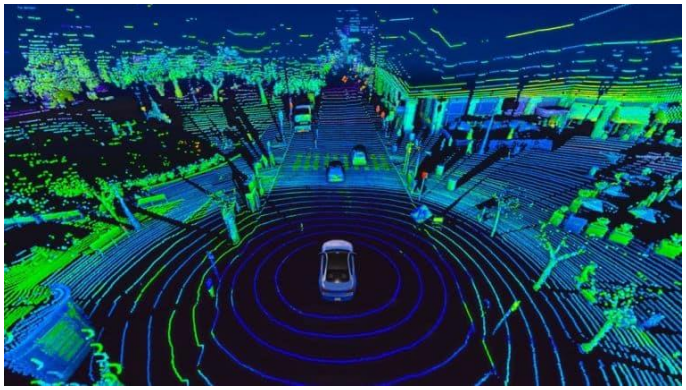
Nota. Imagen obtenida de [21]

1.5.4. Tecnología de Mapeo y Localización simultáneos (SLAM)

La tecnología de mapeo y localización simultáneos (SLAM, por sus siglas en inglés) es una técnica utilizada para mapear en ambiente en el que un robot se encuentra, al mismo tiempo que se localiza al robot dentro del mapa que está siendo construido. Esta tecnología, que involucra sensores y visión por computadora, ha llegado a revolucionar la industria de la robótica, haciendo posible creación de robots móviles autónomos [22]. Los sensores más utilizados en los algoritmos SLAM, son las cámaras y los sensores LiDAR, juntos pueden generar un mapa tridimensional bastante acertado del entorno del robot, lo cual es bastante optimo en un sistema de teleoperación.

Figura 1.8.

Representación visual de la tecnología SLAM aplicada en un vehículo



Nota. Imagen obtenida de [22]

1.5.5. HID

Un sistema HID (Human Interface Device) facilita la interacción entre usuarios y computadoras mediante dispositivos como teclados, ratones, joysticks, y gamepads. Estos dispositivos se comunican con la computadora a través de un estándar que permite transferir datos y controlar funciones, ejecutando acciones en el sistema operativo. Popularizados a finales de los años 90 con la expansión de dispositivos USB, los sistemas HID son ampliamente utilizados en computadoras, dispositivos móviles y consolas de videojuegos. La especificación USB HID, desarrollada por el USB Implementers Forum (USB-IF), estandariza esta comunicación a través de puertos USB.[14]

Figura 1.9.

Conexiones macho-hembra puerto USB

Nota. Imagen obtenida de [16].

1.5.6. Procesamiento de datos para un dispositivo HID

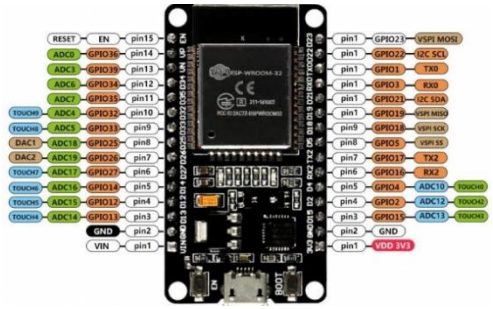
El procesamiento de datos para un dispositivo HID (Human Interface Device) implica la gestión de la información captada por el dispositivo de entrada y su conversión en comandos o acciones entendibles por la computadora o el sistema operativo. Este proceso se lleva a cabo dentro de un sistema informático.

Microcontrolador: Un microcontrolador es un circuito integrado especializado que integra un microprocesador central, memoria, periféricos de entrada/salida y otros componentes esenciales en un solo chip.

ESP32: El ESP32 es un microcontrolador de bajo costo y bajo consumo de energía que integra capacidades de conectividad Wi-Fi y Bluetooth. Es desarrollado por Espressif Systems, una empresa china de semiconductores, y es ampliamente utilizado en una variedad de aplicaciones IoT (Internet de las cosas).

Figura 1.10.

Microcontrolador ESP32 con señalización de pines



Nota. Imagen obtenida de [17].

1.5.7. Diseño electrónico de Joystick

El diseño electrónico de un joystick implica la integración de varios componentes para permitir la detección y transmisión de los movimientos del joystick a un microcontrolador u otro dispositivo electrónico como una computadora, una consola de videojuegos, un dron, un robot, entre otros. Entre estos componentes tenemos principalmente:

Mecánica del joystick: Se refiere al componente mecánico que permite al usuario realizar movimientos en diferentes direcciones. Puede tener un diseño de bola, palanca o tipo de vara, dependiendo de la aplicación y las preferencias del usuario.

Interruptores de botón: Los interruptores de botón se utilizan para detectar la presión del usuario sobre los botones del joystick. Pueden ser interruptores táctiles o interruptores mecánicos simples.

Microcontrolador o dispositivo receptor: El microcontrolador o el dispositivo receptor reciben las señales del joystick a través del circuito de interfaz y las utilizan para determinar la posición y el estado del joystick.

1.5.8. Sistema de retroalimentación háptica

Un sistema de retroalimentación háptica, conocido como haptic feedback en inglés, emplea estímulos táctiles o de fuerza para transmitir información al usuario, mejorando la interacción con dispositivos electrónicos. Esta retroalimentación puede involucrar vibraciones, sensaciones de fuerza, textura, temperatura y resistencia, entre otros efectos sensoriales.

Figura 1.11.

Motor utilizado para simular vibración en joysticks.

Nota. Imagen obtenida de [18]

Capítulo 2

2. Metodología

En el presente capitulo, se describe la metodología utilizada para la elaboración del sistema de teleoperación con un mando que conste con un sistema de retroalimentación sensorial, acorde a los objetivos propuestos. Se aborda la propuesta de varias alternativas de solución y el proceso para seleccionar la más adecuada acorde a la problemática, posteriormente, se presentan los conceptos utilizados y se presenta en profundidad el proceso de diseño de la solución.

2.1. Requerimientos de diseño

Previo a definir alternativas de solución, procederemos a describir los requerimientos de diseño para el sistema de teleoperación, el cual consta de dos componentes principales, la interfaz gráfica de teleoperación, y el joystick de retroalimentación háptica. Posteriormente se tomará cada uno de estos elementos como un componente individual de diseño, por lo tanto, se realizarán procesos individuales para seleccionar la mejor alternativa para cada elemento. Se plantean los siguientes requerimientos propuestos por el cliente:

Tabla 2.1

Requerimientos de diseño

Tipo de requerimiento	Descripción
Software	**Se** debe utilizar el framework de desarrollo ROS para elaborar toda la funcionalidad del sistema de teleoperación.
Funcionalidad	Se desea que el sistema de teleoperación pueda establecer una comunicación exitosa con el robot y recibir métricas de sensores.
Alarma de proximidad	Si el robot llega a una distancia menor a 15 cm de algún obstáculo, se debe emitir una alarma que será mostrada en la interfaz gráfica.
Hardware	Se debe incluir el uso de un sensor LiDAR para mapeo del ambiente donde se encuentra el robot, así como cámaras de profundidad para captar los vectores de dirección del robot-
Interfaz	En la interfaz se debe poder visualizar el video que captura la cámara del robot y el mapa que elabora el sensor LiDAR, también se deben visualizar indicadores de velocidad y dirección del robot
Joystick	El joystick diseñado debe permitir el control teleoperado del robot Hermes, y a su vez retroalimente sensorialmente la inestabilidad superficial del mismo
Compatibilidad del joystick	Compatible con entorno de desarrollo ROS, y posibilidad de recibimiento de datos en la red.
Modelado del Joystick	Proporcionar un modelo de un joystick considerando parámetros de diseño electrónico y de costos.
Controlador del Joystick	Se debe diseñar un controlador que permita el ajuste de una retroalimentación sensorial a partir de los parámetros de inclinación del robot.

Elemento 1. Interfaz gráfica de teleoperación

2.2. Selección de alternativas de solución

Se exponen 3 diferentes alternativas para afrontar el diseño de la interfaz, las cuales se detallan a continuación:

Alternativa 1: Desarrollar la interfaz gráfica utilizando la herramienta rqt Esta solución propone el uso de rqt, una framework propio de ROS, para crear interfaces personalizadas, este software permite implementar nuevos elementos en la interfaz en forma de plugins, es compatible con RViz y permite la integración de varios tipos elementos en la misma interfaz.

Alternativa 2: Desarrollar la interfaz gráfica utilizando pyQt con aplicación de escritorio. Se propone desarrollar una aplicación de escritorio nativa utilizando el lenguaje de programación Python y la herramienta de diseño de interfaces graficas PyQt, la cual permite crear interfaces personalizables y funcionales integrando las capacidades de Qt, una biblioteca de aplicaciones para interfaces graficas.

Alternativa 3: Desarrollar la interfaz gráfica utilizando una aplicación web. En esta alternativa, se propone realizar una interfaz ejecutada sobre una aplicación web, este enfoque, usaría herramientas de desarrollo web como HTML, CSS Y Javascript, para construir una solución altamente personalizable y compatible con protocolos de comunicación HTTP y Websockets.

2.3. Criterios de solución

Tiempo de desarrollo: Se refiere al tiempo que implicaría invertir en el desarrollo de la alternativa, se evalúa tomando en consideración los tiempos requeridos de entrega del proyecto

Estética: Se refiere a la apariencia visual de la solución que vera el usuario, se evaluará el criterio en función de la capacidad de la alternativa de solución para generar interfaces estéticamente agradables.

Baja latencia: Se desea que la opción seleccionada para el proyecto permita tener la menor latencia en la comunicación entre el robot y el operador

Mantenible: Es importante que el código de la solución sea fácil de mantener, apartándose a las nuevas innovaciones en los lenguajes de programación que surjan a lo largo del tiempo

Personalizable: Se espera que la solución elegida permita un nivel alto de personalización de las características de los diferentes elementos de la interfaz, como el tamaño, color, ubicación, entre otros.

Con los criterios de selección establecidos, se procede a determinar la ponderación de cada uno, como se presenta a continuación:

Tabla 2.2

Ponderación de los criterios de solución para el desarrollo de la interfaz de teleoperación

Baja Latencia > Personalizable > Tiempo de desarrollo > Mantenible > Estética							
Criterio	Tiempo de desarrollo	Estética	Baja latencia	Mantenible	Personalizable	$\sum +1$	Ponderación
Tiempo de desarrollo		1	0	0.5	0.5	3	0.21
Estética	0		0	0.5	0	0.5	0.04
Baja latencia	1	1		1	0.5	4.5	0.32
Mantenible	0.5	0.5	0		0	2	0.14
Personalizable	0.5	1	0.5	1		4	0.29
					Suma	14	1

Tabla 2.3

Evaluación de criterio, Baja Latencia

Alternativa 3 > Alternativa 1 > Alternativa 2					
Baja latencia	Alternativa 1	Alternativa 2	Alternativa 3	$\sum +1$	Ponderación
Alternativa 1		1	0	2	0.333
Alternativa 2	0		0	1	0.167
Alternativa 3	1	1		3	0.500
			Suma	6	1

Tabla 2.4

Evaluación de criterio, Personalizable

Alternativa 3 > Alternativa 2 > Alternativa 1					
Personalizable	Alternativa 1	Alternativa 2	Alternativa 3	$\sum +1$	Ponderación
Alternativa 1		0	0	1	0.167
Alternativa 2	1		0	2	0.333
Alternativa 3	1	1		3	0.500
			Suma	6	1

Tabla 2.5

Evaluación de criterio, Tiempo de desarrollo

Alternativa 1 > Alternativa 3 = Alternativa 2					
Tiempo de desarrollo	Alternativa 1	Alternativa 2	Alternativa 3	$\sum +1$	Ponderación
Alternativa 1		1	1	3	0.500
Alternativa 2	0		0.5	1.5	0.250
Alternativa 3	0	0.5		1.5	0.250
			Suma	6	1

Tabla 2.6

Evaluación de criterio, Mantenible

Alternativa 3 > Alternativa 1 = Alternativa 2					
Mantenible	Alternativa 1	Alternativa 2	Alternativa 3	$\sum +1$	Ponderación
Alternativa 1		0.5	0	1.5	0.250
Alternativa 2	0.5		0	1.5	0.250
Alternativa 3	1	1		3	0.500
			Suma	6	1

Tabla 2.7

Evaluación de criterio, Estética

Alternativa 3 > Alternativa 1 > Alternativa 2					
Estética	Alternativa 1	Alternativa 2	Alternativa 3	$\sum +1$	Ponderación
Alternativa 1		1	0	2	0.333
Alternativa 2	0		0	1	0.167
Alternativa 3	1	1		3	0.500
			Suma	6	1

Tras haber realizado la evaluación de cada criterio individualmente para cada una de las tres alternativas de solución propuestas, se puede obtener el ranking de desempeño por criterio de cada solución, que se muestra en las tablas 2.3-2.7 De esta forma, utilizando la ponderación de cada criterio calculada en la tabla 2.2, se puede calcular el valor del peso de cada criterio para cada alternativa, al sumar todos los pesos se obtiene el valor ponderado general de cada solución, y por lo tanto, sus prioridades.

Tabla 2.8

Matriz de decisión

Criterios						Resultados	
Alternativas	Baja Latencia	Personalizable	Tiempo de desarrollo	Mantenible	Estética	$\sum$	Prioridad
1	0.107	0.048	0.105	0.035	0.013	0.308	**2**
2	0.053	0.097	0.053	0.035	0.007	0.245	**3**
3	0.160	0.145	0.052	0.070	0.020	0.447	**1**

En la tabla 2.8 se observa la matriz de decisión, en la que se hicieron los cálculos antes descritos y se llegó a la conclusión de que la solución óptima para el desarrollo de la interfaz de teleoperación es la alternativa 3: Desarrollar la interfaz gráfica utilizando una aplicación web.

Elemento 2. Joystick de retroalimentación háptica

2.4. Selección de alternativas de solución

Se establecieron 3 propuestas de diseño a partir de los diferentes tipos de microcontroladores en el mercado y de los diferentes tipos de retroalimentaciones sensoriales que puede percibir el usuario, las cuales se detallarán en la siguiente sección:

Alternativa 1: Diseño de un joystick similar al Dualshock de SONY con un sistema de retroalimentación háptica por medio de motores vibradores usando un microcontrolador ESP32. El diseño del mando se centraría en 2 thumb joysticks y 8 pulsadores, a su vez 2 motores de baja potencia los cuales retroalimentarían cualquier sensación superficial por la cual el robot esté cursando.

Alternativa 2: Diseño de una palanca de mando con un sistema de retroalimentación visual por medio de leds usando un microcontrolador Arduino Micro. Esta alternativa se centraría en 1 joystick y 4 pulsadores de gatillo, y 4 leds secuenciales que comenzarían a encenderse dependiendo de la fuerza en la sensación superficial detectada del robot.

Alternativa 3: Diseño de un Arcade Gamepad con un sistema de retroalimentación auditivo por medio de buzzers usando un controlador Arduino UNO. Esta solución propone un diseño de un gamepad de Arcade, el cual se centraría en 1 joystick y 8 pulsadores, a su vez 1 buzzer el cual emitiría un pitido con mayor frecuencia dependiendo de los terrenos recorridos por el robot.

Criterios de solución

Para elegir la opción de solución más adecuada, se definieron diversos criterios. Algunos generales enfocándose principalmente a criterios de diseño como el costo, el tiempo de desarrollo y el aspecto estético, sin embargo, otros criterios son más específicos, vinculándose directamente a las necesidades de la solución como la capacidad de conectarse a la red, o los entornos de desarrollo más amigables para la aplicación. A continuación, se detalla cada uno de estos criterios.

Costo: Hace referencia al precio de las componentes tanto electrónicas como mecánicas que se usarían en el desarrollo de la propuesta, a su vez, el costo de un posible prototipo

Compatibilidad con bibliotecas: Hace referencia a la disponibilidad de bibliotecas para el microcontrolador a utilizar, y si estas a su vez pueden realizar la función deseada.

