UNIVERSIDAD RICARDO PALMA

FACULTAD DE INGENIERÍA

PROGRAMA DE TITULACIÓN POR TESIS

ESCUELA PROFESIONAL DE INGENIERÍA MECATRÓNICA

DISEÑO DE UN ROBOT MÓVIL CON DIRECCIONAMIENTO DIFERENCIAL USANDO LÓGICA DIFUSA Y ESCANEO LIDAR PARA DESINFECCIÓN POR RADIACIÓN UV-C DE ESPACIOS CERRADOS

TESIS

PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO MECATRÓNICO

PRESENTADA POR:

Bach. PIMENTEL CALMET GONZALO ANDRÉ

Bach. MEZA SILVERA WALDIR RONALD

ASESOR: Dr. Ing. SOTELO VALER FREEDY

LIMA-PERÚ

2021

DEDICATORIA

Dedico todos mis esfuerzos a mi madre y padre, de quienes siempre he recibido todo el cariño y apoyo incondicional, a Carmen por acompañarme tantas amanecidas en todo el desarrollo de este trabajo, y a Dios por siempre cuidar de todos nosotros.

Pimentel Calmet Gonzalo André

Dedico esta investigación a mis padres quienes me dieron su consejo y apoyo incondicional, en mi formación como profesional y ser humano.

Meza Silvera Waldir Ronald

AGRADECIMIENTO

Nuestros agradecimientos a los profesores de la escuela ingeniería mecatrónica, por darnos sus conocimientos para poder lograr el desarrollo de la presente investigación. Agradecemos también a nuestros asesores de tesis por guiarnos y motivarnos en todo el transcurso y desarrollo de la investigación.

Pimentel Calmet Gonzalo André
Meza Silvera Waldir Ronald

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

El presente trabajo de investigación titulado “Diseño de un robot móvil con direccionamiento diferencial usando lógica difusa y escaneo LiDAR para desinfección por radiación UV-C de espacios cerrados”, permitió mediante la radiación UV-C a través de tiras led ensambladas en un robot móvil la desinfección de espacios cerrados en hospitales. Se dimensionó la estructura mecánica y sistema eléctrico del robot móvil teniendo en cuenta su autonomía energética y velocidad máxima, asimismo se seleccionó un controlador Raspberry Pi para hacer el procesamiento de las señales del sensor LiDAR y lograr hacer un mapeo de la zona a desinfectar. El presente trabajo de investigación se justificó pues tiene valor teórico, beneficio práctico, por su conveniencia y en base a la rentabilidad que generaría la aplicación del diseño de un robot móvil con direccionamiento diferencial para la desinfección de espacios reducidos en hospitales. El marco teórico se basó en las nociones de robots móviles, sistema LiDAR, sistema operativo para robots, control difuso, localización y mapeo simultáneo (SLAM) y desinfección por radiación UV-C.

Como resultado de la investigación, se elaboró una interfaz de simulación mediante Gazebo un simulador de robótica de código abierto, donde se hacen pruebas de evasión de obstáculos del robot móvil con éxito para la desinfección mediante radiación UV-C en espacios cerrados. Se concluyó que el uso de un robot móvil para desinfección por radiación UV-C de espacios reducidos es de mucha utilidad para eliminar el 99,9% de los gérmenes y otros microorganismos como virus de forma autónoma para mayor seguridad del personal médico, lo cual a su vez conlleva a una disminución en la propagación del COVID-19.

Palabras clave: Radiación por UV-C, robot móvil, controlador Raspberry Pi, sensor LiDAR, sistemas de localización y mapeo simultáneo, COVID-19.

ABSTRACT

The present research work entitled "Design of a mobile robot with differential addressing using fuzzy logic and LiDAR scanning for UV-C radiation disinfection of closed spaces", using UV-C radiation through led strips assembled in a mobile robot la disinfection of closed spaces in hospitals. The mechanical structure and electrical system of the mobile robot were dimensioned taking into account its energy autonomy and maximum speed, a Raspberry Pi controller was also selected to process the LiDAR sensor signals and achieve a mapping of the area to be disinfected. The present research work was justified because it has theoretical value, practical benefit, for its convenience and based on the profitability that the application of the design of a mobile robot with differential addressing for the disinfection of confined spaces in hospitals would generate. The theoretical framework was based on the notions of mobile robots, LiDAR system, robot operating system, fuzzy control, simultaneous localization and mapping (SLAM), and UV-C disinfection.

As a result of the research, a simulation interface was developed using Gazebo an open source robotics simulator, where obstacle avoidance tests of the mobile robot are successfully performed for disinfection by UV-C radiation in enclosed spaces. It was concluded that the use of a mobile robot for disinfection by UV-C radiation of reduced spaces is very useful to eliminate 99.9% of germs and other microorganisms such as viruses autonomously for greater safety of medical personnel, which in turn, it leads to a decrease in the spread of COVID-19.

Keywords: UV-C radiation, mobile robot, Raspberry Pi controller, LiDAR sensor, simultaneous localization and mapping systems, COVID-19.

INTRODUCCIÓN

La robótica es una de las ramas más destacadas de la ingeniería mecatrónica debido a su aplicación en diversas actividades del ser humano, por lo cual la presente investigación se orienta en el diseño de un robot móvil con direccionamiento diferencial para desinfección de espacios cerrados. Los robots móviles utilizados en la desinfección de hospitales es un asunto que se encuentra cada vez en crecimiento razón por la cual los fabricantes de robots tienen la tendencia de mejorarlos creando tecnología que aumente su eficiencia energética y autonomía. El objetivo principal de la investigación consiste en el uso de un robot móvil diferencial capaz de desinfectar en espacios cerrados a través de la radiación UV-C, para ello se utilizó un sensor LiDAR que permite el mapeo de la zona a desinfectar y poder evadir obstáculos con éxito. De modo que, para un buen desarrollo, la presente investigación se estructuró de la siguiente manera:

En el capítulo I, se explica la problematización, objetivo general, objetivos específicos, la justificación, el alcance y limitaciones de la investigación.

En el capítulo II, se describe el marco teórico, aquí se explican los antecedentes de la investigación, tanto internacionales como nacionales, asimismo las bases teóricas que sirven de guía para el desarrollo de la investigación. Del mismo modo se detalla nociones robots móviles, sistema LiDAR, sistema operativo para robots, control difuso, localización y mapeo simultáneo (SLAM) y desinfección por radiación UV-C.

En el capítulo III, se describe el diseño del robot móvil, donde se detalla el proceso y requerimientos del sistema, además se detalla el diseño mecánico, eléctrico y electrónico, por último, se describe del diseño del mapeo y navegación del robot móvil.

En el capítulo IV, se detalla y se realiza el análisis de los resultados obtenidos del trabajo de investigación.

Para concluir la investigación se describen las conclusiones, recomendaciones y se menciona la referencia bibliográfica utilizada, de igual manera se consideran los anexos con información que valida la investigación.

CAPÍTULO I: PLANTEAMIENTO DEL PROBLEMA

El presente capítulo explica la problemática de la investigación relacionada a la propagación y permanencia del COVID-19 en superficies sólidas, además se presentan los objetivos generales y específicos, asimismo se describen los alcances, limitaciones, la importancia y justificación de la investigación.

1.1. Descripción de la problemática

En los múltiples estudios de los últimos años realizados a nivel global centrados en los virus del SARS y MERS, demostraron que una de sus variantes denominada popularmente como COVID-19, puede permanecer en superficies sólidas y estar activa a temperaturas bajas hasta un máximo de nueve días continuos si el espacio no es ventilado frecuente y adecuadamente. En otra clase de ambientes de características más agresivas, esta variante tiene a sobrevivir un promedio de entre cuatro a cinco días. Los bajos niveles de temperatura y la alta concentración de humedad del aire alargan más su vida útil. (Kampf, Todt, Pfaender y Steinmann, 2020).

Esta variedad, como todas las demás enfermedades transmitidas por gotitas, tiende a propagarse rápidamente a través de las manos y las superficies que se tocan con frecuencia. Entre ellas se encuentran los pomos de las puertas o las cerraduras de los hospitales, así como los botones de los dispositivos médicos y de acceso a los pacientes, las mesillas de noche, los bordes de las camas y demás materiales que están muy cerca de los usuarios. Ciertamente, en los últimos dos años, se han incrementado los estudios que demuestran la completa efectividad de desinfección por medio de radiación de luz UV-C dentro del espectro entre los 100 y 280 nanómetros, que, si bien es dañina para tejidos orgánicos en cortas y largas exposiciones, podría ser suministrada por aparatos móviles controlados a distancia o con algún tipo de autonomía móvil.

Teniendo en cuenta lo mencionado, se propone en el presente proyecto de tesis el diseño de un de robot móvil con direccionamiento diferencial autónomo para la desinfección de espacios cerrados utilizando emisión de rayos UV-C basado en

lógica difusa y un sistema de escaneo LiDAR, para lo cual se formulan las siguientes preguntas:

1.2. Formulación y delimitación del problema.

1.2.1. Problema General.

¿Cómo diseñar un robot móvil con direccionamiento diferencial usando lógica difusa y escaneo LiDAR para desinfección por radiación UV-C de espacios cerrados?

1.2.2. Problemas Específicos.

El presente proyecto de tesis tiene como objetivo final resolver los siguientes problemas específicos:

a) ¿Cómo diseñar el sistema mecánico del robot móvil, que le permita tener un desplazamiento eficiente en áreas con obstáculos?
b) ¿Cómo diseñar un sistema electrónico para el robot móvil con la autonomía necesaria para asegurar la potencia de emisión de rayos UV-C y el sistema eléctrico de alimentación adecuada?
c) ¿Cómo desarrollar la programación del robot móvil para la evasión de obstáculos basado en inteligencia artificial y sistema de escaneo LiDAR?

1.3. Objetivos

1.3.1. Objetivo General.

Diseñar un robot móvil con direccionamiento diferencial usando lógica difusa y escaneo LiDAR para desinfección por radiación UV-C de espacios cerrados.

1.3.2. Objetivos Específicos.

El proyecto de tesis tiene como objetivos específicos:

a) Diseñar el sistema mecánico del robot móvil, que le permita tener un desplazamiento eficiente en áreas con obstáculos.

b) Diseñar un sistema electrónico para el robot móvil con la autonomía necesaria para asegurar la potencia de emisión de rayos UV-C y el sistema eléctrico de alimentación adecuada.
c) Desarrollar la programación del robot móvil para la evasión de obstáculos basado en inteligencia artificial y sistema de escaneo LiDAR.

1.4. Alcances y Limitaciones de la Investigación

Se presentan a continuación las limitaciones del presente proyecto de tesis:

a) Debido al consumo energético por parte de los motores eléctricos DC del robot móvil y estado de tecnología aún limitada en el campo de baterías de litio o ácido, se considera un tiempo límite de autonomía, lo que se soluciona mediante la implementación de un módulo de carga rápida.
b) Se requiere tener una superficie limpia y clara frente para realizar un escaneo efectivo, especialmente cuando es utilizado en aparatos móviles, esto se podría mejorar mediante la implementación de un sensor con mejores prestaciones tecnológicas.
c) El tiempo de procesamiento del escaneo y mapeado del área en cuestión es ciertamente lenta, lo que podría agilizarse con la implementación de una arquitectura más eficiente y de mayor capacidad de memoria integrada.
d) Los obstáculos en el espacio a desinfectar que se encuentren imposibilitando o limitando la distancia entre la lámpara de LEDS UV-C y las superficies, disminuirán la efectividad de la correcta eliminación de los distintos agentes patógenos, lo que podría solucionarse mediante la implementación de una lámpara de mayor potencia o un control de potencia de esta que regule la intensidad en relación a la distancia a las distintas superficies.

1.5. Importancia y justificación del estudio

1.5.1. Importancia del estudio

La importancia de este trabajo de tesis está en el diseño de un robot móvil con direccionamiento diferencial autónomo que al ser implementado permitirá la desinfección de espacios cerrados utilizando emisión de rayos UV-C basado en lógica difusa y un sistema de escaneo LiDAR, el cual

permitirá dar solución al problema de la desinfección en hospitales, asimismo se lograría ahorrar costos respecto a la contratación de personal, además de eliminar riesgo a infectarse en el proceso de desinfección.

1.5.2. Justificación del estudio

a) Justificación tecnológica: El diseño del robot móvil autónomo sanitario al ser implementado utilizará un microprocesador y programación de Inteligencia Artificial (IA) para el mapeo de áreas automatizado, localización y transporte, además de la desinfección efectiva de superficies mediante la emisión de rayos UV-C en áreas de alta concurrencia de pacientes y personal médico.

b) Justificación económica: El diseño del robot móvil autónomo sanitario al ser implementado permitirá un ahorro en costo de operación debido a que no se requerirá de personal de limpieza y desinfección ya que el robot móvil es autónomo, además de no requerir de ninguna manipulación de un operario durante toda su actividad.

c) Justificación sanitaria: El diseño del robot móvil autónomo sanitario al ser implementado permitirá desinfectar las áreas de alta concurrencia por pacientes y personal médico, disminuyendo considerablemente su exposición y por ende probabilidad a contagiarse de COVID-19.

CAPÍTULO II: MARCO TEÓRICO

En el presente capítulo se describe el marco teórico de la investigación donde se mencionan los antecedentes más relevantes para su desarrollo, del mismo modo se presentan los fundamentos teóricos y científicos que sustentan y complementan las variables que se utilizaron en la investigación.

2.1. Investigaciones relacionadas con el tema

2.1.1. Antecedentes Internacionales

Ortega y Silva (2021). En su tesis para obtener el título de Ingeniero Mecatrónico, titulada "Sistema de navegación autónoma en robot móvil tipo oruga para apoyo en tareas de siembra en campos caficultores". Universidad autónoma de Bucaramanga, Bucaramanga, Colombia. Esta investigación tiene como objetivo el diseño de un sistema de navegación autónomo para un robot móvil instrumentado de tipo oruga, capaz de detectar su posición, calcular las trayectorias y comunicar los datos a través de una interfaz de mando para ayudar en las operaciones de plantación de los campos de café. Para ello se desarrolló una interfaz Hombre-Máquina con la cual se manipulará y evaluará el robot móvil además se desarrolló un algoritmo en base a la planeación de trayectorias para el campo de café establecido. Esta investigación concluye que el sistema de comunicación LORA es más efectivo que el bluetooth, wifi o de radio comunicación básicos, ya que puede enviar datos a largas distancias, sin necesidad de tener línea de vista.

Rodríguez (2019). En su tesis profesional para optar al título de Ingeniero Agrimensor, titulada "Diseño de metodología de trabajo para el escaneo con tecnología láser 3D, aplicada a la arquitectura patrimonial". Universidad de la República, Montevideo, Uruguay. En esta investigación se plantea el objetivo de definir una metodología a aplicar para el uso de equipos láser escáner topográficos terrestres 3D para la conservación y documentación del patrimonio construido. Para ello se hará un análisis de los errores en el registro de la nube de puntos usando los distintos métodos luego de aplicar la metodología para lograr la identificación y documentación de los restos de la antigua construcción denominada "Caserío de los Negros". Se

concluye la obtención de una metodología técnica, clara y ordenada que pueda servir como referencia para futuros trabajos que apliquen la tecnología láser escáner.

2.1.2. Antecedentes Nacionales

Hermoza (2018). Realizó la investigación para obtener el título de Ingeniero Industrial, titulada “Diseño de un sistema de desinfección de envases en el proceso de envasado de agua alcalina “Andea” de la empresa Cervecerías Cusco S.A.C. 2018.”. Universidad Andina del Cusco – Cusco, Perú. Esta investigación tiene como objetivo el desarrollo de un diseño ingenieril de un sistema de desinfección de envases que utiliza la radiación ultravioleta generada por lámparas germicidas UVC para higienizar los envases de la empresa Cerveceras Cusco S.A.C. 2018 en sus dos presentaciones de PET y Vidrio. Se utilizó SolidWorks 2017 en el diseño de un sistema de desinfección de contenedores mediante radiación ultravioleta, así como un sistema de recogida de datos correctamente estructurado según las exigencias y objetivos del trabajo de estudio. La aplicación de la radiación ultravioleta como agente de desinfección resultó ser eficiente y eficaz en este estudio.

Chávez (2020). Desarrolló la investigación para obtener el grado de Ingeniero Electrónico, titulada “Sistema de detección y evasión de obstáculos por medio de un LIDAR 360° para un sistema aéreo no tripulado”. Pontificia Universidad Católica del Perú - Lima, Perú. Esta investigación tiene como objetivo desarrollar un módulo electrónico capaz de detectar y evitar que el sistema aéreo no tripulado colisione con obstáculos durante el vuelo en escenarios como bordear una estructura fija o evitar una colisión inminente con un objeto intermedio, evitando así accidentes y la pérdida de costosos bienes. Para ello se diseñó una estructura compacta y ligera capaz de contener los dispositivos y permitir que el proceso de ensamblaje al sistema aéreo no tripulado sea sencillo además se realizó un arreglo de sensores que permitan obtener información del entorno. Este estudio concluye que se logró desarrollar un módulo electrónico con el propósito de evitar colisiones del sistema aéreo no

tripulado en escenarios difíciles, en consecuencia, se desarrolló un algoritmo de adaptación de rutas el cual permite a la computadora acompañante procesar la información procedente del sensor.

2.2. Estructura teórica y científica que sustenta el estudio

2.2.1. Robots móviles

Aunque no existe un significado universalmente reconocido para la expresión "robot móvil", se entiende comúnmente que se refiere a un dispositivo que puede desplazarse de forma independiente de un lugar a otro para cumplir una serie de objetivos. En las fábricas están los vehículos de guiado automático, en las operaciones militares se puede encontrar los vehículos de reconocimiento terrestre no tripulados, en la asistencia sanitaria está la entrega de productos farmacéuticos, en la búsqueda y el rescate se encuentran como guardias de seguridad, y en los hogares, por ejemplo, se utilizan en la limpieza del suelo y el corte del césped, los robots móviles se utilizan en una amplia gama de aplicaciones. Los robots móviles tienden a contar con desplazamiento y ejecutar tareas en entornos amplios e imprevisibles. Deben hacer frente a circunstancias que no son predecibles de antemano y que evolucionan con el tiempo. En este tipo de situaciones se encuentran criaturas imprevisibles como los seres humanos y los animales. Las aspiradoras robóticas y los vehículos con piloto automático son ejemplos de robots móviles.

Hay tres escenarios principales para los robots móviles que necesitan principios de diseño considerablemente diversos: acuáticos, terrestres y aéreos. Dicha clasificación no es invariable; por ejemplo, hay robots anfibios que pueden desplazarse tanto por tierra como por agua. Los robots terrestres pueden tener patas, ruedas u orugas, mientras que los robots aéreos pueden tener ala fija como aviones o ala rotatoria como helicópteros. Muchos robots móviles se manejan a distancia y realizan trabajos que necesitan un operador, como la inspección de tuberías, fotografía aérea y la desactivación de bombas. Estos robots no son autónomos, sino que utilizan sus sensores para proporcionar a sus operadores acceso remoto a lugares

arriesgados, remotos o inaccesibles. Algunos son semiautónomos y realizan tareas secundarias por sí mismos. El piloto automático de un dron mantiene la estabilidad mientras el piloto determina la ruta de vuelo. Mientras una persona busca los fallos que hay que corregir, un robot en una tubería puede regular su movimiento dentro de la misma.

Los robots móviles totalmente autónomos no necesitan la ayuda de un operador humano; en su lugar, toman decisiones y realizan tareas por sí mismos, como transportar materiales mientras navegan por un terreno incierto (cruces en las calles, paredes y puertas dentro de los edificios) y en un entorno en constante cambio (gente caminando, coches circulando por las calles). Los primeros robots móviles se crearon para tareas sencillas, como la limpieza de piscinas o el manejo de cortacéspedes robotizados. Dado que se ha demostrado que es viable construir robots de coste razonable que puedan atravesar un entorno interior repleto de obstáculos, los robots aspiradores son ahora comúnmente accesibles.

Muchos robots móviles autodirigidos están diseñados para ayudar a los profesionales en zonas organizadas como los almacenes. Un robot para deshierbar campos es un ejemplo interesante, como se observa en la figura 1. Aunque este entorno está algo organizado, es necesario mejorar la detección para ejecutar actividades como la identificación y la eliminación de las malas hierbas. Los robots comparten el entorno con las personas, incluso en industrias muy organizadas, por lo que su detección debe ser extremadamente fiable. El vehículo auto conducido que más atención está recibiendo en estos días es quizá el robot móvil autónomo, debido a su aplicación en entornos complicados e imprevisibles como el tráfico motorizado, donde hay rigurosas normas de seguridad que respetar.

Figura 1: Desbrozadora autónoma de la empresa suiza ecoRobotix
Fuente: Balibouse. (2018)

La robótica móvil es un amplio campo de estudio que suele distinguirse de la robótica convencional sólo por su capacidad de movimiento. Por otra parte, los robots móviles requieren características especiales, como la navegación. A la hora de diseñar un robot móvil, hay que tener en cuenta otras consideraciones limitantes, como la energía, generalmente restringida.

El espacio es un entorno mucho más desafiante y peligroso. Los vehículos de Marte, Sojourner y Curiosity, son robots móviles semiautónomos. En 1997, el Sojourner estuvo activo durante tres meses. El Rover Curiosity, en contraste, ha estado activo desde que aterrizó en Marte en 2012. Si bien las misiones están controladas por un conductor humano en la Tierra (las rutas a recorrer y las investigaciones científicas a realizar), los rovers son capaces de evitar el peligro de forma autónoma.

Gran parte de la investigación y el desarrollo de la robótica actual se centra en hacer que los robots sean más autónomos mejorando los sensores y permitiendo un control más sofisticado de los mismos. La mejora de los sensores permite discernir los entresijos de circunstancias cada vez más complicadas, pero el control del comportamiento del robot en estos entornos requiere flexibilidad y adaptabilidad. Como las cámaras son baratas y los datos que pueden recoger son abundantes, la visión es un tema de investigación especialmente activo. Se están desarrollando sistemas más

flexibles para que puedan aprender de los humanos o adaptarse a nuevas condiciones. La interacción entre las personas y los robots es otra área de investigación activa. Esto requiere tanto la percepción como el intelecto, así como la comprensión de la psicología y la sociología de las relaciones.

Los vehículos con ruedas son la solución más simple y eficiente para conseguir movilidad en terrenos suficientemente duros y libres de obstáculos, permitiendo conseguir velocidades relativamente altas. Como limitación más significativa cabe mencionar el deslizamiento en la impulsión. Dependiendo de las características del terreno pueden presentarse también deslizamientos y vibraciones. La locomoción mediante ruedas es poco eficiente en terrenos blandos. A continuación, se comentarán las características más importantes del sistema de locomoción planteado en este trabajo de tesis.

a) Robot móvil con direccionamiento diferencial

El robot con tracción diferencial cuenta con dos motores que accionan cada rueda como se aprecia en la figura 2. No existe una relación predeterminada entre la velocidad del robot y la potencia del motor. Diferentes relaciones de transmisión pueden vincular el motor a las ruedas, el tipo de neumático de las ruedas influye en la tracción y el terreno arenoso o con barro puede hacer que las ruedas patinen.

En la figura 2 se muestra una vista del robot desde arriba. La parte delantera del robot es la curva de la derecha, que también representa la movilidad hacia delante del robot. En los lados izquierdo y derecho del cuerpo del robot están las ruedas (rectángulos negros). El punto en el eje es la ubicación en el centro del eje a medio camino entre las ruedas. El robot gira a lo largo de un eje que es vertical a este punto mientras gira. En la parte delantera del robot se encuentra una rueda loca de apoyo o no accionada que le da estabilidad.

En ingeniería mecánica, una línea discontinua representa el eje de simetría en un componente como una rueda. La intersección de los dos ejes de simetría representa el eje de rotación que es perpendicular al plano de la página cuando se ve una vista lateral de una rueda. Simplificamos la notación mostrando simplemente líneas discontinuas para el eje de rotación de un componente, como una rueda, para evitar el desorden de los diagramas. Además, la intersección que significa un eje perpendicular se abrevia a veces como una cruz, que puede encontrarse en la rueda o en su eje.

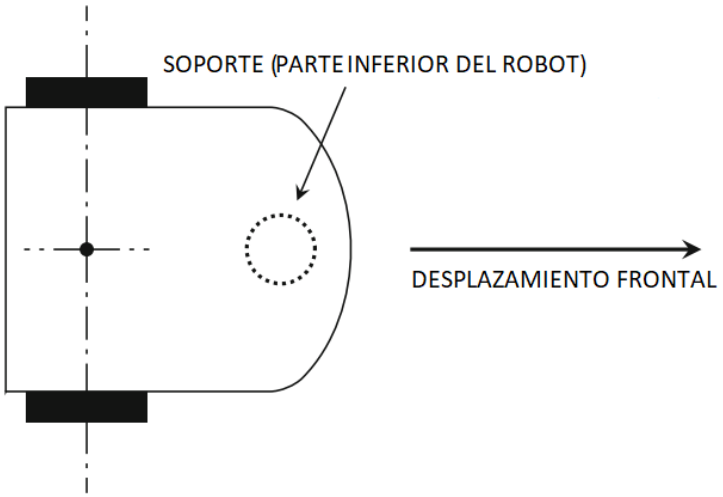


Figura 2: Robot con direccionamiento diferencial.
Fuente: Elaboración propia.

El accionamiento diferencial tiene numerosas ventajas: es sencillo, ya que sólo utiliza dos motores y no requiere ningún componente de dirección adicional, y permite que el robot gire en su sitio. En un automóvil se accionan dos ruedas (o cuatro ruedas por parejas) y existe un complicado sistema de dirección independiente denominado dirección Ackermann, como un coche no puede girar en su sitio, los conductores deben realizar maniobras complejas, como aparcar en paralelo, que los conductores humanos pueden aprender rápidamente pero que son difíciles de dominar para un sistema autónomo. La conducción diferencial es la disposición recomendada para los robots autónomos que necesitan realizar maniobras complicadas con movimientos extremadamente

sencillos: Simplemente puede cambiar su orientación y seguir en esa dirección.

El mayor inconveniente de un sistema de tracción diferencial es que necesita un tercer punto de contacto con el suelo, mientras que un automóvil ya cuenta con cuatro ruedas de apoyo y, por lo tanto, puede circular por terrenos accidentados con facilidad. También tiene la desventaja de no poder conducir lateralmente sin girar. Existen configuraciones que permiten a un robot desplazarse lateralmente, pero son difíciles y caras de implementar. Los vehículos sobre orugas, como los equipos de movimiento de tierras y los tanques militares, también emplean la tracción diferencial. Estos vehículos pueden operar en terrenos muy difíciles, pero sus orugas generan mucha fricción, lo que hace que la movilidad sea lenta e imprevisiblemente precisa.

2.2.2. Sistema LiDAR

Es un sensor activo, lo que significa que envía una onda electromagnética (EM) en longitudes de onda infrarrojas y luego recibe la señal reflejada, de forma similar al radar de microondas, pero con una longitud de onda mucho más corta. Como resultado, tendrá una resolución angular mucho mejor que la del radar, pero no funcionará con niebla o nubes. En cuanto a las longitudes de onda, es comparable a los sensores electroópticos pasivos, salvo que genera su propia radiación en lugar de depender de la luz ambiental, y ofrece muchos más modos de detección gracias a la capacidad de regular la iluminación de la escena. Como un sistema LIDAR tiene su propia iluminación, se puede ver en la oscuridad utilizando longitudes de onda del infrarrojo cercano. La figura 3 representa el espectro electromagnético, que sitúa un sistema LIDAR y los dispositivos electroópticos en general en el contexto de las ondas electromagnéticas. Las longitudes de onda de los espectros visible e infrarrojo son más cortas que las de las ondas de radio y las microondas, pero más largas que las de los rayos X y los rayos gamma. Como la figura anterior está en una escala

logarítmica, la diferencia de longitud de onda es significativa. (McManamon, 2019).

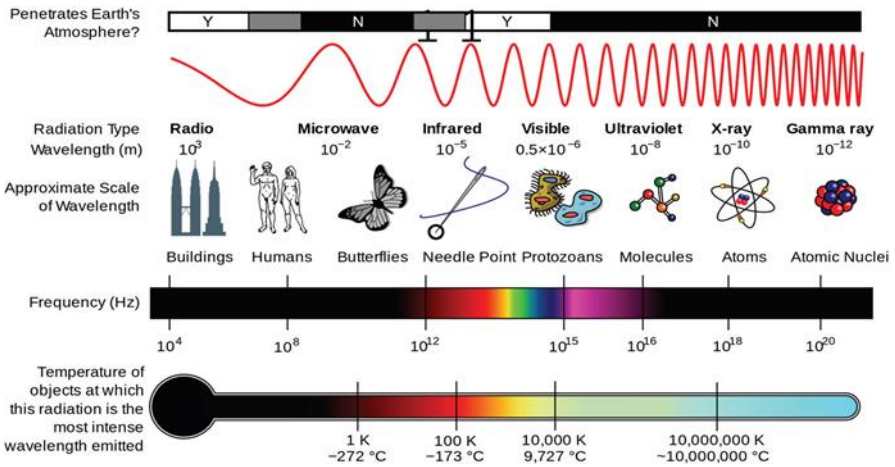


Figura 3: Espectro electromagnético y el tamaño de objetos comunes.
McManamon. (2019)

La figura 4 muestra el concepto principal de funcionamiento del LiDAR. Los principales componentes del sistema están etiquetados con números que van del 1 al 5. El transmisor (1) genera un rayo láser, que luego es cambiado a horizontal por un espejo giratorio reflectante (2). El grupo de haces reflejados vuelve al LiDAR tras encontrar un obstáculo (3). Los haces son redirigidos por el espejo (2) hacia el medio de enfoque (4), que los transmite al detector (5).

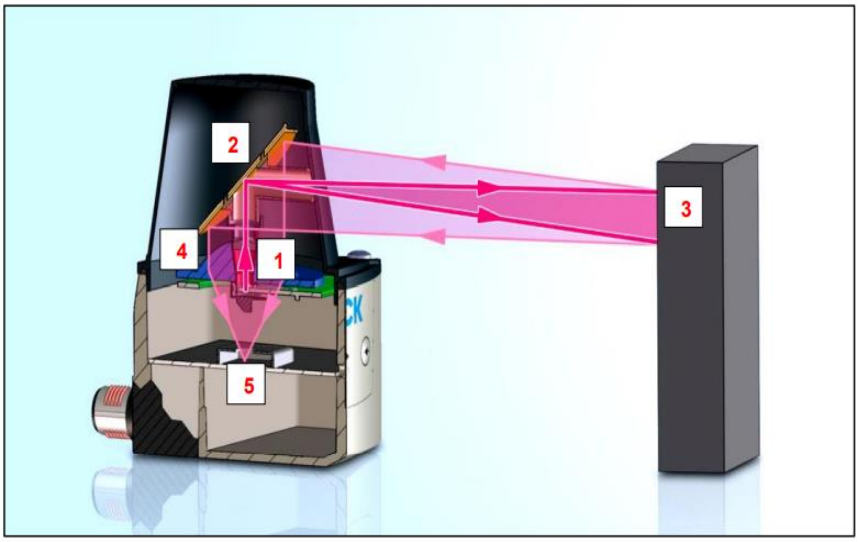


Figura 4: Funcionamiento de un sensor LiDAR.
Fuente: Sick. (2018)

En comparación con los dispositivos convencionales de medición de distancias, los sensores LiDAR tienen la ventaja de realizar mediciones a su alrededor con un amplio ángulo de barrido y una alta resolución angular. Esto implica que el LiDAR puede medir casi rápidamente la distancia entre el LiDAR y los objetos circundantes en el entorno. Para lograr esta funcionalidad se utiliza un espejo giratorio dentro del dispositivo LiDAR. El orden secuencial de los pulsos láser se coordina con la frecuencia de rotación del motor y la resolución angular necesaria. Este método emplea el efecto de un gran número de pulsos individuales, que luego se combinan entre sí utilizando un patrón de transmisión conocido y métodos estadísticos para proporcionar información sobre la distancia y la señal de eco.

El sensor LIDAR realiza la medición láser sin contacto que utiliza pulsos de luz para escanear el perímetro circundante de forma radial en un solo plano. Realizando mediciones bidimensionales emitiendo pulsos láser a su alrededor (ángulo de exploración máximo de 190°) y midiendo las señales recibidas de vuelta. En la localización del elemento, incluyendo su distancia y ángulo, como se ve en la figura 5, no se puede escanear los alrededores a través de los objetos.

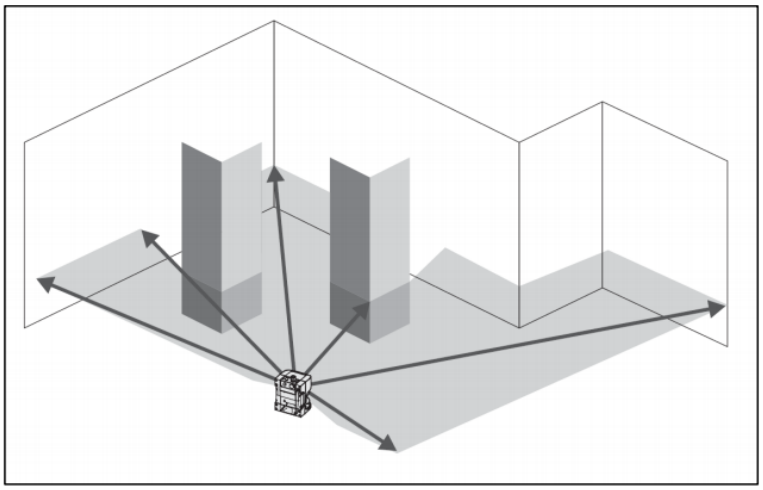


Figura 5:Principio de medición de un sensor LiDAR.
Fuente: Sick. (2018)

- Visión general de los sensores para el sistema de cartografía móvil en interiores.

  La recopilación de datos de la nube de puntos 3D es una condición previa para iniciar el proceso de mapeo. Se debe elegir el sensor adecuado antes de poder generar la solución cartográfica final. En esta sección se describen brevemente varios sensores para el mapeo 3D de entornos interiores.

  El primer sensor es un escáner láser 2D, que mide las distancias mediante luz láser. El rayo láser gira, o más comúnmente, se gira un espejo, para medir las distancias en diferentes direcciones. El ángulo del vértice se discretiza con varias resoluciones, como 1, 0,5 o 0,25 grados, y suele ser de 90, 180, 270 o 360 grados. Se utilizan dos escáneres láser 2D para adquirir datos 3D. Uno de los escáneres se coloca en posición vertical y el otro en posición horizontal en la configuración del sistema. Utilizando la posición actual del robot en 3D, las líneas de escaneo vertical y horizontal se registran en una coordenada común (Zhao y Shibasaki, 2001).