Capacidad de conexión a internet: Hace referencia a la capacidad de la placa microcontroladora de enviar o recibir datos por internet, puede incluir módulos de red, o añadirse módulos adicionales.

Tamaño: Hace referencia a la proporción entre microcontrolador con el joystick, si los componentes son indicados para el diseño.

Estética: Hace referencia al diseño mecánico del joystick, si se adapta de una manera cómoda al usuario y si el prototipo es fácil de entender y manejar.

Los criterios se ponderan de la siguiente manera:

Tabla 2.9

Ponderación de los criterios de solución para el diseño de un joystick con retroalimentación sensorial

Conexión a red > Compatibilidad > Costo > Tamaño > Estética

Criterio	Conexión a red	Compatibilidad	Costo	Tamaño	Estética	∑+1	Ponderación
Conexión a red		0.5	1	1	1	4.5	0.3
Compatibilidad	0.5		0.5	1	1	4	0.27
Costo	0	0.5		1	1	3.5	0.23
Tamaño	0	0	0		0.5	1.5	0.1
Estética	0	0	0	0.5		1.5	0.1
					Suma	15	1

Tabla 2.10

Evaluación de criterio, Conexión a red

Alternativa 1 > Alternativa 1 = Alternativa 2					
Conexión a red	Alternativa 1	Alternativa 2	Alternativa 3	∑+1	Ponderación
Alternativa 1		1	1	3	0.500
Alternativa 2	0		0.5	1.5	0.250
Alternativa 3	0	0.5		1.5	0.250
			Suma	6	1

Tabla 2.11

Evaluación de criterio, Compatibilidad

Alternativa 1 = Alternativa 2 > Alternativa 3					
Compatibilidad	Alternativa 1	Alternativa 2	Alternativa 3	∑+1	Ponderación
Alternativa 1		0.5	1	2.5	0.417
Alternativa 2	0.5		1	2.5	0.417
Alternativa 3	0	0		1	0.166
			Suma	6	1

Tabla 2.12

Evaluación de criterio, Costo

Alternativa 2 > Alternativa 3 > Alternativa 1					
Costo	Alternativa 1	Alternativa 2	Alternativa 3	∑+1	Ponderación
Alternativa 1		0	0.5	1.5	0.250
Alternativa 2	1		0.5	2.5	0.417
Alternativa 3	0.5	0.5		2	0.333
			Suma	6	1

Tabla 2.13

Evaluación de criterio, Tamaño

Alternativa 1 = Alternativa 2 > Alternativa 3					
Tamaño	Alternativa 1	Alternativa 2	Alternativa 3	∑+1	Ponderación
Alternativa 1		0.5	1	2.5	0.417
Alternativa 2	0.5		1	2.5	0.417
Alternativa 3	0	0		1	0.166
			Suma	6	1

Tabla 2.14

Evaluación de criterio, Estética

Alternativa 1 = Alternativa 2 = Alternativa 3					
Estética	Alternativa 1	Alternativa 2	Alternativa 3	∑+1	Ponderación
Alternativa 1		0.5	0.5	2	0.333
Alternativa 2	0.5		0.5	2	0.333
Alternativa 3	0.5	0.5		2	0.333
			Suma	6	1

Se realizó la evaluación de las 3 alternativas respecto a cada uno de los criterios determinantes de selección, con esto se pudo obtener numéricamente cual alternativa resaltará más en cierto criterio, y con ello dar paso a ver cuál alternativa tiene mejor desempeño para nuestra aplicación, con estos datos de las tablas, se podrá detallar una matriz de decisión la cual calculará el peso de cada alternativa.

Tabla 2.15

Matriz de decisión

	Criterios					Resultados	
Alternativas	Conexión a red	Compatibilidad	Costo	Tamaño	Estética	∑	Prioridad
1	0.100	0.083	0.050	0.083	0.066	0.387	**1**
2	0.050	0.083	0.083	0.083	0.066	0.365	**2**
3	0.050	0.033	0.066	0.033	0.066	0.248	**3**

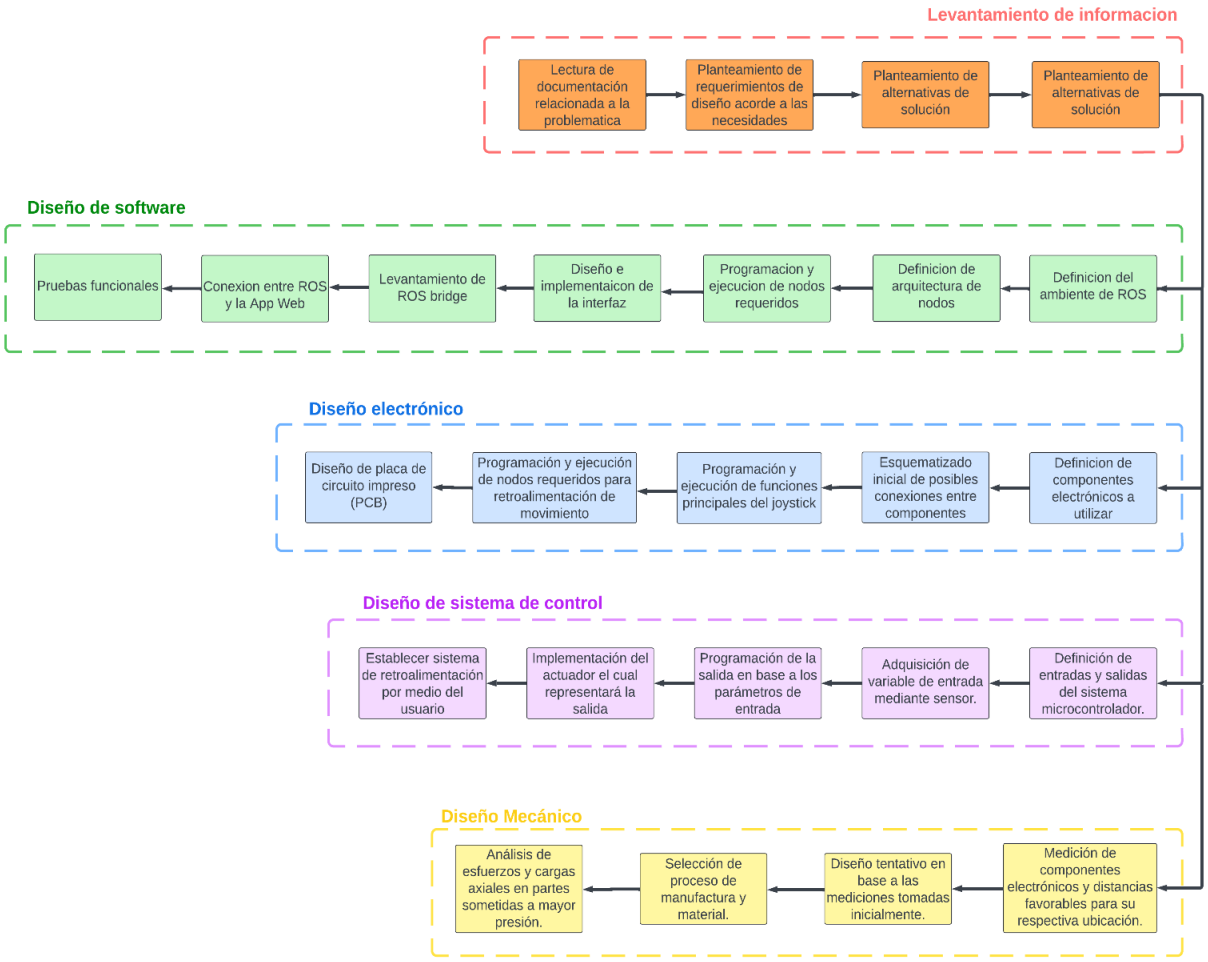
A partir de la matriz de decisión mostrada en la tabla 2.15, se puede observar que la Alternativa 1 tiene la mayor prioridad y, por lo tanto, se la escogió como solución a seguir para abordar el diseño del joystick. Se puede destacar que es la alternativa con menor costo de producción, y mayor conexión a red.

2.5. Proceso de diseño

Una vez que se determinó las soluciones óptimas para llevar a cabo este proyecto, se desarrolló el siguiente diagrama de flujo **(Figura 2.1),** en el cual se pueden visualizar las diferentes etapas a seguir para conseguir ejecutar las soluciones elegidas.

Figura 2.1

Flujograma del proceso de diseño



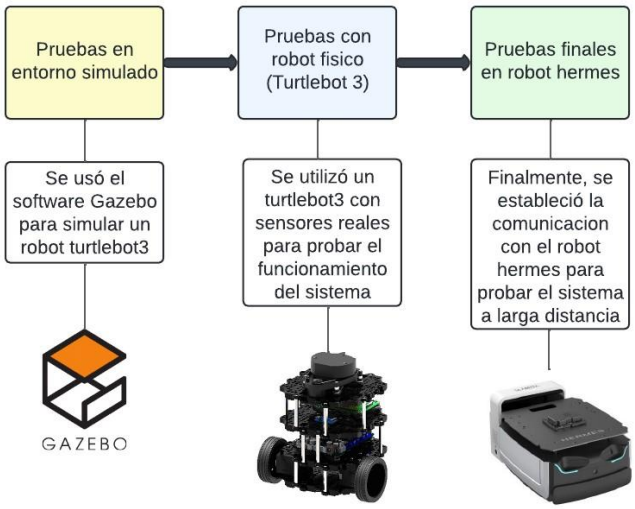
Tal y como se muestra en la figura 2.1 nuestro proceso de diseño comienza con un levantamiento de información necesario para el desarrollo de nuestra alternativa, a partir de este punto, el proceso se separa en 3 procesos principales, el diseño del software el cual se enfocará en el desarrollo de la interfaz gráfica y la obtención de variables obtenibles por nodos de ROS. El diseño electrónico, el cual se centra en buscar el microcontrolador óptimo a la aplicación, la

programación de este y su esquematizado eléctrico, y finalmente el diseño del sistema de control el cual prioriza la lógica del microcontrolador para la retroalimentación sensorial al usuario.

Cabe recalcar que, con la finalidad de testear la funcionalidad de la solución planteada, se realizaron las pruebas funcionales respectivas, sin embargo, estas pruebas estuvieron divididas en tres etapas. Las cuales se definen en la **figura 2.2.** Para empezar, se hicieron pruebas en un ambiente simulado, utilizando el software Gazebo, posteriormente se probó en un robot móvil Turtlebot3 Burger y finalmente se implementó la conexión y pruebas con el robot Hermes, para cual fue diseñada la solución.

Figura 2.2

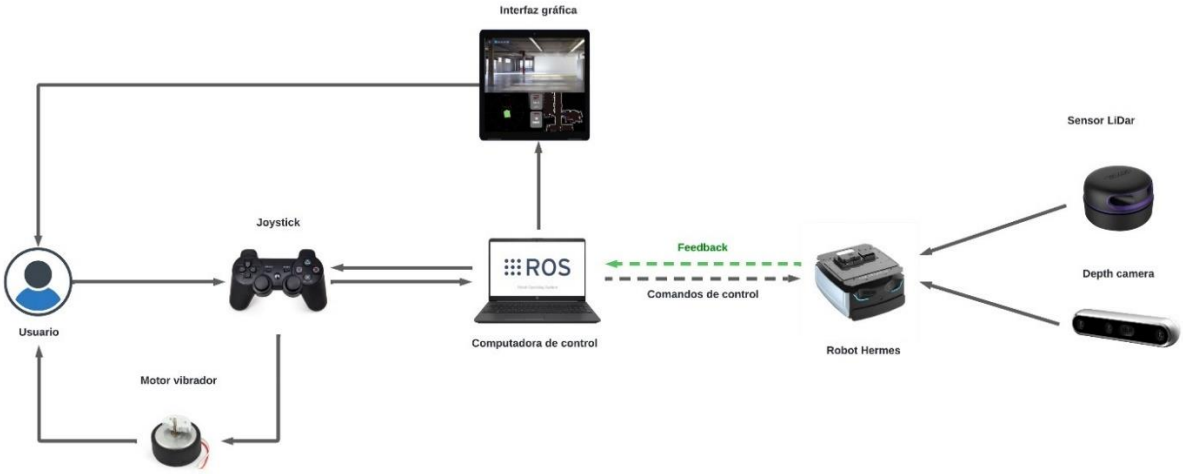
Etapas del proceso de pruebas



2.6. Diseño conceptual

Figura 2.3

Diseño conceptual del proyecto



En la **figura 2.3**, se plantea un diseño conceptual completo a la alternativa de solución escogida, donde se destaca el uso de los sensores para la realización de la interfaz gráfica del robot, como a su vez, el diseño del sistema HID para el control del robot con su respectiva retroalimentación sensorial. En esta figura, se puede apreciar las interacciones entre los diferentes elementos que participan en el sistema de teleoperación con retroalimentación háptica. Empieza por el lado del usuario, el cual por medio de un joystick puede enviar comandos de control hacia la computadora de control, presente físicamente junto al usuario, esta envía estos comandos al robot Hermes, a través de conexión a internet, para que sean ejecutados por este. El robot Hermes, contará con sensores LiDAR y cámaras de profundidad que le ayudaran a visualizar su interno, y transmitirá las señales captadas por estos sensores de vuelta a la computadora de control, a modo de feedback, En la computadora de control, se mostrará una interfaz gráfica, que mostrara los datos

de video, y de los sensores transmitidos desde el robot. El sistema incluye una alarma de proximidad, en caso de que la distancia del robot a un obstáculo sea menor a 15 cm, la computadora de control enviara una señal al joystick para que este vibre, esto se lograra por medio de un motor vibrador instalado dentro del joystick. De esta manera todos los componentes del sistema pueden interactuar armónicamente entre ellos.

2.7. Diseño de software

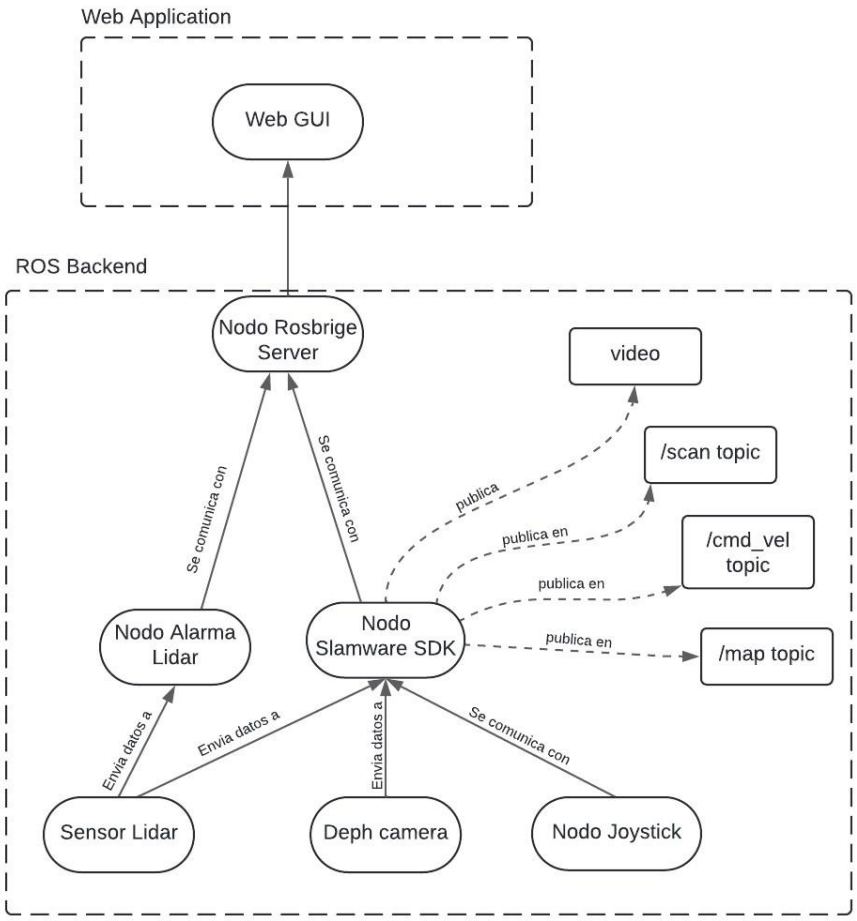
En esta sección se lista cada etapa ejecutada para desarrollar la de la solución planteada, es decir, la programación del sistema de teleoperación, abarcando desde la funcionalidad interna hasta la interfaz gráfica de usuario.