  La figura 6 muestra la nube de puntos adquirida a partir de varios sensores, como el escáner láser SICK LMS200, el sensor RGB-D, el escáner láser 3D RIEGL VZ-400, la cámara térmica Velodyne HDL-32 y el escáner láser SICK LMS200. El Velodyne HDL-32E genera escaneos de alcance en 3D haciendo girar un conjunto de 32 haces alrededor de su eje vertical a 10 Hz, proporcionando hasta 700.000 puntos por segundo o 2.200 puntos por láser en el rango de uno a 70 metros. El conjunto tiene una resolución angular de unos 0,16 grados en la dirección horizontal y un campo de visión (FOV) de 360 grados. Los ángulos de inclinación vertical varían de -30,67 a +10,67 grados, con una precisión angular de 1,33 grados. La precisión de su medición de alcance es generalmente de 2 cm.

  Una de las desventajas de utilizar el Velodyne HDL-32E es el espacio entre las sucesivas líneas de escaneo láser, que aumenta la dispersión

(los puntos adyacentes se separan más) de las nubes de puntos registradas a medida que aumenta la distancia del sensor. Otro problema es la limitada resolución angular vertical, que hace que los métodos estándar de registro de nubes de puntos proporcionen resultados pobres. En consecuencia, deberían explorarse diversas técnicas previas, como el registro grueso con sensores auxiliares, para mejorar los resultados.

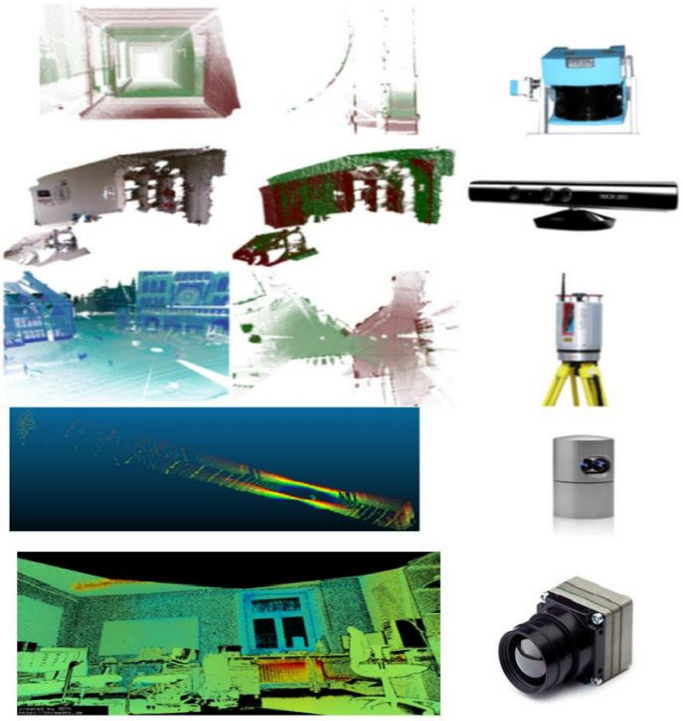


Figura 6: Mapeo de puntos en una nube generado por diferentes sensores.

Fuente: Mostofi. (2015)

2.2.3. Sistema operativo de Robot (ROS)

Es un sistema operativo flexible para escribir software de robot. Es una colección de herramientas, bibliotecas y convenciones que tienen como objetivo simplificar la tarea de crear un comportamiento robótico complejo y robusto en una amplia variedad de plataformas robóticas. Crear un software robótico de propósito general verdaderamente robusto es difícil. Desde la perspectiva del robot, los problemas que parecen triviales para los humanos a menudo varían enormemente entre instancias de tareas y entornos. Hacer frente a estas variaciones es tan difícil que ningún individuo, laboratorio o institución puede esperar hacerlo por sí solo. Como

resultado, ROS se creó desde cero para fomentar el desarrollo de software de robótica colaborativa. Por ejemplo, un laboratorio podría tener expertos en mapeo de ambientes interiores y podría contribuir con un sistema de clase mundial para producir mapas. Otro grupo podría tener expertos en el uso de mapas para navegar, y otro grupo podría haber descubierto un enfoque de visión por computadora que funciona bien para reconocer objetos pequeños en el desorden.

- Componentes principales

  Si bien no se proporciona una lista exhaustiva de lo que hay en el ecosistema ROS, se puede identificar algunas de las partes centrales de ROS y hablar sobre su funcionalidad, especificaciones técnicas y calidad.

  a) Infraestructura de Comunicaciones

     En el nivel más bajo, ROS ofrece una interfaz que proporciona comunicación entre procesos y se conoce comúnmente como middleware. El middleware ROS proporciona estas facilidades:

     - Publicar el paso de mensajes anónimos.
     - grabación y reproducción de mensajes.
     - solicitar y responder llamadas a procedimientos remotos.
     - Sistema de parámetros distribuidos.

  b) Paso de mensajes

     Un sistema de comunicación es a menudo una de las primeras necesidades que surgen al implementar una nueva aplicación de robot. El sistema de mensajería integrado y probado de ROS ahorra tiempo al administrar los detalles de la comunicación entre los nodos distribuidos a través del mecanismo de publicación y suscripción anónima. Otro beneficio de usar un sistema de paso de mensajes es que obliga a implementar interfaces claras entre los nodos de su sistema, mejorando así la encapsulación y promoviendo la

reutilización del código. La estructura de estas interfaces de mensajes se define en el mensaje IDL (Lenguaje de descripción de interfaces).

c) Grabación y reproducción de mensajes

Debido a que el sistema de publicación y suscripción es anónimo y asincrónico, los datos se pueden capturar y reproducir fácilmente sin ningún cambio en el código. Supongamos que tiene la Tarea A que lee datos de un sensor y está desarrollando la Tarea B que procesa los datos producidos por la Tarea A. ROS facilita la captura de los datos publicados por la Tarea A en un archivo y luego vuelve a publicar esos datos desde el archivo en un momento posterior. La abstracción de paso de mensajes permite que la Tarea B sea independiente con respecto a la fuente de los datos, que podría ser la Tarea A o el archivo de registro. Este es un patrón de diseño poderoso que puede reducir significativamente su esfuerzo de desarrollo y promover la flexibilidad y modularidad en un sistema.

d) Llamadas a procedimientos remotos

La naturaleza asincrónica de la mensajería de publicación y suscripción funciona para muchas necesidades de comunicación en robótica, pero a veces desea interacciones de solicitud y respuesta sincrónica entre procesos. El middleware ROS proporciona esta capacidad mediante servicios. Al igual que los temas, los datos que se envían entre procesos en una llamada de servicio se definen con el mismo IDL de mensaje simple.

e) Sistema de parámetros distribuidos

El middleware ROS también proporciona una forma para que las tareas compartan información de configuración a través de un almacén de valores clave global. Este sistema permite modificar fácilmente la configuración de sus tareas e incluso permite que las tareas cambien la configuración de otras tareas.

f) Funciones específicas del robot

Además de los componentes centrales del middleware, ROS proporciona bibliotecas y herramientas comunes específicas para robots que harán que su robot esté listo y funcionando rápidamente. Estas son algunas de las capacidades específicas de los robots que posee ROS:

- Definiciones de mensajes estándar para robots.
- Biblioteca de geometría de robots.
- Lenguaje de descripción del robot.
- Llamadas a procedimientos remotos interrumpibles.
- Diagnósticos.
- Estimación de pose.
- Localización.
- Cartografía.
- Navegación.

g) Mensajes de robot estándar

Años de discusión y desarrollo de la comunidad han llevado a un conjunto de formatos de mensajes estándar que cubren la mayoría de los casos de uso comunes en robótica. Hay definiciones de mensajes para conceptos geométricos como poses, transformaciones y vectores; para sensores como cámaras, IMU y láseres; y para datos de navegación como odometría, rutas y mapas; entre muchos otros. Al usar estos mensajes estándar en su aplicación, su código interactúa sin problemas con el resto del ecosistema ROS, desde herramientas de desarrollo hasta bibliotecas de capacidades.

h) Biblioteca de geometría de robots

Un desafío común en muchos proyectos de robótica es realizar un seguimiento de dónde se encuentran las diferentes partes del robot entre sí. Por ejemplo, si desea combinar datos de una cámara con datos de un láser, necesita saber dónde está cada sensor, en algún marco de

referencia común. Este problema es especialmente importante para los robots humanoides con muchas partes móviles como se observa en la figura 7. Abordamos este problema en ROS con la biblioteca tf (transform), que hará un seguimiento de dónde está todo en su sistema de robot.

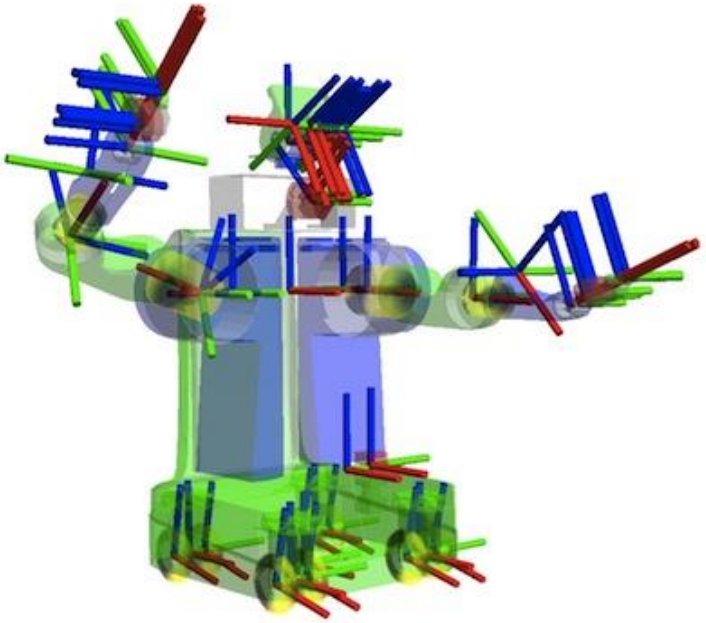


Figura 7:Robot humanoide con muchas partes móviles
Fuente: ROS. (2021)

Diseñada teniendo en cuenta la eficiencia, la biblioteca tf se utiliza para administrar datos de transformación de coordenadas para robots con más de cien grados de libertad y tasas de actualización de cientos de Hertz. La biblioteca tf le permite definir tanto transformaciones estáticas, como una cámara fija a una base móvil, como transformaciones dinámicas, como una articulación en un brazo robótico. Puede transformar los datos del sensor entre cualquier par de marcos de coordenadas del sistema. La biblioteca tf maneja el hecho de que los productores y consumidores de esta información pueden estar distribuidos a través de la red y el hecho de que la información se actualiza a diferentes velocidades.

i) Lenguaje de descripción del robot

Otro problema común de robótica que ROS resuelve es cómo describir su robot de una manera legible por máquina. ROS proporciona un conjunto de herramientas para describir y modelar su robot para que pueda ser entendido por el resto de su sistema ROS, incluidos tf, robot_state_publisher y rviz. El formato para describir su robot en ROS es URDF (Unified Robot Description Format), que consiste en un documento XML en el que describe las propiedades físicas de su robot, desde la longitud de las extremidades y el tamaño de las ruedas hasta la ubicación de los sensores y la Apariencia visual de cada parte del robot. Una vez definido de esta manera, su robot puede usarse fácilmente con la biblioteca tf, renderizarse en tres dimensiones para visualizaciones agradables y usarse con simuladores y planificadores de movimiento.

j) Llamadas a procedimientos remotos interrumpibles

Si bien los temas (publicación y suscripción anónima) y los servicios (llamadas a procedimientos remotos) cubren la mayoría de los casos de uso de comunicación en robótica, a veces es necesario iniciar un comportamiento de búsqueda de objetivos, monitorear su progreso, poder adelantarse a él en el camino y recibir una notificación cuando esté completo. ROS proporciona acciones para este propósito. Las acciones son como los servicios, excepto que pueden informar el progreso antes de devolver la respuesta final, y la persona que llama puede apropiarse de ellas. Entonces, por ejemplo, puede indicarle a su robot que navegue a alguna ubicación, monitorear su progreso mientras intenta llegar allí, detenerlo o redirigirlo en el camino y recibir información cuando haya tenido éxito (o fallado). Una acción es un concepto poderoso que se utiliza en todo el ecosistema ROS.

k) Diagnósticos

ROS proporciona una forma estándar de producir, recopilar y agregar diagnósticos sobre el robot además determinar cómo abordar los problemas a medida que surgen como se observa en la figura 8.

Figura 8:Captura de pantalla de componente de diagnóstico de ROS
Fuente: ROS. (2021)

l) Posesión, localización y navegación

ROS también proporciona algunas capacidades que ayudan a comenzar con un proyecto de robótica. Existen paquetes ROS que resuelven problemas básicos de robótica como estimación de pose, localización en un mapa, construcción de un mapa e incluso navegación móvil como se observa en la figura 9.

Si se busca hacer una investigación y desarrollo rápidos, un investigador de robótica que desea realizar su investigación de manera oportuna o un aficionado que busca aprender más sobre robótica, estas capacidades listas para usar ayudan a hacer más con menos esfuerzo.

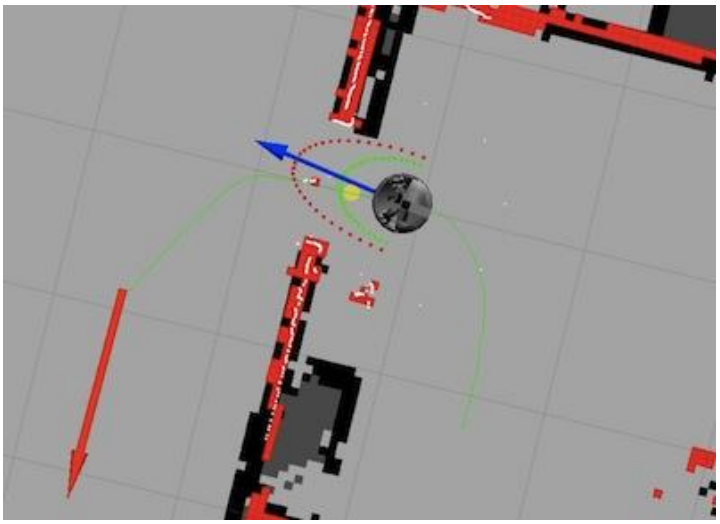


Figura 9:Captura de navegación de robot móvil en ROS
Fuente: ROS. (2021)

m) Instrumentos

Una de las características más sólidas de ROS es el poderoso conjunto de herramientas de desarrollo. Estas herramientas admiten la introspección, la depuración, el trazado y la visualización del estado del sistema que se está desarrollando. El mecanismo de publicación y suscripción subyacente le permite realizar una introspección espontánea de los datos que fluyen a través del sistema, lo que facilita la comprensión y la depuración de problemas a medida que ocurren. Las herramientas ROS aprovechan esta capacidad de introspección a través de una amplia colección de utilidades gráficas y de línea de comandos que simplifican el desarrollo y la depuración.

n) Herramientas de línea de comandos

Todas las funciones básicas y las herramientas de introspección son accesibles a través de una de más de 45 herramientas de línea de comandos. Hay comandos para lanzar grupos de nodos; introspección de temas, servicios y acciones; grabación y reproducción de datos; y muchas otras situaciones. Si prefiere utilizar herramientas gráficas, rviz y rqt proporcionan una funcionalidad similar y extendida.

o) rviz

Quizás la herramienta más conocida en ROS, proporciona visualización tridimensional de propósito general de muchos tipos de datos de sensores y cualquier robot descrito por URDF. Rviz puede visualizar muchos de los tipos de mensajes comunes proporcionados en ROS, como escaneos láser, nubes de puntos tridimensionales e imágenes de cámara como se observa en la figura 10. También utiliza información de la biblioteca tf para mostrar todos los datos del sensor en un marco de coordenadas común, junto con una representación tridimensional del robot. Visualizar todos los datos en la misma aplicación no solo se ve impresionante, sino que también permite ver rápidamente lo que ve el robot e identificar problemas como desalineaciones de sensores o imprecisiones en el modelo del robot.

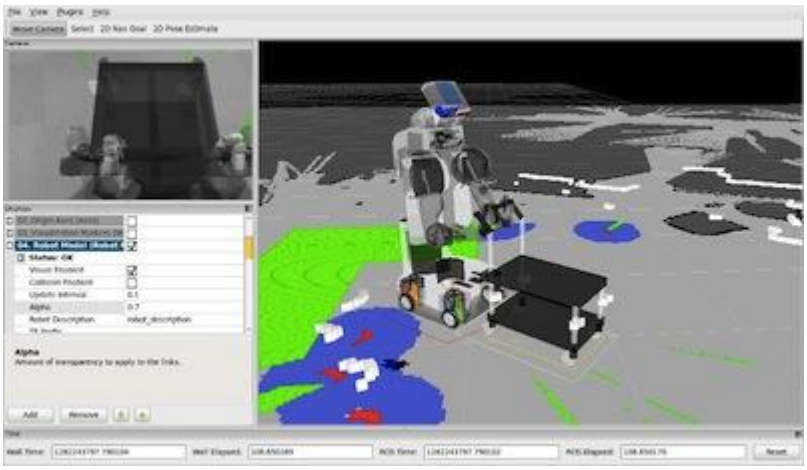


Figura 10: Visualización tridimensional en componente rviz de ROS

Fuente: ROS. (2021)

p) rqt

ROS proporciona rqt, un marco basado en Qt para desarrollar interfaces gráficas para el robot. Puede crear interfaces personalizadas componiendo y configurando la extensa biblioteca de complementos rqt incorporados en diseños con pestañas, pantalla dividida y otros como se observa en la figura 11. También se puede introducir nuevos componentes de interfaz escribiendo complementos propios rqt.

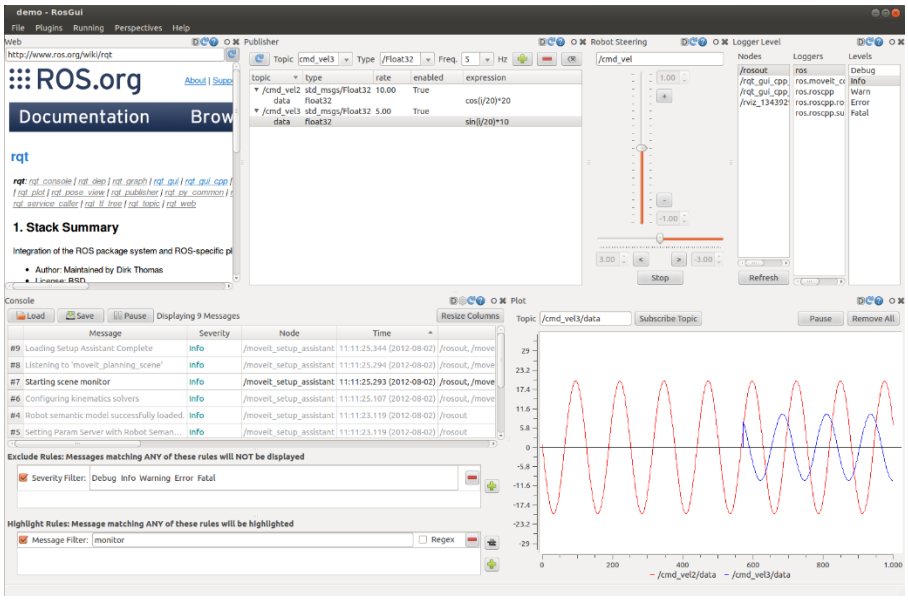


Figura 11:Complemento rqt incorporados en diseños con pestañas.

Fuente: ROS. (2021)

El complemento rqt_graph proporciona introspección y visualización de un sistema ROS en vivo, mostrando los nodos y las conexiones entre ellos como se observa en la figura 12, y permitiendo depurar y comprender fácilmente el sistema en ejecución y cómo está estructurado.

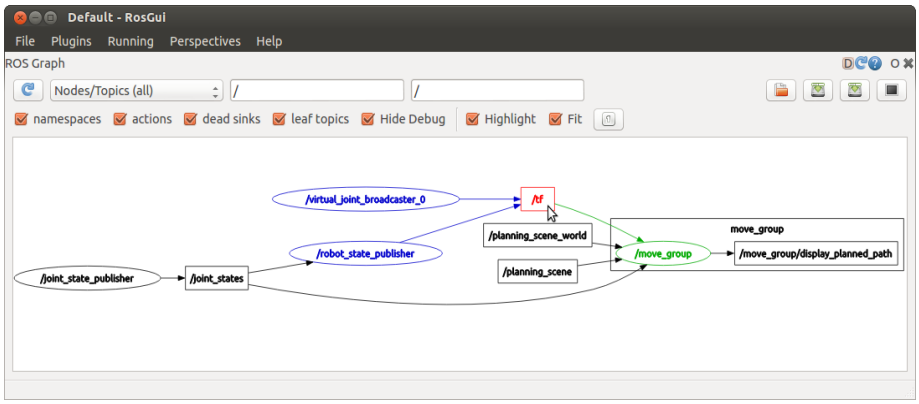


Figura 12: Captura de nodos y conexiones en ROS

Fuente: ROS. (2021)

Con el complemento rqt_plot observado en la figura 13 se puede monitorear codificadores, voltajes o cualquier variable que pueda representarse como un número que varía con el tiempo. El complemento rqt_plot permite elegir el backend de trazado (por

ejemplo, matplotlib, Qwt, pyqtgraph) que mejor se adapte a las necesidades.

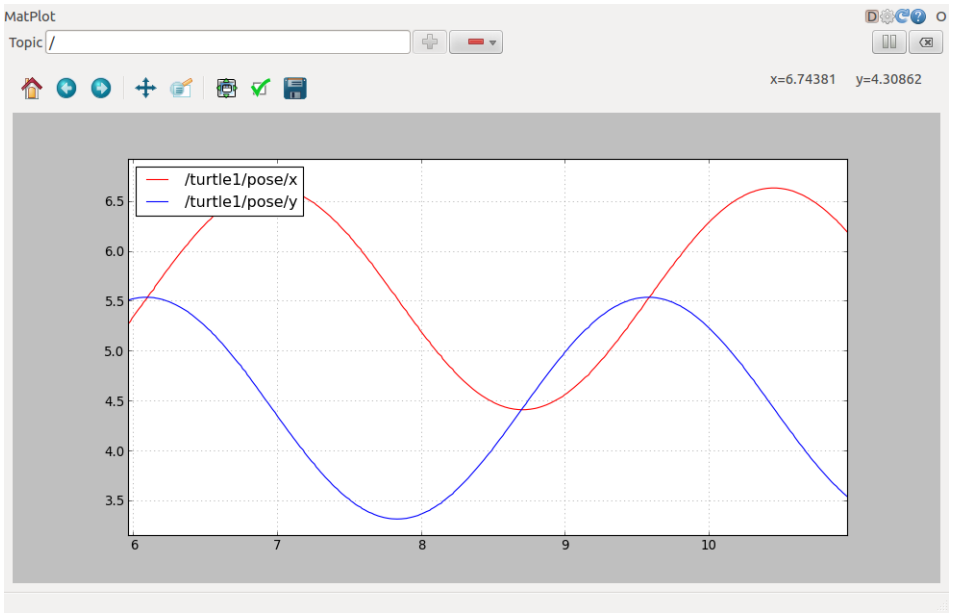


Figura 13: rqt_plot para monitorear respecto del tiempo.

Fuente: ROS. (2021)

Para monitorear y usar temas, hay complementos rqt_topic y rqt_publisher. El primero permite monitorear y revisar cualquier cantidad de temas que se publiquen dentro del sistema como se observa en la figura 14. Este último permite publicar mensajes propios sobre cualquier tema, facilitando la experimentación con el sistema.

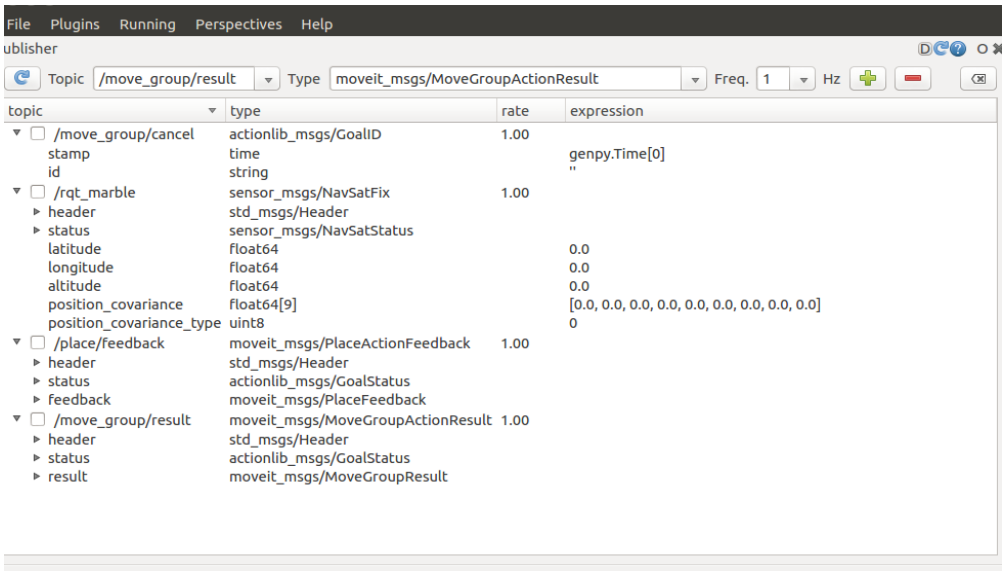


Figura 14: Monitoreo para cualquier tema que se publique en el sistema.

Fuente: ROS. (2021)

Para el registro y la reproducción de datos, ROS usa el formato de bolsa. Los archivos Bag se pueden crear y acceder a ellos gráficamente a través del complemento rqt_bag. Este complemento puede registrar datos en bolsas, reproducir temas seleccionados de una bolsa y visualizar el contenido de una bolsa, incluida la visualización de imágenes y el trazado de valores numéricos a lo largo del tiempo como se observa en la figura 15.

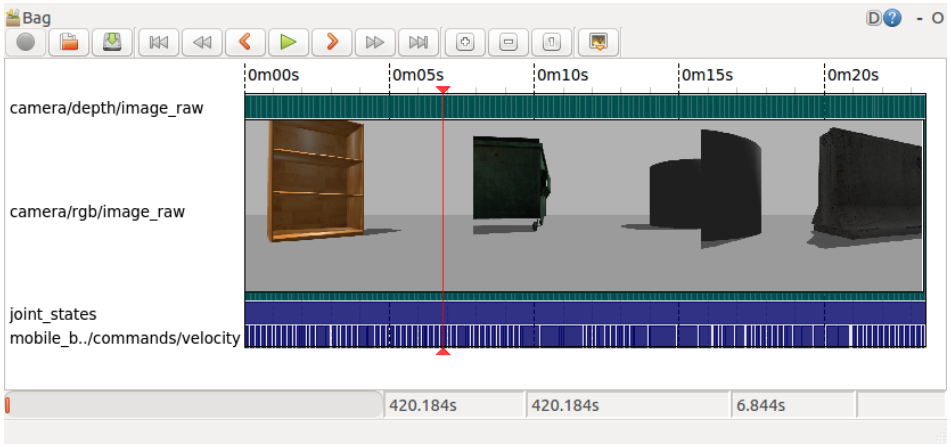


Figura 15: Complemento Bag

Fuente: ROS. (2021)

2.2.4. Inteligencia Artificial

La inteligencia artificial (IA) es un campo de la informática que trabaja con símbolos y técnicas no algorítmicas de resolución de problemas. La IA puede considerarse un dialecto simbólico compuesto por cadenas de caracteres que representan nociones del mundo real. Los procesos simbólicos son, por supuesto, una característica clave de la Inteligencia Artificial. Hay varios elementos que componen la ciencia de la IA, con tres categorías principales: la lógica difusa, las redes neuronales artificiales y los algoritmos genéticos. Cada uno de ellos tiene cualidades únicas, así como una función distinta. (Ponce,2010).

- Lógica Difusa

  La lógica difusa es un campo de la inteligencia artificial que permite a un ordenador evaluar datos del entorno natural en una relación de falso a verdadero. En la década de 1920, los matemáticos que trabajaban en la

lógica desarrollaron un concepto vital: todo es una cuestión de grado. Los ingenieros pueden construir televisores, aparatos de aire acondicionado y otros artilugios que evalúan información difícil de definir utilizando la lógica difusa, que manipula nociones imprecisas como "caliente" o "húmedo". Los sistemas difusos adoptan un enfoque diferente a los antiguos conceptos griegos de pertenencia y lógica. Nociones no precisas como "hace frío" o "el precio es alto" se manejan con el lenguaje natural. La profundidad del significado se pierde cuando el lenguaje humano se traduce al entorno de la lógica clásica, lo que podría ser significativo a la hora de crear un sistema inteligente. Teniendo en cuenta que un sistema inteligente pretende imitar la capacidad de diagnóstico de un médico, el ingeniero es consciente de que, mientras el médico depende de mediciones precisas, el diagnóstico y la prescripción de medicamentos están cargados de incertidumbre. (Ponce, 2010).

En la lógica booleana, las operaciones básicas (unión, intersección y complemento) se especifican mediante operadores binarios; en la lógica difusa, se presentan los operadores utilizados para realizar las operaciones básicas (unión, intersección y complemento). Como se ilustra en la figura 16, la intersección es una norma triangular (norma T), la unión es una conorma T (o norma S), y la intersección es una operación de álgebra y binaria en el rango [0,1].

```
Gráfica triangular

for time=1:10,
if (time<alfa)
      y(time)=0;
elseif and(alfa<=time,time<=beta)
      y(time)=(time-alfa)/(beta-alfa);elseif and (beta <= time, time <= gamma)
      y(time)=((gamma-time)/(gamma-beta));
elseif (time>gamma)
      y(time)=0;
end
end
plot (y)
```

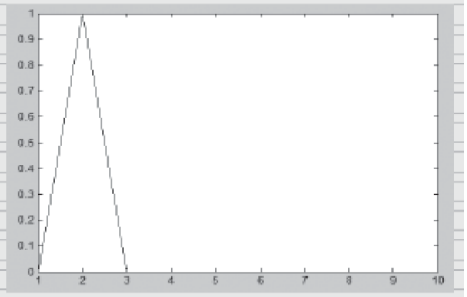


Figura 16: Operación difusa triangular elaborada con MATLAB.
Fuente: Ponce. (2010)

2.2.5. Localización y Mapeo simultáneo (SLAM)

El SLAM (acrónimo de Simultaneous Localization and Mapping) comprende la estimación simultánea del estado de un robot equipado con sensores integrados, y la construcción de un modelo (mapa) del entorno que los sensores están percibiendo (Cadena et al., 2016).

En casos simples, el estado del robot es descrito por su pose (posición y orientación), aunque se puede incluir una mayor información en dicho estado, como la velocidad del robot, errores del sensor y parámetros de calibración. El mapa, por otro lado, es una representación de aspectos de interés, como la posición de puntos de referencia u obstáculos, que describe el entorno en el que opera el robot. La necesidad de usar un mapa del entorno es doble. Primeramente, se requiere el mapa para realizar otras tareas. Por ejemplo, un mapa puede proveer la información necesaria al planificador de trayectorias o mostrar de forma intuitiva la presencia de un operador humano. En segundo lugar, el mapa permite limitar el error cometido al estimar el estado del robot. En ausencia de un mapa, la navegación basada exclusivamente en la odometría deriva rápidamente con el tiempo, mientras que usando un mapa el robot puede" restablecer" su error de localización al volver a visitar. Por lo tanto, las aplicaciones del SLAM se encuentran en

todos los escenarios en los que un mapa a priori no está disponible y necesita ser construido.

La arquitectura básica de un sistema SLAM moderno incluye dos componentes principales: el front-end y el back-end. El front-end abstrae los datos recogidos por los sensores formando modelos completos sobre los que realizar estimaciones, mientras que el back-end lleva a cabo inferencias sobre estos datos producidos por el front-end.

El estándar de los sistemas SLAM en la actualidad basa su formulación en un problema de estimación máxima a posteriori, MAP, usando a menudo el formalismo de factor graph para relacionar la interdependencia de las variables. Supóngase que se pretende estimar una variable X que representa la trayectoria del robot, como un conjunto de poses, y la posición de ciertos puntos de referencia en el entorno. Dadas una serie de mediciones:

$$Z = \{zk : k = 1, \ldots, m\} \quad (1)$$

tal que cada medición puede expresarse como función de X,

$$zk = hk(Xk) + sk \quad (2)$$

Donde:

- Xk ⊆ X es un subconjunto de las variables
- hk es una función conocida
- sk es una medida de ruido aleatoria.

En la estimación MAP, se estima la variable X calculando la asignación de variables.

$X*$ que consigue una posterior probabilidad máxima p(X|Z) (ecuación 3):

$$X^* = argmax\, p(X|Z) = argmax\ p(Z|X)p(X) \quad (3)$$

Donde:

- La igualdad sigue el teorema de Bayes.
- p(Z|X) es la probabilidad de las medidas Z dadas las asignaciones X,
- p(X) es la probabilidad anterior sobre X.

Esta probabilidad incluye todo el conocimiento previo sobre X, y si no lo hubiera, se convertiría en una constante (distribución uniforme). Al contrario de lo que ocurre con el filtro de Kalman, la estimación MAP no requiere una distinción explícita entre los modelos de movimiento y observación. Ambos son tratados como factores e incluidos en el proceso de estimación. Asumiendo que las mediciones Z son independientes y que los ruidos no están correlacionados,

la ecuación 2 se factoriza en:

$$X^* = argmax\, p(X) \prod_{k=1}^{m} p(z_k|X) = argmax\, p(X) \prod_{k=1}^{m} p(z_k|X_k) \dots \quad (4)$$

Donde:

- zk solo depende del subconjunto Xk.

La ecuación 3 puede ser interpretada en términos de inferencia sobre un factor graph. Las variables corresponden a los nodos y los términos p(zk|Xk) y p(X) son los factores. Un factor graph es un modelo gráfico que codifica la dependencia entre el factor k-ésimo (con su correspondiente medida zk) y las variables Xk.

La formulación del SLAM como un factor graph se expresa visualmente en la figura 17. Los círculos azules representan las poses de robot en los instantes de tiempo (x1, x2, ...), los círculos verdes denotan las posiciones de los puntos de referencia (l1, l2, ...) y el círculo rojo, la variable asociada con los parámetros de calibración. Los factores se representan como cuadrados negros: la etiqueta" u" identifica a los factores de las restricciones de la odometría," v" para las observaciones de la cámara (o LiDAR, en el caso del presente trabajo)," c" para los cierres de bucle y" p" para los factores anteriores.

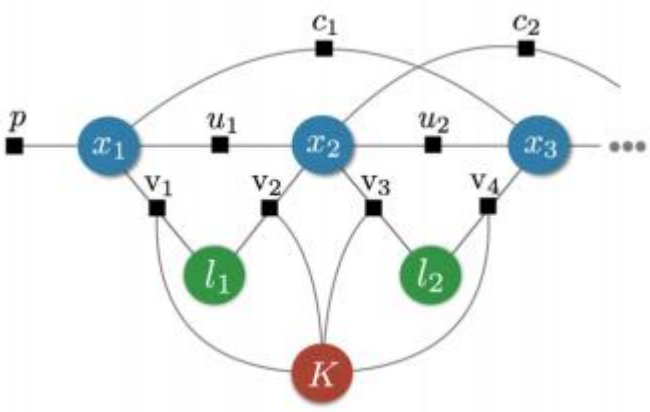


Figura 17: SLAM como un factor graph.
Fuente: Cadena et al., (2016).

Con la irrupción de nuevos sensores cada vez más asequibles y precisos, el problema del mapeo y localización simultáneos se ha adaptado para integrarlos y combinar la información recogida por cada uno de ellos. Así, surgen nuevas formulaciones para el SLAM con cámaras basadas en eventos. Al incorporar más de un sensor para esta tarea, aparecen nuevos enfoques basados en una estructura modular que permite el empleo de LiDAR, cámaras monoculares y estéreo, odometría, IMU y GPS.

De igual forma, los grandes avances en Machine Learning y Deep Learning aportan una visión novedosa a este problema. Si bien no significa reemplazar los algoritmos actuales bien conocidos, se ha demostrado la capacidad de estas nuevas herramientas para relacionar los marcos de referencia entre imágenes y la localización de los 6 grados de libertad de una cámara mediante regresión forest y redes neuronales. Una solución completa sería el popular algoritmo FastSLAM basado en redes neuronales.

2.2.6. Desinfección UV-C

Durante años, la desinfección con UVC se ha utilizado para ayudar a prevenir la propagación de enfermedades adquiridas en los hospitales. Las infecciones son un problema continuo, y se ha identificado un número creciente de bacterias resistentes a los medicamentos. La desinfección con UVC es eficaz para inactivar los gérmenes en el agua, el aire y las superficies, según varios estudios. El uso de la UVC para la desinfección de superficies será el tema de las próximas secciones. Uno de los aspectos más

importantes de la desinfección con UVC es determinar la dosis de exposición a la radiación UV que debe aplicarse en las zonas contaminadas. La necesidad de una técnica de desinfección que no implique el contacto directo con las superficies contaminadas ha aumentado recientemente a raíz de la epidemia de Covid-19. Los sistemas de desinfección UVC no requieren un contacto directo con las superficies enfermas. Además, la desinfección UVC es un método de desinfección no químico, lo que supone una ventaja adicional respecto a los métodos de desinfección tradicionales.

Dado que el nuevo coronavirus puede permanecer infeccioso entre 24 y 72 horas en las superficies, el enfoque más eficaz para combatir la pandemia del Covid-19, así como muchas otras epidemias y pandemias, es la prevención, que incluye la limpieza de lugares y superficies contaminadas.

El espectro ultravioleta (UV) se divide en numerosos subtipos, como UVA, UVB, UVC y VUV. La figura 18 muestra las definiciones. La luz UVB, UVC y VUV con longitudes de onda inferiores a 320 nm se denomina actínica, lo que implica que provoca reacciones fotoquímicas.

El rango germicida se define como el rango de longitudes de onda entre 200 y 320 nm, que incluye la UVB y la UVC. La radiación UV dentro de este rango es absorbida por los ácidos nucleicos de los microorganismos, produciendo alteraciones estructurales en el ADN y el ARN, lo que puede dar lugar a la inactivación del microorganismo. Cuando un microorganismo se inactiva, pierde su capacidad de reproducción y proliferación. Aunque esté metabólicamente vivo, no puede causar infecciones o enfermedades si no puede multiplicarse.

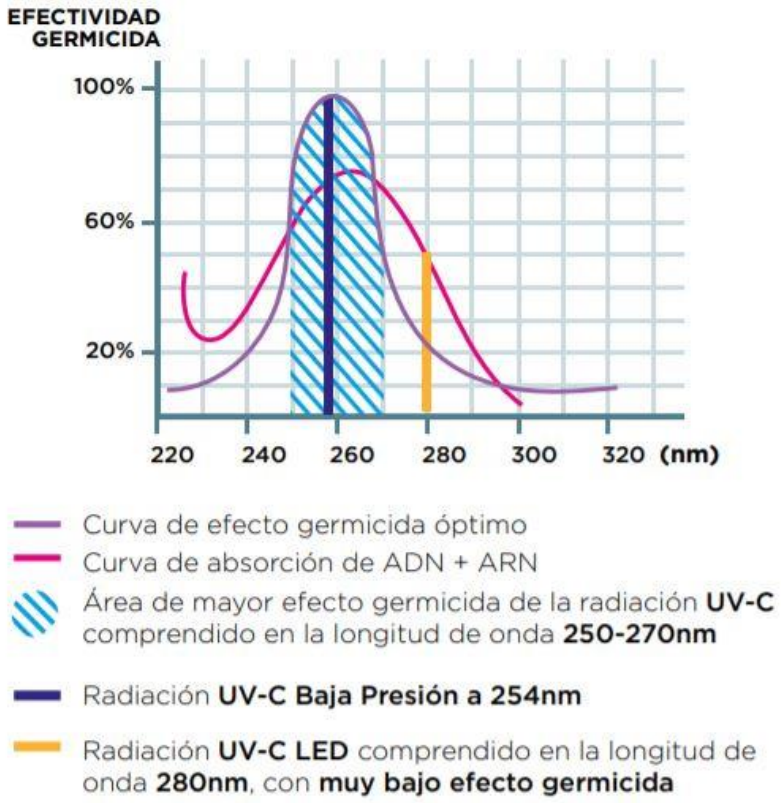


Figura 18: Espectro UV-C y su efectividad germicida

Fuente: Luxiona. (2021)

La desinfección implica una disminución de la población microbiana, que puede lograrse mediante luz UV germicida.

- Ventajas

  La radiación UVC es muy eficaz en la inactivación de microorganismos y tiene una serie de beneficios que la convierten en una buena opción para los sistemas de desinfección. Una de sus principales ventajas es su gran eficacia en la inactivación de microorganismos, lo que ayuda a prevenir y limitar la propagación de enfermedades infecciosas causadas por virus o bacterias. Se trata de una característica crítica dada la actual epidemia mundial producida por el nuevo coronavirus.

  Otra ventaja significativa es que la radiación UVC se desvanece rápidamente, lo que hace que sea seguro entrar en un espacio que ha sido limpiado poco después de que se haya completado el proceso.

Esto es fundamental en caso de que alguien entre por error en una sala que se está desinfectando (al admitir el sistema de desinfección UVC detecta a la persona y se apaga automáticamente).

La desinfección química es el método convencional de desinfección de salas y equipos. Los profesionales que utilizan desinfectantes químicos se exponen a sustancias químicas cáusticas y venenosas que son perjudiciales para la salud humana.

La desinfección por UVC no requiere el uso ni el transporte de ningún agente tóxico o corrosivo, y aunque la desinfección por UVC puede ser más eficaz cuando se sigue un proceso de desinfección tradicional, la desinfección por UVC sigue reduciendo o eliminando el riesgo de que los trabajadores sanitarios se puedan exponer a agentes nocivos. La desinfección UVC tampoco produce residuos químicos ni subproductos.

Además, tiene la ventaja de no requerir el contacto con las superficies contaminadas, lo que ayuda a reducir la transmisión de enfermedades. En comparación con otros medios alternativos, la desinfección de superficies tiene la ventaja de atacar el desarrollo microbiano en su origen. Cabe destacar que las superficies contaminadas son con frecuencia una fuente de microorganismos en el aire.

La desinfección UVC es sencilla de utilizar y requiere poco mantenimiento (aunque el mantenimiento también depende del tipo de sistema de desinfección UVC). Por último, el procedimiento de desinfección es rápido, ya que sólo requiere unos segundos. (cuando las superficies están muy cerca de la fuente de luz, como en los estudios presentados).

- Desventajas

  Aunque la utilización de la UVC para la desinfección presenta varias ventajas, es fundamental tener en cuenta los inconvenientes del empleo de la luz UV y de los equipos de desinfección UVC. Para empezar, el mayor inconveniente de la radiación UV es el peligro que supone para la salud humana. Incluso si se tolera durante cortos periodos de tiempo, la radiación UVC sin protección es perjudicial para la salud humana.

  Los daños oculares, las quemaduras de la piel e incluso el cáncer de piel son sólo algunas de las consecuencias secundarias de la exposición a los rayos UVC sin protección. El más importante de estos efectos secundarios es el deterioro de los ojos. La exposición prolongada a los rayos UV puede provocar fotoqueratitis, conjuntivitis, lesiones en la córnea e incluso cataratas.

  El eritema, la fotosensibilidad, el envejecimiento de la piel, los daños en el sistema inmunitario y, como ya se ha señalado, el cáncer de piel, son algunas de las repercusiones del daño cutáneo. Las longitudes de onda más pequeñas, como la de 222 nm, son menos peligrosas para la salud humana y no provocan los efectos descritos. Dado que la radiación UVC es invisible a simple vista, puede dar lugar a una exposición desprotegida y prolongada.

  Cuando se utilizan lámparas UV de mercurio de media o baja presión, que se emplean ampliamente en los sistemas de desinfección UVC, surge otro posible inconveniente. El inconveniente de utilizar estas lámparas es la presencia de mercurio. Cuando una lámpara se rompe, existe el riesgo de envenenamiento por mercurio. Si la lámpara se rompe mientras se está utilizando, el mercurio estará en forma de vapor, lo que supone un peligro de inhalación. Estas lámparas también tienen una estabilidad mecánica limitada, y algunas de ellas pueden generar ozono. Las lámparas UV-LED podrían ser una alternativa superior.

  Otros inconvenientes son la imprevisibilidad de la influencia de la radiación UV de una especie a otra, su dependencia del medio de cultivo

y las propiedades de los microbios, como la cepa y la densidad, entre otros.

Por último, la desinfección con UVC sólo es eficaz cuando la radiación UV se dirige directamente al objetivo, lo que implica que el diseño de la superficie, las sombras, los poros y los orificios influyen en la eficacia germicida de la UVC.

- Medidas de Seguridad

Ya se han discutido varios inconvenientes de la desinfección con UVC, incluidos los peligros para la salud humana. La radiación UV con longitudes de onda entre 180 y 320 nm es una fuente de preocupación para la seguridad biológica y la salud. Los rayos UVB y UVC se incluyen en esta categoría. Por ello, a la hora de proyectar y manejar un sistema de desinfección UVC, la seguridad es un factor crítico a tener en cuenta. Los controles de ingeniería, los protocolos operativos y el equipo de protección personal son los tres tipos de precauciones de seguridad que deben aplicarse.

La exposición a la UVC sin protección puede tolerarse durante cortos periodos de tiempo, pero debe evitarse. Los daños producidos por la radiación UV sin protección afectan sobre todo a los ojos y la piel, y sus consecuencias pueden tardar muchas horas en manifestarse, entre 1 y 12 horas. Muchas lesiones oculares son temporales y pueden desaparecer en cuestión de horas o días. Aunque la radiación UVB es más peligrosa que la UVC en términos de daños en la piel, cualquier parte del entorno susceptible de recibir radiación UV debe estar debidamente etiquetada y debe haber señales de advertencia.

La dosis de exposición a la luz UV sin protección con longitudes de onda entre 180 nm y 400 nm no debe superar los 30 J=m2. Cuando se utilizan lámparas UV de mercurio, se pueden alcanzar temperaturas extremadamente altas durante su funcionamiento, lo que supone un peligro de quemaduras. Por ello, hay que dejar que las lámparas se enfríen por completo antes de proceder a su mantenimiento.

CAPÍTULO III: DISEÑO DEL ROBOT MÓVIL

En el presente capítulo se describe el desarrollo del diseño de un robot móvil usando lógica difusa y escaneo LiDAR para desinfección por radiación UV-C de espacios cerrados, se describen las etapas de diseño mecánico, el proceso de diseño y características. Además, se realiza el diseño del sistema eléctrico y electrónico, la descripción de los componentes a utilizarse, el cálculo del motor. Finalmente, se elabora el diseño del sistema de control, el cual describe el desarrollo de la programación que se realizó, como también las condiciones de mapeo y localización que se tomaron en cuenta.

3.1. Descripción del proceso

Se utilizó luz ultravioleta en la desinfección de habitaciones y equipamiento, con un apagado automático cuando se detectan personas por seguridad, para ello el robot contó con ocho luces LED UV-C dispuestas alrededor de una columna central de manera vertical, y otras cuatro luces LED UV-C en la parte superior instaladas de manera horizontal. La columna fue montada en una base móvil con un sensor LiDAR para medir la distancia, identificar y evitar obstáculos mediante el análisis del entorno y elaboración de un mapa digital.

3.2. Requerimientos del sistema

Se especifica los requerimientos del sistema en la tabla 1 y en la lista

Tabla 1. Requerimientos del sistema

Requerimientos	
Masa total del robot	16.255kg
Pendiente máxima de subida	12%
Mínima autonomía	4 h
Velocidad máxima	40 cm/s

Fuente: Elaboración propia

El robot móvil cumple con la finalidad de desarrollar de manera exitosa las actividades planteadas en la presente tesis con las siguientes características funcionales:

- Robot móvil cuenta con un sistema de locomoción con direccionamiento diferencial.
- La batería posee una autonomía de 4 horas de funcionamiento continuo, garantizando así un periodo variable de trabajo entre reposo y funcionamiento de 1 día como mínimo.
- El robot móvil desinfecta de noche para no tener contacto con personas.
- Está equipado con motorreductores DC para que cumpla con las necesidades y características del momento.
- El material que se usa en la estructura del robot móvil es de aluminio. Una de las propiedades más relevantes de este material es que tiene una buena resistencia a la tracción y compresión, además de ser muy ligero.

En la figura 19 se muestra la representación gráfica del funcionamiento interno del diseño, los datos tomados por el sensor LiDAR son enviados al Raspberry Pi para hacer procesamiento de mapeo, posteriormente se envían los datos procesados al Arduino Uno el cual aplica un método de control con lógica difusa para que mueva las ruedas del robot a través de los motores DC y encienda las luminarias UV-C mediante una interfaz de potencia.

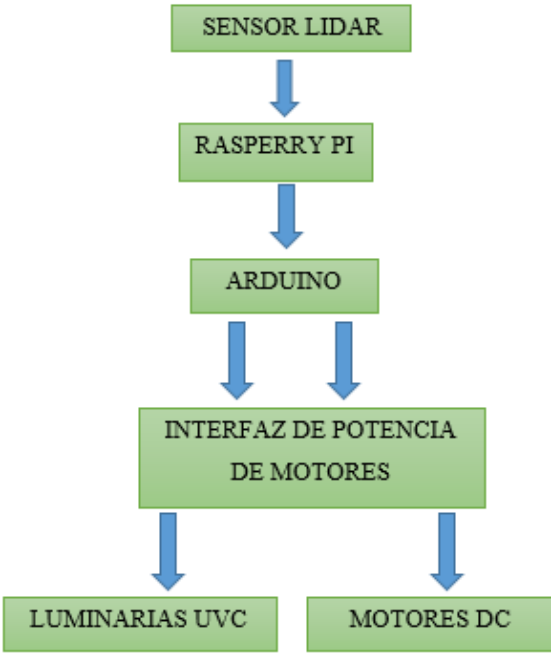


Figura 19: Funcionamiento interno del robot móvil.

Fuente: Elaboración propia.

3.3. Diseño mecánico

3.3.1. Selección del material de la estructura

Elegir el material ideal en el diseño de la estructura mecánica del robot móvil es muy importante ya que, si se hubiera elegido un material muy pesado se hubiera requerido motores más potentes para movilizar la estructura, del mismo modo si se hubiera elegido un material poco resistente, el chasis hubiera presentado una falla estructural y ocasionando accidentes.

Debido a la fácil accesibilidad, alta resistencia y ligereza, tal como se observa en la tabla 2 se optó por considerar aluminio como mejor opción para el diseño de la estructura del chasis del robot móvil y estructura de soporte de las luminarias led UV-C.

Tabla 2. Característica de materiales.

Nombres	Accesibilidad	Resistencia	Peso	Precio
Aluminio	Alta	Alta	Liviano	Alto
Acero	Alta	Alta	Muy Pesado	Medio
Titanio	Alta	Alta	Muy Liviano	Muy Caro
Fibra de carbono	Baja	Baja	Muy Liviano	Muy Caro

Fuente: Elaboración propia.

3.3.2. Extrusiones de aluminio TSLOTS

Para la construcción de la estructura del chasis del robot se utilizó extrusiones de aluminio TSLOTS ya que permite realizar un ensamble de la estructura del robot más rápido que estructuras soldadas debido a que solo utiliza sujetadores, este tipo de extrusión cuenta con las siguientes dimensiones en una vista de perfil como lo observamos en la figura 20.

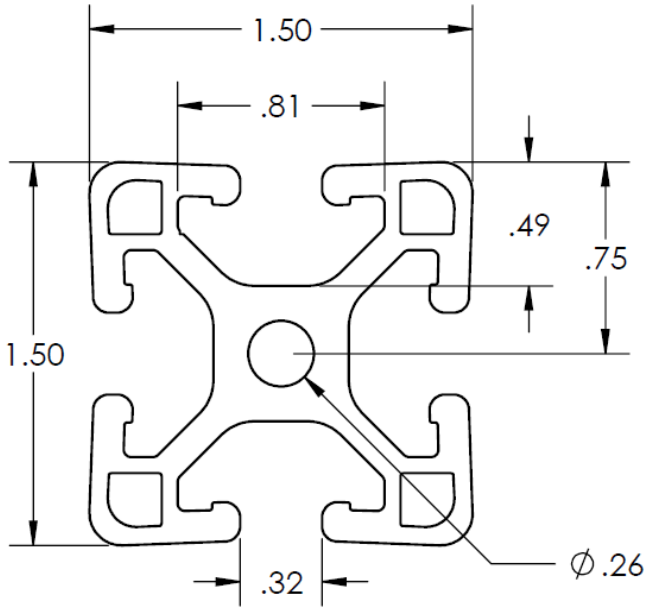


Figura 20: Dimensiones de extrusión TSLOTS

Fuente: Bonnell. (2021)

A continuación, se presenta las especificaciones técnicas de la extrusión TSLOTS como se muestra en la tabla 3 donde se especifica el largo de la estructura, el peso por metro, área estimada de la sección transversal y momentos de inercia, información que fue necesaria para el diseño estructural en el software CAD de modelado mecánico SolidWorks.

Tabla 3. Especificaciones técnicas de una extrusión TSLOT

Especificaciones técnicas	
Largo	609.6 cm
Masa por metro	$1.627 kg/m$
Área de sección transversal estimada	$5.968\ cm^2$
Momento de inercia en el eje x	$8.075\ cm^4$
Momento de inercia en el eje y	$8.075\ cm^4$

Fuente: Bonnell. (2021)

Para saber la capacidad de carga de la barra extruida TSLOTS se tomó en cuenta la información proporcionada por el fabricante en un gráfico de carga vs largo de la barra extruida TSLOTS como se aprecia en la figura 21, donde se observa que para barras extruidas TSLOTS más largas hay una capacidad de carga menor, por lo tanto, tener una barra de menor longitud en el chasis del robot permitirá que tenga una mejor resistencia a la aplicación de una carga uniformemente distribuida.

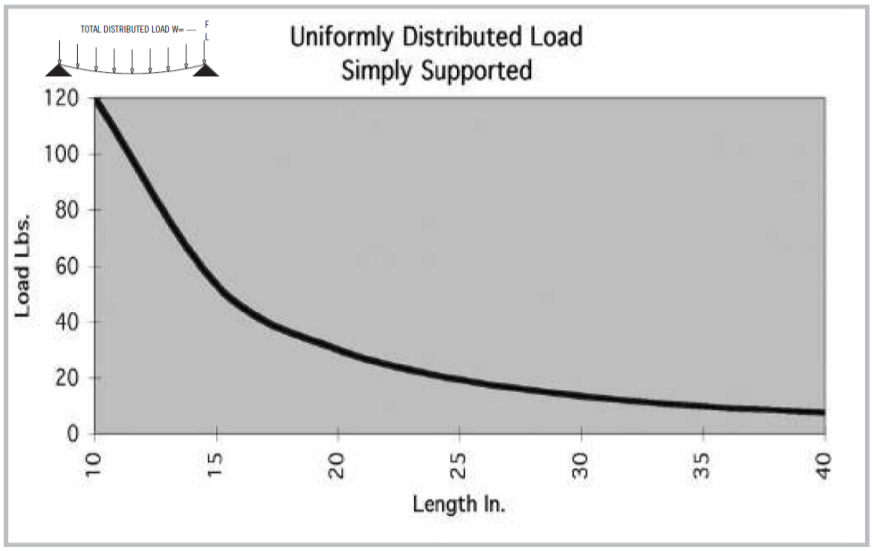


Figura 21: Gráfico de carga distribuida aplicada vs largo de la barra TSLOTS

Fuente: Bonnell. (2021)

3.3.3. Chasis del robot móvil

En primer lugar, se diseñó la extrusión de aluminio TSLOTS, sus dimensiones se observan en la figura 22, en una vista frontal y de planta. De acuerdo a los datos proporcionados por el fabricante este tipo de material tiene una masa de 1.627 kg por metro y para el cálculo de la masa de una barra de 30.48cm hicimos una regla de tres simple dándonos como resultado 0.5 kg.

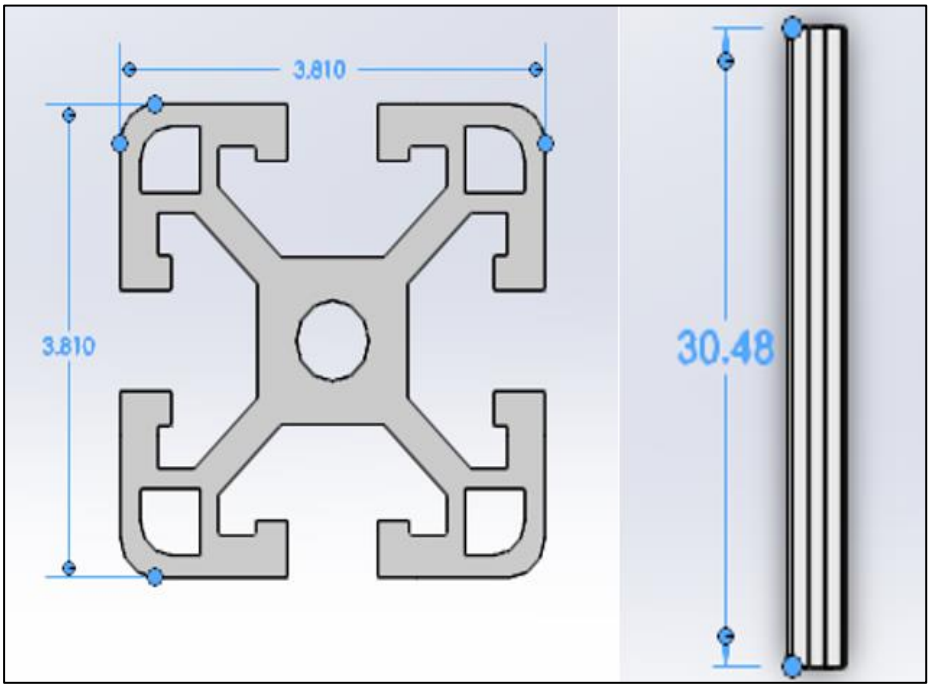


Figura 22: Vista frontal y de planta de extrusión TLOTS.
Fuente: Captura de pantalla de diseño realizado en SolidWorks

Se utilizó 4 barras de extrusión TSLOTS diseñadas previamente para la implementación del chasis del robot móvil, se observa en figura 23 que como elemento de unión para el ensamble de las barras se empleó soportes tipo L de 30 milímetros y se acopló ruedas locas a mediante tornillos milimétricos M2.5x10 en las ranuras de extrusión.

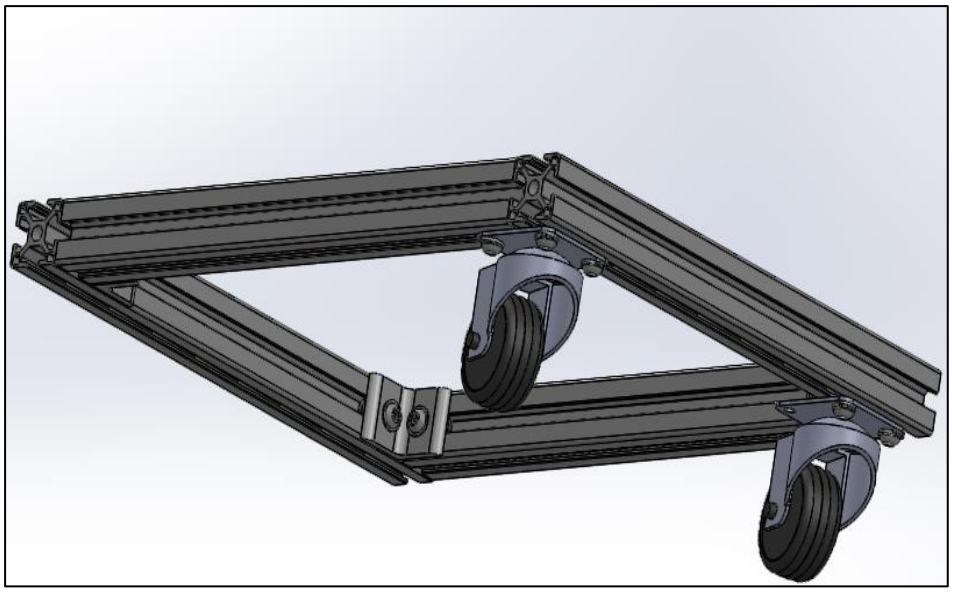


Figura 23: Visualización 3D del chasis para el robot móvil.
Fuente: Captura de pantalla de diseño realizado en SolidWorks.

3.3.4. Sistema de propulsión

Para iniciar se seleccionó el motorreductor de la marca JSUMO de 200 RPM, con reducción de 60 vueltas a 1, accionada por un motor de 12V de corriente directa y con un torque de 10.35 kg-cm, su diseño CAD observado en la figura 24 se descargó de la página web del fabricante.

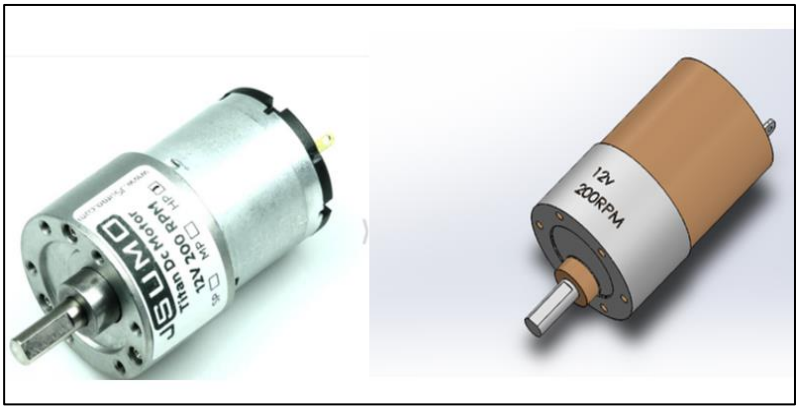


Figura 24: Motorreductor de la marca SUMO y su diseño CAD.
Fuente: JSUMO. (2021)

Luego se diseñó un sistema de transmisión por poleas mediante una faja dentada de 9 mm de ancho por 250 mm de largo como se observa en la figura 25.

Figura 25: Correa de hule de 50 dientes

Fuente: Bestorq. (2021)

Para el cálculo de distancias de centros de poleas lo cual que permite que la faja trabaje de manera eficiente se utilizó la fórmula siguiente con datos de los diámetros de las poleas:

$$L = 2C + \frac{\pi}{2}(D_1 + D_2) + \frac{(D_2 - D_1)^2}{4C} \quad (5)$$

Donde:

L: Longitud de la correa que es de 250mm

C: Distancia entre centros a calcular.

D1: Diámetro de polea menor de que es de 22.9mm

D2: Diámetro de polea mayor que es de 50.94mm

Despejando C de la ecuación (5) y reemplazando datos tenemos:

$$C = 65.5mm$$

Luego se utilizó la distancia de centros “C” para el diseño en SolidWorks como se observa en la figura 26.

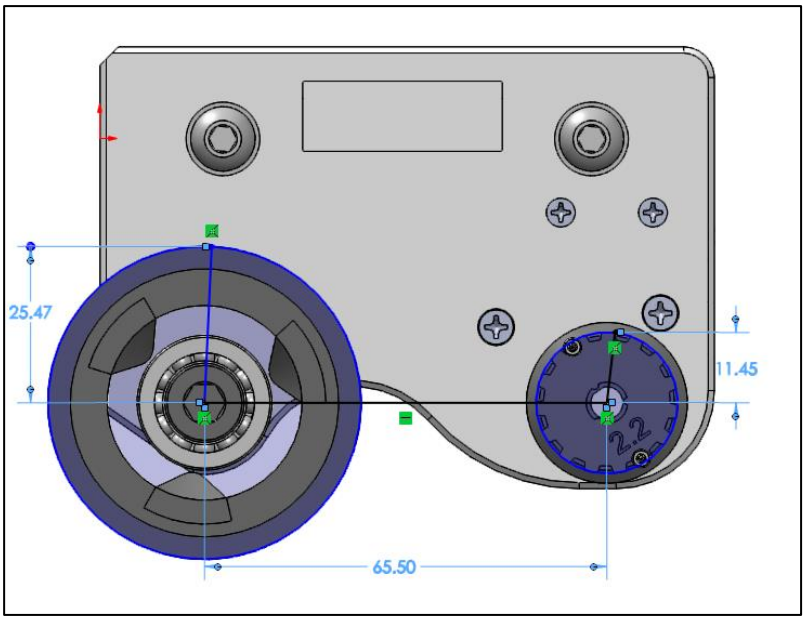


Figura 26: Vista Frontal de sistema de transmisión por faja dentada.

Fuente: Captura de pantalla de diseño realizado en SolidWorks.

Se utilizó una llantas de 80 mm de diámetro y 56 mm de ancho para darle mayor estabilidad al robot móvil, como se observa en la figura 27.

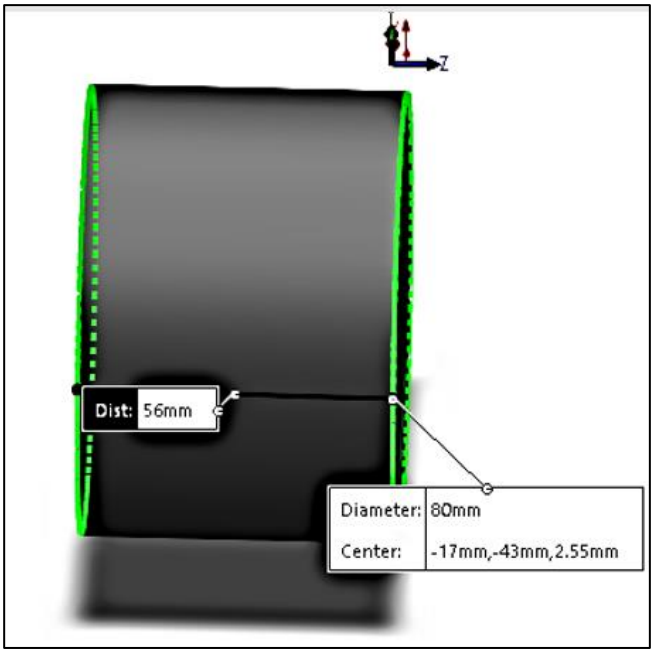
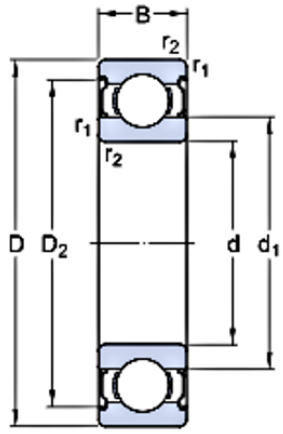


Figura 27: Dimensiones de Llantas del robot móvil.

Fuente: Captura de pantalla de diseño realizado en SolidWorks.

Para permitir la unión entre las llantas y el eje montado en el chasis se utilizó rodamientos rígidos de bolas modelo 608-2Z de la marca SKF, ya que tienen una baja fricción y están optimizados para un nivel de ruido bajo y baja

vibración, lo que permite altas velocidades de giro. Soportan cargas radiales y axiales en ambos sentidos, son fáciles de montar y requieren menos mantenimiento que otros tipos de rodamientos. Las dimensiones del rodamiento se muestran en la figura 28.



DIMENSIONES

d	8 mm	Diámetro del agujero
D	22 mm	Diámetro exterior
B	7 mm	Ancho
d1	≈12.15 mm	Diámetro del resalte del aro interior
D2	≈19.2 mm	Diámetro del rebaje del resalte del aro exterior
r1,2	min.0.3 mm	Dimensión del chaflán del aro exterior

Figura 28: Dimensiones del rodamiento rígido de bolas.
Fuente: SKF.(2021)

En la tabla 4 se observa la capacidad de carga dinámica de 3.45kN y estática de 1.37kN del rodamiento que fue más que suficiente para soportar la estructura del robot móvil que tuvo un peso de 159.3N, también se observa la velocidad de giro de referencia de 75 000 r/min lo que permitió que las ruedas giren sin ningún problema ya que la velocidad máxima del robot fue de 40 cm/s.

Tabla 4. Ficha técnica de Rodamiento rígido de bolas

Ficha técnica rodamiento 608-2Z	
Capacidad de carga dinámica básica	3.45 kN
Capacidad de carga estática básica	1.37 kN
Carga límite de fatiga	0.057 kN
Velocidad de referencia	75 000 r/min
Velocidad límite	38 000 r/min
Factor de cálculo	0.025
Masa del rodamiento	0.013 kg

Fuente: SKF.(2021)

Asimismo se descargó un modelo CAD desde la página web del fabricante tal como se observa en la figura 29.

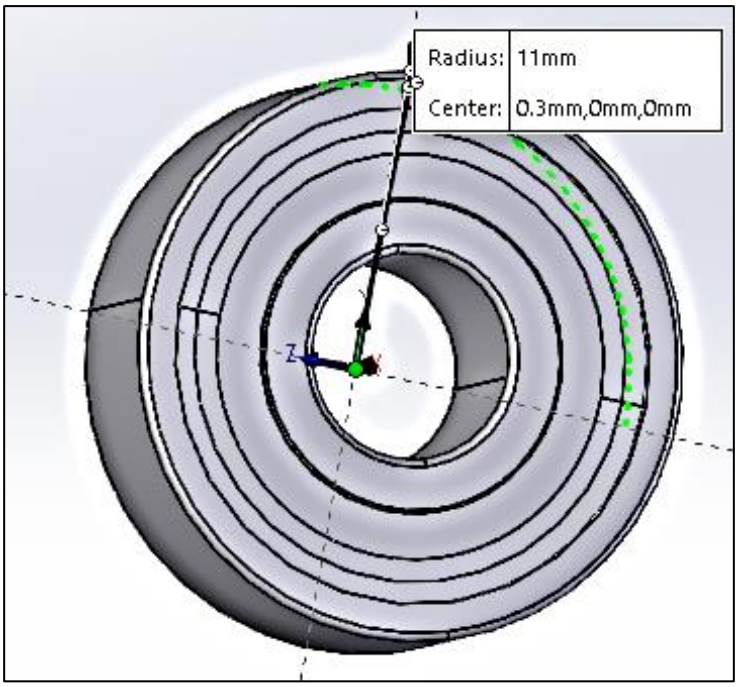


Figura 29: Modelo 3D rodamiento rígido de bolas.