Definición del ambiente de ROS: Se define la versión de ROS con la cual se trabajará, para este proyecto se eligió la versión ROS 1 Noetic, compatible con el sistema operativo Ubuntu 20,.04 LTS. Se eligió esta versión de ROS por dos factores principales: su facilidad de integración con el SDK de Slamware, el cual se utiliza para interactuar con el robot Hermes, y debido a que cuenta con una gran cantidad de documentación y paquetes compatibles, por ser la última versión de ROS1.

Definición de arquitectura de nodos: El siguiente paso para desarrollar el sistema de teleoperación, consiste en definir las diferentes funcionalidades y procesos que lo conforman. De esta manera podemos encapsular cada funcionalidad como un nodo individual de ROS, y comunicar los distintos nodos entre ellos. Esto nos permitirá dividir toda la funcionalidad del sistema en una arquitectura de nodos interconectados, lo que nos facilitará la programación, entendimiento y mantenibilidad del código.

Figura 2.4

Diagrama de arquitectura de nodos del sistema de teleoperación



En la **figura 2.4**, se muestra el diagrama general de la arquitectura concebida los nodos a utilizar para el sistema de teleoperación del robot. Se puede observar cómo estos nodos se relacionan entre ellos, con los sensores periféricos, y con la interfaz web. Como vemos en el diagrama toda la funcionalidad del sistema se ejecuta en el backend de ROS, es decir, el procesamiento se hará nativamente en la computadora de control, mientras que la interfaz web,

actuará meramente como una herramienta visual que ayudará al operador a tomar decisiones sobre el control del robot.

Programación y ejecución de nodos requeridos: Una vez definidos funcionalidades a usar en los nodos, se procede con su implementación, para ello se utilizarán tanto nodos predefinidos propios de paquetes disponibles en ROS, como nodos creados desde cero para funcionalidades más específicas, para la programación de nodos se usará el lenguaje de programación Python.

Diseño e implementación de la interfaz: En esta fase del desarrollo se creará y se implementará el diseño estético de la interfaz web, así como la estructura de sus elementos. Para ello se utilizarán herramientas de desarrollo web, tales como HTML y CSS. De la misma forma se crearán las referencias para que tanto ROS como el usuario puedan interactuar con los elementos de la interfaz posteriormente, eso se hará mediante el uso del framework de Javascript Vue.js.

Levantamiento de ROS bridge: Durante esta etapa del desarrollo se establecerá la comunicación entre el motor de ROS nativo de la computadora de control y la interfaz web, para ello se hará uso de ROS bridge, que actuará como enlace entre estos dos elementos permitiendo el intercambio de información como comandos y feedback a través de los tópicos de ROS.

Configuración de conexión entre ROS y la App Web: Una vez levantado el puente de comunicación entre la app web y ros, se debe proceder a configurar esta conexión, esto implica definir los suscriptores y publicadores a los tópicos de ROS necesarios, y definir la relación de cada tópico con los elementos de la interfaz.

Pruebas funcionales: Finalmente una vez desarrollada la solución, se procede a realizar las pruebas de funcionamiento para poder evaluar el desempaño de la misma y detectar puntos de

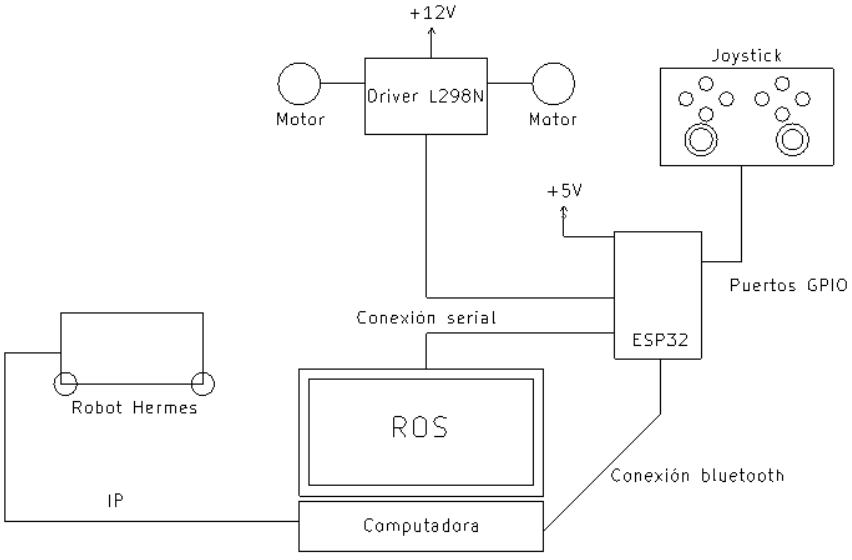
mejora. Como se explicó en el proceso de diseño, se procedió a realizar pruebas tanto simuladas, como en dos tipos diferentes de robots físicos, en este caso el trutlebot3 y la plataforma Hermes.

2.8. Diseño electrónico

Para el diseño electrónico, se deben tomar en consideración todas las componentes de mayor importancia utilizadas para el desarrollo de nuestra alternativa, elementos que destaquen principalmente por su funcionalidad indispensable, a continuación, se detallará el proceso esquemático de cómo debería funcionar el sistema.

Figura 2.5

Esquemático de procesos necesarios para el joystick.



Si bien el esquemático detalla los tipos de comunicación para la adquisición y envío de datos, es importante tener en cuenta que tipo de parámetros o características requerirá nuestro controlador con el fin de optar por el más indicado del mercado, se realizará una conexión por medio de bluetooth para reconocer un dispositivo de entrada como un nodo de ROS, el envío de datos del sensor se hará por internet, por ello se requerirá un microcontrolador con acceso a WIFI, a su vez, estos datos serán los encargados de ser filtrados y validar la activar del sistema de retroalimentación sensorial, finalmente la interfaz de usuario a utilizar deberá constar con entradas analógicas para que el usuario pueda hacer cambios graduales en la dirección y avance del robot al momento de controlarlo, teniendo en cuenta estos parámetros, se buscará los componentes más adecuados para cumplir la función de controlador, de entradas analógicas, digitales y del sistema de retroalimentación a utilizar.

2.9. Diseño mecánico

El diseño mecánico para la implementación del joystick será similar a los que se encuentran en el mercado actual, la ligera diferencia será la cantidad de botones operacionales con los cuales esta contará, el siguiente diseño propuesto muestra una parte superior la cual consta con espacios para los pulsadores y los joysticks, y una parte inferior la cual servirá de depósito para los motores vibradores y la placa de circuito impresa, el método de sujeción no permanente de estas 2 piezas, idealmente sería a partir del uso de tornillos M6 los cuales se adaptan de mejor manera a las dimensiones del joystick y sus componentes.

Figura 2.6

Parte superior del joystick

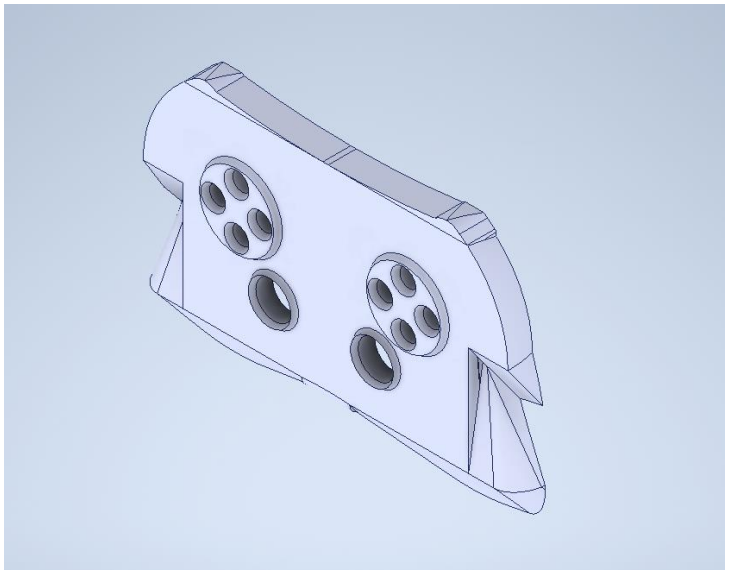


Figura 2.7

Parte inferior del joystick.

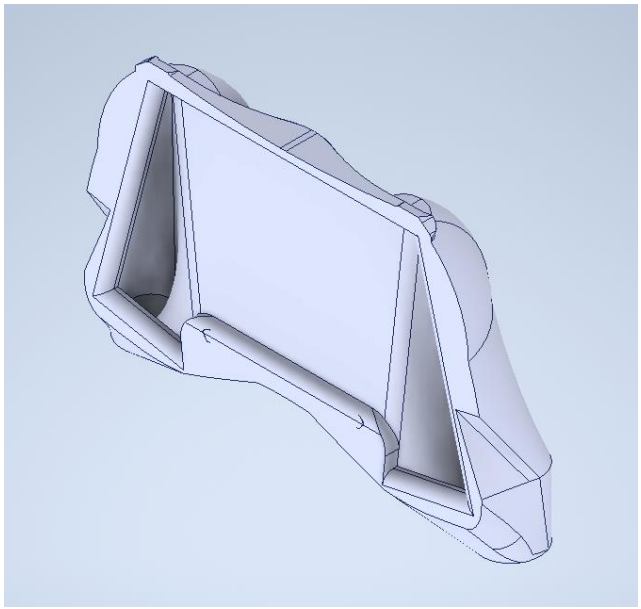


Figura 2.8

Diseño electrónico del joystick.

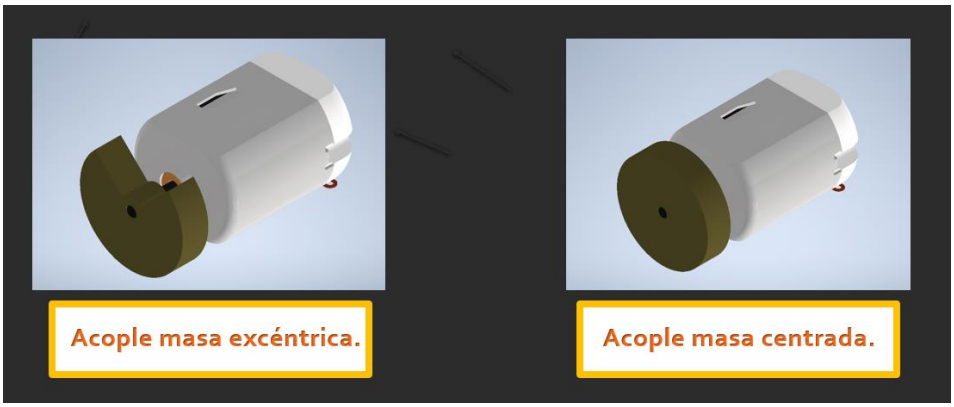
Si bien el diseño del joystick debe proporcionar al usuario comodidad al usarlo, y a su vez estabilidad ante las vibraciones del motor, el material más indicado para la implementación de la carcasa del joystick sería el PLA (*ácido poliláctico*) el cual mediante impresiones 3D nos permite la obtención de las partes del joystick a las medidas deseadas correspondientes directamente de la placa PCB, al no necesitar realmente un material con una resistencia a la fluencia alta o a cualquier tipo de comportamiento mecánico destructivo, podemos elegir este tipo de material que nos permite un mecanizado rápido y económico.

Por otro lado, tenemos el diseño para generar la vibración del motor, sabemos que la aplicación de una carga descentrada produce una inercia diferente a la conocida normalmente en un motor con un acople de disco, por ello se realizará un diseño de un acople de masa excéntrica

teniendo en consideración que tan fuerte deben ser las vibraciones en el mando, donde considerando diseños de mandos en el mercado actual, como el dualshock 4 propuesto por Sony, estos mandos vibran aproximadamente a los 0.1 N, y considerando la forma y las dimensiones acorde al motor a utilizar, se realizaran las estimaciones de masa para el acople.

Figura 2.9

Comparación entre masa excéntrica y masa centrada.



A través de la siguiente ecuación, se propone el cálculo de una fuerza de vibración equivalente a los 0.1 N, que gire a una velocidad de 200 rpm, un con un radio de giro de 1.55 mm, donde con la siguiente ecuación:

$$Fuerza\ centrífuga = mw^2r$$

Se obtiene una masa equivalente de 140 gramos, la cual se podrá diseñar con diferentes tipos de materiales propuestos en los resultados.

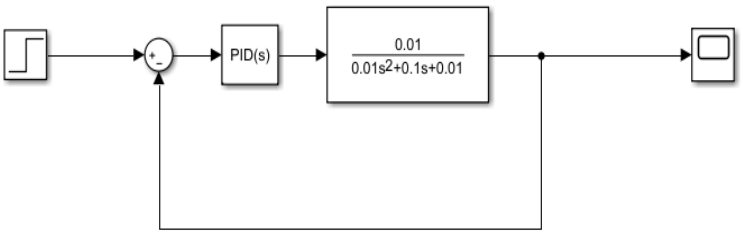
2.10. Sistema de control

Si bien el sistema propuesto en las alternativas de solución abarca una conversión del dato de entrada respecto al dato de salida de una manera escalar, en este caso siendo el aumento escalar

de la velocidad de los motores de vibración respecto al grado de inclinación del robot, al no existir una compensación para el grado de inclinación del robot, no se propone un sistema de control de lazo cerrado, por ello, se propone un modelo adicional para la compensación de la entrada manteniendo el valor de la velocidad del motor constante, donde tras varias pruebas, se detectó que la velocidad del motor práctica para una percepción sensorial adecuada es de 200 RPM, por ello se presenta el siguiente esquema en la figura 2.10.

Figura 2.10

Diseño del Sistema de control.



El sistema de lazo cerrado propuesto se centra en una entrada de voltaje de referencia para la activación del motor a un valor seteado para alcanzar una velocidad de estabilidad de 200 rpm, para la implementación de este sistema se realiza la colocación de una retroalimentación unitaria de velocidad a la entrada a través de un encoder o una herramienta de medición de velocidad, este valor de voltaje se tomará directamente del sensor de inclinación, el cual pasará por una razón para convertir este valor a niveles de voltaje manejables por el sistema de control propuesto.

La función de transferencia se fundamenta por el modelo matemático de un motor DC de baja potencia, el cual, tras realizar el respectivo cálculo, se obtiene la siguiente función:

$$\frac{\omega(s)}{V(s)} = \frac{K_t}{(Js + B)(R + Ls)(K_t * K_e)}$$

Esta función relaciona el comportamiento de la velocidad angular de salida en base al voltaje de entrada en el dominio de la frecuencia, las constantes son relacionadas directamente con las características del motor, siendo estas:

U(s) = Tensión aplicada.

R = Resistencia de armadura.

L = Inductancia de armadura

Ke = Constante de fuerza contraelectromotriz.

Kt = Constante de torque.

J = Momento de inercia.

B = Coeficiente de fricción viscosa.

w(s) = Velocidad angular.

Capítulo 3

3. Resultados y análisis

En el presente capitulo, se presentan los resultados obtenidos en las diferentes secciones de diseño, tomando en consideración el ajuste de piezas, los entornos de simulación utilizados, el diseño y parámetros considerados para la interfaz gráfica para finalmente estructurar la síntesis entre las componentes de diseño para la teleoperación del robot Hermes.

3.1. Selección de componentes electrónicas

Tabla 3.1

Características de microcontrolador ESP32 WROOM 32D

Figura	Características
	Voltaje: 3.0 V ~ 3.6 V Núcleo: ESP32-D0WD SPI flash: 32 MB Protocolos: WiFi Radio, Bluetooth LE Radio

Nota. La tabla de información sobre el controlador fue sacada de su respectiva hoja de datos.