Fuente: Captura de pantalla de diseño realizado por SKF en SolidWorks.

Por último, todos los componentes del sistema de propulsión que fueron diseñados es SolidWorks se ensamblaron al chasis del Robot móvil por lo tanto se obtuvo una configuración con direccionamiento diferencial, como se observa en la figura 30.

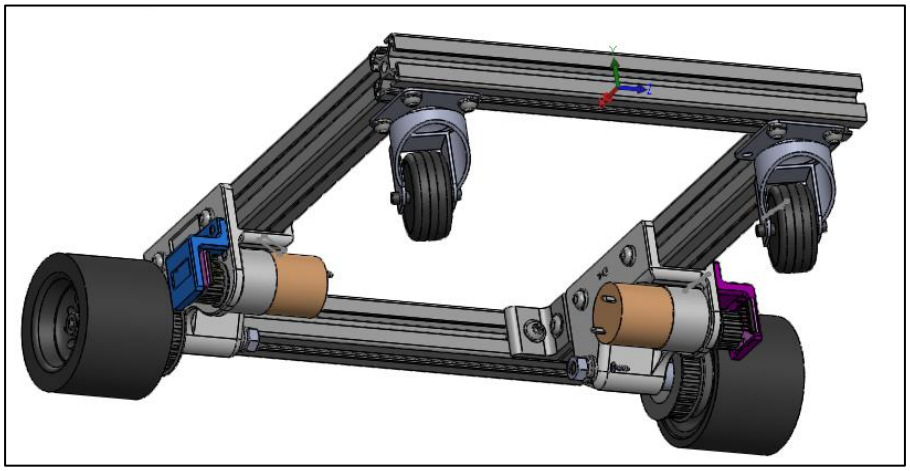


Figura 30: Sistema de propulsión unido al chasis.

Fuente: Captura de pantalla de diseño realizado en SolidWorks.

3.3.5. Estructura para las Luminarias UV-C

Se implementó una estructura prismática con barras extruidas TSLOTS y se ensamblaron con soportes tipo L de 30 milímetros, mediante tornillos

milimétricos M2.5x8 como se observa en la figura 31, además se hizo un cálculo de la masa total de esta estructura sumando todo el largo de cada barra que midió en total 1.84 m y con la información del fabricante de la masa por metro se obtuvo una masa total de 3 kg. En esta estructura se colocaron las tiras led UV-C de forma distribuida.

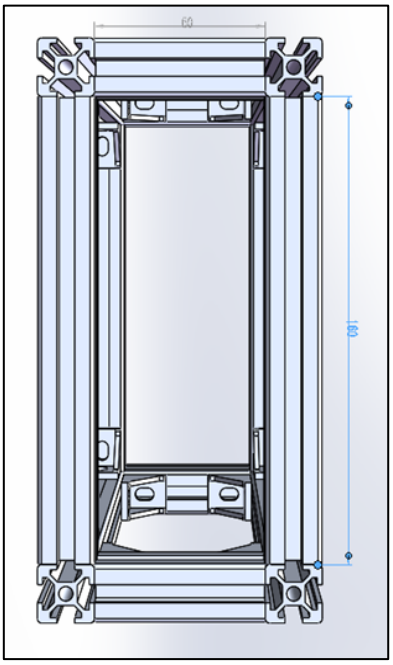


Figura 31: Estructura para las luminarias UV-C

Fuente: Captura de pantalla de diseño realizado en SolidWorks

Para finalizar, se ensambló la estructura para las luminarias UV-C al chasis del robot móvil que está unido al sistema de propulsión como se observa en la figura 32.

Figura 32: Estructura completa del robot móvil.

Fuente: Captura de pantalla de diseño realizado en SolidWorks

3.3.6. Montaje del sensor LiDAR:

El diseño propio de la estructura del montaje donde se encuentra el sistema LiDAR cuenta con las siguientes características:

- Ser estable y robusta.
- Minimizar en lo posible los golpes y vibraciones.
- Libre de obstáculos que puedan imposibilitar el escaneo del ambiente.

La ubicación donde fue instalado acorde a los puntos antes mencionados permite que el sensor LiDAR pueda registrar todos los obstáculos en la parte inferior cercanos a las llantas y chasis que pudieran afectar el desplazamiento y funcionamiento del robot. Por lo antes expuesto se ha optado por escoger en la parte central frontal del robot, entre las 02 llantas más específicamente como se observa en la figura 33, de manera que ninguno de los componentes de este se intercepte con la serie de rayos láser que emite el sensor LiDAR.

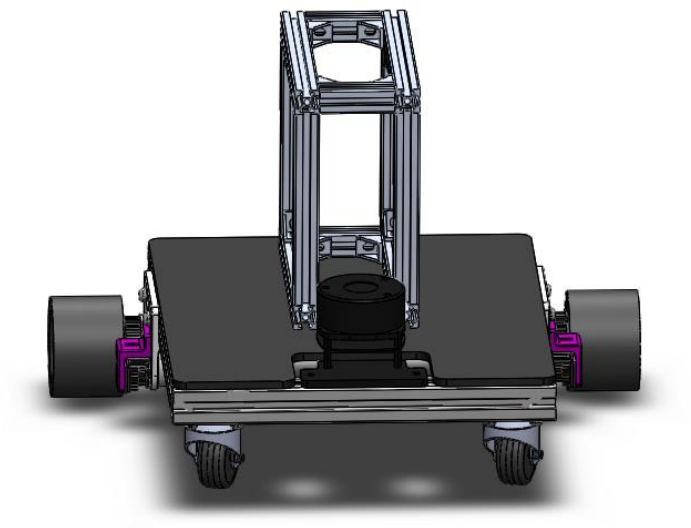


Figura 33:Montaje del sensor LiDAR

Fuente: Captura de pantalla de diseño realizado en SolidWorks

Es importante denotar que un montaje del LiDAR muy cercano al suelo podría conllevar a ciertas limitaciones, considerando el más molesto la aparición de un punto ciego en ángulo y distancia específica, como se puede apreciar en la siguiente figura 34:

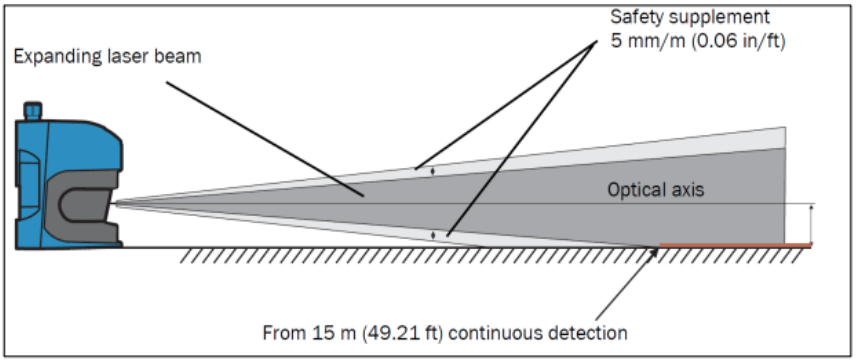


Figura 34: Incremento en tamaño del haz de rayos denotando área ciega

Fuente: YDLIDAR. (2021)

El fenómeno anterior es provocado por la dispersión del haz de rayos a lo largo de la distancia. Es por esto por lo que, para conseguir una señal confiable para la navegación del robot, el radio del haz de rayos no deberá exceder la distancia entre el suelo y el eje óptico como se muestra en la figura anterior. Este requerimiento es debido a que la detección continua del piso podría generar falsos registros que el sistema de navegación podría

tomar como válidos, implicando un comportamiento erróneo en el proceso de localización.

De acuerdo a las instrucciones del sensor LiDAR utilizado, la distancia entre la parte frontal hasta un punto con un cierto diámetro de rayo puede ser calculado con la siguiente fórmula:

$$l = \frac{D-d}{a+s} \tag{6}$$

Donde:

- l es la distancia desde la parte frontal en milímetros
- D es el diámetro del rayo en milímetros
- d es el tamaño del punto del rayo en milímetros
- a es la divergencia en radianes
- s es el factor de seguridad del sensor, ~0.005 rad

Se pudo emplear esta ecuación para obtener la máxima distancia real; el valor del diámetro del rayo fue reemplazado por "la distancia entre el suelo y el eje óptico multiplicado por 2". Los valores de las constantes adicionales, como el tamaño del punto de rayo en la parte frontal y divergencia son extraídos de la hoja técnica del sensor LiDAR en cuestión. Reemplazando estos valores por lo mencionado obtenemos lo siguiente:

$$l = \frac{(250 * 2 - 13.6)mm}{(11.9 * 10 + 0.005)rad} = 28781\ mm \sim 28\ m$$

De esto podemos denotar que el valor máximo confiable de la distancia de escaneo es significativamente menor que el valor máximo declarado por el fabricante. Por estos resultados iniciales podemos concluir que el sistema de soporte o estructura del montaje deberá ser localizado en una posición más elevada que la antes considerada para incrementar la distancia máxima confiable de escaneo.

3.3.7. Análisis de esfuerzos y deformaciones de la estructura del robot

En el cálculo de esfuerzos y deformación de la estructura del robot móvil se analizó la barra de obstrucción TSLOTS de 30.48 cm en la herramienta de simulación por cálculo de elementos finitos de Solidworks. En primer lugar, se introdujo el tipo de material que es una aleación de aluminio 6061 en la ventana de SolidWorks Simulation como se observa en la figura 35, el cual definió el comportamiento de la pieza, así como sus propiedades físicas.

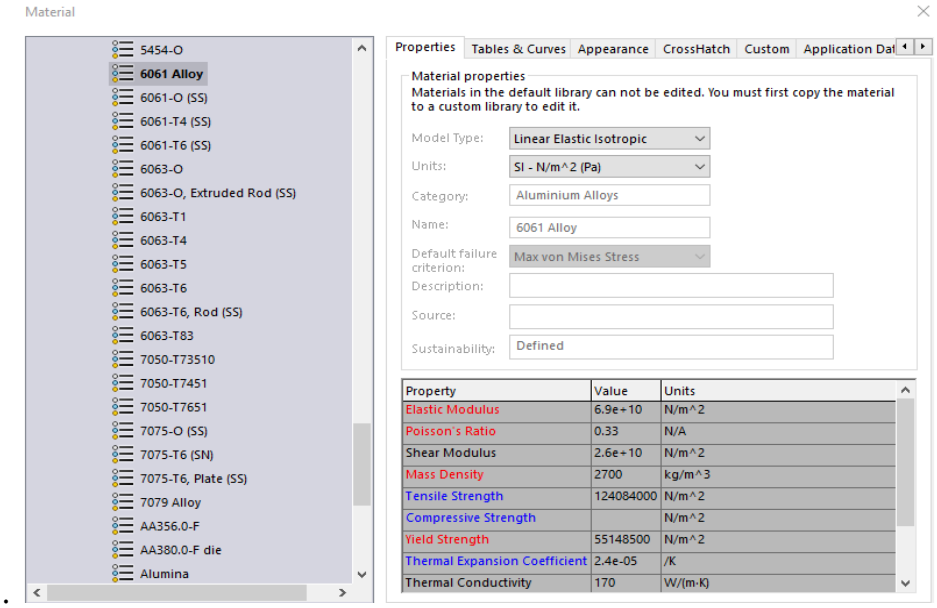


Figura 35:Asignación del tipo de material en Solidwork Simulation

Fuente: Elaboración propia hecha en Solidworks

Para la verificación de esfuerzos y deformaciones de la barra extruida del chasis se necesitó hacer un cálculo de la masa total aplicada al robot móvil como se aprecia en la tabla 5, en donde la masa total que se obtuvo es de 16.255Kg

Tabla 5. Cálculo de la masa total aplicada sobre el robot móvil.

Parte	Cantidad	Masa (kg)	Masa Total(K6)
Batería	1	9.5	9.5
Sensor LiDAR	1	0.1	0.1
Cubierta	1	1.2	1.2
Arduino	1	0.025	0.025
Raspberry Pi	1	0.045	0.045
MDD10A	1	0.025	0.025
Barra de chasis	4	0.5	2
Soporte UV-C	1	3	3
Tira Led UV-C	12	0.03	0.36
		Total	16.255

Fuente: Elaboración Propia

Una vez calculado la masa total aplicada al robot móvil se calculó el peso total que dió un resultado de 159.3 N, se divide este resultado en cuatro, debido a que el peso total se distribuye aproximadamente igual entre las cuatro barras extruidas del chasis, lo que dió un resultado de 39.82 N, dato que se introdujo en el simulador de Solidworks, asignando fijaciones en los extremos de la barra como se observa en la figura 36.

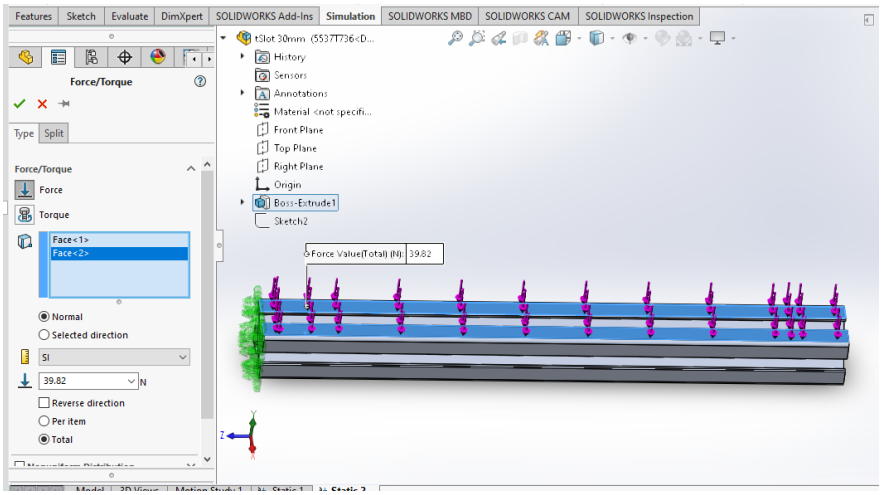


Figura 36: Introducción de fuerza aplicada y sujeciones en la barra extruida.

Fuente: Elaboración propia hecha en Solidworks

Asimismo se ejecutó la simulación y dió un diagrama de esfuerzos donde el esfuerzo máximo de la barra extruida producto del peso de 39.82N aplicado es de $8.717x10^5 \frac{N}{m^2}$ que no sobrepasa el límite elástico del aluminio 6061 utilizado de $5.515x10^7 \frac{N}{m^2}$, (Ver figura 37) lo que indicó que la barra de aluminio soportó sin problemas el peso aplicado sobre el mismo.

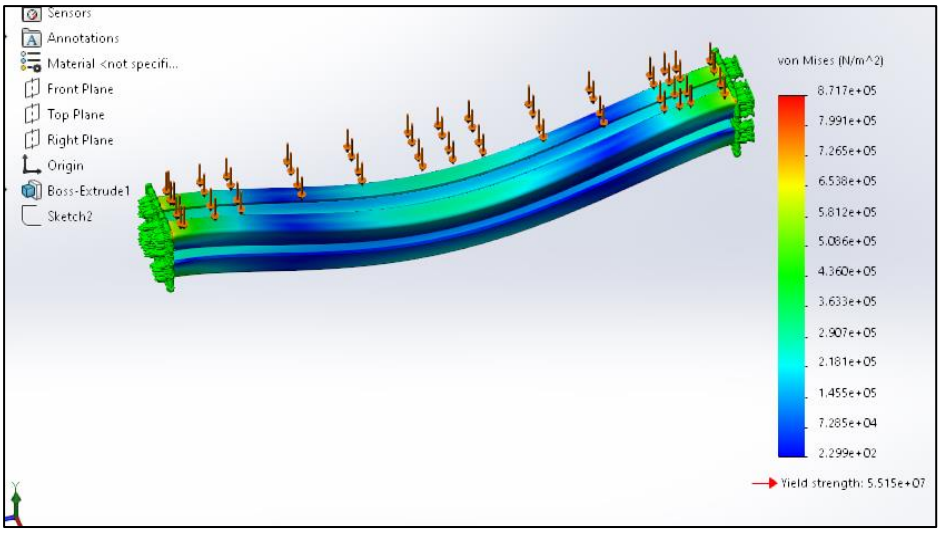


Figura 37: Diagrama de esfuerzos de la barra extruida de aluminio TSLOTS

Fuente: Elaboración propia hecha en Solidworks

Por último, el simulador permitió analizar las deformaciones a lo largo de toda la barra, donde se ve que la mayor deformación se da en el centro de la barra con un valor de $2.121x10^{-3}mm$, como se observa en la figura 38, y en los extremos de la barra no hay deformaciones, debido a estas deformaciones pequeñas la barra podrá soportar pesos aún mayores a la asignada.

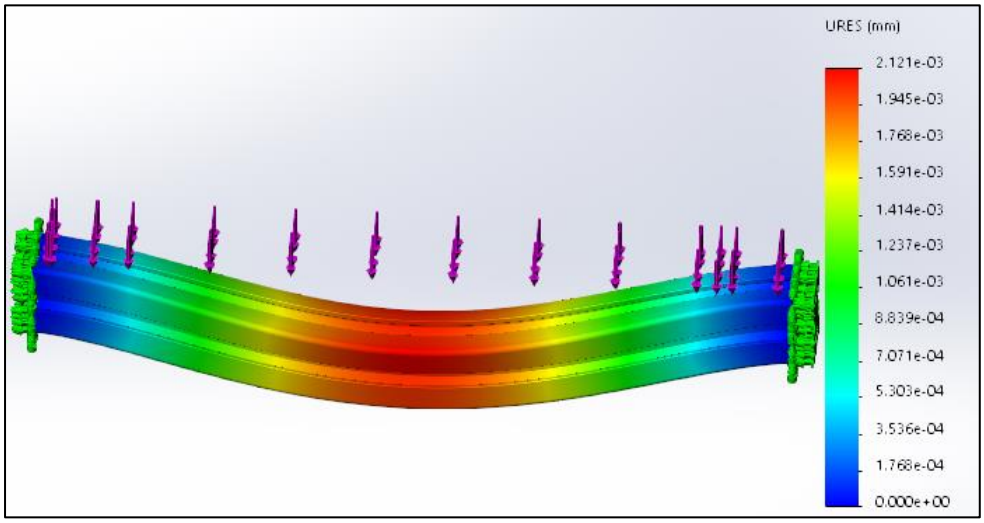


Figura 38: Deformaciones de la barra extruida TSLOTS

Fuente: Elaboración propia hecha en Solidworks

3.4. Diseño eléctrico y electrónico

3.4.1. Conexiones del Raspberry Pi Modelo 4

El popular microordenador desarrollado por Raspberry Pi ® indicado en la figura 39 es uno de los más usados en proyectos de estudiantes y en el mundo de la robótica debido a su versatilidad y gran soporte entre su comunidad, para el diseño de la investigación se utilizó los pines de propósito general para comunicarlo con el controlador Arduino y para el enlace con el sensor LiDAR. y se utilizó la ranura Micro SD para insertar una memoria SD y cargar el sistema operativo Ubuntu.

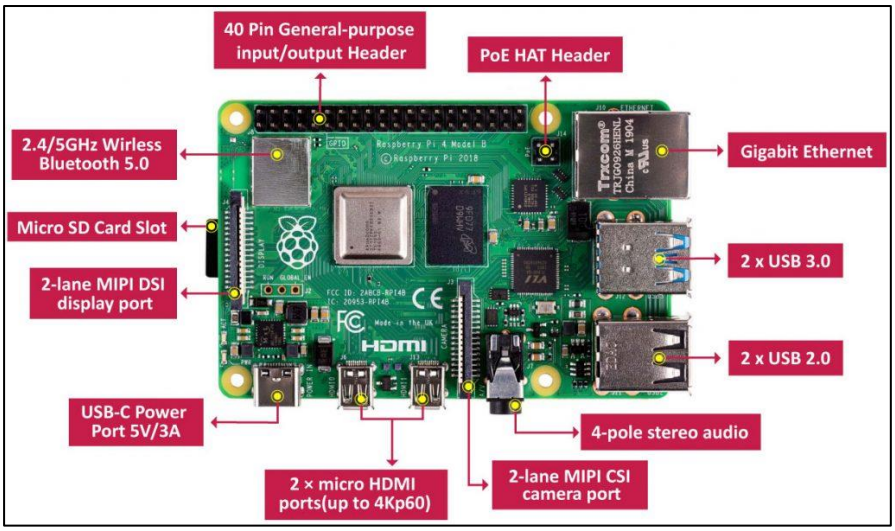


Figura 39: Raspberry Pi Modelo 4

Fuente: Raspberry Pi. (2021)

Las especificaciones de este modelo, según su fabricante (Raspberry Pi Foundation), se aprecian en la tabla 6:

Tabla 6. Especificaciones técnicas del microordenador Raspberry Pi

Familia	Broadcom BCM2711
Procesador	ARM Cortex-A72/64 bits/4 núcleos a 1,5 GHz
Fuente de alimentación	5 Voltios-DC / 3 Amperios
Puertos USB 3.0	2
Puertos USB 2.0	2
Memoria RAM LPDDR4	2GB, 4GB o 8GB
Conexiones micro-HDMI	2
Temperatura de funcionamiento	0 - 50 grados Centígrados
Comunicación Inalámbrica	Bluetooth 5.0

Estándar Inalámbrico	2.4 GHz y 5.0 GHz IEEE 802.11ac
Pines de conexión	Cabecera GPIO Raspberry Pi / 40 pines
Soporte de tarjeta SD	Ranura para tarjeta micro SD
Interfaz de programación de aplicaciones	OpenGL ES 3.1

Fuente: Raspberry Pi (2021).

3.4.2. Conexiones del controlador ARDUINO

El Arduino y el IDE de Arduino son grandes herramientas para programar hardware de forma rápida y sencilla. Usando el paquete rosserial_Arduino, se puede usar ROS directamente con el IDE de Arduino. rosserial proporciona un protocolo de comunicación ROS que funciona sobre la UART de la tarjeta Arduino. Permite que tu Arduino sea un nodo directo de ROS que permite publicar y suscribirse directamente a los mensajes ROS, ejecutar transformaciones TF, y obtener la hora del sistema ROS.

Los enlaces a ROS se implementan como una biblioteca de Arduino. Como todas las librerías de Arduino, ros_lib tiene que ubicarse en la carpeta de librerías del Arduino.

Para llevar a cabo la comunicación entre ambos microcontroladores utilizamos esta línea de código al inicio de nuestra programación: #include <ros.h>

La conexión de pines propuesta en este proyecto de tesis, correspondiente específicamente al microcontrolador Arduino UNO y el controlador Cytron, y el controlador Cytron a los actuadores y batería, todo esto representado según la figura 40 a continuación:

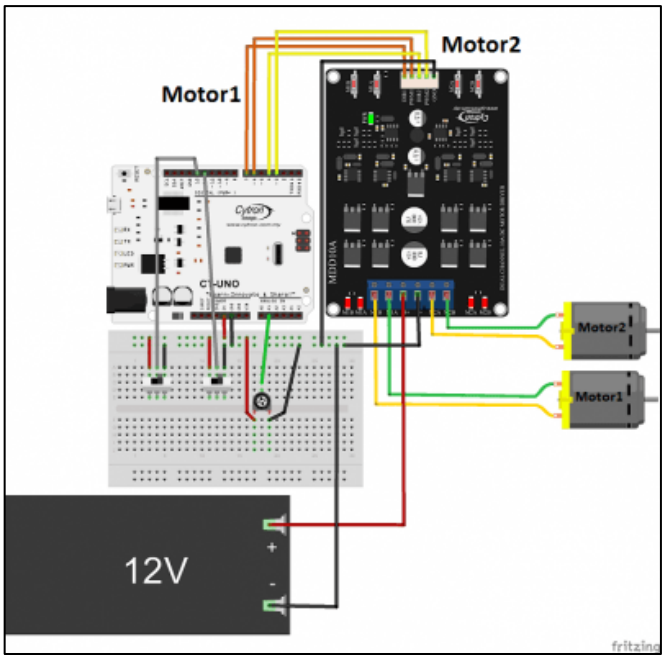


Figura 40: Conexión Etapa de Potencia con Lógica de Arduino UNO

Fuente: Cytron. (2021)

De igual manera el Arduino UNO es conectado al Raspberry Pi por los puertos USB disponibles, aprovechando la simplicidad, rápida comunicación y conexión, como se muestra a continuación en la figura 41:

Figura 41:Conexión USB entre Etapas lógicas: Raspberry Pi – Arduino

Fuente: Cytron. (2021)

3.4.3. Interfaz de potencia de motores con el controlador Cytron

Es un controlador de motor de DC de dos canales mostrado en la figura 42. En donde la corriente directa por canal es de 10 A (momentánea de 30 A). La tensión de alimentación es de 5 V a 30 V. El controlador trabaja con tensión lógica de 3,3 V y 5 V (PWM y DIR) por ello es compatible por un microcontrolador Arduino.

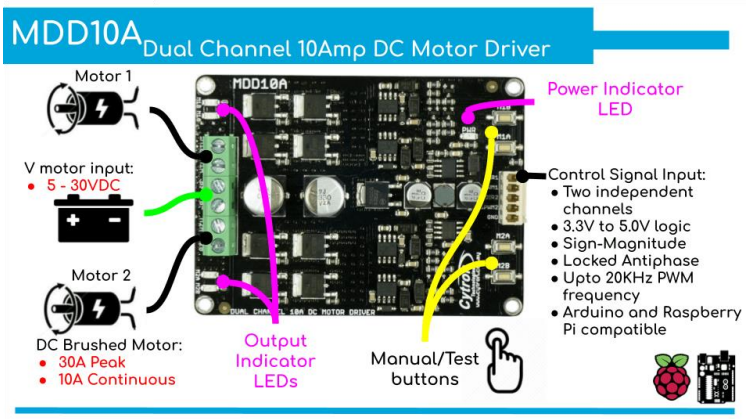


Figura 42: Características del controlador Cytron

Fuente: Cytron. (2021)

El controlador está equipado con una entrada de 5 pines de 2,54 mm desde la que recibió información del microcontrolador Arduino sobre el sentido y la velocidad de giro de los motores de 12V DC de la marca JSUMO y 6 conectores de tornillo a los que se conectan los motores y la fuente de alimentación. El Cytron MDD10A puede ser controlado a través de las entradas PWM y DIR. La tensión lógica es de 3,3 V y 5V y es compatible con la mayoría de los controladores como Arduino, Raspberry Pi, PLC. A continuación, se observa en la tabla 7 las características técnicas del controlador:

Tabla 7. Características de controlador CYTRON

Tensión de alimentación para motores.	5V-30V
Tensión de la parte lógica.	3.3V-5V
Corriente por canal.	10A
Corriente momentánea.	30A
Número de canales	2
Puente H	Tipo NMOS
Frecuencia PWM	20 kHz
Dimensiones	84,5 x 62 mm
Peso	33,5 g
Botones para comprobación del sistema	2

Fuente: Cytron. (2021)

3.4.4. Tiras LED UV-C

Las cantidad y modelo de tiras LED UV-C propuestas para este trabajo de tesis son 04 tiras LED Klaran LE, son módulos "plug and play" para que los fabricantes puedan añadir rápidamente la desinfección con LEDs UVC a las

aplicaciones que requieren tratamiento de superficies o del aire. Klaran LE utiliza LEDs UVC de alto rendimiento de la serie WD de Klaran que emiten luz UV en el rango germicida ideal (260 nm a 270 nm). Klaran LE incluye un controlador LED de 12V montado en la placa para que el motor pueda integrarse convenientemente en una variedad de aplicaciones, como se muestra en la figura 43.

Figura 43: Tira de Luces LED UV-C – Klaran LE
Fuente: KLARAN. (2021)

La gama de longitudes de onda de 260 nm a 270 nm de Klaran WD de longitud de onda para la desinfección asegura que los productos construidos con Klaran LED requieren menos potencia UVC para lograr la certificación de certificación de rendimiento del producto que LEDs UV con longitudes de onda máximas superiores a 270 nm que las lámparas de mercurio a 254 nm.

En comparación con las lámparas UV, que están funcionando constantemente y requieren sustitución anual, los LEDs UVC se encienden sólo cuando se necesita la desinfección. Esto conserva la vida útil de los LEDs UVC y ayuda a prolongar los ciclos de mantenimiento de la desinfección por muchos años. La matriz de nitruro de aluminio de alta tenacidad elimina la necesidad de un costoso y voluminoso paquete de cuarzo sellado, lo que permite el funcionamiento de la más alta intensidad en un paquete compacto de bajo coste. Podemos encontrar las características

del LED propuesto en su hoja de datos (ver anexo 2), ubicada en la página web del fabricante, como se muestra en la Tabla 8.

Tabla 8. Características técnicas de LED Klaran LE

Voltaje de alimentación	11.4V-12.6V
Consumo energético	45W
Longitud de Onda	260nm-270nm
Modelo	LE-12V-9U-HC
Diseño de tira LED	9 und. De 60mW c/uno
Largo	250mm
Ancho	15mm
Rango de Temperatura	5°C-50 °C

Fuente: KLARAN. (2021)

3.4.5. Interfaz de potencia de tiras LED UV-C.

Debido al sumamente bajo consumo energético aproximados a los 300 mW por parte de las tiras LED UV-C, se considera más que suficiente la energía provista por las baterías de 11,1V que utilizamos en conjunto de un módulo elevador de voltaje a 12V, como se muestra en la figura 44.

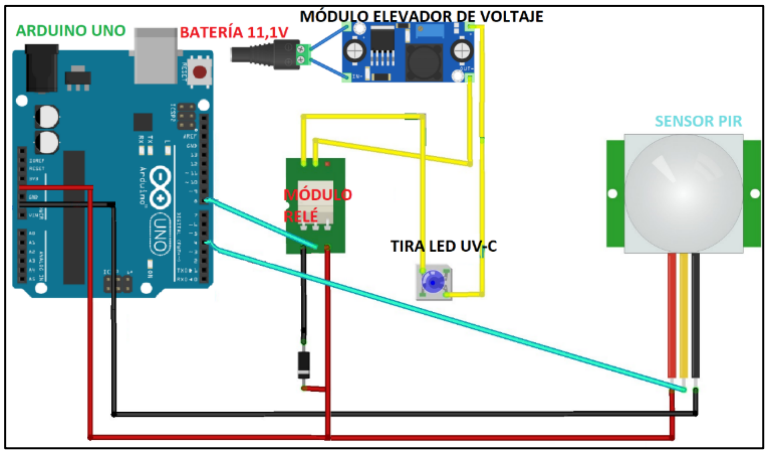


Figura 44: Conexión Arduino UNO con sistema de luces LED UV-C y PIR

Fuente: Elaboración propia

En el presente trabajo de investigación se plantea utilizar el mismo Arduino UNO que controla los actuadores, en conjunto de un módulo relé el cual va conectado directamente a la salida de 12V del elevador de voltaje y a la tira LED UV-C, así como también se ha implementado un sensor PIR que detectará el movimiento de personal médico o pacientes en el área a

desinfectar, previniendo cualquier posible daño inmediato por la exposición a estos rayos UV-C.

3.4.6. Selección del sensor LiDAR

- YDLIDAR G2

  Un *LiDAR* (acrónimo del inglés LIDAR, Laser Imaging Detection And Ranging) es un dispositivo que permite obtener la distancia comprendida entre el punto del emisor y un cuerpo o superficie lejana. Esta distancia se calcula midiendo el desfase comprendido entre la emisión del haz láser y la recepción de la señal reflejada. La tecnología LiDAR tiene aplicaciones en geología, sismología y robótica, especialmente en robótica móvil. Un uso muy recurrente en los últimos años es su incorporación en prototipos de vehículos autónomos. A grandes rasgos, los LiDAR pueden ser de dos tipos: mecánicos y sólidos. En el primer caso, emplean un engranaje móvil para crear un amplio campo de visión como en la figura 45. Sin embargo, cargan una gran ratio señal-ruido y su construcción es bastante voluminosa. En cambio, los LiDAR de estado sólido no tienen componentes móviles, reduciendo así el ruido. Este tipo, no obstante, reduce también su campo de visión, por lo que se suelen emplear múltiples canales para poder competir en este aspecto con los mecánicos. En el caso del presente trabajo, el dispositivo empleado será el YDLIDAR G2 de la firma YDLIDAR.

Figura 45: YDLIDAR G2

Fuente: YDLIDAR. (2021)

Su hoja técnica proporcionada por el fabricante YDLIDAR muestra su alcance, resolución y principales características como se muestra en la tabla 9. (ver Anexo 1)

Tabla 9. Características de sensor YDLIDAR G2

Item	Min	Typical	Max	Unit	Remarks
Ranging frequency	-	5000	-	Hz	Support customization
Motor frequency	5	7	12	Hz	Software speed control, can be customized
Ranging distance	0.12	-	12	m	80% reflectivity object
Scanning angle	-	0~360	-	Deg	-
Absolute error	-	2	-	cm	Distance≤0.5m
Relative error	-	1.0%	-	-	0.5m<Distance≤6m

Fuente: YDLIDAR. (2021)

El funcionamiento de este dispositivo se basa en el principio de triangulación láser como se observa en la figura 46 e incorpora un sistema de visión y adquisición de datos de alta velocidad con el que consigue una frecuencia de muestreo de 16000 muestras por segundo.

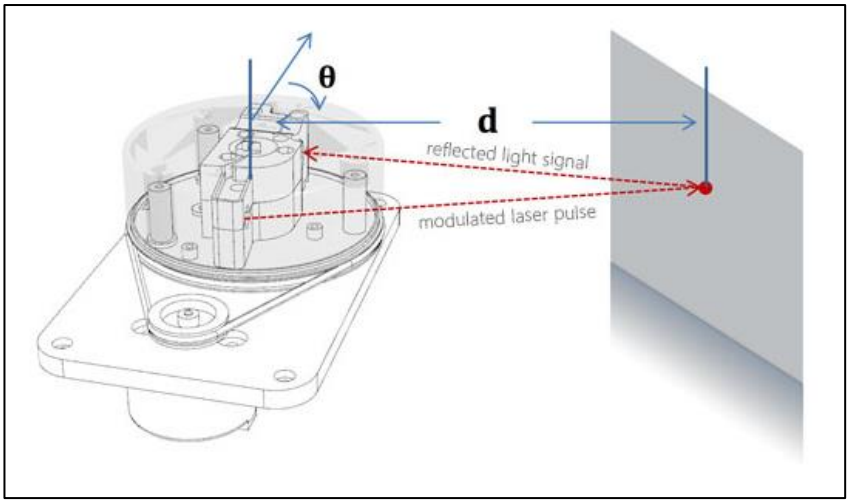


Figura 46:Triangulación láser del sistema LiDAR

Fuente: YDLIDAR. (2021)

Un aspecto relevante, sobre todo a la hora de su disposición en el robot es la ubicación de los sensores ópticos como se aprecia en la figura 47. Para que el LiDAR ofrezca todo su rango de detección es básico que esta quede libre de obstáculos, como podrían ser el resto de los periféricos montados

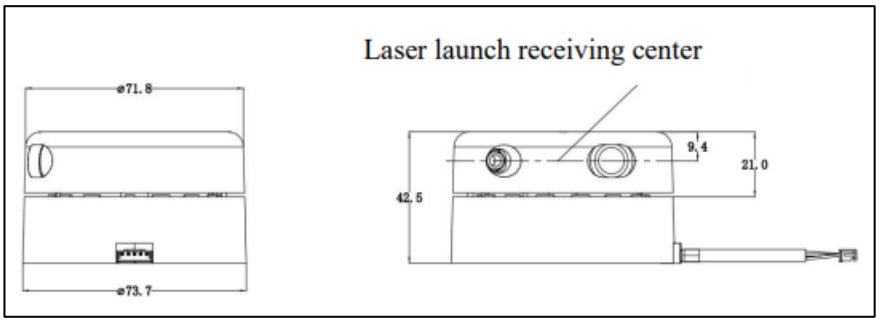


Figura 47:Vista de Perfil del sistema LiDAR y sensores ópticos

Fuente: YDLIDAR. (2021)

3.4.7. Selección de la Batería

Para la selección de la batería se hizo un cálculo de potencia independiente de cada componente del robot móvil, como los motores, las tiras LED UV-C, Arduino y el Raspberry Pi, luego se sumaron todas las potencias de cada componente y se obtuvo la potencia total del sistema como se observa en la tabla 10.

Tabla 10. Cálculo de potencia eléctrica del robot móvil.

Componentes	**Cantidad**	**Voltaje (V)**	**Corriente (A)**	**Potencia (W)**
Tira LED UV-C	4	12	3.75	180
Motores	2	12	4.1	98.4
Arduino	1	5	0.046	0.23
Raspberry Pi	1	5	3	15
Sensor LiDAR	1	5	0.5	2.5
			Potencia Total	296.13

Fuente: Elaboración Propia

Luego teniendo en cuenta que la potencia total del sistema es de 296.13W y que se necesita una autonomía del robot de 4 horas para cumplir el

requerimiento del sistema mencionado en la tabla 1, se procedió a hacer el cálculo siguiente:

$$I = \frac{P}{V} \tag{7}$$

Donde:

P: Potencia total del sistema en Watts

V: Tensión nominal de la batería a seleccionar en voltios.

I: Corriente consumida por el sistema en amperios.

De la ecuación (7):

$$I = \frac{296.13W}{12V}$$

Entonces:

$$I = 24.68\ Amperios$$

Luego se procedió a hacer el cálculo de carga eléctrica requerido que debe tener la batería para una corriente de 24.68 amperios que consume el sistema durante 4 horas de funcionamiento continuo del robot móvil.

$$t = \frac{C_b}{I_s} \tag{8}$$

Donde:

t: Tiempo de funcionamiento continuo del robot móvil en horas.

Cb: Carga eléctrica de la batería en Amperio-hora.

Is: Corriente eléctrica consumida por el sistema en Amperios.

De la ecuación (8):

$$4\ h = \frac{C_b}{24.68\ A}$$

Entonces:

$$C_b = 98.72Ah$$

En base a estos datos se hizo la elección de la batería de Litio "SuperPack" tomando como criterio principal la carga eléctrica nominal de 100Ah que posee, como se aprecia en la tabla 11, lo cual se aproxima muy bien a la carga de 98.72Ah calculado. (ver Anexo 3).

Tabla 11. Características técnicas de la batería de Litio SuperPack

Química	LiFePO4
Tensión nominal	12.8 V
Carga eléctrica nominal	100 Ah
Cantidad de ciclos	2500 ciclos
Peso	9.5Kg
dimensiones	213x229x138mm
Clase de protección	IP 43
Configuración en serie	No
Configuración en paralelo	Si, sin limitación
Máxima corriente de descarga	30A

Fuente: Victron Energy. (2021)

La batería de Litio SuperPack (ver figura 48) es fácil de instalar y no necesita componentes adicionales. El interruptor interno desconectará la batería en caso de descarga excesiva, sobrecarga o temperatura demasiado alta. La batería solo puede conectarse en paralelo, lo cual no es un problema ya que los motores y las tiras LED UV-C trabajan con una tensión nominal de 12V y no es necesario la conexión en serie para elevar el voltaje.

Figura 48: Batería LiFePO4 SuperPack de 12.8V y 100Ah

Fuente: Victron Energy. (2021)

3.4.8. Simulación del sistema electrónico del robot móvil en Proteus 8

Para hacer simular el sistema electrónico se introdujo variables del motor en Proteus como la resistencia de carga del motor para ello se hizo un cálculo previo:

$$R = \frac{V}{I} \tag{9}$$

Donde:

R: Resistencia de carga del motor DC en Ohmios (Ω)

V: Tensión nominal del motor DC en voltios (V)

I: Corriente máxima del motor bloqueado en amperios (A)

De la ecuación (7):

$$R = \frac{12V}{4.1A}$$

Entonces:

$$R = 2.927\ \Omega$$

A continuación, se procedió a introducir los datos del motor en el simulador Proteus como se observa en la figura 49:

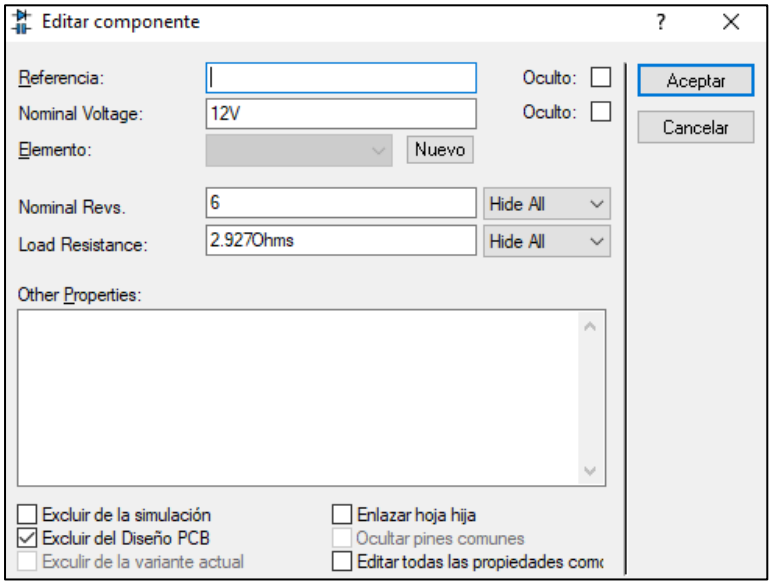


Figura 49: Introducción de parámetros del motor DC en Proteus 8.

Fuente: Elaboración propia realizada en software Proteus 8.

Luego también se modelaron las tiras led como una resistencia de 3.2 Ω (Ver figura 50) para que tenga un consumo aproximado de 3.75 Amperios a 12 voltios.

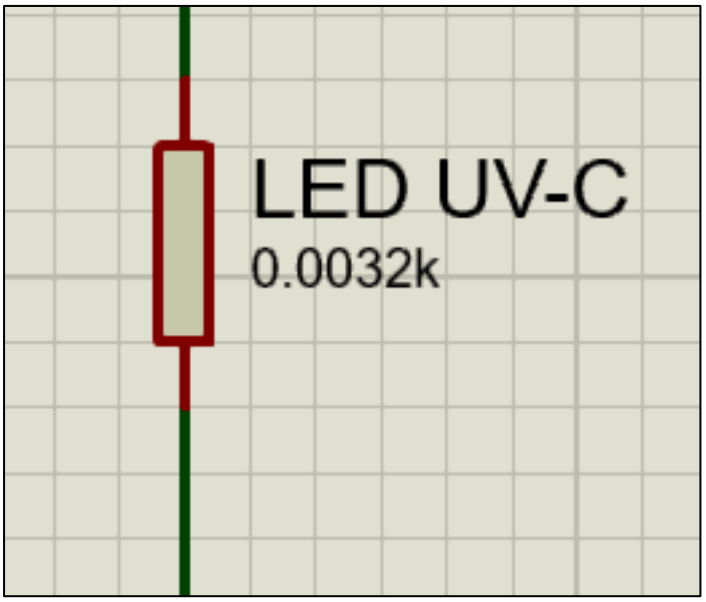


Figura 50: Modelo de LED UV-C en una resistencia de 32Ω

Fuente: Elaboración propia realizada en software Proteus 8.

Además, se configuró la resistencia interna a la batería de 0.4Ω como se observa en la figura 51 con los datos de voltaje de 12V y corriente de descarga máxima de 30Amperios obtenidos de la tabla 11.

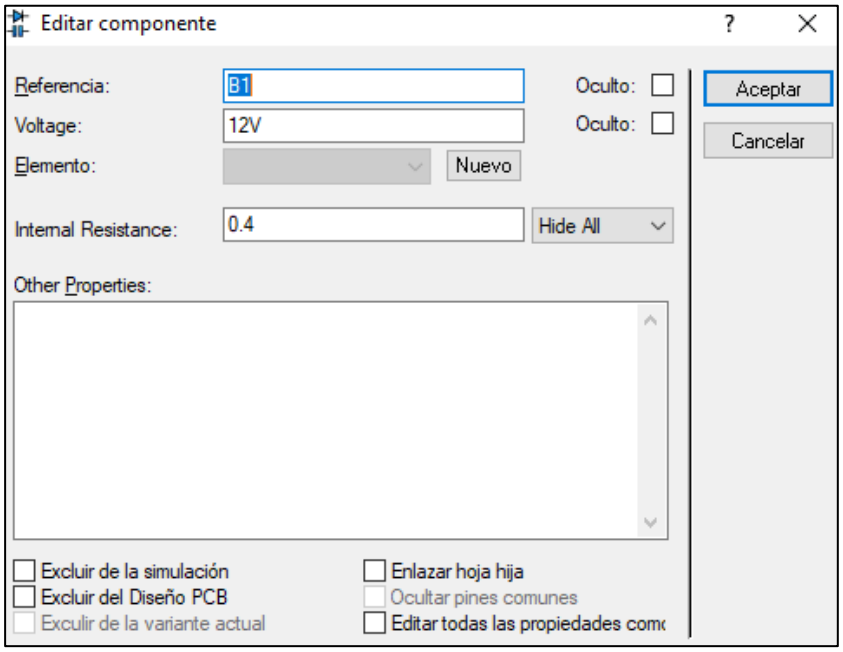


Figura 51:Configuración de resistencia interna de la batería.

Fuente: Elaboración propia realizada en software Proteus 8.

Teniendo todo lo mencionado en consideración se procedió a diseñar y simular un circuito electrónico en el software Proteus 8 como se aprecia en la figura 52 para asegurar la autonomía necesaria para asegurar la potencia de rayos UV-C y el sistema de alimentación adecuada.

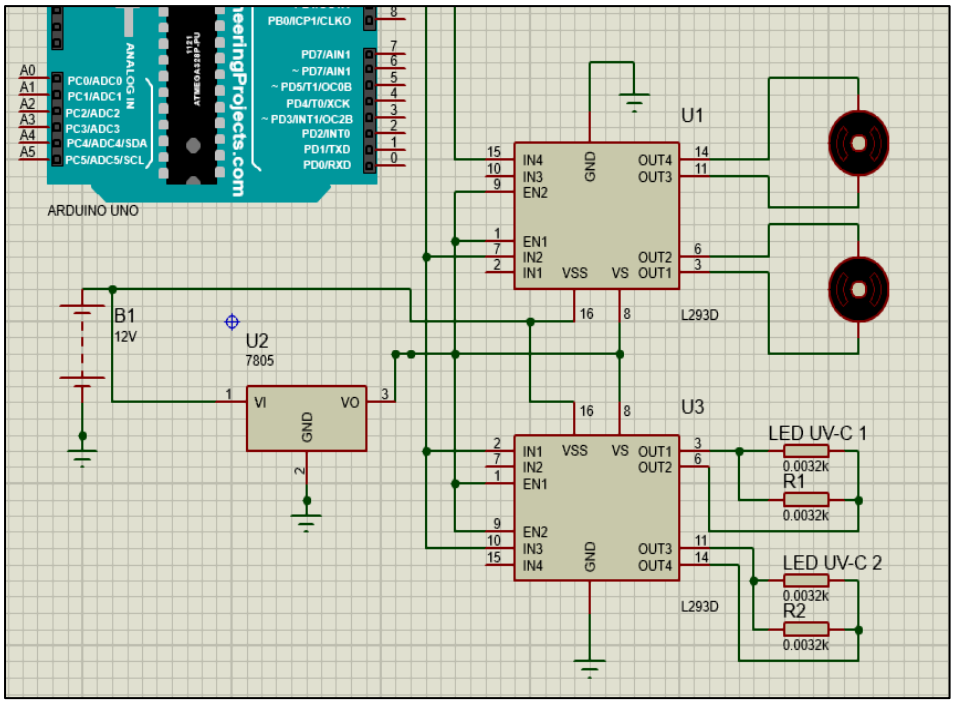


Figura 52: Circuito electrónico del sistema de alimentación del robot móvil.

Fuente: Elaboración propia realizada en software Proteus 8.

3.5. Diseño del sistema de control del Robot Móvil.

3.5.1. Descripción control de detección de obstáculos mediante el sensor LiDAR

En la Figura 53 se puede contemplar la arquitectura propuesta para el robot móvil, el sistema recopila información del mundo exterior a través del sensor LiDAR, se ha usado el sensor LiDAR modelo YDLIDAR G2, que utiliza luz infrarroja para medir distancias. Puede detectar distancias desde los 20 cm hasta los 120 cm con una precisión de 1.cm. Los datos recogidos son tratados en el microordenador que luego de procesarlos da órdenes al microcontrolador, que contiene el control basado en lógica difusa, y permite actuar sobre los motores que determinan la velocidad de cada rueda. El microcontrolador utilizado para este sistema es Arduino Uno, basado en el microcontrolador ATmega328. La placa de desarrollo consta de catorce pines E/S digitales y seis pines E/S analógicos, con una memoria flash de 32kB y una memoria EEPROM de 1 kB, con una frecuencia de 16 MHz.

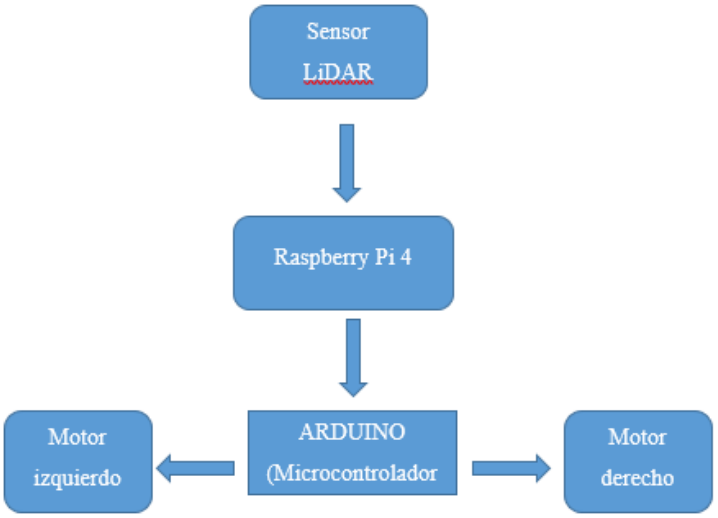


Figura 53: Arquitectura de hardware del robot móvil

Fuente: Elaboración propia

3.5.2. Instalación y Acondicionamiento de Ubuntu y ROS:

Para llevar a cabo lo planteado teóricamente y el diseño ingenieril que se describe en los puntos anteriores se empleó el sistema operativo Ubuntu 18.04.6 LTS (Bionic Beaver) principalmente debido a su estabilidad y compatibilidad con los distintos repositorios y dependencias utilizadas en el trabajo de tesis actual:

Se inicia esta etapa del trabajo al descargar Ubuntu 18.04.6 desde su página web oficial.

Se optó por configurar una máquina virtual en VMware Workstation 16 Player, descargado de su dirección web. cómo se observa en la figura 54 y no depender de una estación de trabajo independiente a las ya utilizadas para realizar los diseños, lo cual ciertamente incrementa los gastos financieros.

VMware Workstation 16 Player al igual que Ubuntu y todos los paquetes informáticos utilizados en nuestro presente trabajo de tesis son utilizados con la licencia gratuita o abierta, lo cual nos permite aligerar los gastos financieros que representan paquetes informáticos de esta índole, así como logísticos en los trámites de adquisición de estos.

Seguimos los pasos de instalación de VMware Workstation 16 Player y una vez ejecutado este programa seleccionamos la opción de crear una nueva máquina virtual la cual solicitó el archivo iso de Ubuntu 18.04.6 descargado antes.

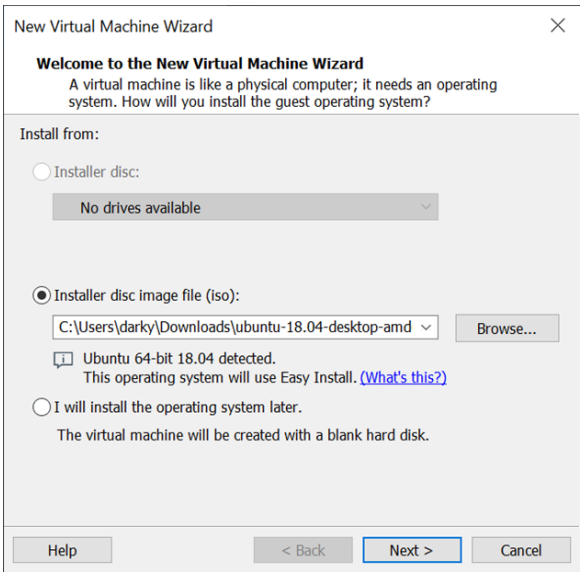


Figura 54: Ventana de asistente de máquina Virtual

Fuente: Captura de pantalla de Software VMware Workstation

Posteriormente se continuó con la instalación del sistema operativo Ubuntu en la máquina virtual, como se observa en la figura 55 el programa solicita un nombre de usuario, y una contraseña para mantener segura la información de la unidad virtual:

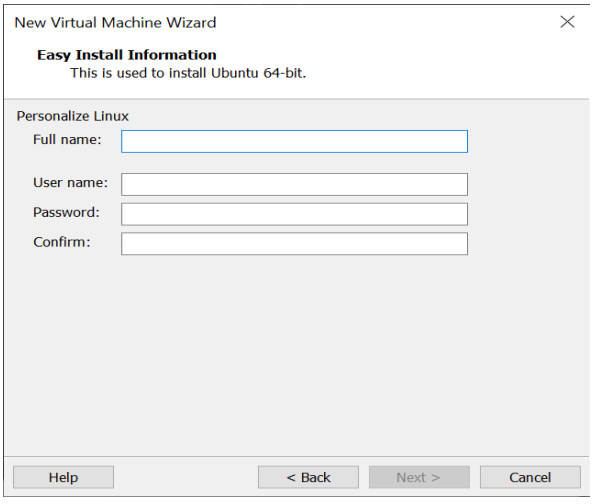


Figura 55: Ventana de asistente de máquina Virtual para creación de usuario.

Fuente: Captura de pantalla de Software VMware Workstation

Luego, como podemos observar en la figura 56, nos solicita ingresar el nombre de la máquina virtual en el entorno de VMware y la ubicación donde se deseara mantener los archivos de esta máquina virtual.

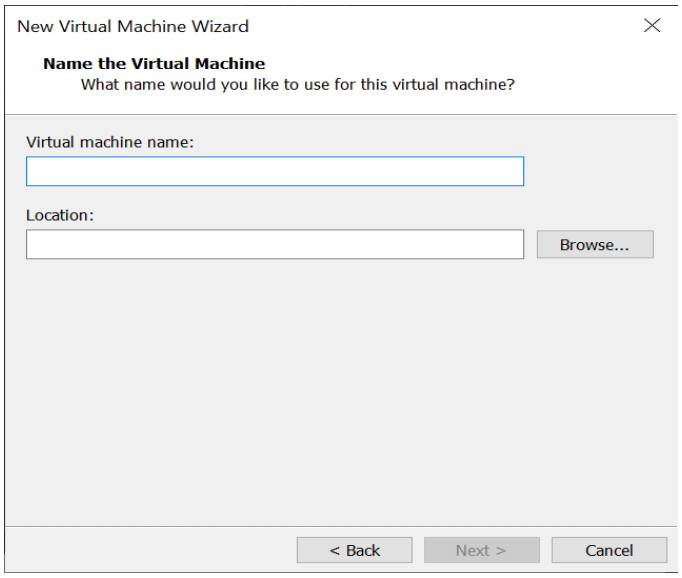


Figura 56: Asistente de máquina Virtual para localización de instalación.

Fuente: Captura de pantalla de Software VMware Workstation

En la cuarta ventana de este proceso se ingresaron los valores correspondientes al tamaño de disco virtual en donde se ejecutarán todos los procesos del sistema operativo Ubuntu. Asimismo, seleccionamos la opción "Store virtual disk as a single file".

Finalmente, como se observa en la figura 57 se configuran parámetros con los que se desea trabajar esta máquina virtual, se ingresaron los valores correspondientes al tamaño de disco duro virtual, memoria RAM virtual, adaptador de red y dispositivos conectados, en donde se ejecutarán todos los procesos del sistema operativo Ubuntu.

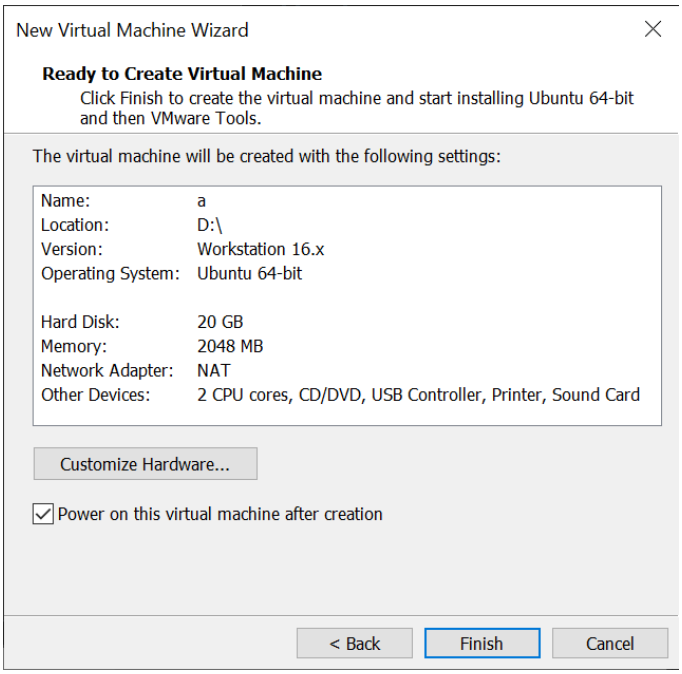


Figura 57: Ventana de configuraciones y parámetros de máquina virtual.

Fuente: Captura de pantalla de Software VMware Workstation

Antes de seleccionar “Finish” se procedió a seleccionar la opción “Power on this virtual machine after creation” como se aprecia en la figura 57, la cual inicializa el sistema operativo inmediatamente después de haberse apagado la máquina virtual. Es en este momento cuando el sistema operativo se instala en los discos de la máquina virtual, una vez finalizada la instalación se procedió a actualizar todas los subpaquetes informáticos que conforman nuestro sistema operativo Ubuntu mediante la aplicación preinstalada llamada “Software Updater”, la cual buscó y solicitó instalar todas las actualizaciones necesarias para dejar el sistema lo más estable y con máximo rendimiento posible. En esta etapa es cuando actualizamos los repositorios disponibles de Ubuntu Melodic con el fin de permitir la instalación de paquetes de información de ROS, mediante la siguiente línea de código ejecutada en la interfaz textual terminal de Ubuntu apreciada en la figura 58:

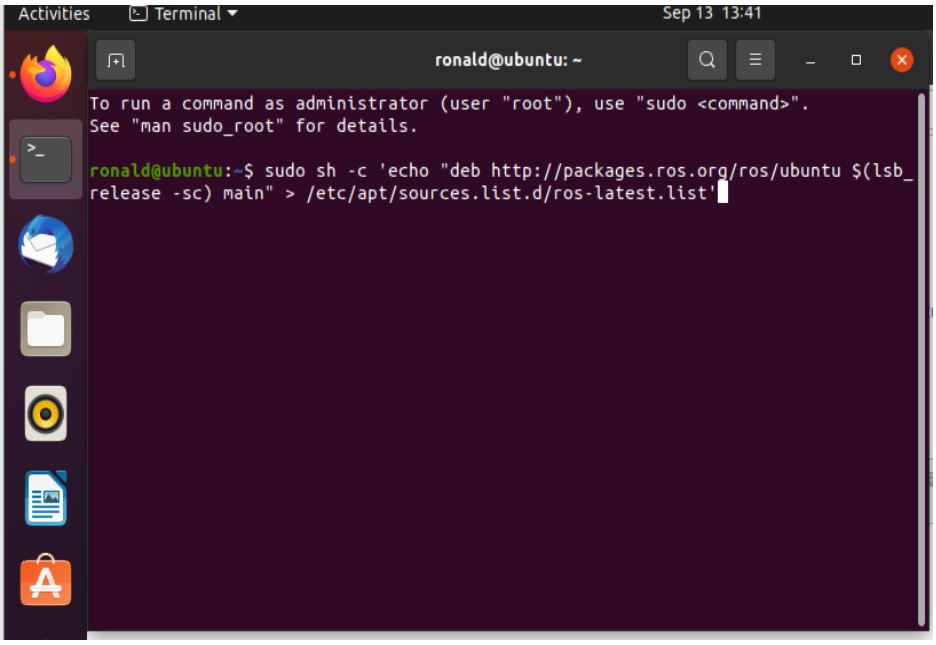


Figura 58: Instalación de paquete ROS.

Fuente: Captura de pantalla de interfaz textual en Ubuntu.

Primero, asegúrese de que los repositorios indexados de su paquete Debian estén actualizados (Ver figura 59):

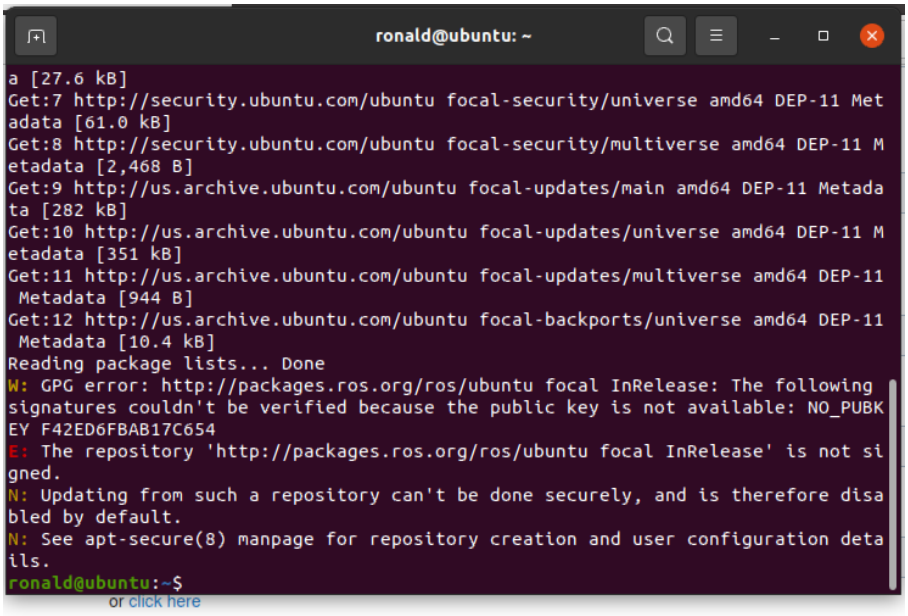


Figura 59: Instalación de paquete Debian

Fuente: Captura de pantalla de terminal en Ubuntu

Se agrega el repositorio de ROS y llaves necesarias con la siguiente línea de código: "curl -s https://raw.githubusercontent.com/ros/rosdistro/master/ros.asc | sudo apt-key add –"

Las "llaves de autenticación" suelen obtenerse del mantenedor del repositorio de software. Estas son utilizadas para poder validar la fuente original del paquete de donde se cree que se está obteniendo, con el fin de evitar que terceros introduzcan paquetes dañinos en el paquete a descargar.

Luego se procede a verificar si el índice del paquete DEBIAN se encuentra actualizado ya reconociendo los repositorios instalados antes, con la siguiente línea de código: “Sudo apt update”

Una vez llegado a este punto, es seguro proceder con la instalación del paquete completo de ROS (Desktop-Full Install), con el que se instalarán de manera automática también los paquetes de rqt, rviz, librerías robot-genetic, simuladores y percepción 2D/3D (ejem. Gazebo), todo esto mediante la siguiente línea de código: “sudo apt install ros-melodic-desktop-full”

Una vez finalizado este último paso de instalación, se procede a configurar el entorno de ROS, primero ejecutando las líneas de códigos para agregar la dirección de ROS a la ruta principal con lo que las variables del entorno de ROS se añadan automáticamente a la sesión de bash cada vez que se lanza un nuevo Shell: “echo "source /opt/ros/melodic/setup.bash" >> ~/.bashrc (enter) source ~/.bashrc”. Se verifica la nueva ruta principal con el código: “echo $ROS_PACKAGE_PATH”

Hasta ahora hemos instalado lo que se necesita para ejecutar los paquetes centrales de ROS. Para crear y gestionar nuestros espacios de trabajo ROS propios, hay varias herramientas y requisitos que se distribuyen por separado. Por ejemplo, rosinstall es una herramienta de línea de comandos de uso frecuente que nos permite descargar fácilmente muchas fuentes para paquetes ROS con un solo comando. Para instalar esta herramienta y otras dependencias para construir paquetes ROS, ejecutamos: “sudo apt install python-rosdep python-rosinstall python-rosinstall-generator python-wstool build-essential”

Antes de que se pudieran utilizar muchas herramientas y como último paso de la instalación y configuración de ROS, se tuvo que inicializar rosdep. rosdep nos permitió instalar fácilmente las dependencias del sistema para la

fuente por compilar y es necesario para ejecutar algunos de los componentes básicos en ROS, para instalar esta herramienta basta con ejecutar la siguiente línea de código: “sudo apt install python-rosdep”

Una vez instalado, lo inicializamos ejecutando:

- sudo rosdep init
- rosdep update

Adicionalmente, y como algo totalmente no indispensable, instalamos Python 3 para posteriormente no tener problemas al ejecutar otros programas.

- sudo apt-get install python3-pip python3-yaml
- sudo pip3 install rospkg catkin_pkg

Una vez finalizada la instalación de ROS con todos los componentes y repositorios correspondientes se procedió con la integración del repositorio Turtlebot con ROS, primero ejecutando todos paquetes de dependencia para ROS mediante la ejecución del siguiente código en una nueva ventana de terminal como se observa en la figura 60:

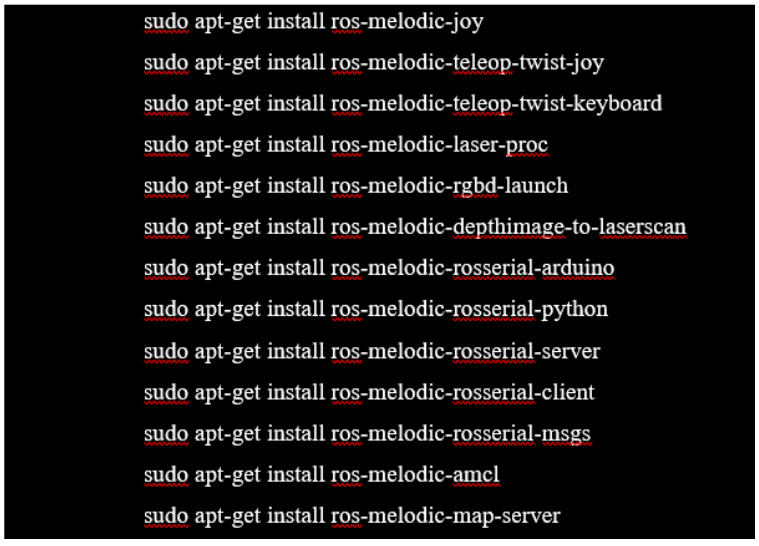


Figura 60: Instalación de paquetes en Turtlebot en ROS
Fuente: Elaboración propia

Ahora vamos a crear un espacio de trabajo para ROS con la siguiente línea de código:

- mkdir -p ~/catkin_ws/src
- regresamos al fichero anterior:
- cd ~/catkin_ws/
- y compilamos:
- catkin_make

Al ser ejecutado por primera vez se generó automáticamente un enlace CMakeLists.txt en la carpeta 'src'.

catkin es el sistema de construcción oficial de ROS y el sucesor del sistema de construcción original de ROS, rosbuild. catkin combina macros de CMake y scripts de Python para proporcionar algunas funcionalidades sobre el flujo de trabajo normal de CMake. catkin fue diseñado para ser más convencional que rosbuild, permitiendo una mejor distribución de los paquetes, un mejor soporte de compilación cruzada, y una mejor portabilidad. El flujo de trabajo de catkin es muy similar al de CMake, pero añade soporte para la infraestructura automática de "encontrar paquetes" y la construcción de múltiples proyectos dependientes al mismo tiempo.

Para terminar con la incorporación de este paquete nos dirigimos al fichero fuente de catkin, en donde están todos los repositorios a utilizar “cd ~/catkin_ws/src/”

y clonaremos los repositorios del GitHub de Turtlebot, para finalizar con la línea de código:

- git clone -b melodic-devel https://github.com/ROBOTIS-GIT/DynamixelSDK.git
- git clone -b melodic-devel https://github.com/ROBOTIS-GIT/turtlebot3_msgs.git
- git clone -b melodic-devel https://github.com/ROBOTIS-GIT/turtlebot3.git

Y finalizamos con la compilación y ejecución de los repositorios antes clonados: “cd ~/catkin_ws && catkin_make”. Una vez llegado a instalar todos los repositorios y dependencias, proseguimos con la simulación mediante la herramienta Gazebo, un entorno de desarrollo de simulación que puede ser programado y desarrollado con un robot virtual en tiempo real sin necesidad de detener la simulación en algún momento.