Placa controladora que nos permitirá el manejo de las variables, y que a su vez interconecta las demás componentes, posee un total de 38 pines, donde 34 pines se reparten entre entradas tanto digitales como analógicas, a su vez permite la programación para el controlador y posee conexión a Wi-Fi y Bluetooth.

Tabla 3.2

Características de motores de baja potencia RF-300FA-12350

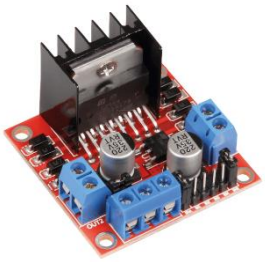
Figura	Características
	Voltaje nominal: 3.0 V ~ 5.9 V Velocidad: 2830 RPM Corriente: 0.093 A Torque: 0.48 Nm Datos máxima eficiencia

Nota. La tabla de información sobre el motor fue sacada de su respectiva hoja de datos.

Motor de baja potencia, se utilizarán 2 para recrear la moción del robot Hermes en ciertas superficies inestables, sus parámetros de máxima eficiencia se detallan en la tabla.

Tabla 3.3

Características de Driver L298N

Figura	Características
	Voltaje de habilitación: -0.3 V ~ 7 V Temperatura de funcionamiento de la unión: -25 °C ~ 130 °C Voltaje de alimentación: hasta 50 V Cantidad de salidas: 4

Nota. La tabla de información sobre el driver fue sacada de su respectiva hoja de datos.

Este controlador nos permitirá realizar la activación de 2 motores para el microcontrolador que estamos utilizando, debido al voltaje que se requiere para arrancar ambos motores, se utilizará una fuente externa, el driver L298N permite la activación de 4 diferentes salidas.

Tabla 3.4

Características del joystick

Figura	Características
	Voltaje: 3.3 V ~ 5 V Rango en X: 206 ~ 798 en sistema de coordenadas Rango en Y: 203 ~ 797 en sistema de coordenadas Incluye un pulsador

Nota. La tabla de información sobre el joystick fue sacada de su respectiva hoja de datos.

Mecanismo que nos permitirá recibir un valor analógico al cual podremos realizarle su respectivo mapeo, consta de 2 potenciómetros que cambian su valor conforme el eje x o y se muevan, a su vez, posee un pulsador para el envío de una señal digital pulsante.

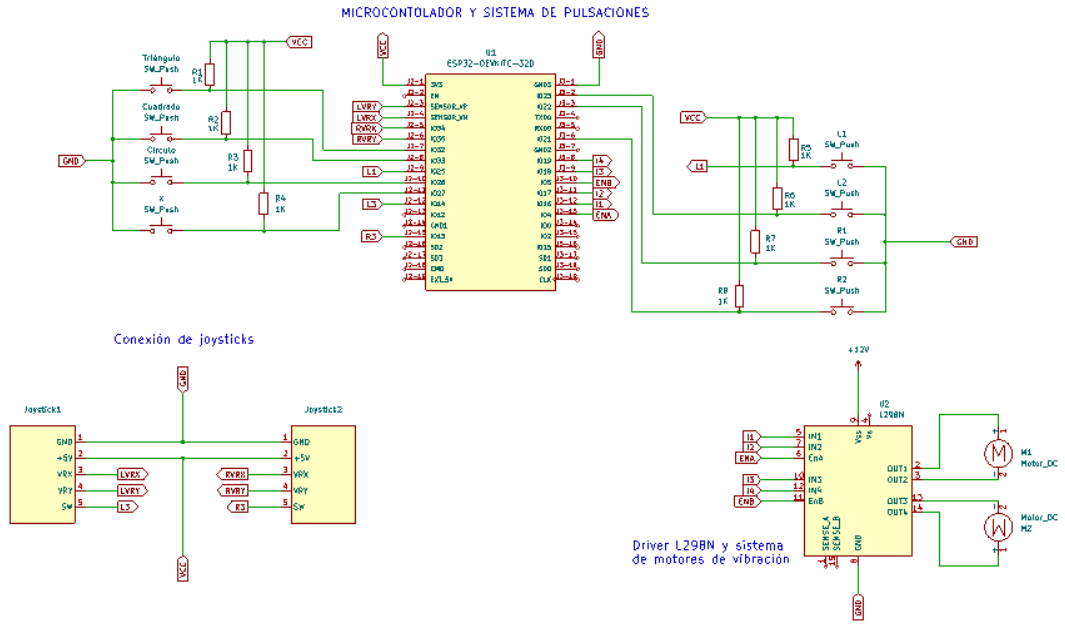
3.2. Diseño final de circuito eléctrico.

Para el diseño final eléctrico, se utilizarán las componentes seleccionadas anteriormente para cada una de las entradas en un modo analógico o digital que el microcontrolador nos proporciona, para el uso de los botones pulsadores, se tomarán los pines GPIO 21, GPIO 22, GPIO

23, GPIO 25, GPIO 26, GPIO 27, GPIO 32, GPIO 33, GPIO 14 y GPIO 16 como variables de entrada digitales para el microcontrolador, siendo respectivamente los pulsadores, estos se encontraran en una configuración pull-up con el fin de evitar posibles ruidos o fallos en reconocer el cambio de estado en los pulsadores, mientras que las entradas analógicas GPIO 34, GPIO 35, GPIO 36 y GPIO 39 serán vinculadas a cada eje de cada joystick.

Figura 3.1

Diseño electrónico del joystick.



El diagrama de entradas y salidas propuesto en la figura 3.1 nos presenta la inclusión del driver para el manejo de la velocidad de los motores de vibración, la conexión de este driver requiere una fuente externa, capaz de suministrar un voltaje en corriente continua de un máximo de 12VDC para la activación de ambos motores a la vez, a su vez constará con 6 pines que se conectaran de manera digital a los pines GPIO 4, GPIO 16, GPIO 17, GPIO 5, GPIO 18 y GPIO 19 para la activación del flujo de corriente, y de la polaridad de los motores, tras realizar las respectivas conexiones, se presenta la siguiente placa PCB.

Figura 3.2

Diseño de la placa PCB para la conexión de componentes (parte superior).

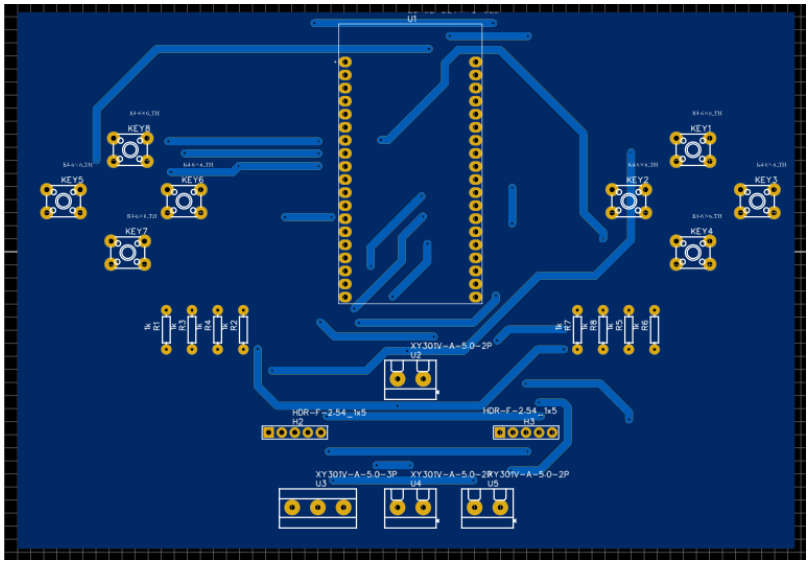
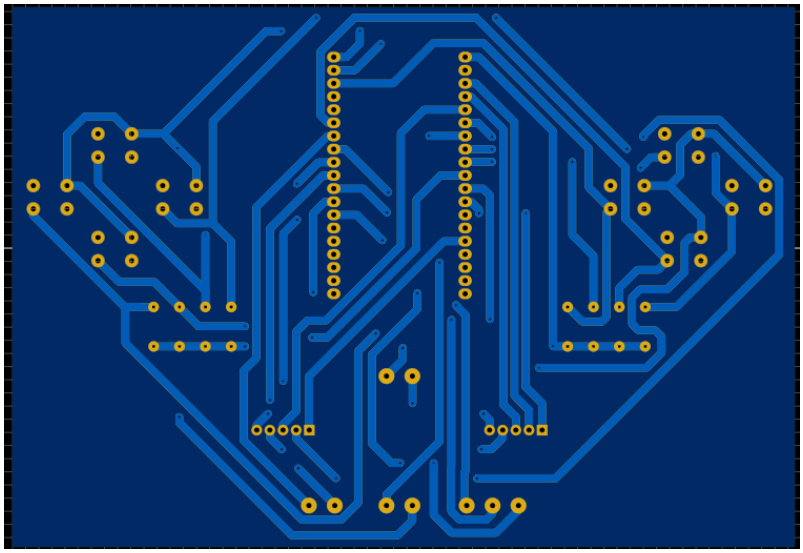


Figura 3.3

Diseño de la placa PCB para la conexión de componentes (parte inferior).



El siguiente diseño propone una síntesis entre todas las componentes electrónicas a utilizar en nuestro sistema, donde para el diseño de la placa se tuvo en consideración ciertos parámetros de diseño impuestos por el fabricante.

Tabla 3.5

Parámetros de diseño PCB

Parámetro	Medida mínima
Ancho de pista	1.5 mm
Ancho entre las pistas	0.5 mm
Diámetro de la vía	0.41 mm
Diámetro del agujero	0.35 mm

Tal como se muestra en la tabla 3.5, los parámetros de diseño impuestos por el fabricante justifican las dimensiones de la placa al tener mayoritariamente pistas distribuidas por cada uno de los pines que dan funcionalidad a los botones y a los joysticks, finalmente las dimensiones de la placa son de 151.638 mm X 104.521 mm.

3.3. Sistema de control planteado.

Se implementa el siguiente sistema de control basado en las características técnicas del motor a utilizar, considerado uno de baja potencia, el cual nos presenta la siguiente función de transferencia.

$$\frac{\omega(s)}{V(s)} = \frac{0.01}{0.01s^2 + 0.1s + 0.01}$$

Tras diferentes etapas de selección de constantes para el controlador PID a diseñar, y que tipo de respuesta estamos buscando, se consideraron las siguientes constantes:

Kp: 50.0

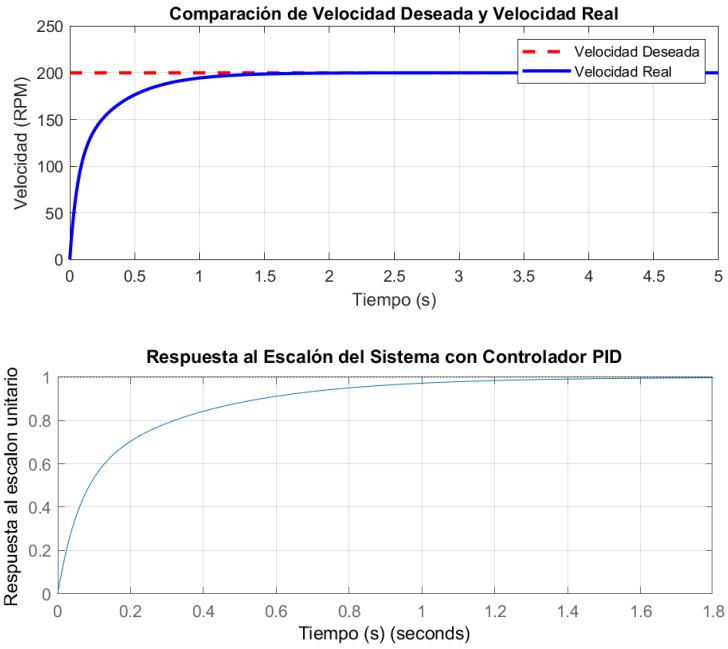
Ki: 5.0

Kd: 10.0

Estas constantes nos permitieron la obtención de la siguiente respuesta en el dominio del tiempo, donde tal como se muestra en la figura 3.4, se obtuvo una respuesta de carácter críticamente amortiguado, lo cual nos permite la obtención de un tiempo de respuesta pequeño, y permitiendo la activación del motor de una manera suave y sin tantas ondulaciones o variaciones en su velocidad, esto permitirá al usuario obtener una respuesta inmediata al cambio de los valores del sensor, y que sea constante en casos de activación de las alarmas de proximidad.

Figura 3.4

Respuesta del sistema de control ante una entrada escalón unitario.



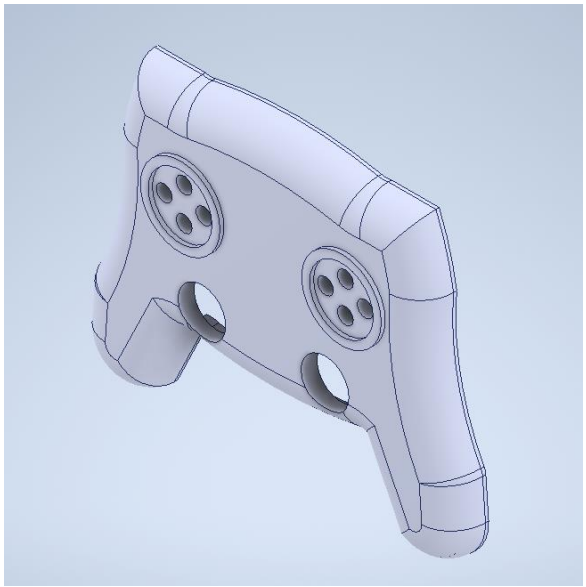
3.4. Diseño final de sistema mecánico.

Se realizaron mejoras constantes con el fin de una mejor dinámica y comodidad para el usuario, de tal manera que se respetarán los puntos de diseño clave destacadas en la metodología de diseño mecánico.

Para el diseño de la parte superior se ajustaron los agujeros de los botones de la placa y del joystick a las dimensiones de las componentes finales, para realizar la impresión de esta sección, de dividió a la mitad y se realizaron acoples para la conexión entre las partes.

Figura 3.5

Parte superior del diseño final, ajustado a las componentes de la placa.

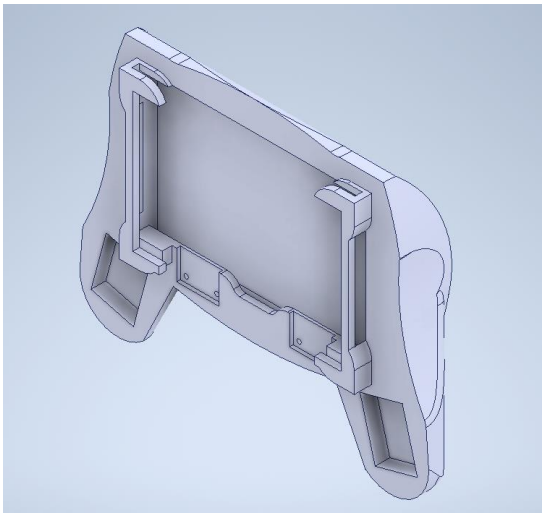


Para el diseño de la parte inferior, se tomaron en consideración unos seguidores para ajustar la placa en esta sección, esto permitió rigidez y estabilidad al manipular los botones, a su vez, en

esta parte se incluyeron los cajetines para la inclusión de los motores de vibración en las zonas de agarre del operador.

Figura 3.6

Parte inferior del diseño final, con acople para la placa, joysticks y motores.



Además, para una mayor rigidez, y como método de sujeción de la placa del joystick con la parte inferior, se implementaron unos cajetines que permiten el ajuste de la placa correctamente en la sección izquierda y derecha del joystick.

Figura 3.7

Sujeción de placa de joystick con tornillos M3

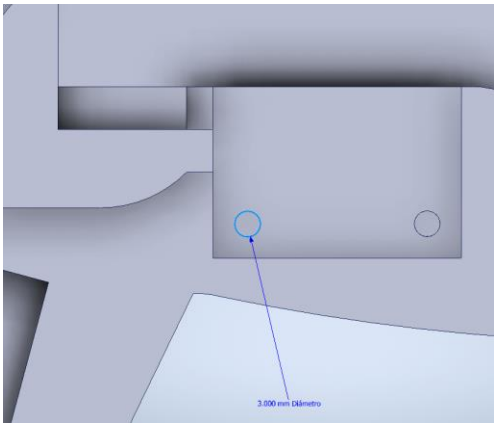
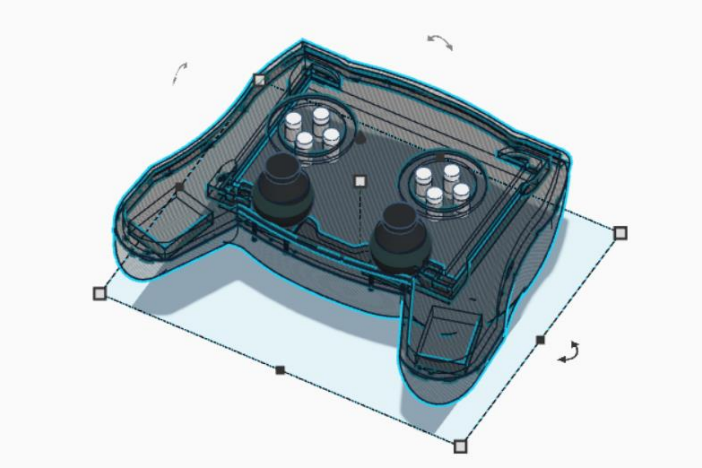


Figura 3.8

Vista interna de joystick.



Finalmente, se realizaron las impresiones 3D de las piezas por secciones, si bien las piezas se diseñaron completas, por motivos de duración de la impresión y principalmente el tamaño de impresión, se decidió realizar un plano cortante por cada pieza, para seccionarla en 2, a su vez se agregaron acoples entre piezas para evitar separaciones o inestabilidades entre las piezas impresas.

Figura 3.9

Impresión de la parte inferior, sección izquierda

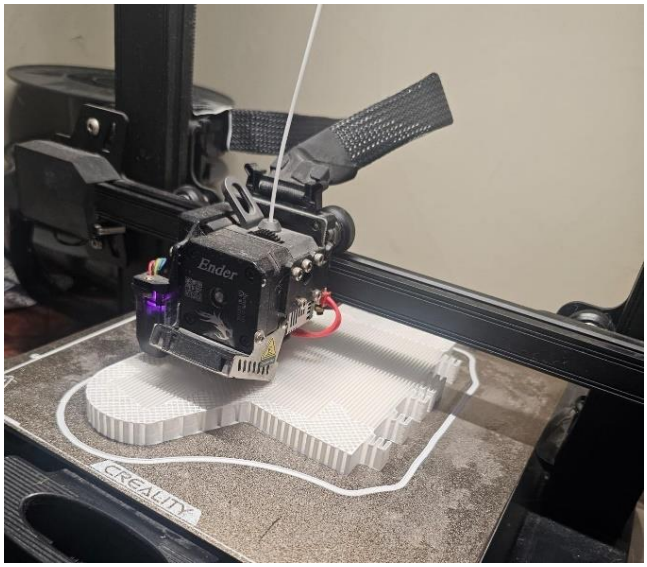


Figura 3.10

Cajetín para PCB y motores.

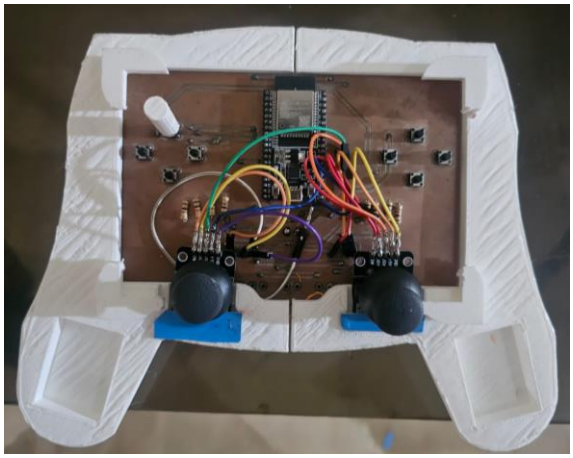
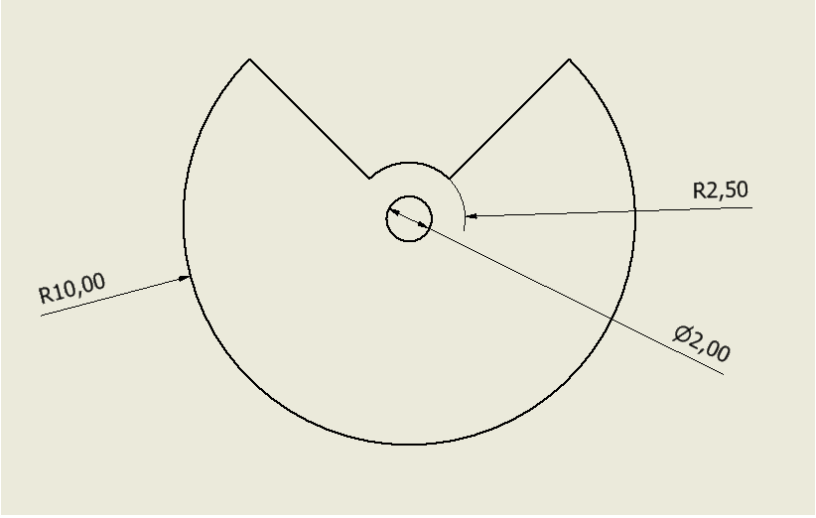


Figura 3.11

Prototipo final.

Figura 3.12

Dimensiones transversales del acople en mm.



Se diseño un acople a dimensiones respectivas para el motor DC, estas dimensiones permitieron el cálculo del centroide, el cual salió a un desplazamiento de 1.55 mm en x respecto al centro del eje de giro, utilizando la siguiente fórmula.

$$Rtotal = Arestante \cdot (0,0) + Atrapecio \cdot (6,0) / Arestante + Atrapecio$$

Se obtuvo una masa de 140 gramos, la cual se propone realizar el proceso por mecanizado para realizar el acople en el motor, debido a que las dimensiones no permiten llegar al peso ideal con algún proceso de manufactura aditiva, sin embargo, se pueden explorar técnicas de modelado en resina, para alcanzar la masa deseada.

3.5. Funcionalidad del sistema de teleoperación

Para empezar a elaborar el sistema de teleoperación, se realizó la instalación de ROS 1 Noetic, en una laptop con sistema operativo Ubuntu 20.04 Focal Fossa. Una vez completada la instalación de ROS, se procedió a crear y compilar un espacio de trabajo (WORKSPACE) utilizando un nuevo directorio y el comando de compilación catkin_make, en este workspace se alojó todo el código del proyecto. Como se mencionó en capítulos anteriores, nuestro sistema debe ser capaz de cumplir con algunos requerimientos, los cuales son:

- El robot debe poder ser controlado utilizando un periférico, en este caso joystick, y el mismo debe ser reconocible por el sistema ROS.
- El sistema debe emitir alarmas de proximidad cuando el robot se ubique a menos de 15 cm de algún obstáculo, entiéndase por obstáculo cualquier cuerpo que no sea el mismo robot.
- El sistema debe poder recibir video de la cámara del robot, así como datos de posición y velocidad.

- El sistema debe emitir alarmas de desestabilización, cuando el sistema de referencia del robot haya girado más de 15 grados en el eje Y, esto advertirá acerca de irregularidades en el terreno.
- Se debe construir un mapa del ambiente de robot a partir de las medidas tomadas por el sensor LiDAR.
- Toda la información recolectada por el sistema (Video de cámara, mapa del LiDAR, etc) debe ser consolidada en una interfaz gráfica para uso del operador.

  Una vez definidas las funcionalidades, se procedió a programar los nodos requeridos.

3.6. Control por joystick

Para realizar la conexión del Joystick hacia ROS, se verifico que este pueda ser reconocido por el sistema Linux, una vez que el sistema fue capaz de reconocer el mando, se iniciaron dos nodos de ROS para interactuar con el mismo, estos nodos son los siguientes:

Nodo Joy: Este nodo se utiliza para captar las entradas de los diferentes botones y ejes del joystick y publicar sus valores en el tópico /joy desde este tópico, otros nodos pueden suscribirse, leer estos valores, y ejecutar algún proceso con ellos les convenga

Nodo Teleop_Twist_Joy:_ Este nodo sirve para convertir las entradas del joystick en comandos de velocidad, en el ecosistema ROS, los comandos tienen un tipo de mensaje denominado Twist, este nodo lee los mensajes que llegan al tópico /joy, los convierte en mensajes de tipo Twist y los publica en el tópico /cmd_vel, este nodo es configurable y puede recibir varios parámetros para configurar la forma en la que procesa los mensajes del tópico /Joy, por ejemplo, se pueden definir que ejes del joystick definirán la velocidad lineal y cuales definirán la angular,

además, se define el factor de sensibilidad del joystick, es un parámetro proporcional al aumento de velocidad del joystick conforme se mueve el eje del Joystick-

Una vez iniciados los dos nodos necesarios para ejecutar el control por joystick, se procedió a realizar pruebas para verificar que, en efecto, se estén publicando correctamente y con sincronía las entradas del Joystick en el tópico /Joy y posteriormente ser convertidas a mensajes twist en el tópico /cmd_vel

3.7. Alarmas de proximidad

Esta funcionalidad se implementó utilizo un nodo de ROS programado desde cero, el nodo funciona de la siguiente manera:

Nodo LiDAR_Alarm: Este nodo se subscribe al tópico /scan en el cual se publican todas las lecturas que enviar en sensor LiDAR conectado al robot, el nodo toma estas lecturas y las procesa para obtener la mínima distancia hacia un objeto, si esta distancia es menor al umbral de 15cm, el nodo emite una alarma publicando un arreglo de dos elementos en el tópico /lidar_alarm, estos elementos son un valor booleano que indica si la alarma esta activa o no, y un valor flotante que contiene la distancia más cercana al obstáculo.

3.8. Alarmas de desestabilización

Para crear esta alarma se utilizó un nodo programado desde cero al igual que para el caso anterior, el objetivo de esta alarma es dar aviso cuando la desviación en el eje Y sea mayor a 15 grados. Para lograr esto se utilizaron los mensajes que llegan al tópico /odom de ROS. El tópico /odom sirve para proporcionar la estimación de la posición y orientación del robot o dispositivo al que está conectada. Específicamente, este tópico publica datos sobre la **odometría**, que incluyen:

Posición: Las coordenadas espaciales (x, y, z) del dispositivo o robot en relación con un marco de referencia inicial.

Orientación: Representada mediante cuaterniones, indicando la rotación del dispositivo o robot en el espacio.

Velocidades: La velocidad lineal y angular del dispositivo en los tres ejes.

Una vez conociendo la información que este tópico entrega, se procedió a definir el siguiente nodo:

Nodo Euler_Angle: Este nodo toma la orientación del robot que el tópico /odom brinda representada en quaterniones y la transforma a valores en ángulos de Euler, los ángulos de Euler son 3 ángulos conocidos como roll, pitch y yaw, que representan las rotaciones de un cuerpo en el eje X, Y y Z respectivamente. Si el valor de pitch, ese decir el valor de rotación en el eje Y, es mayor a 15 grados, se emite una alarma y el nodo publica un arreglo de dos elementos en el tópico /Euler_alarm, este arreglo contiene un valor booleano que indica si la alarma esta activa o no, y el valor medido del ángulo Pitch. Este nodo también publica un vector de 3 elementos en el tópico /x, este vector contiene los valores en grados sexagesimales de los ángulos de Euler roll, pitch y yaw.

3.9. Video de la cámara del robot:

El video de la cámara se obtuvo utilizando una cámara de profundidad colocada en el robot, esta se conectó a ROS mediante los mediante los drivers correctos desarrollados por el fabricante, en este proyecto se utilizó la cámara Inter Realsense T265, la cual cuenta con el SDK de Intel para iniciar un nodo de comunicación con la cámara, esta cámara publica video en el tópico /camera/image_raw, y a partir de aquí, se pueden usar diferentes métodos para visualizar el video,

en este proyecto se utilizó RVIZ y posteriormente herramientas de desarrollo web para visualizar el video en un navegador.

3.10. Mapa del ambiente (SLAM):

El mapa del ambiente se creó a partir de las mediciones del sensor LiDAR publicadas en el tópico /scan, con estos valores, se utilizó un nodo predefinido capaz de crear mapas del ambiente, representados como mensajes de tipo nav_msgs/OccupancyGrid publicados en el tópico /Map, el nodo usado fue el nodo /turtlebot3_slam_gmapping el cual es parte del SDK del Turtlebot, uno de los robots utilizados para probar el presente proyecto.

3.11. Interfaz grafica

Para mejorar la experiencia del operador al controlar el robot, se resolvió que todas las funcionalidades descritas anteriormente debían incorporarse en una sola interfaz gráfica donde la presencia de estas permitiría establecer un control más preciso y seguro del robot.

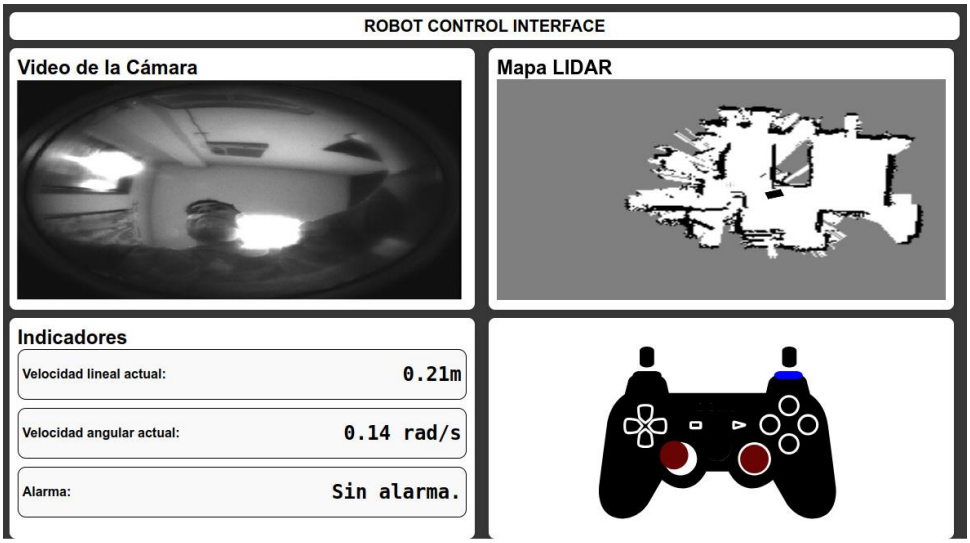
La interfaz gráfica se desarrolló utilizando herramientas de desarrollo web: HTML, CSS y Javascript. HTML se utilizó para definir la estructura de la interfaz web, CSS para estilizarla y agregarle elementos estéticos y finalmente Javascript para darle dinamismo a sus elementos y programar su comportamiento y funcionalidad, lógicamente esta interfaz debió vincularse con el ecosistema ROS para lo cual también se utilizó Javascript. Concretamente para una mejor

solvencia se utilizó el framework Vue.js, el cual es framework progresivo de JavaScript que se utiliza para construir interfaces de usuario interactivas.

Para el diseño de la interfaz gráfica se incluyeron 4 secciones para mostrar la mayor cantidad de operación posible al operador, estas secciones fueron: el video tomado por la cámara del robot, el mapa del sistema construido a partir de la información del sensor LiDAR, indicadores del estado general del robot (posición, velocidad linear y angular) y de alarmas, y finalmente una sección donde se incluye un joystick digital que emula los movimientos.

Figura 3.13

Diseño de la interfaz gráfica.