Cabe resaltar que, si bien hay otros entornos de desarrollo que forman parte de ROS de manera predeterminada, se ha preferido utilizar Gazebo por su integración pre incorporado con SLAM o Navegación manual, además que soporta sensores como IMU, LDS y cámara.

Para llevar a cabo esta simulación abriremos una nueva ventana de terminal y ejecutaremos la siguiente línea de código con la que nos ubicamos en el fichero fuente de nuestro entorno de trabajo catkin: “cd ~/catkin_ws/src/”

Luego clonamos el repositorio de simulación:

- git clone -b melodic-devel https://github.com/ROBOTIS-GIT/turtlebot3_simulations.git

Y finalizamos con la compilación y ejecución del repositorio antes clonado:

- cd ~/catkin_ws && catkin_make

Antes de proceder con las simulaciones en Gazebo, se procedió a utilizar la siguiente variable de entorno para reducir el conjunto de instrucciones gráficas a OpenGL2.

- echo "export SVGA_VGPU10=0" >> ~/.profile

Y a instalar los últimos repositorios para ROS de los sensores a simular en GAZEBO: “sudo apt-get install ros-melodic-tf2-sensor-msgs”

Adicionalmente utilizaremos el algoritmo RRT para que el robot planifique su trayectoria hacia todos los puntos finales alcanzables dentro de la cercanía del sensor, también conocidos como puntos fronterizos, lo que a su vez hace que el robot mapee nuevas regiones continuamente utilizando

SLAM mientras intenta alcanzar sus nuevos puntos finales lejanos. La aplicación de RRT a los robots móviles nos permite crear un robot autónomo auto explorador sin necesidad de intervención humana. La naturaleza del algoritmo tiende a inclinarse hacia las regiones no exploradas, lo que resulta muy beneficioso para las tareas de exploración del entorno. En este trabajo de tesis se puede encontrar información más detallada y el flujo de esta estrategia en el capítulo de marco teórico.

Para dar por finalizada la instalación de repositorios y dependencias adicionales instalamos los siguientes módulos de python para inicializar el paquete del algoritmo RRT:

- sudo apt-get install python-opencv
- sudo apt-get install python-numpy
- sudo apt-get install python-scikits-learn

Ahora nos dirigimos a la carpeta raíz de nuestra área de trabajo o workspace mediante código de terminal: “cd catkin_ws/src” y clonamos el repositorio del algoritmo RRT mediante: “git clone -b melodic-devel https://github.com/hasauino/rrt_exploration.git”, luego procederemos a compilar en nuestro sistema operativo ROS mediante: “cd ~/catkin_ws && catkin_make”

Abrimos una nueva ventana de terminal y procedemos a ejecutar los siguientes códigos de pilares de navegación.

- sudo apt-get install ros-melodic-navigation
- cd catkin_ws/src
- git clone -b melodic-devel https://github.com/ros-planning/navigation
- cd ..
- cd ~/catkin_ws && catkin_make

3.5.3. Integración de ROS y LiDAR

La integración del sensor LiDAR escogido con el entorno del sistema operativo robótico (ROS) fue realizado con el paquete YDLIDAR G2, desarrollado por EAIROBOT. El paquete fue descargado directamente del repositorio de GitHub donde se encuentran toda la información y notas relacionadas al uso de este. Algunos de los valores que se obtienen con este paquete son PointCloud2 y LaserScan, los cuales son de mucho provecho para poder llevar a cabo el presente proyecto. Una vez se haya instalado el sistema operativo y drivers del sensor, se pueden empezar las distintas pruebas del sistema propuesto. Para llevar a cabo esto de manera correcta se tuvo que ingresar las siguientes instrucciones que aparecen en la figura 61 en orden en el terminal del Raspberry Pi:

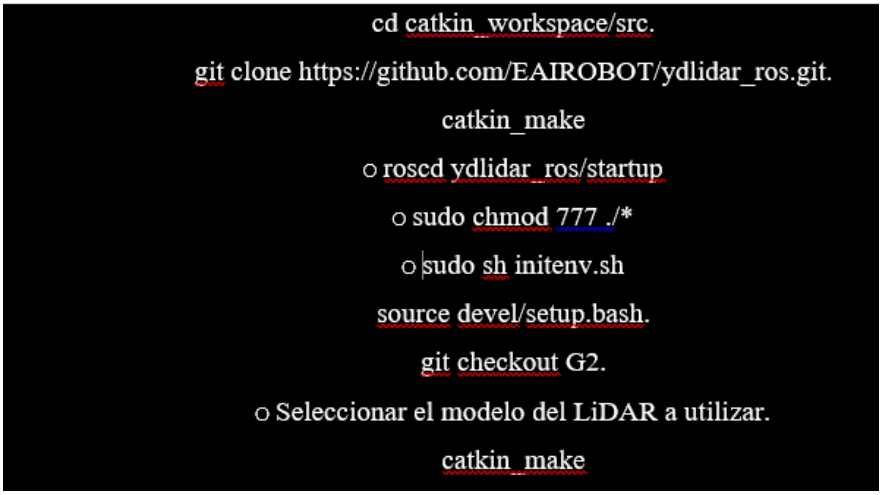


Figura 61: Instrucciones de integración con Raspberry Pi
Fuente: Elaboración propia.

Podremos realizar nuestra primera prueba con la función “roslaunch YDLiDAR_ros LiDAR.launch”, con lo que se visualiza las lecturas de escaneo en Rviz, solamente ingresando “ /scan”.

3.5.4. Comunicación Raspberry Pi (ROS) con Arduino:

Instalamos rosserial, un módulo de ROS que permite la comunicación Arduino-ROS, tanto en la Raspberry Pi como en el Arduino, para llevar a cabo eso ejecutamos las siguientes líneas de código en un terminal del Raspberry Pi:

- sudo apt-get install ros-melodic-rosserial-Arduino
- sudo apt-get install ros-melodic-rosserial
- cd catkin_ws/src
- git clone https://github.com/ros-drivers/rosserial.git
- cd catkin_ws
- cd ~/catkin_ws && catkin_make
- catkin_make install

En el IDE de Arduino instalamos la librería rosserial. Lo más fácil es hacerlo desde el propio IDE: buscamos rosserial en el Library Manager e instalamos, una vez finalizado nos muestra todos los ejemplos de la librería como en la Figura 62.

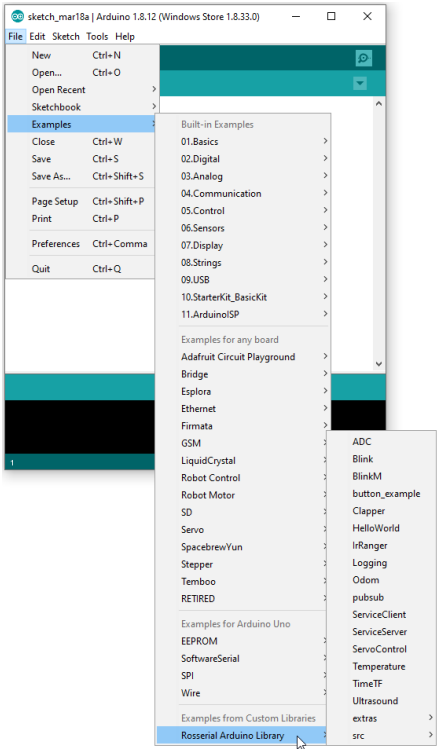


Figura 62:Arduino IDE con las bibliotecas Rosserial

Fuente: Captura de pantalla de Arduino IDE.

Para verificar que todo esté funcionando correctamente, probamos el ejemplo HelloWorld, de los ejemplos incluidos con la librería rosserial. Compilamos esta programación de ejemplo en el Arduino y lo conectamos al Raspberry Pi. Para ejecutarlo:

- En el Raspberry Pi, abrimos una ventana de terminal y ejecutamos el siguiente código: "roscore"
- En un segundo terminal del Raspberry Pi ejecutamos "rosrun rosserial_python serial_node.py /dev/ttyACM0"
- Cambiamos ttyACM0 por el puerto de nuestro Arduino, que se puede obtener navegando a ~/dev/, y observando qué archivos desaparecen y vuelven a aparecer cuando el Arduino se desconecta y se conecta.
- En un tercer terminal del Raspberry Pi ejecutamos "rostopic echo chatter" para ver los mensajes que se envían entre ambos microprocesadores, es decir que hay comunicación efectiva entre ambos dispositivos.

3.5.5. Control de velocidad de las llantas con la aplicación de lógica difusa

Para un desplazamiento más estable y menos caótico del robot móvil con direccionamiento diferencial se desarrolló un código de programación en Arduino aplicando principios de Lógica Difusa para el control de la velocidad de los motores DC mediante la modulación PWM.

Arduino recibe señales digitales de orden para mover los motores e información de distancia a la que se encuentran los obstáculos por parte del microordenador Rasberry Pi, por consiguiente, se instaló un módulo ROS en Raspberry Pi que habilita la comunicación serial con Arduino, llamado “rosserial”, para ello se ejecutó sus paquetes de instalación en el terminal textual de Ubuntu como se observa en la imagen 63.

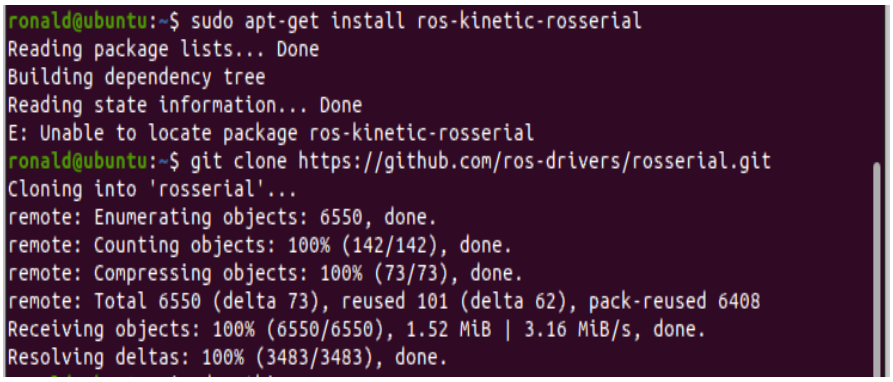


Figura 63: Instalación de paquete “Rosserial” en terminal de Ubuntu.

Fuente: Captura de pantalla de ventana terminal de Ubuntu.

Algo semejante ocurrió con la instalación de la librería rosserial en el Entorno de desarrollo integrado IDE de Arduino desde su gestor de librerías como se observa en la figura 64.

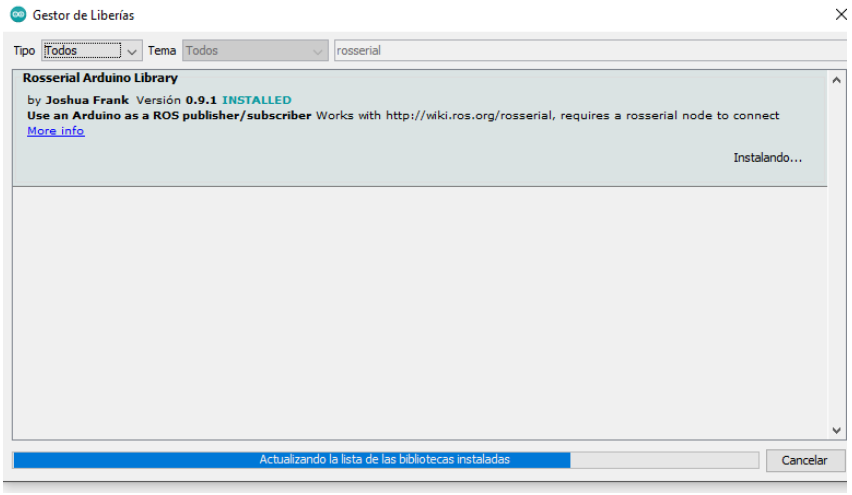


Figura 64: Instalación de librería rosserial en el IDE de Arduino

Fuente: Captura de pantalla de IDE Arduino

Una vez enlazado Raspberry Pi con Arduino se procedió a desarrollar el programa en el IDE del microcontrolador (Ver anexo 6), donde se tuvo 3 variables lingüísticas para la distancia a la que está el robot móvil de un obstáculo como “pequeño”, “seguro” y “grande”, cada una con su correspondiente rango, la cual fue calculada por el microordenador Raspberry Pi, con una función de membresía tipo trapezoidal como se observa en la figura 65.

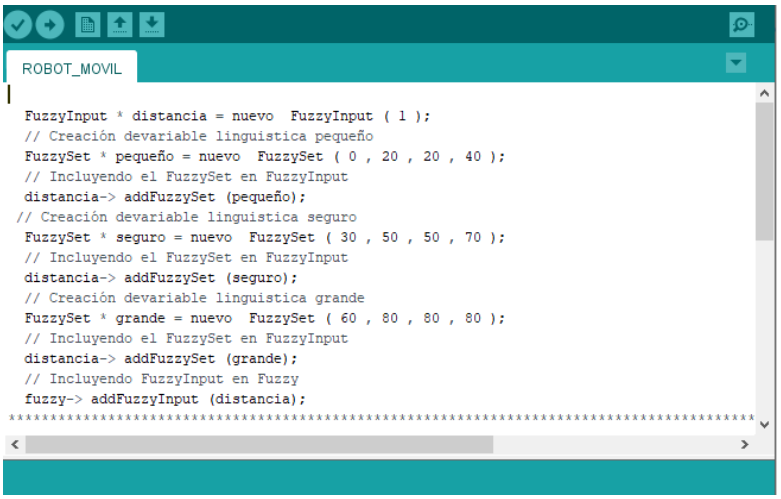


Figura 65: Creación de variables lingüísticas para la distancia

Fuente: Captura de pantalla IDE Arduino

Luego se crearon 3 variables lingüísticas para la velocidad que toma el robot de acuerdo a la distancia a la que esté del obstáculo como “lento”, “promedio”, “rápido” como se observa en la figura 66.

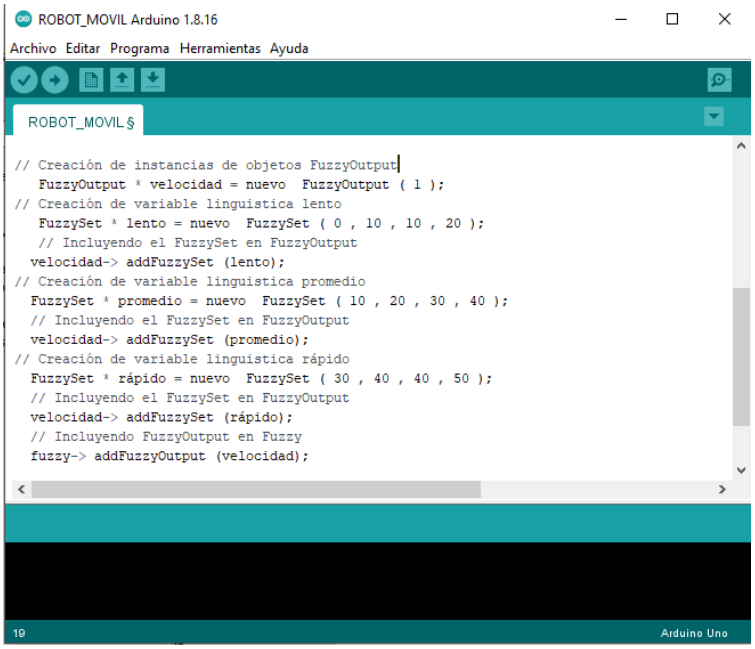


Figura 66: Creación de variables lingüísticas para la velocidad de los motores

Fuente: Captura de pantalla IDE Arduino

Por último, se crearon 3 funciones de membresía también llamadas reglas difusas con las variables ya declaradas en el código Arduino, las cuales son: “si la distancia es pequeña entonces se asigna la velocidad lenta”, "si la distancia es segura entonces se asigna la velocidad promediada" y "Si la

distancia es grande entonces se asigna la velocidad alta", tal como se observa en la figura 67.

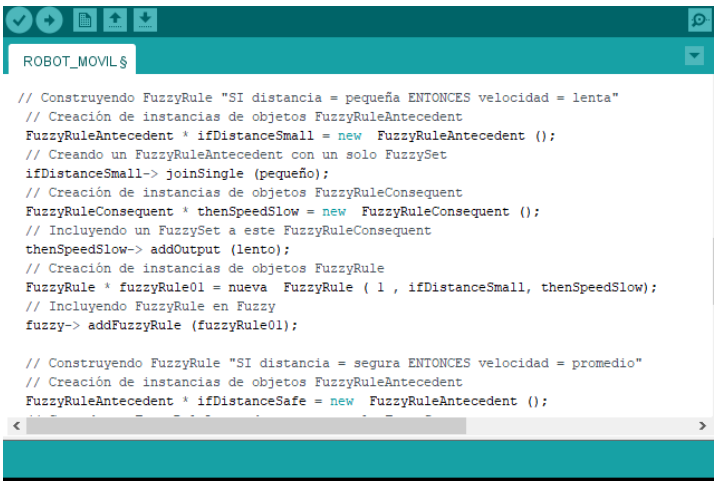


Figura 67: Creación de reglas difusas en Arduino

Fuente: Captura de pantalla IDE Arduino

Se empleó la función defuzzify "defusificar" para obtener la velocidad deseada y finalmente se logró enviar la señal por medio de PWM a los pines del Arduino (figura 68).

Figura 68: Programación para él envió de señal PWM al controlador Cytron.

Fuente: Captura de pantalla IDE Arduino.

CAPÍTULO IV: ANÁLISIS DE RESULTADOS

En el presente capítulo, con el objetivo de validar el diseño mecánico, electrónico y el desarrollo de la programación del robot móvil se muestran tablas de resultados obtenidos de las simulaciones correspondientes. En el diseño mecánico se utilizó la herramienta Simulation del Software Solidwork, en diseño electrónico se utilizó el programa Proteus 8, y en el desarrollo de la programación se utilizó el entorno de desarrollo Integrado (IDE) de Arduino y el software de simulación Gazebo.

4.1. Resultados de la simulación del sistema mecánico del robot móvil.

Para la verificación de la resistencia de la estructura del robot móvil frente a diferentes fuerzas externas como el peso total del robot, se hicieron simulaciones en la herramienta Simulation del software de diseño Solidworks con diferentes fuerzas aplicadas sobre la barra de la estructura del chasis como se observa en la figura 69.

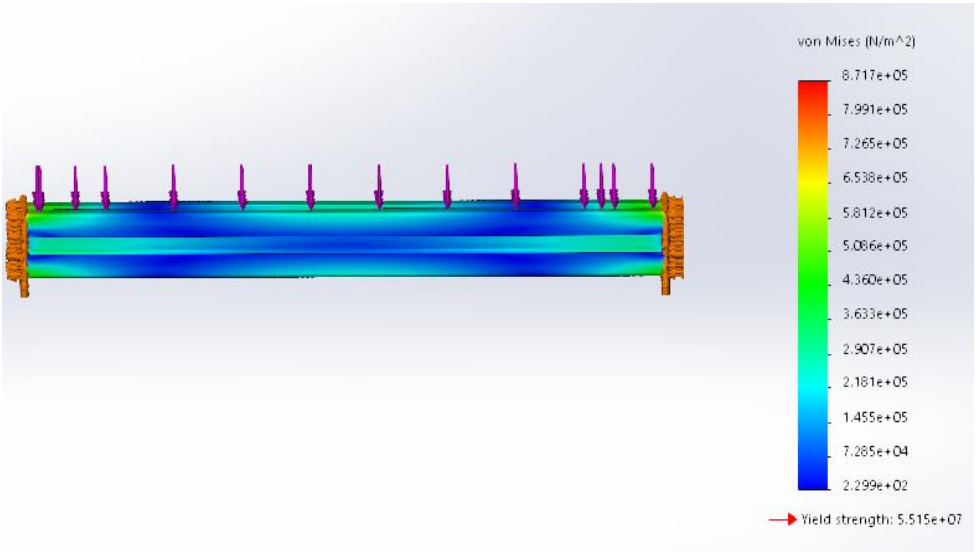


Figura 69: Prueba de simulación de esfuerzos con diferentes fuerzas.

Fuente: Elaboración propia hecha en Solidworks

Asignando diferentes pesos aplicados sobre la barra extruida TLOTS del chasis se muestran los datos tomados en la tabla 12 a partir de las simulaciones de esfuerzos y deformaciones hechas en Solidworks simulation.

Tabla 12. Esfuerzos y deformaciones máximas de la barra extruida.

Masa Total aplicada(Kg)	Masa aplicada a la barra (Kg)	Peso aplicado (N)	Esfuerzo máximo (N/m2)	Deformación máxima (mm)
16.255	4.06375	39.82475	871700	0.002121
20	5	49	1073000	0.00261
25	6.25	61.25	1341000	0.003263
30	7.5	73.5	1609000	0.003916
40	10	98	2145000	0.005221
60	15	147	3218000	0.007831
120	30	294	6436000	0.01566
240	60	588	12870000	0.03132
900	225	2205	48270000	0.1175
1000	250	2450	53630000	0.1305
1028.166	257.0415	2519.0067	55140000	0.1342

Fuente: Elaboración propia.

De estos resultados obtenidos a través del simulador, se dedujo que el peso aplicado sobre la barra de aluminio es insignificante respecto a los esfuerzos máximos que puede soportar de hasta 60 kg con una deformación mínima de 0.007831mm y un esfuerzo máximo de $1.287x10^7 \frac{N}{m^2}$. Para un análisis porcentual respecto del límite elástico del aluminio que es de $5.515x10^7 \frac{N}{m^2}$. se adjuntan datos en la tabla 13.

Tabla 13. Análisis porcentual respecto del límite elástico del aluminio 6061

Esfuerzo máximo (N/m2)	% respecto del Límite elástico
871700	1.6
1073000	1.9
1341000	2.4
1609000	2.9
2145000	3.9
3218000	5.8
6436000	11.7
12870000	23.3
48270000	87.5
53630000	97.2
55140000	100.0

Fuente: Elaboración propia.

Como se observa en la tabla 13 el esfuerzo de $8.717x10^5 \frac{N}{m^2}$ aplicado sobre la barra de la estructura del chasis del robot móvil fue solo el 1.6% del límite elástico del aluminio 6061, lo que significa que el robot móvil se desplazó sin problemas en la evasión de obstáculos, además con la misma estructura se puede soportar cargas mucho mayores como por ejemplo de 40kg sin problemas, siempre en cuando se utilice motores y baterías de mayor capacidad.

4.2. Resultados de la simulación del sistema electrónico del robot móvil

Para la comprobación de autonomía necesaria de la batería y asegurar la potencia de emisión de los rayos UV-C de los LED, y asegurar el desplazamiento del robot móvil mediante los motores durante 4 horas de funcionamiento continuo se procedió a hacer la simulación en Proteus, primero se simulo con una resistencia de carga máxima del motor de 2.9 Ω y se conectó en serie un amperímetro para la medición de corriente que fue de 19.3 Amperios como se aprecia en la figura 70.

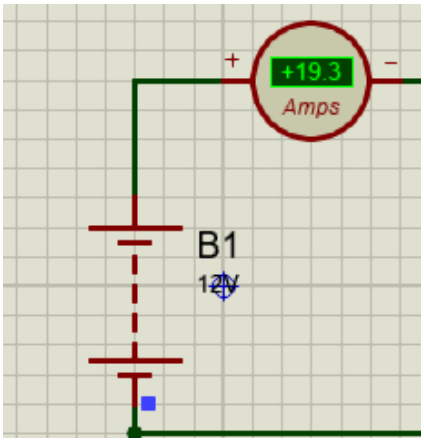


Figura 70: Medición de Amperaje consumido en Proteus.
Fuente: Elaboración propia realizada en software Proteus 8.

Posteriormente se hicieron varias pruebas con diferentes resistencias de carga del motor para de esa manera ver la duración de la batería frente a cambios de parámetros en los motores, para mayor detalle todos los datos se adjuntan en la tabla 14.

Tabla 14. Duración de batería según cambios en la resistencia de carga.

Resistencia de carga (Ω)	Intensidad (Amperios)	Carga eléctrica Ah	Tiempo de duración(horas)
2.9	19.3	100	5.18134715
3.5	18.4	100	5.434782609
4.5	17.3	100	5.780346821
5.5	16.6	100	6.024096386
6.5	16.1	100	6.211180124

Fuente: Elaboración propia

De acuerdo a la tabla 14 para una resistencia de carga mayor de los motores se obtiene una mayor duración de la batería, y esta resistencia disminuye cuando se somete al motor a grandes esfuerzos por ello se diseñó al robot móvil lo más ligero posible para que así se someta a un menor esfuerzo a los motores. En el caso de las tiras LED UV-C el amperaje es constante sin variaciones lo cual no afecta los resultados obtenidos.

4.3. Resultados en Matlab para regulación de velocidad aplicando lógica difusa

Lo desarrollado en Arduino se puede visualizar gráficamente en el software Matlab ingresando los variables lingüísticas con sus correspondientes funciones de membresía como se aprecia en la figura 71.

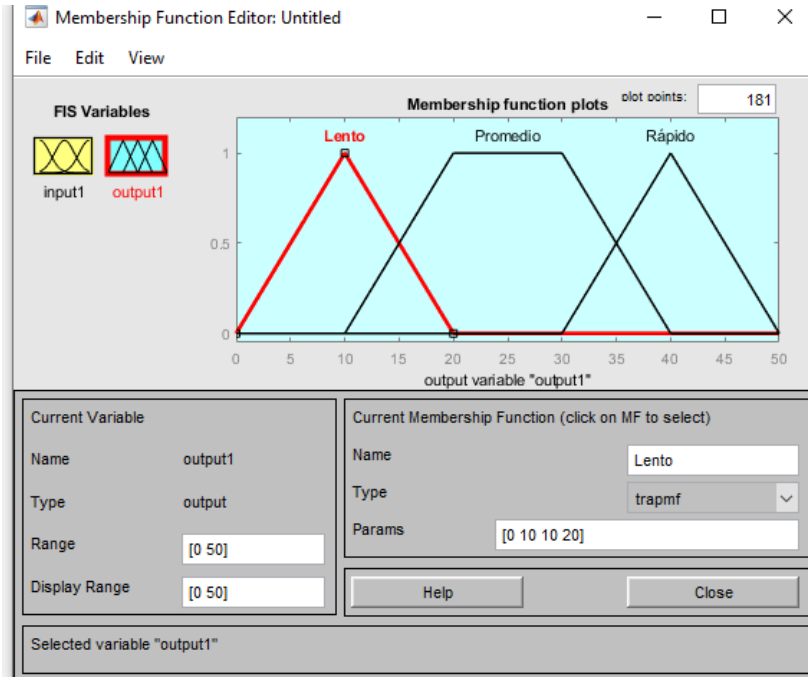


Figura 71: Introducción de entradas y salidas difusas en Matlab.

Fuente: Captura de pantalla de Matlab

Asignamos reglas difusas como se observa en la figura 72.

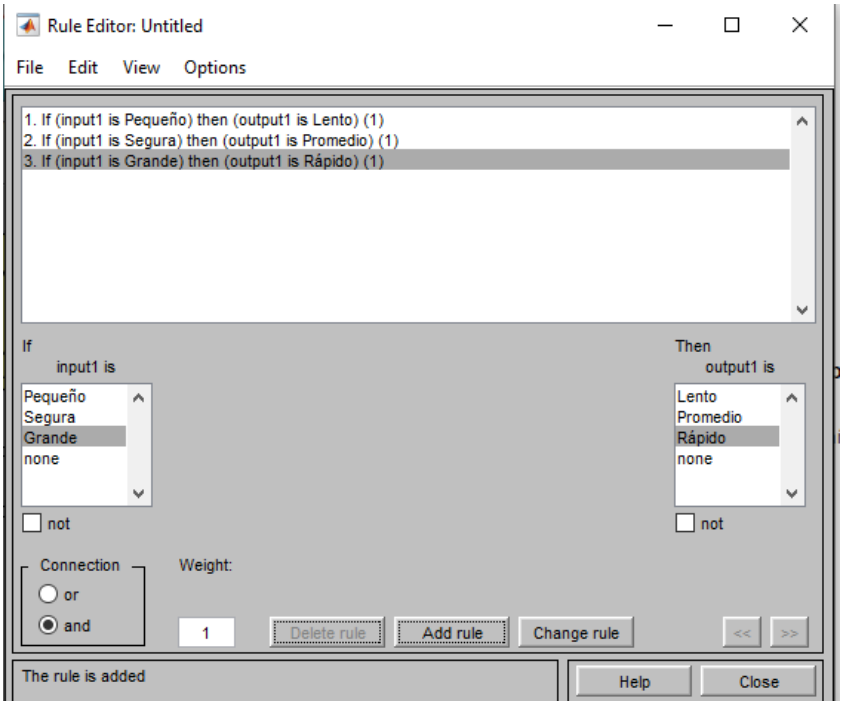


Figura 72: Asignación de reglas difusas

Fuente: Captura de pantalla de Matlab

Aquí se obtuvo que para distancias largas tenemos valores de velocidad altos y para distancias de 10, 20 y 30 pequeñas tenemos una velocidad constante de 10 como se observa en la figura 73.

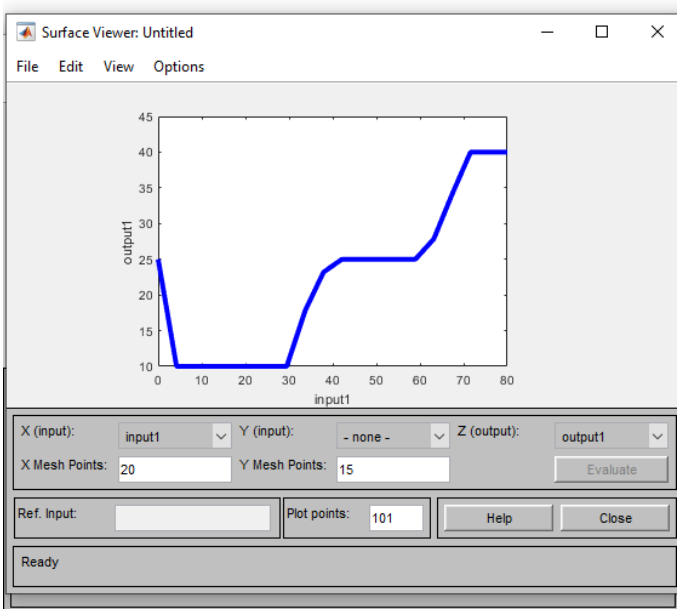


Figura 73: Gráfica de distancia vs velocidad según reglas difusas

Fuente: Captura de pantalla de Matlab

Analizando la gráfica de la figura 73 se elaboró la tabla 15 donde se obtuvo la velocidad mínima de 10 cm/s para distancias menores de 30 cm entre el sensor y el obstáculo, y una velocidad máxima de 40 cm/s para distancias mayores de 70 cm, por consiguiente, se desinfectará mejor le área asignada, debido a las bajas velocidades.

Tabla 15. Resultado del control difuso de velocidades

Distancia (cm)	**Velocidad (cm/s)**
10	10
20	10
30	10
40	24
50	25
60	25
70	37
80	40
90	40
100	40

Fuente: Elaboración propia

4.4. Simulación de evasión de obstáculos con GAZEBO en ROS

En este punto contamos con todo lo necesario para llevar a cabo las pruebas y simulaciones correspondientes a este trabajo de tesis, donde se evaluó el comportamiento y obtuvieron resultados en diferentes escenarios simulados con el sensor LIDAR. Para llevar a cabo esta simulación se procedió a ejecutar una serie de líneas de código en una nueva ventana de terminal, comenzando con la asignación del modelo de robot de movimiento diferencial a utilizar en nuestras pruebas utilizando el siguiente comando: "export TURTLEBOT3_MODEL=waffle_pi". Luego seleccionamos el entorno o mapa simulado en 3D en donde se designaron las tareas al robot mediante el comando: "roslaunch turtlebot3_gazebo turtlebot3_house.launch" y se designó la programación de movimiento autónomo para el robot, al cual fue cargada al robot durante la simulación mediante la siguiente línea de código: "roslaunch turtlebot3_gazebo turtlebot3_simulation.launch" cargada en la ventana Terminal como se aprecia en la figura 74.

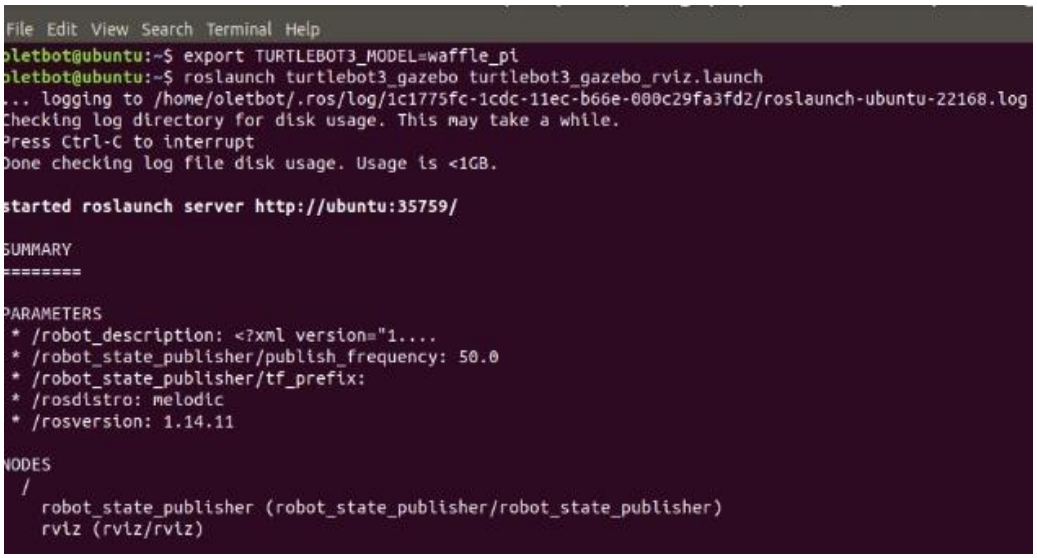


Figura 74: Código de programación cargada a Gazebo.

Fuente: Captura de pantalla del terminal de Ubuntu.

Una vez cargado el código el simulador Gazebo nos permite visualizar el entorno a escanear en una vista de planta como se aprecia en la figura 75.