3.12. Simulación de manejo del robot con el joystick.

Para la verificación del correcto mapeo de las variables de entrada de nuestro joystick, se hará una prueba /jstest desde la terminal del dispositivo al cual estamos conectado la placa, esto permitirá observar que eje del joystick se está moviendo, que valores está tomando en ese movimiento, y que pulsadores se presionaran.

Figura 3.14

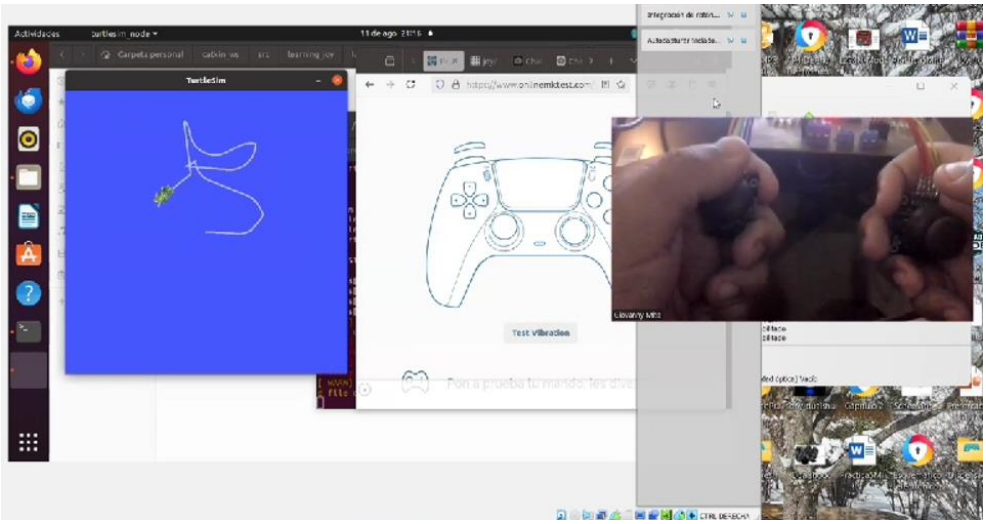
Prueba de reconocimiento del tópico Joy.

```
dmoran@dmoran-HP-245-14-inch-G9-Notebook-PC:~$ rostopic echo /joy
header:
  seq: 281
  stamp:
    secs: 1724631506
    nsecs: 202039293
  frame_id: "/dev/input/js0"
axes: [-0.0, -0.0, 1.0, -0.0, -0.0, 1.0, 0.0, 0.0]
buttons: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
---
header:
  seq: 282
  stamp:
    secs: 1724631506
    nsecs: 354277309
  frame_id: "/dev/input/js0"
axes: [-0.0, -0.0, 1.0, -0.0, -0.0, 1.0, 0.0, 0.0]
buttons: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
---
header:
  seq: 283
  stamp:
    secs: 1724631507
    nsecs:  38043148
  frame_id: "/dev/input/js0"
axes: [-0.0, -0.0, 1.0, -0.0, -0.0, 1.0, 0.0, 0.0]
buttons: [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

Tras realizar la verificación de los ejes y de los pulsadores, se podrá realizar el uso de ROS noetic para ajustar el tópico del joystick al tópico del ajuste de velocidad del robot a utilizar, en este caso, por motivos prácticos, se realiza la simulación de un turtlebot3 modelado en gazebo ajustando la velocidad lineal con el eje de movimiento vertical del joystick izquierdo, y la velocidad angular con el eje de movimiento horizontal del joystick derecho.

Figura 3.15

Control de turtlesim a través del joystick.



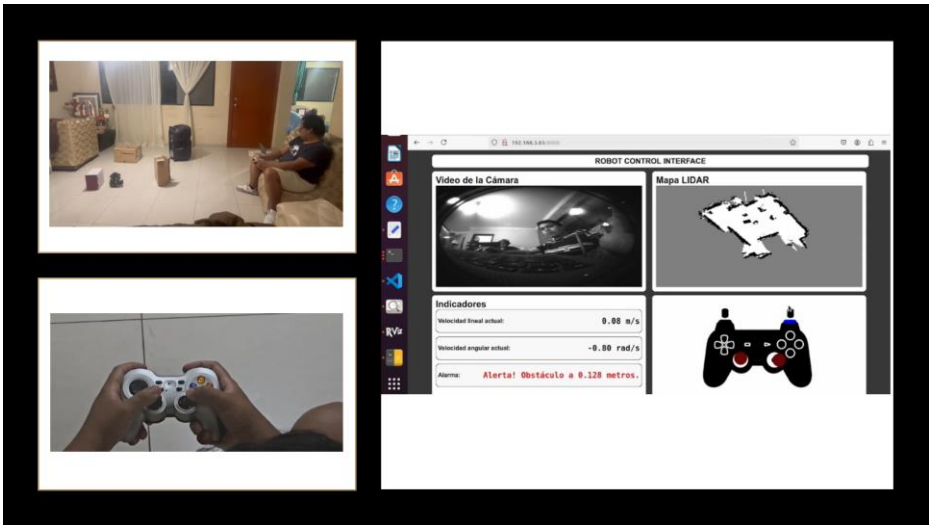
3.13. Pruebas funcionales

Tras implementar todas las funcionales requeridas se procedió a realizar pruebas funcionales, tanto simuladas como físicas del sistema del teleoperación, para realizar las pruebas físicas se usaron un turtlebot3 y el robot Hermes, ambos escenarios resultaron en intentos exitosos.

A continuación, en la imagen se puede apreciar al sistema en funcionamiento, utilizaron un turtlebot3 real con el operador realizando el control con el joystick a través de un ambiente con diversos obstáculos, con la finalidad de probar el funcionamiento de la alarma de proximidad descrita en secciones anteriores.

Figura 3.16

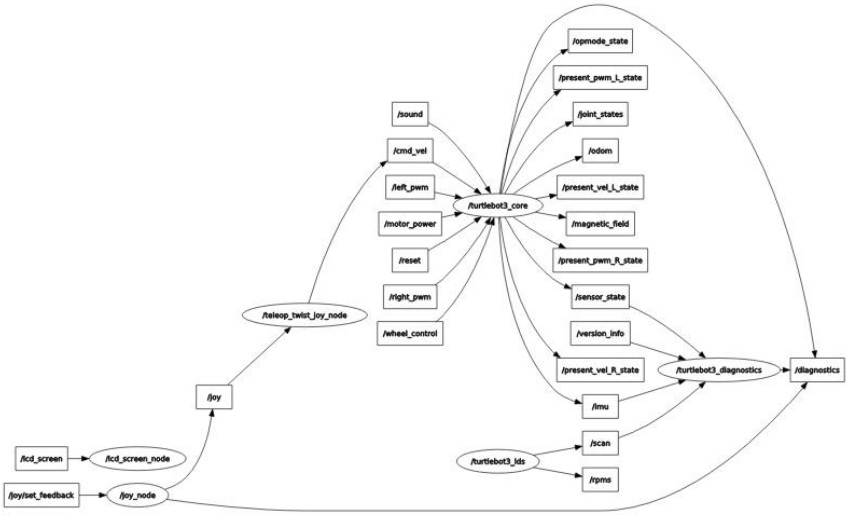
Funcionamiento del sistema de teleoperación



La arquitectura de nodos de ROS final, utilizada en las pruebas con robot físicos, quedó de la siguiente manera:

Figura 3.17

Arquitectura de nodos de ROS



3.14. Análisis de costos.

Para la implementación del proyecto, se deberán tomar en cuenta todos los posibles costos de la fabricación y elaboración completa del proyecto, si bien existen algunos equipos y elementos suministrados por el cliente, es importante saber que la elaboración de este proyecto no es aplicada únicamente para la resolución de esta problemática en particular, se puede realizar la implementación a menor escala para diferentes tipos de sensores y robots.

En la Tabla 3.6 se especifican los rubros detallados para la elaboración de la parte mecánica y electrónica del proyecto.

Tabla 3.6

Análisis de costos.

Modelo proyecto	**Costo**
Microcontrolador ESP32	$12.00
Motores DC 5.9 V	$3.00 c/u
Driver L298N (Puente H)	$6.00
Joysticks con placa	$3.00 c/u
Pulsadores de 4 pines	$0.50 c/u
Baterías AA	$5.00
Fabricación de placa PCB	$50.00
Impresión PLA	$32.00
Tornillos	$2.00
Kit de soldadura básico	$25.00
TOTAL	$145.00

Si bien en análisis de costos se tomó en consideración algunos elementos para realizar el trabajo por uno mismo, en el análisis de rubros, que se detalla en la Tabla 3.7, se especifican ciertos gastos adicionales que se podrían cruzar al buscar optimizar tiempos de elaboración o al buscar

asegurar cierta garantía en procesos de soldadura, impresiones o incluso el diseño de la interfaz gráfica.

Tabla 3.7

Análisis de rubros.

Actividad	**Costo**
Diseño de la placa PCB.	$10.00
Fabricación de la placa PCB.	$45.00
Soldadura de componentes en placa PCB.	$15.00
Diseño de joystick con dimensiones de la placa.	$20.00
Diseño de interfaz gráfica de control.	$50.00
Impresión de joystick.	$32.00
Configuración de joystick con robot.	$10.00
Interconexión entre nodos de ROS.	$20.00
TOTAL	$202.00

Capítulo 4

4.1 Conclusiones y recomendaciones

En base a los objetivos de este proyecto, y a los resultados e información recolectada durante el desarrollo de este, se obtuvieron las siguientes conclusiones:

4.1.1 Conclusiones

- Se logró realizar la teleoperación exitosa del robot Hermes, ubicado en Seúl, Corea del Sur desde Guayaquil, Ecuador. La arquitectura de software desarrollada en ROS tuvo un buen desempeño al ejecutar con mínima latencia las distintas funcionalidades del sistema, y el protocolo de comunicación websocket usado para la comunicación con el robot fue una elección eficiente ya que nos permitió una transmisión de información en tiempo real, siendo la opción de comunicación con la mayor velocidad.

- El uso de la interfaz gráfica de teleoperación, nos permitió consolidar en un solo lugar y de forma eficiente, la retroalimentación entregada por los sensores, en el caso de este proyecto sensores LiDAR y cámaras de profundidad. De esta manera la interfaz se vuelve una herramienta fundamental para mejorar la interacción humano-robot, ya que le transmite al operador información en tiempo real, y organizada de una manera visualmente eficiente, logrando reducir el esfuerzo cognitivo del operador y ayudándolo a tener un control más preciso del robot operado.

- Utilizar diversos tipos de sensores para sensar al robot que desea controlar y al entorno que lo rodea es una práctica que trae muchos beneficios, ya que le brinda al operador información útil que le permite comprender mejor la situación por la cual pasa el robot, para tomar mejores decisiones en cuanto al manejo de este. Mientras más sensores se vean involucrados se podrá conseguir una retroalimentación más acorde a la realidad, pero se debe manejar esta información de forma organizada ya que un mal uso de ello podría aumentar mucho la carga cognitiva del operador y tener un efecto contraproducente.

- Se implementó un sistema de alarmas transmitidas mediante la vibración del motor del joystick de retroalimentación háptica para mejorar el control de robots, optimizando así el tiempo de reacción del usuario y aumentando la seguridad operacional. Esta tecnología permite que el operador reciba señales de alerta a través de vibraciones directas en el joystick, proporcionando una respuesta inmediata y clara a eventos críticos. Al reducir la dependencia de señales visuales o auditivas y ofrecer una comunicación más directa y tangible, se facilita una reacción más rápida y precisa, minimizando el riesgo de errores y mejorando la seguridad general en los entornos operativos.

- La ejecución de este proyecto nos permitió comprobar que la teleoperación de robots, es una rama muy importante de la robótica en actualidad, ya nos permite actuar frente a situaciones específicas donde el control tradicional de robots no es de utilidad. Con el constante desarrollo en la ciencia robótica actual, cada vez más presente en la industria, también es imperativo elaborar sistemas de teleoperación cada vez más robustos que se

ajusten a las necesidades del mercado y que sean capaces de mejorar el rendimiento en las operaciones del robot teleoperado.

4.1.2 Recomendaciones

- Se recomienda tener una buena conexión a internet al momento de realizar las pruebas de comunicación con el robot, para una comunicación de manera fluida se debe trabajar con un ancho de banda de 200 mb/s o superior, conexiones de menor ancho de banda son propensas a sufrir de latencia, lo cual causaría un bajo rendimiento del sistema de teleoperación.

- Para el desarrollo de la interfaz web, se recomiendo optimizar lo máximo posible el código funcional de la interfaz, es decir, el código Javascript, es mejor utilizar frameworks de desarrollo como Vue.js o React que permiten codificar más rápida y organizadamente y optimizando los recursos computacionales. Al momento de interactuar con los tópicos de ROS, se puede usar librerías y herramientas existentes en la web, que nos permitirá reducir la complejidad del código y usar funciones pre creadas para evitar escribir tareas repetitivas, sino enfocarnos más en las funcionalidades específicas de nuestro proyecto, en el caso de este proyecto se usó roslibjs, y roslibjs, herramientas de código abierto mantenidas por la comunidad.

- Al momento de interactuar con los tópicos de ROS desde la interfaz web, se debe tener cuidado puesto que generalmente estos tienen una frecuencia de llegada de mensajes bastante alta, lo que podría llegar a ralentizar la interfaz si no se maneja esta información

entrante de una forma eficiente, también se debe tener cuidado de no hacer una suscripción al mismo tópico más de una vez ya que esto de igual manera vuelve más lenta la interfaz.

- La implementación del sistema de control para el manejo de la velocidad de los motores a una en específico se vuelve sustancial para salvaguardar la vida útil de los motores de vibración, además de brindar una respuesta más apta al comportamiento deseado para generar la vibración, recordar que, al tener una carga excéntrica, un cambio brusco en la velocidad puede generar una contrafuerza.

Referencias

[1] P. Miranda and Anrrango Alex, “Diseño de un robot móvil autónomo para entregas de alimentos en un campus universitario.”

[2] Y. Uchuari and R. Valdez, “Diseño y construcción de un compañero robótico de asistencia para niños hospitalizados (Yaren)”.

[3] “Top 5 Robot Trends 2024 - International Federation of Robotics.” Accessed: Jun. 13, 2024. [Online]. Available: https://ifr.org/ifr-press-releases/news/top-5-robot-trends-2024

[4] S. Lichiardopol, “A survey on teleoperation,” 2007. [Online]. Available: www.tue.nl/taverne

[5] G. Yang *et al.*, “Keep Healthcare Workers Safe: Application of Teleoperated Robot in Isolation Ward for COVID-19 Prevention and Control,” *Chinese Journal of Mechanical Engineering (English Edition)*, vol. 33, no. 1, pp. 1–4, Dec. 2020, doi: 10.1186/S10033-020-00464-0/FIGURES/7.

[6] Mauricio Serfatty Godoy, “Los robots cirujanos competirán en el primer gran estudio que los compara | WIRED.” Accessed: Jun. 10, 2024. [Online]. Available: https://es.wired.com/articulos/robots-cirujanos-competiran-en-primer-gran-estudio-comparativo

[7] NASA/JPL/Cornell University, “PIA04413: Artist’s Concept of Rover on Mars.” Accessed: Jun. 10, 2024. [Online]. Available: https://photojournal.jpl.nasa.gov/catalog/PIA04413

[8] D. J. Rea and S. H. Seo, “Still Not Solved: A Call for Renewed Focus on User-Centered Teleoperation Interfaces,” *Front Robot AI*, vol. 9, p. 704225, Mar. 2022, doi: 10.3389/FROBT.2022.704225/BIBTEX.

[9] S. Avgousti *et al.*, "Medical telerobotic systems: current status and future trends," *BioMedical Engineering OnLine 2016 15:1*, vol. 15, no. 1, pp. 1–44, Aug. 2016, doi: 10.1186/S12938-016-0217-7.

[10] SLAMTEC, "Slamtec Hermes - SLAMTEC Global Network." Accessed: Jun. 09, 2024. [Online]. Available: https://www.slamtec.ai/product/slamtec-hermes/

[11] "LiDAR: what the future looks like for this sensor | by Alex Undermoore | The Startup | Medium." Accessed: Jun. 09, 2024. [Online]. Available: https://medium.com/swlh/lidar-what-the-future-looks-like-for-this-sensor-be3caf917e53

[12] V. Juan and C. Segura, "Módulo de detección y seguimiento de personas para el guiado de un robot móvil dotado de un sensor LIDAR," May 2023, Accessed: Jun. 24, 2024. [Online]. Available: https://upcommons.upc.edu/handle/2117/390657

[13] G. A. Avella Neira, "Diseño y construcción de un prototipo funcional de vehículo terrestre teledirigido para el mapeo de espacios interiores usando un lidar," 2021, Accessed: Jun. 24, 2024. [Online]. Available: https://openaccess.uoc.edu/handle/10609/126266

[14] "Evolution in the use of range and artificial vision sensors for... | Download Scientific Diagram." Accessed: Jun. 09, 2024. [Online]. Available: https://www.researchgate.net/figure/Evolution-in-the-use-of-range-and-artificial-vision-sensors-for-morphological_fig5_320296507

[15] "Robots and Robotics Jobs in 2024: Career Outlook + FAQ | Coursera." Accessed: Jun. 09, 2024. [Online]. Available: https://www.coursera.org/articles/robotics-jobs

[16] "The Future of Jobs Report 2020 O C T O B E R 2 0 2 0," 2020.

[17] J. Wardini, "27+ Astonishing Robotics Industry Statistics You Should Know in 2024." Accessed: Jun. 09, 2024. [Online]. Available: https://techjury.net/blog/robotics-industry-statistics/

[18] “The Benefits of Using ROS in Robotics - Awe Robotics.” Accessed: Jun. 10, 2024. [Online]. Available: https://www.awerobotics.com/the-benefits-of-using-ros-in-robotics/

[19] “Home - SLAMWARE ROS SDK.” Accessed: Jun. 10, 2024. [Online]. Available: https://developer.slamtec.com/docs/slamware/ros-sdk-en/2.8.2_rtm/

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

[21] “How Nikola Tesla Inspired The Field of Remote Operation.” Accessed: Jun. 12, 2024. [Online]. Available: https://phantom.auto/phantom-auto-blog/how-nikola-tesla-inspired-the-field-of-remote-operation

[22] “History of Telepresence Robots | TRinE.” Accessed: Jun. 12, 2024. [Online]. Available: https://www.trine-platform.com/project/history-of-telepresence-robots/

[23] A. C. Correa, “SISTEMAS ROBOTICOS TELEOPERADOS Teleoperated Robotics Systems.”

[24] E. P. Superior and S. Alfaro Ballesteros, “Universidad Carlos III de Madrid Sistema de teleoperación mediante una interfaz natural de usuario Tutor: Moisés Martínez Muñoz,” 2012.

[25] “¿Qué sensores se utilizan en los robots industriales y móviles? | Robotnik ®.” Accessed: Jun. 12, 2024. [Online]. Available: https://robotnik.eu/es/tipos-de-sensores-en-robotica-movil/

[26] J. S. Lee, Y. Ham, H. Park, and J. Kim, “Challenges, tasks, and opportunities in teleoperation of excavator toward human-in-the-loop construction automation,” *Autom Constr*, vol. 135, Mar. 2022, doi: 10.1016/j.autcon.2021.104119.

[27] Y. P. Su, X. Q. Chen, C. Zhou, L. H. Pearson, C. G. Pretty, and J. G. Chase, “Integrating Virtual, Mixed, and Augmented Reality into Remote Robotic Applications: A Brief Review of Extended Reality-Enhanced Robotic Systems for Intuitive Telemanipulation and Telemanufacturing Tasks

in Hazardous Conditions," *Applied Sciences 2023, Vol. 13, Page 12129*, vol. 13, no. 22, p. 12129, Nov. 2023, doi: 10.3390/APP132212129.

[28] K. Youssef, S. Said, S. Al Kork, and T. Beyrouthy, "Telepresence in the Recent Literature with a Focus on Robotic Platforms, Applications and Challenges," *Robotics 2023, Vol. 12, Page 111*, vol. 12, no. 4, p. 111, Aug. 2023, doi: 10.3390/ROBOTICS12040111.

[29] "Sensor de LiDAR mecánico 3D de 32 líneas para escáner V2X 3D - AliExpress 1420." Accessed: Jun. 13, 2024. [Online]. Available: https://es.aliexpress.com/item/1005005719161358.html

[30] "Simultaneous Localization and Mapping | Baeldung on Computer Science." Accessed: Jun. 12, 2024. [Online]. Available: https://www.baeldung.com/cs/slam

[31] "What Is Simultaneous Localization and Mapping? What Is Simultaneous Localization and Mapping? | NVIDIA Blog." Accessed: Jun. 24, 2024. [Online]. Available: https://blogs.nvidia.com/blog/what-is-simultaneous-localization-and-mapping-nvidia-jetson-isaac-sdk/

[32] "Interfaz remota para la teleoperación del robot móvil | Download Scientific Diagram," Caicedo Eduardo, Bacc Bladimir. Accessed: Jun. 12, 2024. [Online]. Available: https://www.researchgate.net/figure/Figura-6-Interfaz-remota-para-la-teleoperacion-del-robot-movil_fig3_266854304

[33] Gómez García Félix Fabián and Entrena Arrontes Luis, "Diseño de un Joystick USB con licencias libres.," Madrid, Dec. 2011.

[34] López Javier, "Periféricos ¿Te acuerdas del puerto PS/2 en tu teclado? ¿Por qué no se usa?"

[35] Carrillo Jaramillo Alex, "¿Qué significan los colores en los puertos USB?," Jun. 28, 2022.

[36] DEL ROSARIO EDISON, "Grupo de Investigación de Redes de información Inalámbricas," *1.4 IDE Arduino con ESP32*, Dec. 2018.

[37] jorgebjm, “Como soldar motores vibradores de un mando joystick a un mouse?” Foros de electrónica, Cusco, Feb. 05, 2021.

[38] “es/ROS/Introduccion - ROS Wiki.” Accessed: Jun. 24, 2024. [Online]. Available: https://wiki.ros.org/es/ROS/Introduccion

[39] “ROBOT OPERATING SYSTEM (ROS) – Tknika.” Accessed: Jun. 24, 2024. [Online]. Available: https://tknika.eus/cont/robot-operating-system-ros-3/

[40] “ROS – ¿Un sistema operativo diseñado para robots? | Autracen Soluciones Industriales 4.0.” Accessed: Jun. 24, 2024. [Online]. Available: https://www.autracen.com/blog/viajes-1/ros-un-sistema-operativo-disenado-para-robots-67

Anexos

Código de archivo para iniciar nodos de control por joystick

```
<launch>

    <!-- Launch the joy node -->
    <node name="joy_node" pkg="joy" type="joy_node" output="screen">
        <!-- Example of a parameter specific to joy_node -->
        <!-- <param name="dev" value="/dev/input/js0" /> -->
    </node>

    <!-- Load parameters from the YAML file -->
    <rosparam file="$(find joy_turtlesim)/config/teleop_params.yaml" command="load"/>

    <!-- Launch the teleop_twist_joy node with topic remapping -->
    <node name="teleop_twist_joy_node" pkg="teleop_twist_joy" type="teleop_node" output="screen">

    </node>
</launch>
```

Código para nodo de Alarma de proximidad

```
#!/usr/bin/env python3
import rospy
from sensor_msgs.msg import LaserScan
from my_gui.msg import LidarAlarm

def publish_alarm(distance, alarm_status):
    """Publishes the alarm message with the given distance and status."""
    msg = LidarAlarm()
    msg.distance = distance
    msg.alarm = alarm_status
    alarm_pub.publish(msg)
    rospy.loginfo(f"Alarm status: {'Activated' if alarm_status else 'Deactivated'}, Distance: {distance:.2f} m")

def callback(data):
    """Callback function for LaserScan messages."""
    try:
        min_distance = min(data.ranges)
        global StatusAlarma
        # Check if the detected distance is less than the threshold
        if min_distance < 0.15:
            if not StatusAlarma:  # Only update if status has changed
                rospy.logwarn(f"Obstacle too close! Distance: {min_distance:.2f} m")
                publish_alarm(min_distance, True)
                StatusAlarma = True
        else:
            if StatusAlarma:  # Only update if status has changed
                publish_alarm(min_distance, False)
                StatusAlarma = False
    except ValueError:
        rospy.logwarn("Data ranges were empty, no minimum distance found.")
```

Código para nodo de conversión de quaterniones a ángulos de Euler

```
#!/usr/bin/env python3
import rospy
import tf
from nav_msgs.msg import Odometry
from std_msgs.msg import String
from geometry_msgs.msg import Vector3
import tf.transformations


def quaternion_to_euler(quaternion):
    euler = tf.transformations.euler_from_quaternion(
        [quaternion.x, quaternion.y, quaternion.z, quaternion.w]
    )
    return euler

def odometry_callback(data):

    quaternion = data.pose.pose.orientation

    roll,pitch,yaw = quaternion_to_euler(quaternion)

    # Create a Vector3 message
    rpy_vector = Vector3()
    rpy_vector.x = roll
    rpy_vector.y = pitch
    rpy_vector.z = yaw

    # Publish the Vector3 message
    pub.publish(rpy_vector)

    if pitch > 0.26 or pitch < -0.26:
        rospy.logwarn("ALERTA. ROBOT DESESTABILIZADO, HA GIRADO MAS DE 15 GRADOS")
```

Código HTML para estructura de la interfaz web

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Robot Control Interface</title>

    <script type="text/javascript" src="https://cdn.jsdelivr.net/npm/roslib@1/build/roslib.js"></script>
    <script type="text/javascript" src="https://cdn.jsdelivr.net/npm/easeljs@1/lib/easeljs.js"></script>
    <script type="text/javascript" src="https://cdn.jsdelivr.net/npm/eventemitter2@6/lib/eventemitter2.js"></script>
    <script type="text/javascript" src="https://cdn.jsdelivr.net/npm/ros2d@0/build/ros2d.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/vue@3.4.31/dist/vue.global.js"></script>


</head>

<body>

    <div class="grid-container" id="app">

        <div class="grid-item-header">
            ROBOT CONTROL INTERFACE
        </div>

        <!--
FIRST DIV: ROBOT CAMERA-->
```

Código HTML para la sección de indicadores de la Interfaz

```
<h2>Indicadores</h2>
<div class="indicators">
    <div class="indicator-box">
        <span class="indicator-label">Velocidad lineal actual:</span>
        <span class="indicator-value">{{ v_linear }}</span>

    </div>
    <div class="indicator-box">
        <span class="indicator-label">Velocidad angular actual:</span>
        <span class="indicator-value">{{ v_angular }}</span>
    </div>

    <div class="indicator-box">
        <span class="indicator-label">Alarma:</span>
        <span class="indicator-value">
            <span v-if="alarm">Alerta! Obstáculo a {{ distance_obs }} metros.</span>
            <span v-else>Sin alarma.</span>
        </span>
    </div>
</div>
```

Código Arduino IDE configuración Joystick.

```
#include <Arduino.h>
#include <BleGamepad.h>

// ABXY BUTTONS
#define X_BUTTON 27
#define CIRCLE_BUTTON 26
#define TRIANGLE_BUTTON 32
#define SQUARE_BUTTON 33

// TRIGGERS
#define R1_BUTTON 22
#define R2_BUTTON 21
#define L1_BUTTON 25
#define L2_BUTTON 23

// MENU BUTTONS
#define START_BUTTON 19
#define SELECT_BUTTON 18
#define PS_BUTTON 17

// JOYSTICKS BUTTONS
#define R3_BUTTON 14
#define L3_BUTTON 16

// JOYSTICKS
#define LEFT_VRX_JOYSTICK 39  // SN
#define LEFT_VRY_JOYSTICK 36  // SP
```

```
#define RIGHT_VRX_JOYSTICK 35
#define RIGHT_VRY_JOYSTICK 34

#define NUM_BUTTONS 13

int buttonsPins[NUM_BUTTONS] = {X_BUTTON, CIRCLE_BUTTON, TRIANGLE_BUTTON, SQUARE_BUTTON,
                                R1_BUTTON, R2_BUTTON, L1_BUTTON, L2_BUTTON,
                                START_BUTTON, SELECT_BUTTON, PS_BUTTON,
                                R3_BUTTON, L3_BUTTON};

int PCGamepadButtons[NUM_BUTTONS] = {1, 2, 4, 3, 6, 8, 5, 7, 10, 9, 0, 12, 11};

float leftVrxJoystickLecture = 0;
float leftVryJoystickLecture = 0;
float rightVrxJoystickLecture = 0;
float rightVryJoystickLecture = 0;

uint16_t leftVrxJoystickValue = 0;
uint16_t leftVryJoystickValue = 0;
uint16_t rightVrxJoystickValue = 0;
uint16_t rightVryJoystickValue = 0;

BleGamepad bleGamepad("JOYSTICK", "ESPOL JOYSTICK");
BleGamepadConfiguration bleGamepadConfig;

void setup() {
  delay(1000);

  Serial.begin(115200);
  pinMode(LEFT_VRX_JOYSTICK,INPUT);
  pinMode(LEFT_VRY_JOYSTICK,INPUT);
  pinMode(RIGHT_VRX_JOYSTICK,INPUT);
  pinMode(RIGHT_VRY_JOYSTICK,INPUT);
  for (int i = 0; i < NUM_BUTTONS; i++) {
    pinMode(buttonsPins[i], INPUT_PULLUP);
  }

  bleGamepadConfig.setAutoReport(false);
  bleGamepadConfig.setControllerType(CONTROLLER_TYPE_GAMEPAD);
  bleGamepadConfig.setVid(0xe502);
  bleGamepadConfig.setPid(0xabcd);
  bleGamepadConfig.setHatSwitchCount(4);
  bleGamepad.begin(&bleGamepadConfig);
}

void loop() {
  if (bleGamepad.isConnected()) {
    // Joysticks lecture
    leftVrxJoystickLecture = analogRead(LEFT_VRX_JOYSTICK);
    leftVryJoystickLecture = analogRead(LEFT_VRY_JOYSTICK);
    rightVrxJoystickLecture = analogRead(RIGHT_VRX_JOYSTICK);
    rightVryJoystickLecture = analogRead(RIGHT_VRY_JOYSTICK);

    // Compute joysticks value
    leftVrxJoystickValue = 32767-(leftVrxJoystickLecture/4095)*32767;
```

```
      leftVryJoystickValue = (leftVryJoystickLecture/4095)*32767;
      rightVrxJoystickValue = 32767-(rightVrxJoystickLecture/4095)*32767;
      rightVryJoystickValue = (rightVryJoystickLecture/4095)*32767;

      for (int i = 0; i < NUM_BUTTONS; i++) {
        if (!digitalRead(buttonsPins[i])) {
          bleGamepad.press(PCGamepadButtons[i]);
        } else {
          bleGamepad.release(PCGamepadButtons[i]);
        }
      }

      joysticksHandlerForPC(leftVrxJoystickValue, leftVryJoystickValue, rightVrxJoystickValue, rightVryJoystickValue);

      bleGamepad.sendReport();
    }
    delay(100);
  }

  void joysticksHandlerForPC(uint16_t leftVrx, uint16_t leftVry, uint16_t rightVrx, uint16_t rightVry) {
    bleGamepad.setX(leftVrx);
    bleGamepad.setY(leftVry);
    bleGamepad.setZ(rightVrx);
    bleGamepad.setRX(rightVry);
  }
```