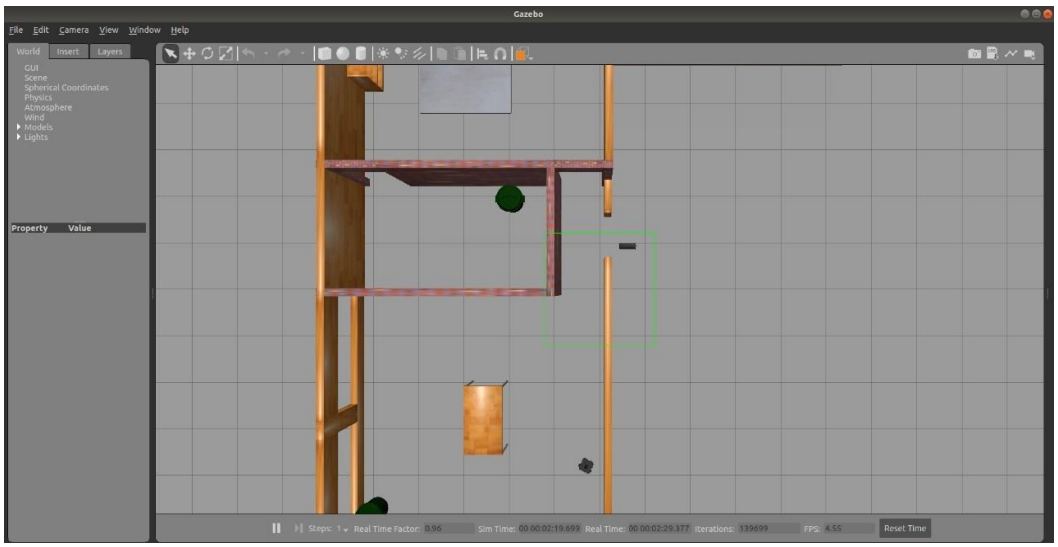


Figura 75: Vista de Planta de entorno a escanear.

Fuente: Captura de pantalla del simulador Gazebo.

Asimismo, se pudieron visualizar los resultados de los sensores mientras se ejecutaba la simulación como se aprecia en la figura 76, (Ver anexo 7), mediante la herramienta RViz en una nueva ventana de terminal introduciendo el siguiente comando: “roslaunch turtlebot3_gazebo turtlebot3_gazebo_rviz.launch”

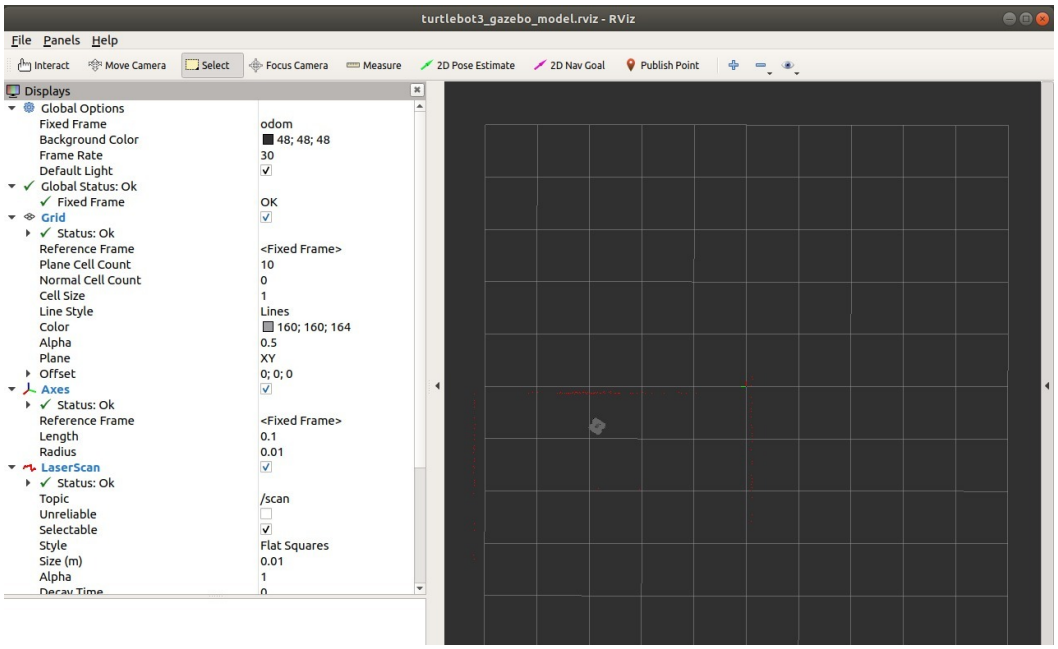


Figura 76: Visualización del resultado de los sensores.

Fuente: Captura de pantalla del simulador Gazebo.

4.5. Simulación de algoritmo RRT con GAZEBO en ROS

Luego de observar los vagos resultados de la simulación del evitamiento de obstáculos, debido a que no cuenta con un planeamiento de ruta, se decidió implementar un algoritmo más complejo que nos brinda la posibilidad de generar posibles rutas para realizar el mapeado de cada parte del espacio a desinfectar, por ende, la efectividad de este es mucho mayor. El algoritmo a escoger fue el RRT, por su facilidad en la implementación, así como los buenos resultados que se obtuvieron, como veremos más adelante. Cabe resaltar que esta herramienta viene preinstalada en el sistema ROS completo, por lo que no fue necesario ninguna instalación de repositorios ni dependencias previas. Asimismo, realizamos las pruebas respectivas con el algoritmo SLAM-RRT para generar un mapa y sus diferentes posibles trayectorias, en caso se desee brindarle una mayor autonomía al crear objetivos específicos dentro de un área previamente escaneada con la finalidad de realizar una mayor desinfección en determinados puntos. Cerramos todas las ventanas previamente abiertas para la simulación anterior y procedemos a abrir nuevos terminales para cada siguiente instrucción:

- export TURTLEBOT3_MODEL=waffle_pi
- roslaunch ros_autonomous_slam turtlebot3_world.launch

Abrimos otro terminal y ejecutamos:

- export TURTLEBOT3_MODEL=waffle_pi
- roslaunch ros_autonomous_slam autonomous_explorer.launch

La exploración RRT requiere que se defina una región rectangular alrededor del robot en la ventana RVIZ utilizando cuatro puntos (1-4) y un punto de partida (5) como se aprecia en la figura 77, para la exploración dentro de la región conocida del robot. El total de los cinco puntos debe definirse en la secuencia exacta que se indica en la Figura 40 utilizando la opción “Publish Points” de RVIZ.

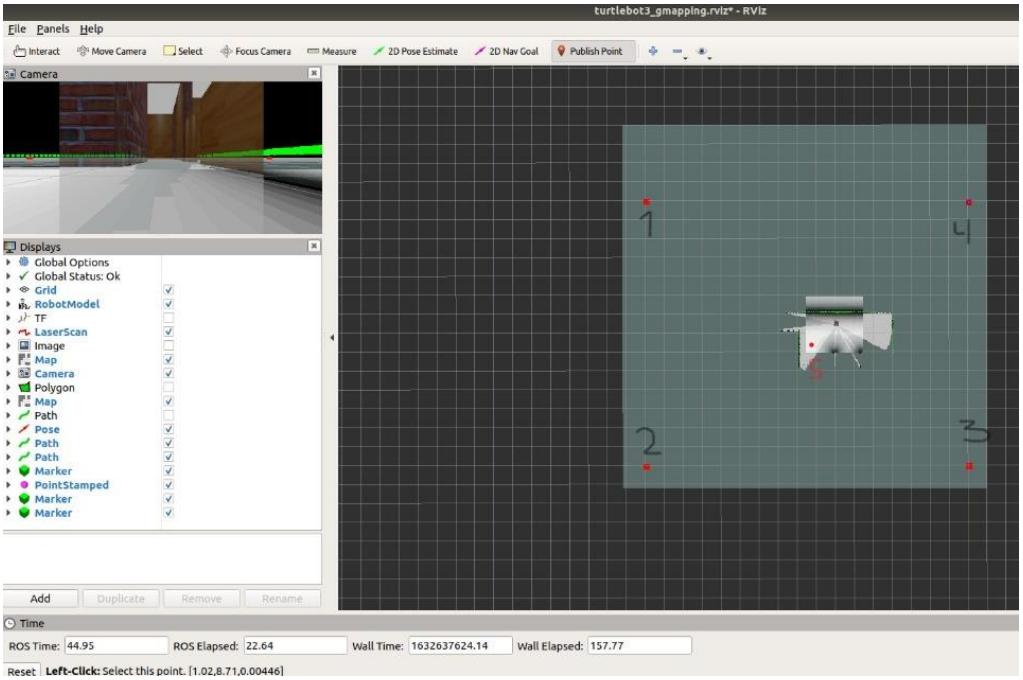


Figura 77:Escaneo de robot móvil entre 4 puntos definidos

Fuente: Captura de pantalla de simulación en Gazebo

Inmediatamente el robot empezó a escanear toda su área alrededor, recopilando todos los distintos puntos de escaneo LIDAR, que no solo permite guardar todos los parámetros del mapa, sino además va actualizando el mapa en tiempo real de acuerdo a nuevos obstáculos o cambios en su entorno conforme va desplazándose por este como se observa en la figura 78 (Ver anexo 8).

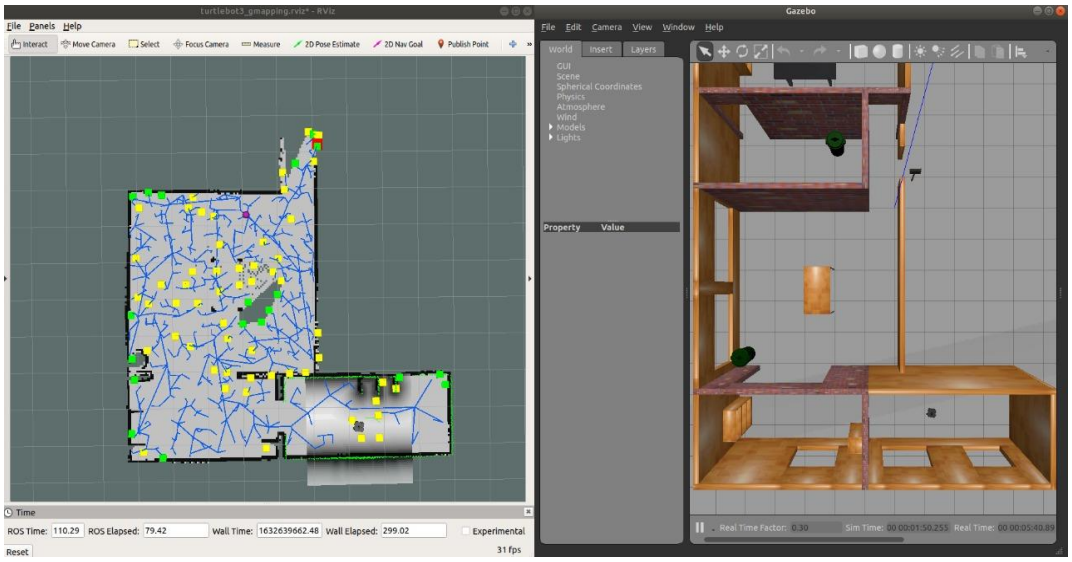


Figura 78: Escaneo total de un espacio cerrado

Fuente: Captura de pantalla de simulación en Gazebo.

Esto lo realizamos con la finalidad de poder obtener posteriormente un mapa en nuestro directorio, mediante el siguiente comando de línea: “rosrun map_server map_saver -f my_map”. de este procedimiento se obtuvo el mapa mostrado en la figura 79:

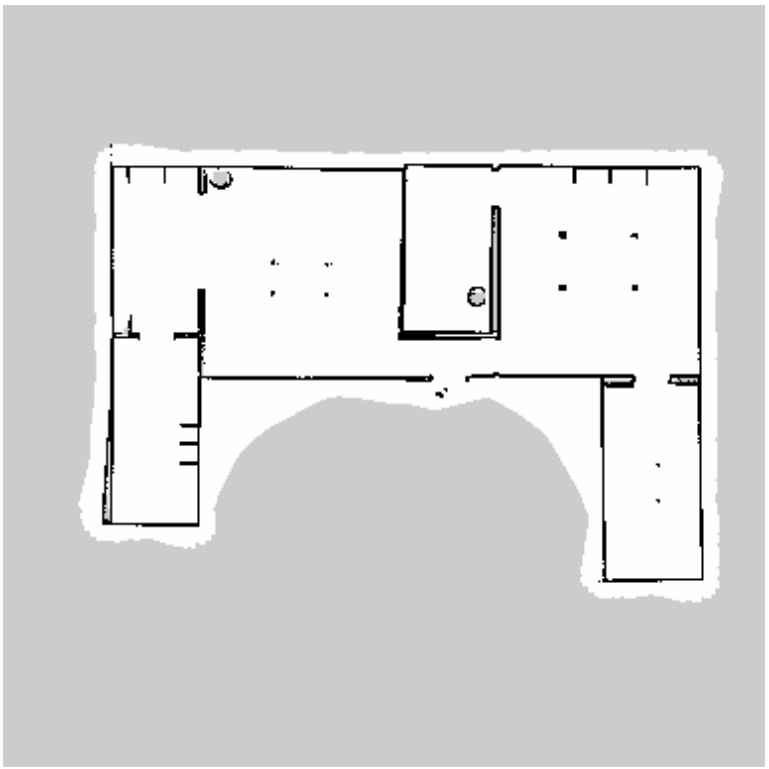


Figura 79: Mapa guardado del primer barrido de escaneo con LiDAR

Fuente: Captura de pantalla de escaneo LiDAR

Los archivos “my_map.pgm” y “my_map.yaml” se guardan en el directorio principal, por lo que movimos estos archivos a la carpeta de mapas del paquete mediante el comando “(catkin_ws\src\ros_autonomous_slam\maps)”. Ahora para definir objetivos específicos y su ruta planificada hacia estos dentro de nuestro mapa utilizamos los pilares de navegación que fueron generados previamente con el algoritmo RRT. Abrimos una ventana de RVIZ que nos mostró el mapa actual en tiempo real que nuestro robot estaba escaneando con el LIDAR con la superposición del mapa guardado previamente, por ende, obteniendo la localización del robot dentro del espacio, mediante el siguiente código: “roslaunch ros_autonomous_slam turtlebot3_navigation.launch”

Para definir los objetivos específicos utilizamos primero la estimación de la posición inicial en 2D del robot, herramienta del programa RVIZ, con esta herramienta seleccionada ubicamos el cursor encima del robot, con lo cual nuestro sistema determina la localización y orientación actual del robot. Una vez realizado este paso, definimos la posición del objetivo específico con la siguiente herramienta del programa RVIZ, con esta herramienta seleccionada ubicamos el cursor encima del objetivo que deseamos desinfectar con mayor rigurosidad como se observa en la figura 80, con lo cual nuestro sistema planea buscar la ruta más rápida y corta para llegar a dicho objetivo en el mapa.

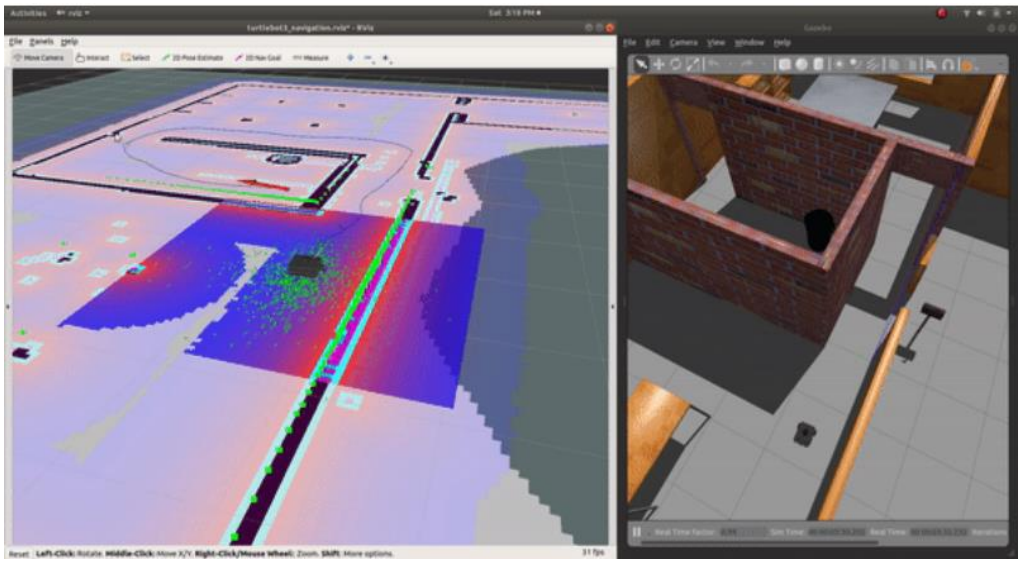


Figura 80: Escaneo en sensor LiDAR para encontrar la ruta más rápida.

Fuente: Elaboración propia.

Se realizaron múltiples intentos de escaneo y movilidad autónoma mediante el algoritmo RRT, el cual ciertamente presentó algunos inconvenientes de atasco que notamos más frecuentemente en las esquinas de las habitaciones en donde se ubicaban objetos estáticos tales como mueblería cercana como se muestra en la Figura 81, a continuación podremos ver estos resultados de casos de atasco en la tabla 16:

Tabla 16. Efectividad de escaneo ante un atasco por un determinado tiempo.

Área total escaneada (%)	Atasco por fallo de sensor	Tiempo transcurrido (segundos)
67	Si	5.18134715
37	Si	5.434782609
65	Si	5.780346821
86	Si	6.024096386
66	Si	6.211180124

Fuente: Elaboración propia

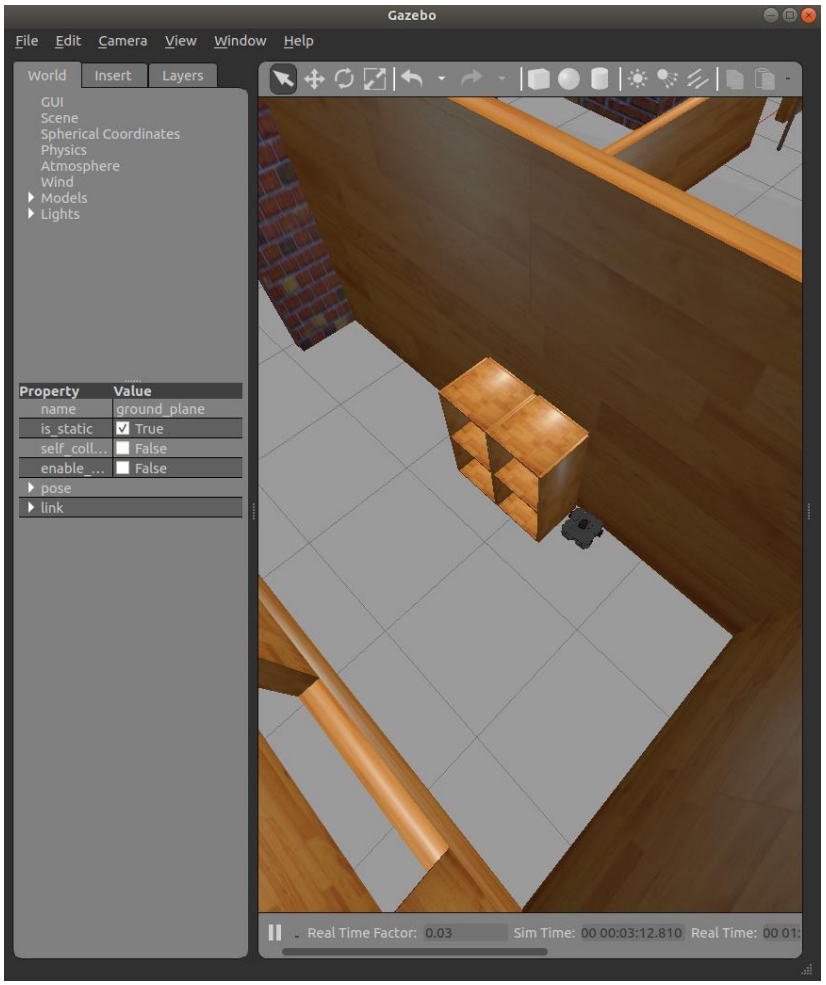


Figura 81:Atasco de robot autónomo en la esquina.

Fuente: Elaboración propia

De las 10 simulaciones continuas que se realizaron solamente en 02 oportunidades se logró de manera efectiva y autónoma el escaneo del total del área definida inicialmente, obteniéndose como resultado archivos en formato PGM y YAML que contienen la información del mapa tal como se muestra en la Figura 79. Asimismo durante las pruebas se realizó el seguimiento continuo de la velocidad angular y aceleración lineal del modelo simulado, valores que se obtuvieron de mediante la ejecución de los siguientes comandos en una ventana de terminal paralela a las de Gazebo y Rviz:

```
rostopic echo /imu
```

De la implementación de estas líneas de código se obtuvieron los resultados expuestos en la tabla 17 y figura 82.

Tabla 17. Simulaciones de velocidad y aceleración en un instante determinado.

Tiempo (Segundos)	Aceleración lineal (x;y;z)	Velocidad Angular (x;y;z)
02	-0.0214;-0.0029;0.0060	-0.000109;-0.000115;0.081000
04	0.0426;0.0462;-0.0023	0.000423;0.000767;-0.239781
08	-0.1436;0.5722;0.0010	-0.000592;-0.000846;0.363544
16	0.1052;0.4120;-0.0023	-0.000412;0.000957;0.537490
32	0.0470;0.5281;0.0060	0.000385;0.000720;-0.014503

Fuente: Elaboración propia

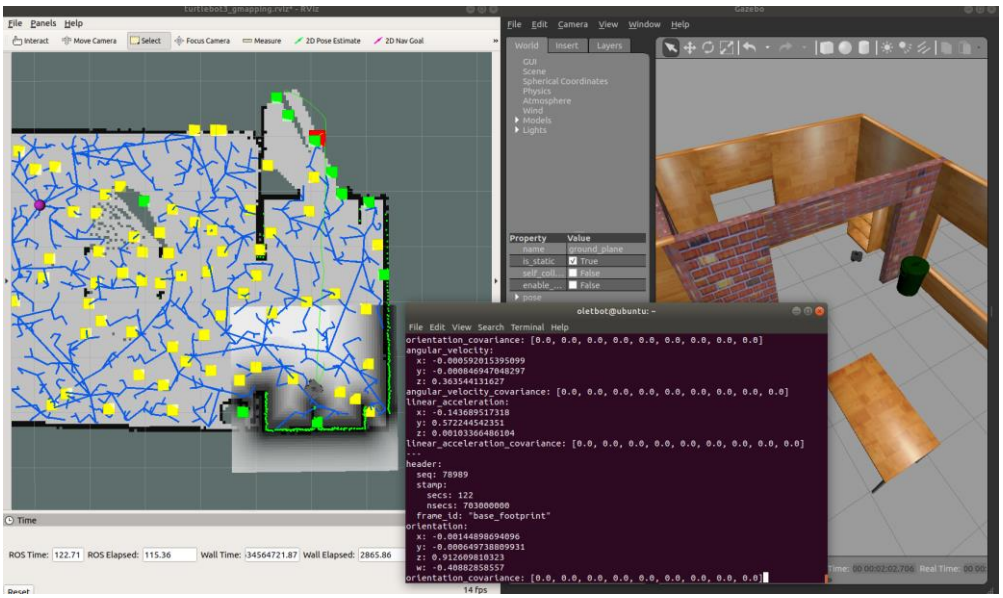


Figura 82: Visualización de variables de velocidad y aceleración del robot

Fuente: Elaboración propia

Como prueba final agregamos al mapa nuevos obstáculos cúbicos, esféricos y cilíndricos para evaluar el comportamiento del robot y verificar la eficacia en la evasión de los mismos mediante el algoritmo slam-rrt. Los resultados fueron satisfactorios al no colisionar con ninguno de los objetos cercanos de cualquiera de las formas antes mencionadas, por lo cual tuvo un desplazamiento y trayectoria continua, a continuación los resultados en la figura 83.

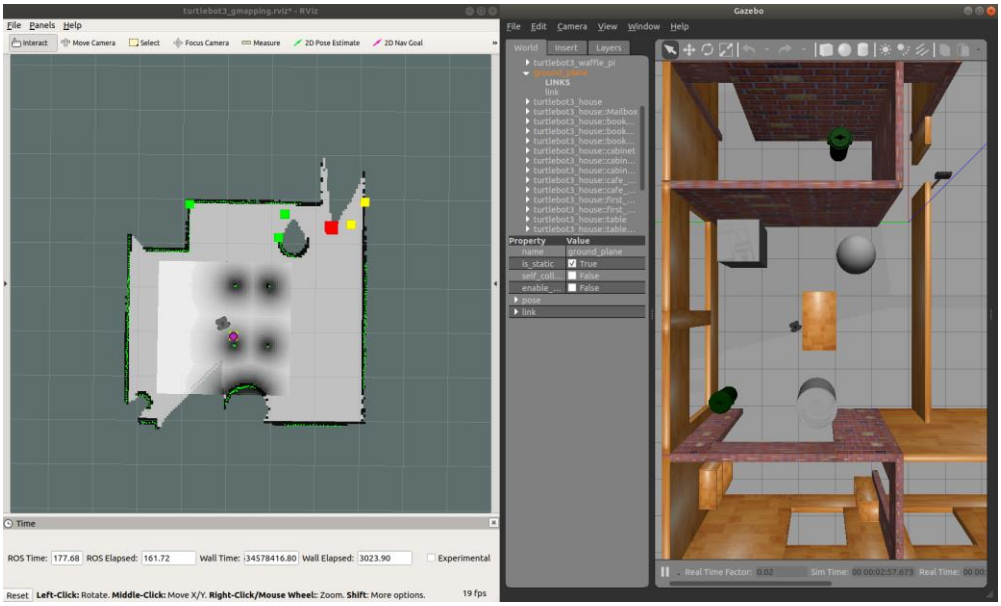


Figura 83:Escaneo e identificación de nuevos obstáculos de diferentes formas

Fuente: Elaboración Propia

CONCLUSIONES

1. Fue posible diseñar un sistema mecánico del robot móvil, que permita tener un desplazamiento eficiente en áreas con obstáculos, para ello se hizo un análisis de esfuerzos y deformaciones de la estructura de aluminio 6061 que conforma el chasis del robot con lo que se comprobó que el sistema mecánico soporta sin problemas esfuerzos de tracción y compresión producto de movimientos bruscos.

2. Fue posible diseñar un sistema electrónico con la autonomía necesaria para asegurar la potencia de emisión de rayos UV-C y los motores con el sistema eléctrico de alimentación adecuada debido a que se utilizó una batería con capacidad de carga eléctrica de 100Ah que cumple con los requisitos suficientes para que el robot móvil tenga un funcionamiento de 4 horas de funcionamiento continuo.

3. Fue posible desarrollar la programación del robot móvil para la evasión de obstáculos basado en inteligencia artificial y sistema de escaneo LiDAR, para ello se desarrolló un sistema de evasión de obstáculos con el sensor LiDAR utilizando el simulador Gazebo, en donde el robot móvil escanea más del 66% del área de la habitación antes de encontrar un inconveniente de atasco en las esquinas, además se desarrolló un control difuso en Arduino para el manejo de los motores DC, de tal manera que el robot móvil maneje velocidades bajas y haga efectiva una mejor desinfección en el área asignada.

4. Se comprobó la versatilidad de ROS, un programa que actualmente cuenta con un amplio apoyo de la comunidad robótica debido a sus características, en particular su capacidad para reutilizar y compartir el código. Por estas razones, se puede afirmar que Gazebo es una herramienta convincente para el trabajo que se ha realizado, proporcionando seguridad para el progreso del proyecto.

RECOMENDACIONES

1. Para la carga de la batería del robot móvil se recomienda utilizar un sistema autónomo que monitoree el nivel de voltaje hasta que descienda al mínimo predefinido, y luego a través de un algoritmo de programación, detenga la ejecución de tareas asignadas y dirija al robot móvil a la estación de carga mediante el mapeo y localización que hace el sensor LiDAR con la ayuda de ROS.

2. Para futuros trabajos de investigación se recomienda utilizar un comando por voz para evitar el contacto manual con el robot y así evitar posibles transmisiones de virus o baterías entre pacientes y personal médico.

3. Todavía se tienen que hacer muchas mejoras para perfeccionar el robot de desinfección. El control y la navegación son las dos partes fundamentales del robot. En el siguiente paso construiremos nuestra placa de navegación personalizada, desde el hardware y la integración del sistema operativo hasta los algoritmos. ROS proporciona comodidad para los desarrolladores de programas de robots. Sin embargo, ¿es lo suficientemente fiable para desarrollar un producto de uso comercial? ¿Cómo aplicar los algoritmos para una localización más exacta por coordenadas? Estas son cuestiones que se tratarán en el futuro.

REFERENCIAS BIBLIOGRÁFICAS

Balibouse, D. (22 de mayo de 2018). Los robots luchan contra las malas hierbas en desafío a los gigantes de los agroquímicos. *REUTERS*. Recuperado de https://www.reuters.com/news/picture/robots-fight-weeds-in-challenge-to-agroc-idUSKCN1IN0IK

Bestorq. (2021). *Tamaños de la correa de distribución.* Nebraska, EU.: BESTORQ inc. Recuperado de https://bestorq.com/timingsizelist.asp

Bonnell Aluminum. (2021). T-SLOTTED ALUMINUM EXTRUSION. Georgia, EU.: Kriesi. Recuperado de https://www.tslots.com/

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., y J. Leonard, J. (2016). Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age. *IEEE Transactions on Robotics*, *32*(6), 1309-1332. doi: 10.1109/TRO.2016.2624754

Chávez, A. (2020). *Sistema de detección y evasión de obstáculos por medio de un LIDAR 360° para un sistema aéreo no tripulado* (Tesis de pregrado). Pontificia Universidad Católica del Perú, Lima, Perú.

Cytron. (2021). *Usando MDD10A con ARDUINO UNO*. Penang, Malasia.: Cytron Technologies. Recuperado de https://tutorial.cytron.io/2015/04/05/using-mdd10a-Arduino-uno/

Hermosa, J. (2018). *Diseño de un sistema de desinfección de envases en el proceso de envasado de agua alcalina "Andea" de la empresa Cervecerías Cusco S.A.C. 2018* (Tesis de pregrado). Universidad Andina del Cusco, Cusco, Perú.

JSUMO. (2021*). Titan gearhead DC motor*. Estambul, Turquía.: T-Soft E-Commerce. Recuperado de https://www.jsumo.com/robotus-titan-dc-gearhead-motor-12v-200-rpm-601

Kampf, G., Todt, D., Pfaender, S., y Steinmann, E. (2020). Persistence of coronaviruses on inanimate surfaces and their inactivation with biocidal agents. *Journal of Hospital Infection,* Recuperado de

https://www.journalofhospitalinfection.com/action/showPdf?pii=S0195-6701%2820%2930046-3

KLARAN. (2021). *Su socio para LED UVC de alto rendimiento y soluciones para desinfección de agua, superficies y aire.* New York, EU.: Crystal IS. Recuperado de https://www.klaran.com/

Luxiona, (2021). Espacios limpios y seguros de Covid: Luminarias y dispositivos UV-C. Barcelona, España.: Grupo LUXIONA, S.L. Recuperado de: https://luxiona.com/es/noticias/41/clean-and-save-spaces-covid-uv-c-luminaires-and-devices/

McManamon, P. (2019). LiDAR Technologies and Systems. Recuperado de https://www.spiedigitallibrary.org/ebooks/PM/LiDAR-Technologies-and-Systems/eISBN-9781510625402/10.1117/3.2518254

Mostofi, N. (2015). *3D Indoor Mobile Mapping using Multi-Sensor Autonomous Robot* (Tesis doctoral). Universidad de Calgary, Calgary, Canada.

Ortega, L. & Silva, N. (2021). *Sistema de navegación autónoma en robot móvil tipo oruga para apoyo en tareas de siembra en campos caficultores* (Tesis de pregrado). Universidad autónoma de Bucaramanga, Bucaramanga, Colombia.

Ponce, P. (2010). *Inteligencia artificial con aplicaciones a la ingeniería*. Ciudad de México, México: Alfaomega.

Raspberry Pi. (2021). *Especificaciones técnicas de Raspberry Pi 4*. Cambridge, Reino Unido.: Raspberry Pi Press. Recuperado de https://www.raspberrypi.com

Rodríguez, M. (2019). *Diseño de metodología de trabajo para el escaneo con tecnología láser 3D, aplicada a la arquitectura patrimonial* (Tesis de pregrado). Universidad de la república, Montevideo, Uruguay.

ROS. (2021). *About ROS*. California, EU.: Element Public Relations. Recuperado de https://www.ros.org/news/

Sick. (2018). *Soluciones de localización basadas en LiDAR*. Baden-Wurtemberg, Alemania.: SICK AG. Recuperado de https://www.sick.com/

SKF. (2021). *Rodamientos rígidos de bolas*. Gotemburgo, Suecia.: AB SKF (publ.). Recuperado de https://www.skf.com/pe/products/rolling-bearings/ball-bearings/deep-groove-ball-bearings

Victron Energy. (2021). *Baterías de litio SuperPack de 12,8 V & 25,6 V*. Flevoland, Países Bajos.: CDNetworks. Recuperado de https://www.victronenergy.com.es/batteries/12,8v-lithium-superpack

YDLIDAR. (2021). *LiDARes de triangulación*. Shenzhen, China.: Shenzhen EAI Technology. Recuperado de https://www.ydLiDAR.com/LiDARs/Triangulation.html

Zhao, H., y Shibasaki,R. (2001). A robust method for registering ground-based laser range images of urban outdoor objects, *PE&RS, 67*(10), 1143-1153. Recuperado de http://www.poss.pku.edu.cn/Data/publications/pers2001.pdf

ANEXOS

Anexo 1: Ficha técnica de sensor LiDAR, marca YDLIDAR modelo G2.

YDLIDAR Make machines serve people more intelligently

1.3 Installation and Dimensions

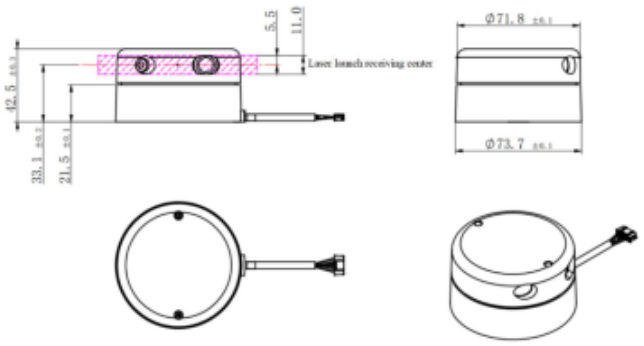


FIG 1 YDLIDAR G2 INSTALLATION&MECHANICAL SIZE

2 SPECIFICATIONS

2.1 Product Parameter

CHART 1 YDLIDAR G2 PRODUCT PARAMETER

Item	Min	Typical	Max	Unit	Remarks
Ranging frequency	/	5000	/	Hz	Ranging 5000 times per second
Motor frequency	5	7	12	Hz	Software control, Factory setting 7Hz
Ranging distance	0.12	/	12	m	80% Reflectivity
Fileld of view	/	0-360	/	°	/
Systematic Error	/	2	/	cm	Range≤1m
Statistical Error	/	2.0%	/	/	1m<Range ≤8m
Luminous intensity range	0	/	1023	/	laser intensity
Laser inclination	0.25	1	1.75	°	Laser pitch angle
Angle resolution	0.36 (Frequency @5Hz)	0.504 (Frequency @7Hz)	0.864 (Frequency @12Hz)	°	/

Note1: The range and relative accuracy in the above table are factory values,statistical error changes with the actual distance value;

Note2: In the above table, the relative error value represents the accuracy of the radar measurement, Statistical Error = (Ranging distance - actual distance)/ actual distance * 100%. LiDAR is a precision equipment, which needs to be protected during use. In the use scenarios of high temperature, high and low temperature or strong vibration, the parameter index of relative error will be relatively larger, and the error may exceed the than Typical value.

2.2 Electrical Parameter

CHART 2 YDLIDAR G2 ELECTRICAL PARAMETER

Item	Min	Typical	Max	Unit	Remarks
Supply voltage	4.8	5.0	5.2	V	Excessive voltage might damage the Lidar while low affect normal performance
Startup current	1000	/	/	mA	Instantaneous peak current at start-up
Sleeping current	/	/	50	mA	System sleep, motor stops
Working current	/	350	500	mA	System work, motor speed=7Hz

2.3 Interface Definition

G2 provides PH2.0-5P female connector to realize power and data communication function.

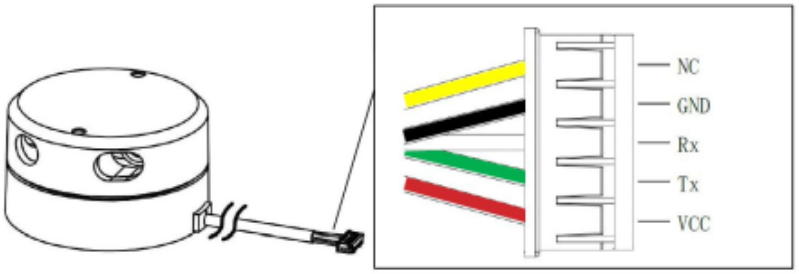


FIG 2 YDLIDAR G2 INTERFACES

CHART 3 YDLIDAR G2 INTERFACE DEFINITION

Pin	Type	Description	Defults	Range	Remarks
VCC	Power Supply	Positive	5V	4.8V-5.2V	/
Tx	Output	System serial port output	/	/	Data stream: Lidar→Peripherals
Rx	Input	System serial port Input	/	/	Data stream: Peripherals→Lidar
GND	Power Supply	Negative	0V	0V	/
NC	Reserve	Reserved pin	/	/	/

2.4 Data Communication

With a 3.3V level serial port (UART), users can connect the external system and the product through the physical interface. After that, you can obtain the real-time scannedpoint cloud data,device information as well as device status.The communication protocol of parameters are as follows：

CHART 4 YDLIDAR G2 SERIAL SPECIFICATION

Item	Min	Typical	Max	Unit	Remarks
Baud rate	/	230400	/	bps	8-bit data bit,1 stop bit,no parity
High Signal Level	2.4	3.3	3.5	V	/
Low signal Level	0	0.3	0.6	V	/

2.5 Optical Characteristic

G2 uses an infrared laser that meets FDA Class I eye safety standards. The laser and optical lens finish the transmission and reception of the laser signal to achieve high-frequency ranging while working. To ensure system ranging performance, please keep the laser and optical lens clean. The detailed optical parameters are as follows:

CHART 5 YDLIDAR G2 LASER OPTICAL PARAMETERS

Item	Min	Typical	Max	Unit	Remarks
Laser Wavelength	775	792	800	nm	Infrared band
Laser Power	/	3.5	6	mw	Average power
FDA	⚠ Class I				

2.6 Polar Coordinate System Definition

In order to facilitate secondary development, G2 internally defines a polar coordinate system. The polar coordinates of the system take the center of the rotating core of G2 as the pole, and the specified Angle is positive clockwise (top view). The zero Angle is located in the direction of the outlet of the G2 PH2.0-5P interface line. Due to individual differences, there is a deviation of +/-3°, as shown in the figure:

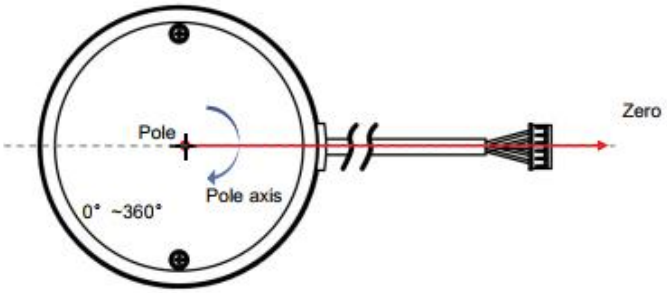


FIG 3 YDLIDAR G2 POLAR COORDINATE SYSTEM DEFINITION

Note1: When the Lidar is in production, the deviation of the assembly may cause the individual deviation of the radar zero angle of ±3°

Note2: When the Lidar is assembled on the machine, the deviation of the assembly may cause slight difference in the consistency of the zero position. For the convenience of use, we provide a zero calibration software, which can customize the zero direction and assemble to the machine. The Lidar is calibrated twice. For details, please refer to the User manual.

2.7 Others

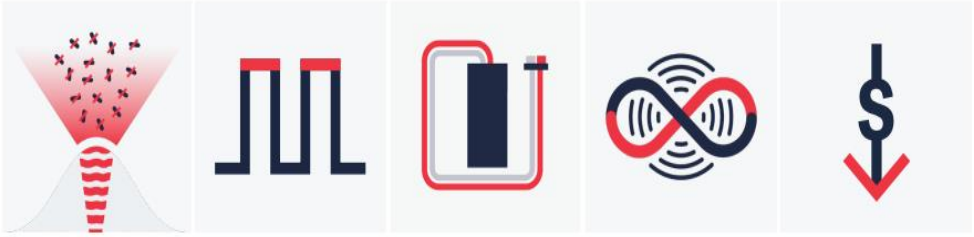
CHART 6 YDLIDAR G2 OTHERS

Item	Min	Typical	Max	Unit	Remarks
Operating temperature	0	20	50	℃	Long-term working in a high temperature environment will reduce the life span
Storage temperature	-10	/	60	℃	/
Lighting environment	0	2000	40000	Lux	For reference only
weight	/	185	/	g	N.W.

Anexo 2: Ficha técnica de LEDs UVC marca KLARAN modelo LE

Klaran® LE

Klaran LE (Light Engines) are plug and play modules for manufacturers to quickly add UVC LED disinfection to applications that require surface or air treatment. Klaran LE uses Klaran WD Series high-performance UVC LEDs which emit UV light in the ideal germicidal range (260 nm to 270 nm). Klaran LE includes a board mounted 12V LED driver so the engine can be conveniently integrated into a variety of applications.



260 NM TO 270 NM: MOST EFFECTIVE FOR DISINFECTION	ON DEMAND PERFORMANCE: GET LONGER LIFE DISINFECTION	INTEGRATED 12VDC DRIVER: PLUG AND PLAY UVC PURIFICATION	PREDICTABLE SERVICE LIFE: PERFORMANCE YOU CAN COUNT ON	HIGH PERFORMANCE ALUMINUM NITRIDE: LOW COST DISINFECTION
Emitting UV light at the peak germicidal wavelength, Klaran LE (Light Engine) can help prevent the spread of dangerous viral and bacterial pathogens such as MRSA and C. diff.	Compared to UV lamps, which are constantly operating and require annual replacement, UVC LEDs are on only when disinfection is needed. This conserves UVC LED lifetime and helps extend disinfection maintenance cycles to many years.	Klaran LE modules can be conveniently added to new or existing products. Manufacturers can leverage Klaran's international team of Application Engineers to explore the best options for product integration.	Robustly characterized, Klaran WD UVC LEDs provide confidence that your disinfection solution perform reliably for its design lifetime.	Klaran WD Series LEDs are produced from Crystal IS' proprietary Aluminum Nitride substrates which support high yields of UVC LEDs for cost effective solutions.

Product Nomenclature

Part Number	Peak Wavelength	LED Arrangement
LE-12V-9U-HC – Engineering Sample	260 nm - 270 nm	Qty 9 Klaran WD 60 mW LEDs (KL265-50U-SM-WD)

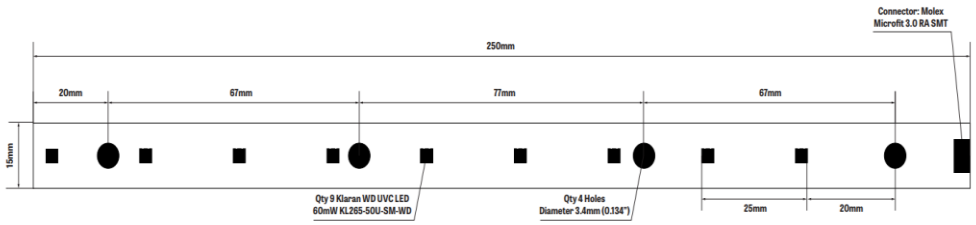
Electrical Characteristics

Characteristic	Unit	Min	Type	Max	Note
Power Input Voltage (VCC)	V	11.4	12	12.6	Constant DC Voltage
Power Consumption (LED ON)	W	-	45	-	

Physical Characteristics

Characteristic	Unit	Min	Type	Max	Note
Length	mm		250		
Width	mm		15		
Ambient Temperature Range (LED ON)	°C	5	–	50	
Ambient Temperature Range (LED OFF)	°C	5	–	85	
Relative Humidity	%	40	–	75	

Mechanical Dimensions



Anexo 3: Características técnicas de la batería Litio SuperPack

Litio SuperPack	12,8/20	12,8/60	12,8/100	12,8/100 Alta corriente	12,8/200	25,6/50
Química	LiFePO4					
Tensión nominal	12,8 V					25,6 V
Capacidad nominal a 25°C	20 Ah	60 Ah	100 Ah		200 Ah	50 Ah
Capacidad nominal a 0°C	16 Ah	48 Ah	80 Ah		160 Ah	40 Ah
Energía nominal a 25°C	256 Wh	768 Wh	1280 Wh		2560 Wh	1280 Wh
Cantidad de ciclos al 80 % DoD y 25°C	2500 ciclos					
CARGA Y DESCARGA						
Máxima corriente de descarga continua [1]	30 A	30 A	50 A	100 A	70 A	50 A
Corriente máxima de descarga (10 segundos)	80 A	80 A	100 A	150 A	100 A	100 A
Tensión de final de descarga	10 V					20 V
Tensión de carga, absorción [2]	14,2 V - 14,4 V					28,4 V – 28,8 V
Tensión de carga, flotación	13,5 V					27 V
Máxima corriente de carga continua	15 A	30 A	50 A	100 A	70 A	50 A
CONDICIONES DE TRABAJO						
Configuración en paralelo	Sí, sin limitación					
Configuración en serie	No					
Temperatura de trabajo	Descarga: -20 °C a +50 °C Carga: Entre +0 °C y +45 °C [3]					
Temperatura de almacenamiento	Entre -40°C y +65°C					
Periodo máximo de almacenamiento estando completamente cargada	1 año ≤ 25 °C 3 meses ≤ 40 °C					
Humedad (sin condensación)	Máx. 95 %					
Clase de protección	IP 43					
OTROS						
Conexión eléctrica (inserciones roscadas)	M5	M6	M8		M8	M8
Dimensiones (Al x An x Pr) mm	167 x 181 x 77	213 x 229 x 138	220 x 330 x 172		208 x 520 x 269	220 x 330 x 172
Peso	3,5 kg	9,5 kg	14 kg		21 kg	14 kg

1 La batería podría desconectarse en caso de que se conectara una carga con una gran capacidad de entrada, como un inversor. Sin embargo, la batería intentará conectarse de nuevo pasados unos 10 segundos.
2 Es mejor que el periodo de absorción no supere las 4 horas. Un periodo de absorción más largo puede reducir ligeramente la vida útil.
3 Número de serie HQ2040 y más recientes: la carga se bloqueará automáticamente cuando la temperatura de la celda caiga por debajo de 0±3°C. Volverá a aceptar la carga cuando suba por encima de 3±3 °C. La descarga se bloqueará automáticamente cuando la temperatura de la celda caiga

Anexo 4: Programación de Acondicionamiento para comunicación entre etapa Lógica de Potencia y Lógica SLAM

```
#if (ARDUINO >= 100)
#include <Arduino.h>
#else
#include <WProgram.h>
#endif
#include <ros.h> // Incluimos la librería de rosserial que permite la comunicación entre microcontroladores
#include <geometry_msgs/Twist.h>
// Definimos los pines de conexión a utilizar para la comunicación con nuestros motores:
const int derecha_pwm_pin = 3;
const int derecha_dir_pin = 2;
const int izquierda_pwm_pin = 5;
const int izquierda_dir_pin = 4;
const bool izquierda_fwd = true;
const bool derecha_fwd = false;

// Velocidad estándar.
const int vel_estandar = 201;
ros::NodeHandle  nh;
void Avanzar(const size_t speed) {
  digitalWrite(derecha_dir_pin, derecha_fwd);
  digitalWrite(izquierda_dir_pin, izquierda_fwd);
  analogWrite(derecha_pwm_pin, speed);
  analogWrite(izquierda_pwm_pin, speed);
}
void MoverDetener() {
  digitalWrite(derecha_dir_pin, derecha_fwd);
  digitalWrite(izquierda_dir_pin, izquierda_fwd);
  analogWrite(derecha_pwm_pin, 0);
  analogWrite(izquierda_pwm_pin, 0);
```

```
}
void cmd_vel_cb(const geometry_msgs::Twist & msg) {
  // Se lee el mensaje para determinar comportamiento.
  // Solamente nos es útil el lineal x y el rotacional z.
  const float x = msg.linear.x;
  const float z_rotation = msg.angular.z;

  // Decidimos el estado de motor que necesitamos, de acuerdo al comando de ROS.
  if (x > 0 && z_rotation == 0) {
    Avanzar(vel_estandar);
  }
  else {
    MoverDetener();
  }
}
ros::Subscriber<geometry_msgs::Twist> sub("cmd_vel", cmd_vel_cb);
void setup() {
  pinMode(derecha_pwm_pin, OUTPUT);    // define el pin 13 como salida PWM
  pinMode(derecha_dir_pin, OUTPUT);
  pinMode(izquierda_pwm_pin, OUTPUT);
  pinMode(izquierda_dir_pin, OUTPUT);
  // Dfine los valores iniciales para la dirección. Ambos en dirección frontal.
  digitalWrite(derecha_dir_pin, derecha_fwd);
  digitalWrite(izquierda_dir_pin, izquierda_fwd);
  nh.initNode();
  nh.subscribe(sub);
}
void loop() {
  nh.spinOnce();
  delay(1);
}
```

Anexo 5: Programación de Sistema de Luces LED UV-C y PIR

```
const int PIN_TO_SENSOR = 6;  // pin conectado a la salida del sensor PIR
int pinStateCurrent   = LOW;  // estado actual del PIN
int pinStatePrevious  = LOW;  // estado anterior del PIN

void setup()
{  Serial.begin(9600);           // inicializamos el serial
  // ponemos el pin del Arduino en modo de entrada para que lea el valor del pin OUTPUT del sensor
  pinMode(PIN_TO_SENSOR, INPUT);
}
void loop()
{

 // Configuramos un pin de Arduino al modo de entrada digital
 pinMode(PIN_TO_SENSOR, INPUT);

 //Leemos el estado del pin OUTPUT del sensor
 pinStateCurrent = digitalRead(PIN_TO_SENSOR);

 //Detect motion start (pin's state change from HIGH to LOW)
 pinStatePrevious = pinStateCurrent; // guardamos el estado anterior
 pinStateCurrent = digitalRead(PIN_TO_SENSOR);    // leemos el nuevo estado

 // pin state change: HIGH -> LOW
 if (pinStatePrevious == HIGH && pinStateCurrent == LOW)
 {
   Serial.println("Movimiento Detectado!");
   // podemos colocar una alarma para advertir a los humanos que se acerquen
 }
```

```
 else
 // pin state change: LOW -> HIGH
 if (pinStatePrevious == LOW && pinStateCurrent == HIGH)
 {
   Serial.println("Movimiento Detenido");
   // la alarma se detendrá al no detectarse la presencia humana
 }
}
```

Anexo 6: Programación de Control por Lógica Difusa aplicada en motores DC

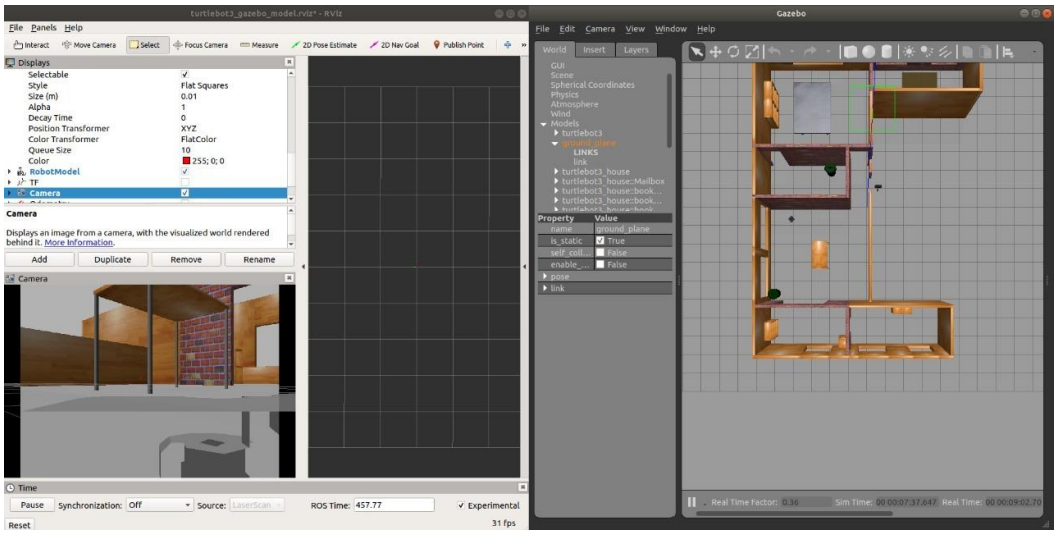
```
#include <Fuzzy.h>

// Instantiating a Fuzzy object
Fuzzy *fuzzy = new Fuzzy();
*****************************************

#if (ARDUINO >= 100)
#include <Arduino.h>
#else
#include <WProgram.h>
#endif

#include <ros.h>
#include <geometry_msgs/Twist.h>

*************************************

void setup()
{
  FuzzyInput * distancia = nuevo  FuzzyInput ( 1 );
  // Creación devariable linguistica pequeño
  FuzzySet * pequeño = nuevo  FuzzySet ( 0 , 20 , 20 , 40 );
  // Incluyendo el FuzzySet en FuzzyInput
  distancia-> addFuzzySet (pequeño);
 // Creación devariable linguistica seguro
  FuzzySet * seguro = nuevo  FuzzySet ( 30 , 50 , 50 , 70 );
  // Incluyendo el FuzzySet en FuzzyInput
  distancia-> addFuzzySet (seguro);
  // Creación devariable linguistica grande
  FuzzySet * grande = nuevo  FuzzySet ( 60 , 80 , 80 , 80 );
  // Incluyendo el FuzzySet en FuzzyInput
  distancia-> addFuzzySet (grande);
```

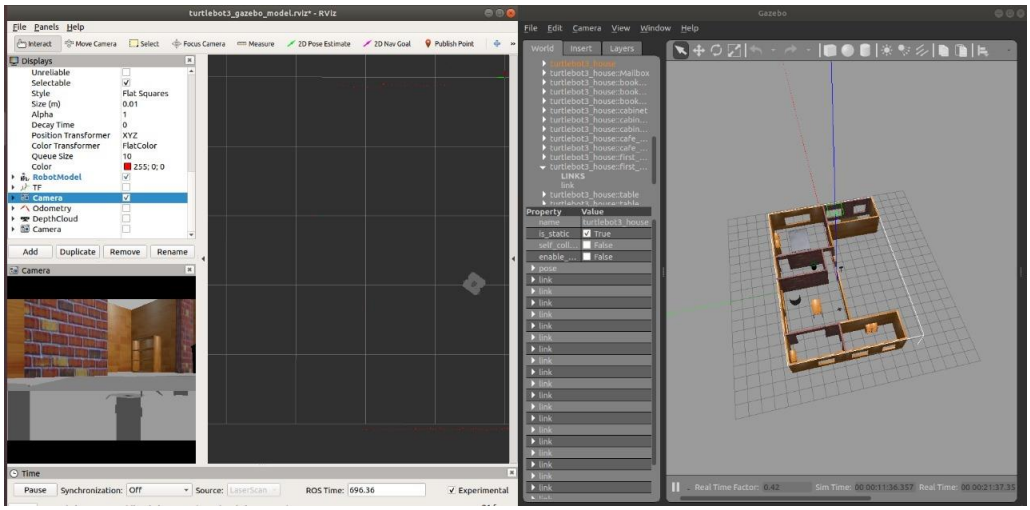
```
  // Incluyendo FuzzyInput en Fuzzy
  fuzzy-> addFuzzyInput (distancia);
**************************************************************************
**************************************************************

// Creación de instancias de objetos FuzzyOutput
   FuzzyOutput * velocidad = nuevo  FuzzyOutput ( 1 );
// Creación de variable linguistica lento
   FuzzySet * lento = nuevo  FuzzySet ( 0 , 10 , 10 , 20 );
   // Incluyendo el FuzzySet en FuzzyOutput
  velocidad-> addFuzzySet (lento);
// Creación de variable linguistica promedio
  FuzzySet * promedio = nuevo  FuzzySet ( 10 , 20 , 30 , 40 );
  // Incluyendo el FuzzySet en FuzzyOutput
  velocidad-> addFuzzySet (promedio);
// Creación de variable linguistica rápido
  FuzzySet * rápido = nuevo  FuzzySet ( 30 , 40 , 40 , 50 );
  // Incluyendo el FuzzySet en FuzzyOutput
  velocidad-> addFuzzySet (rápido);
  // Incluyendo FuzzyOutput en Fuzzy
  fuzzy-> addFuzzyOutput (velocidad);


**************************************************************************
*******


 // Construyendo FuzzyRule "SI distancia = pequeña ENTONCES velocidad = lenta"
  // Creación de instancias de objetos FuzzyRuleAntecedent
  FuzzyRuleAntecedent * ifDistanceSmall = new  FuzzyRuleAntecedent ();
  // Creando un FuzzyRuleAntecedent con un solo FuzzySet
  ifDistanceSmall-> joinSingle (pequeño);
  // Creación de instancias de objetos FuzzyRuleConsequent
  FuzzyRuleConsequent * thenSpeedSlow = new  FuzzyRuleConsequent ();
```

```
// Incluyendo un FuzzySet a este FuzzyRuleConsequent
thenSpeedSlow-> addOutput (lento);
// Creación de instancias de objetos FuzzyRule
FuzzyRule * fuzzyRule01 = nueva  FuzzyRule ( 1 , ifDistanceSmall, thenSpeedSlow);
// Incluyendo FuzzyRule en Fuzzy
fuzzy-> addFuzzyRule (fuzzyRule01);

// Construyendo FuzzyRule "SI distancia = segura ENTONCES velocidad = promedio"
// Creación de instancias de objetos FuzzyRuleAntecedent
FuzzyRuleAntecedent * ifDistanceSafe = new  FuzzyRuleAntecedent ();
// Creando un FuzzyRuleAntecedent con un solo FuzzySet
ifDistanceSafe-> joinSingle (seguro);
// Creación de instancias de objetos FuzzyRuleConsequent
FuzzyRuleConsequent * thenSpeedAverage = new  FuzzyRuleConsequent ();
// Incluyendo un FuzzySet a este FuzzyRuleConsequent
thenSpeedAverage-> addOutput (promedio);
// Creación de instancias de objetos FuzzyRule
FuzzyRule * fuzzyRule02 = nueva  FuzzyRule ( 2 , ifDistanceSafe, thenSpeedAverage);
// Incluyendo FuzzyRule en Fuzzy
fuzzy-> addFuzzyRule (fuzzyRule02);

// Construyendo FuzzyRule "SI distancia = grande ENTONCES velocidad = alta"
// Creación de instancias de objetos FuzzyRuleAntecedent
FuzzyRuleAntecedent * ifDistanceBig = new  FuzzyRuleAntecedent ();
// Creando un FuzzyRuleAntecedent con un solo FuzzySet
ifDistanceBig-> joinSingle (grande);
// Creación de instancias de objetos FuzzyRuleConsequent
FuzzyRuleConsequent * thenSpeedFast = new  FuzzyRuleConsequent ();
// Incluyendo un FuzzySet a este FuzzyRuleConsequent
thenSpeedFast-> addOutput (rápido);
// Creación de instancias de objetos FuzzyRule
FuzzyRule * fuzzyRule03 = nueva  FuzzyRule ( 3 , ifDistanceBig, luego SpeedFast);
// Incluyendo FuzzyRule en Fuzzy
fuzzy-> addFuzzyRule (fuzzyRule03);
```

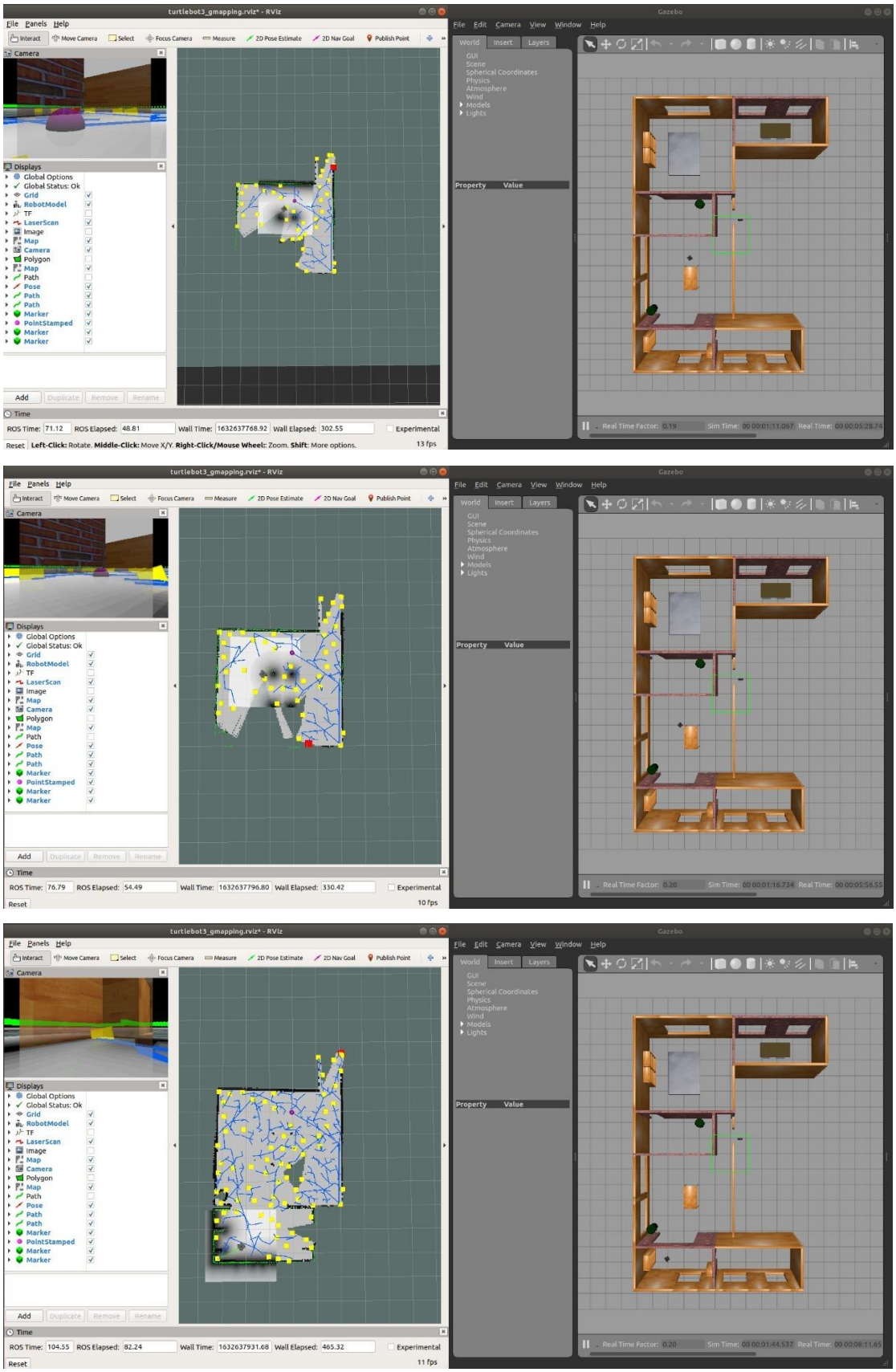
```
}
 void loop ()
{
  // Obteniendo un valor aleatorio
  int entrada = aleatorio ( 0 , 80 );
  // Imprimiendo algo
  De serie. println ( " \ n \ n \ n Entrada: " );
  De serie. print ( " \ t \ t \ t Distancia: " );
  De serie. println (entrada);
  // Establecer el valor aleatorio como entrada
  difuso-> setInput ( 1 , entrada);
  // Ejecutando la Fuzzificación
  difuso-> fuzzify ();
  // Ejecutando la Defuzzification
  float output = difusa-> defuzzify ( 1 );
  // Imprimiendo algo
  De serie. println ( " Resultado: " );
  De serie. print ( " \ t \ t \ t Velocidad: " );
  De serie. println (salida);
  // espera 12 segundos
  retraso ( 12000 );
}
```

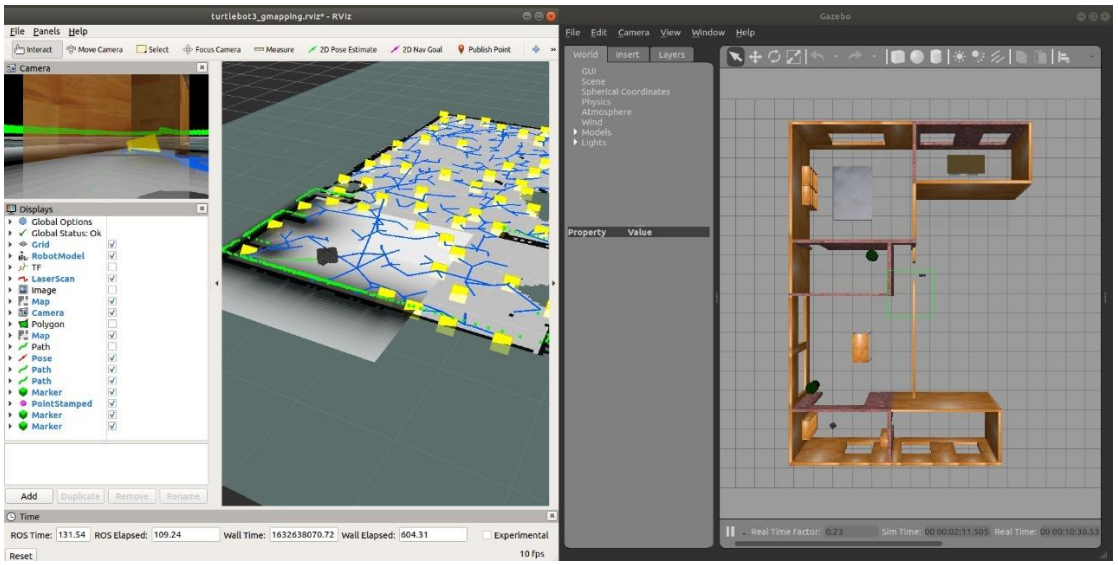
Anexo 7: Visualización del resultado de los sensores en el software de simulación Gazebo.

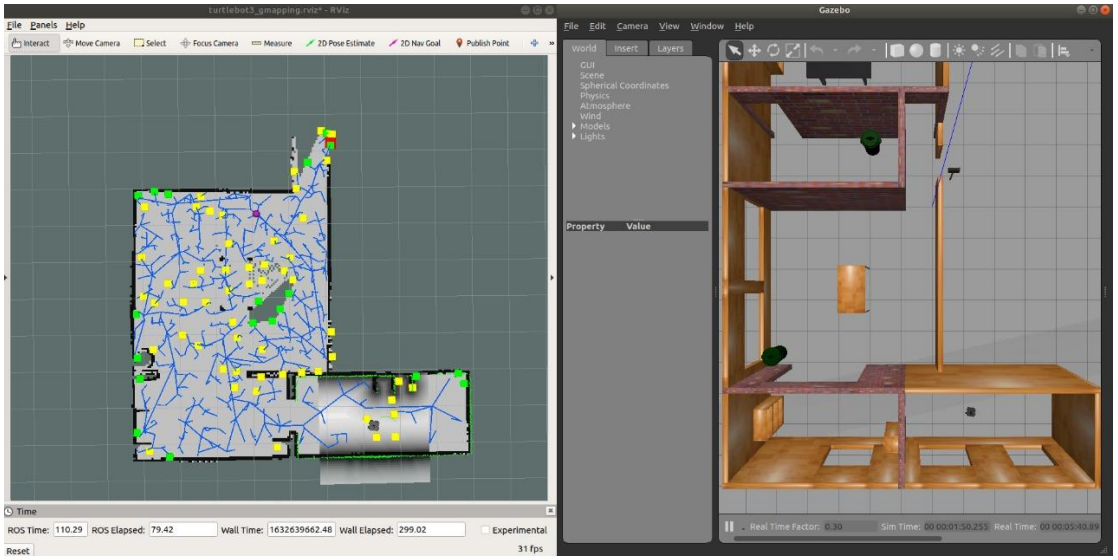




Anexo 8: Escaneo LiDAR completo del robot alrededor de 4 puntos.




turtlebot3_gmapping.rviz* - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Camera
Displays
Global Options
Global Status: Ok
Grid
RobotModel
TF
LaserScan
Image
Map
Camera
Polygon
Map
Path
Pose
Path
Path
Marker
PointStamped
Marker
Marker
Add Duplicate Remove Rename
Time
ROS Time: 131.54 ROS Elapsed: 109.24 Wall Time: 1632638070.72 Wall Elapsed: 604.31 Experimental
Reset
10 fps
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Real Time Factor: 0.23 Sim Time: 00 00:02:11.505 Real Time: 00 00:10:30.53


turtlebot3_gmapping.rviz* - RViz
File Panels Help
Interact Move Camera Select Focus Camera Measure 2D Pose Estimate 2D Nav Goal Publish Point
Time
ROS Time: 110.29 ROS Elapsed: 79.42 Wall Time: 1632639662.48 Wall Elapsed: 299.02 Experimental
Reset
31 fps
Gazebo
File Edit Camera View Window Help
World Insert Layers
GUI
Scene
Spherical Coordinates
Physics
Atmosphere
Wind
Models
Lights
Property Value
Real Time Factor: 0.30 Sim Time: 00 00:01:50.255 Real Time: 00 00:05:40.89