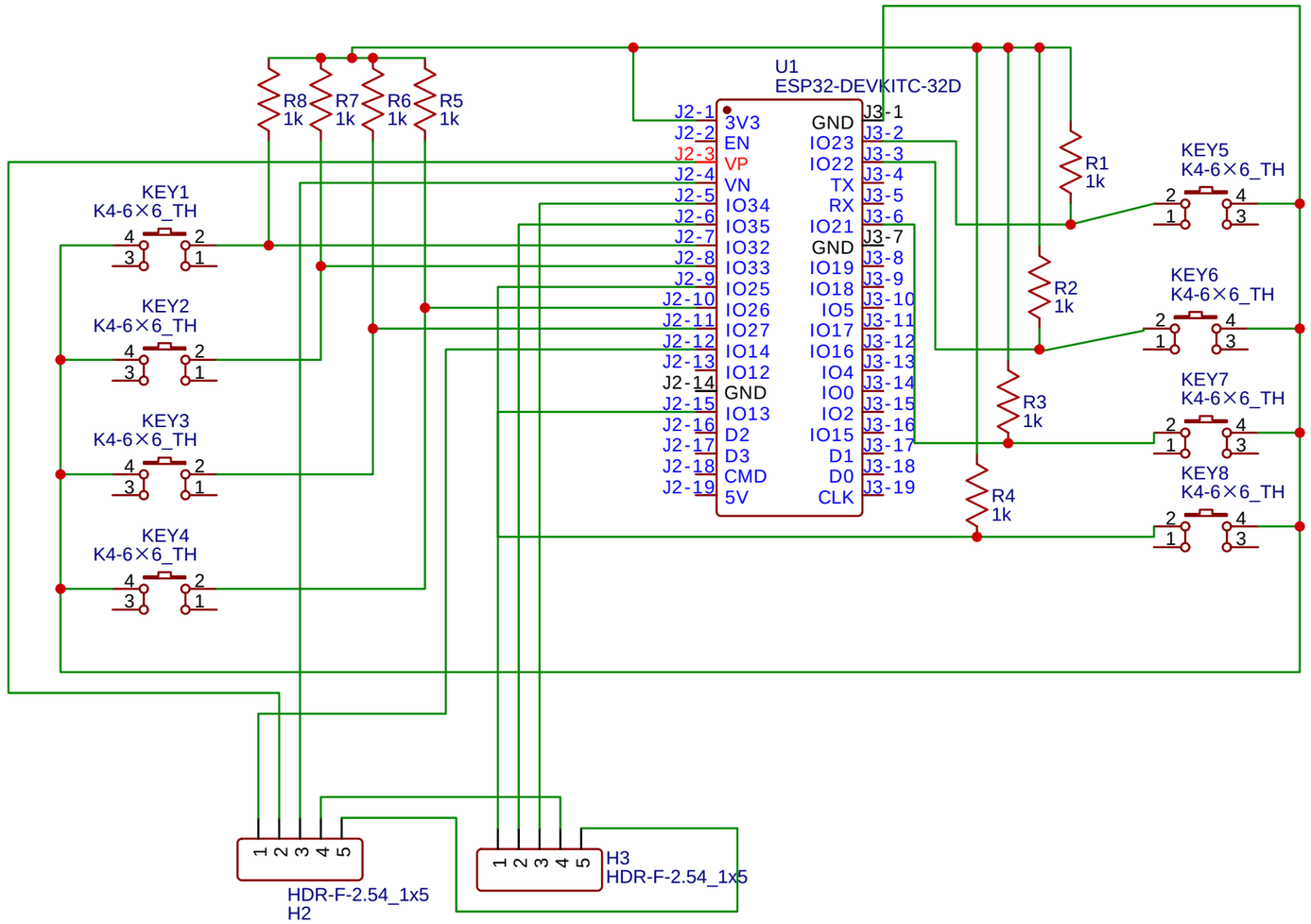
Código Arduino IDE Motores de vibración.

```
const int enA = 4;
const int enB = 5;
const int in1 = 16;
const int in2 = 17;
const int in3 = 18;
const int in4 = 19;

void setup() {
  // Configuration
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  // Inicializing pins
  digitalWrite(enA, LOW);
  digitalWrite(enB, LOW);
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}


void motorAForward(int speed) {
```

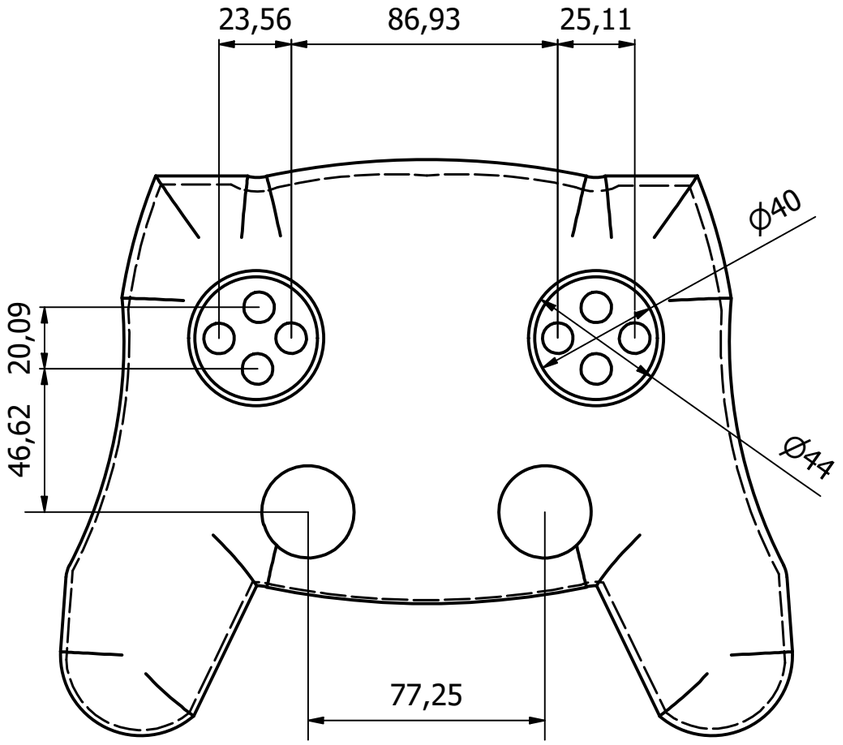
```
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  analogWrite(enA, speed);
}

void motorABackward(int speed) {
  digitalWrite(in1, LOW);
  digitalWrite(in2, HIGH);
  analogWrite(enA, speed);
}

void motorBForward(int speed) {
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enB, speed);
}

void motorBBackward(int speed) {
  digitalWrite(in3, LOW);
  digitalWrite(in4, HIGH);
  analogWrite(enB, speed);
}

void stopMotors() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);

  digitalWrite(in4, LOW);
  analogWrite(enA, 0);
  analogWrite(enB, 0);
}

void analogWrite(int pin, int value) {
  // Function to emulate AnalogWrite in ESP32
  ledcAttachPin(pin, pin); // Associate pin to channel
  ledcSetup(pin, 5000, 8); // Determine a frecuency and resolution to channel
  ledcWrite(pin, value);   // write PWM signal in channel


void loop() {
//Simulation of vibration in the motors, potencia value will change depending on what we obtain in the tracking camera.
  int potencia = 250;
  int tiempo = 100;

  motorAForward(potencia);
  motorBBackward(potencia);
  delay(tiempo);

  stopMotors();
  delay(tiempo);

  motorABackward(potencia);
  motorBForward(potencia);
  delay(tiempo);


  stopMotors();
  delay(tiempo);
}
```

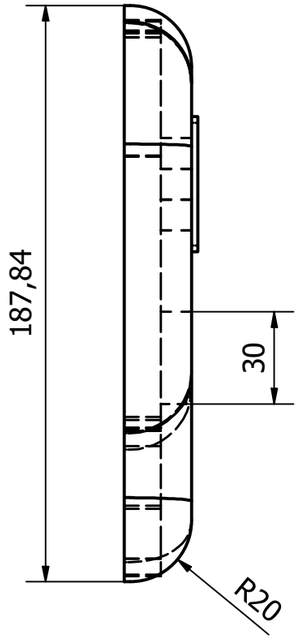
Código Sistema de control para motores de vibración.

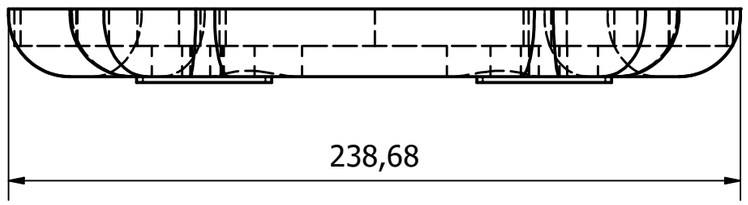
```
J = 0.01;
b = 0.1;
K = 0.01;
R = 1;
L = 0.5;

Kp = 50.0;
Ki = 5.0;
Kd = 10.0;

desired_rpm = 200;
desired_omega = desired_rpm * (2 * pi / 60);

s = tf('s');
P_motor = K / (J * s^2 + b * s + K);

C_pid = Kp + Ki / s + Kd * s;

T_cl = feedback(C_pid * P_motor, 1);

t = 0:0.01:5;
step_input = desired_omega * ones(size(t));
[y, t_out] = lsim(T_cl, step_input, t);

y_rpm = y * (60 / (2 * pi));

tolerance = 0.05 * desired_omega;

final_value = desired_omega;
settling_time = find(abs(y - final_value) < tolerance, 1, 'first');
if ~isempty(settling_time)
    settling_time = t(settling_time);
else
    settling_time = NaN;
end

peak_value = max(y);
overshoot = (peak_value - final_value) / final_value * 100;

figure;

subplot(2, 1, 1);
step(T_cl);
title('Respuesta al Escalón del Sistema con Controlador PID');
xlabel('Tiempo (s)');
ylabel('Respuesta al escalón unitario');
grid on;

subplot(2, 1, 2);
plot(t_out, step_input * (60 / (2 * pi)), 'r--', 'LineWidth', 2);
hold on;
plot(t_out, y_rpm, 'b-', 'LineWidth', 2);
title('Comparación de Velocidad Deseada y Velocidad Real');
xlabel('Tiempo (s)');
ylabel('Velocidad (RPM)');

legend('Velocidad Deseada', 'Velocidad Real');
grid on;

fprintf('Tiempo de estabilización: %.2f segundos\n', settling_time);
fprintf('Porcentaje de sobreimpulso: %.2f%%\n', overshoot);
```



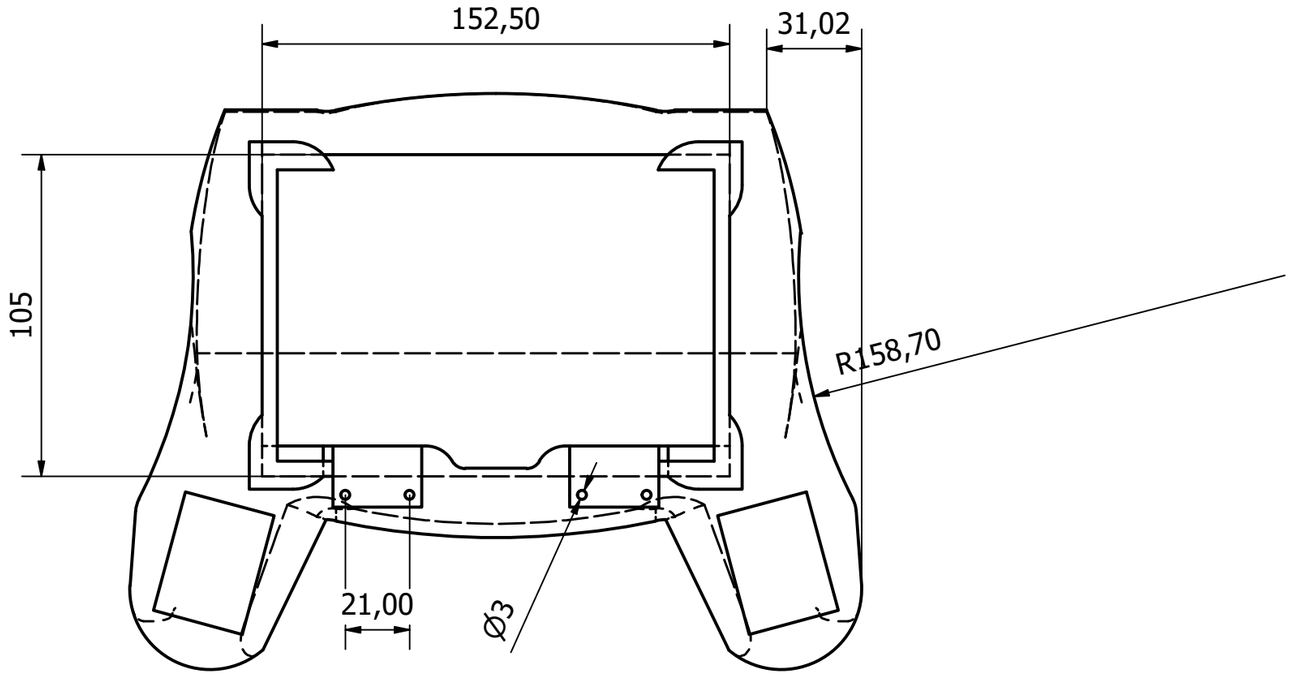
TITLE: Joystick configuración			REV: 1.0
EasyEDA	Company: ESPOL		Sheet: 1/1
	Date: 2024-07-15	Drawn By: Giovanny Mite	

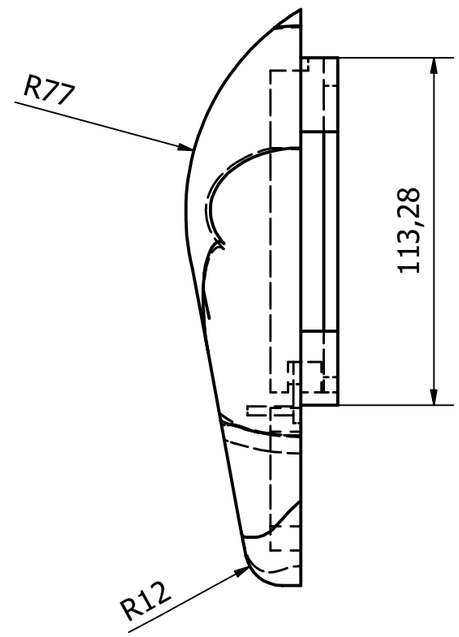


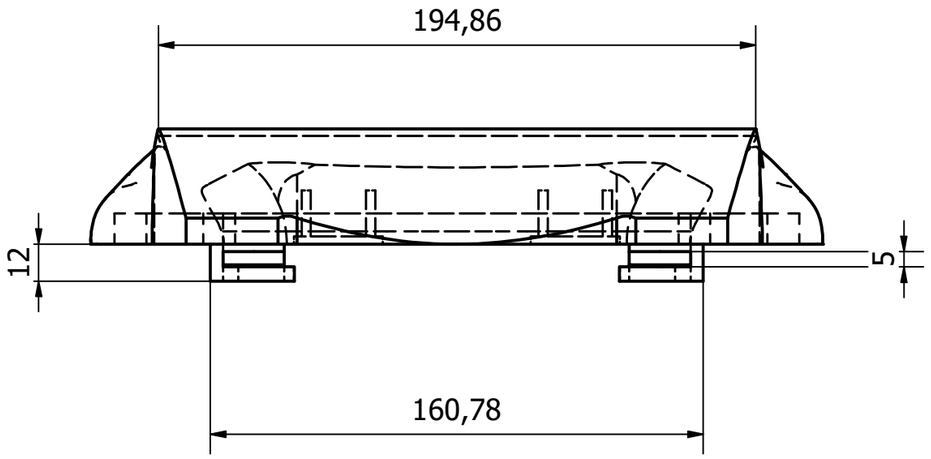




Diseño de	Vista	Escala		Fecha	
Giovanny Mite	ISO A	1/2		9/9/2024	
ESPOL			Vista Superior		
		1		Edición	Hoja 1 / 1







Diseño de: Giovanny Mite	Vista: ISO A	Escala: 1/2		Fecha: 9/9/2024	
ESPOL			Vista Inferior		
			1	Edición	Hoja: 1 / 1