UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE CUENCA

CARRERA DE INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo
a la obtención del título de
Ingeniero Electrónico

PROYECTO TÉCNICO CON ENFOQUE INVESTIGATIVO:

"LEVANTAMIENTO DE MAPAS DE AMBIENTES PARA LA NAVEGACIÓN AUTÓNOMA DE ROBOTS MÓVILES"

AUTORES:

JAVIER ESTÉBAN JARA JIMBO

JUAN PABLO URUCHIMA CALLE

TUTOR:

ING. RENÉ SEVERO ÁVILA CAMPOVERDE

CUENCA - ECUADOR

2021

CESIÓN DE DERECHOS DE AUTOR

Nosotros, Javier Estéban Jara Jimbo con documento de identificación N° 0104228341 y Juan Pablo Uruchima Calle con documento de identificación N° 0104582168, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de titulación: **"LEVANTAMIENTO DE MAPAS DE AMBIENTES PARA LA NAVEGACIÓN AUTÓNOMA DE ROBOTS MÓVILES",** mismo que ha sido desarrollado para optar por el título de: *Ingeniero Electrónico*, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, noviembre de 2021.

Javier Estéban Jara Jimbo

C.I. 0104228341

Juan Pablo Uruchima Calle

C.I. 0104582168

CERTIFICACIÓN

Yo declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **"LEVANTAMIENTO DE MAPAS DE AMBIENTES PARA LA NAVEGACIÓN AUTÓNOMA DE ROBOTS MÓVILES"**, realizado por Javier Estéban Jara Jimbo y Juan Pablo Uruchima Calle, obteniendo el *Proyecto Técnico con enfoque investigativo* que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana.

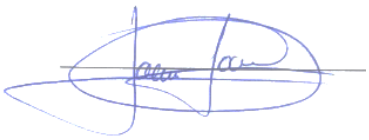
Cuenca, noviembre de 2021.

Ing. René Severo Ávila Campoverde
C.I. 0102257920

DECLARATORIA DE RESPONSABILIDAD

Nosotros, Javier Estéban Jara Jimbo con documento de identificación N° 0104228341 y Juan Pablo Uruchima Calle con documento de identificación N° 0104582168, autores del trabajo de titulación: **"LEVANTAMIENTO DE MAPAS DE AMBIENTES PARA LA NAVEGACIÓN AUTÓNOMA DE ROBOTS MÓVILES",** certificamos que el total contenido del *Proyecto Técnico con enfoque investigativo*, es de nuestra exclusiva responsabilidad y autoría.

Cuenca, noviembre de 2021.



Javier Estéban Jara Jimbo

C.I. 0104228341

Juan Pablo Uruchima Calle

C.I. 0104582168

ÍNDICE

INDICE DE FIGURAS

INDICE DE TABLAS

AGRADECIMIENTOS

Agradecemos a Dios por guiarnos para lograr esta meta, a la Universidad Politécnica Salesiana y a todos los docentes que nos ayudaron a formarnos como profesionales.

Un agradecimiento especial a nuestros tutores, Ing. René Ávila e Ing. Julio Montesdeoca quienes, con su apoyo, conocimientos y aptitudes, nos han permitido desarrollar este tema y consolidar los conocimientos adquiridos durante toda la carrera.

.

DEDICATORIA

Este proyecto va dedicado principalmente a Dios, por brindarme fuerza y sabiduría para recorrer este camino y cumplir todas mis metas.

A mis padres por darme todo el apoyo, por su amor incondicional y por siempre creer en mí, a mis hermanos y a toda mi familia por haber aportado un granito de arena y ayudarme a hacer esto posible.

Javier Estéban Jara Jimbo

El presente proyecto está dedicado principalmente a Dios por guiarme y darme la sabiduría necesaria, ser el apoyo y fortaleza en aquellos momentos difíciles.

A mis padres por el amor, apoyo y sacrificio durante toda la carrera universitaria ya que gracias a ellos he logrado llegar hasta aquí para convertirme en lo que soy y porque han sido el pilar fundamental en el desarrollo de este proyecto.

Agradezco de manera especial a mi Hermano Patricio por su ejemplo, palabras de apoyo, motivación desde el día uno y confianza en mí.

Extiendo mi agradecimiento a mis tutores de tesis el Ing. Julio Montesdeoca y el Ing. René Ávila por el aporte brindado durante todo este proceso, quienes con su dirección conocimiento, enseñanza y colaboración permitieron el desarrollo de este proyecto; así como también a la Universidad politécnica Salesiana por ofrecernos las herramientas necesarias para la preparación de nuestra profesión.

Juan Pablo Uruchima Calle

GLOSARIO

Autónomo, que tiene autonomía. También, aplicado al mundo del trabajo, significa que trabaja por cuenta propia.

Cinemática, es la rama de la mecánica que describe el movimiento de los objetos sólidos sin considerar las causas que lo originan (las fuerzas) y se limita, principalmente, al estudio de la trayectoria en función del tiempo.

Framework, o marco de trabajo es un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

Lemniscata, es una cualquiera de varias curvas con figura en forma de ocho (8) o de símbolo de infinito (∞).

Odometría, es el estudio de la estimación de la posición de vehículos con ruedas durante la navegación.

PLA, es un polímero o bioplástico constituido por elementos similares al ácido láctico, con propiedades semejantes a las del tereftalato de polietileno (PET).

ROS, es un framework para el desarrollo de software para robots que provee la funcionalidad de un sistema operativo en un clúster heterogéneo.

SLAM, del inglés Simultaneous Localization and Mapping, o en español: Localización y Mapeado Simultáneo

RESUMEN

En el siguiente documento se presenta el diseño y la construcción de un robot móvil de tracción diferencial para la exploración de ambientes y levantamiento de mapas.
La aplicación que se le dará al robot será la de transportar mensajería de un lugar a otro.

El robot móvil contiene 4 ruedas independientes, la cuales son accionadas por motores sin escobillas que poseen sensores de efecto hall los cuales facilitan el control de bajo nivel. Para cada rueda se tiene implementado un controlador PID para el control de velocidad. De la misma manera, mediante un sensor Lidar y una cámara Intel Real Sense se implementan algoritmos para la percepción del ambiente. La implementación de los algoritmos del robot se realiza en el framework ROS (Robot Operating System), el cual está instalado en una raspberry PI 4. Así también, el robot puede ser teleoperado mediante un dispositivo de control externo. Por otra parte, el diseño mecánico está constituido por una estructura que contiene todo el hardware del robot móvil, los cuales están construidos en acero e impreso con tecnología 3D.

INTRODUCCIÓN

El mundo de la robótica actualmente experimenta un desarrollo acelerado, por impulso de los avances de computación y electrónica, así como también de softwares y comunicaciones.

La aplicación de la robótica es multivariable y se relaciona a la vez con tareas nocivas o riesgosas para el ser humano, dentro de ellas aquellas tareas que incluyen el manejo de material explosivo; exploración subterránea, tareas de mantenimiento en reactores nucleares, transporte y almacenamiento de materiales, etc.

El robot autónomo debe ser capaz de evitar obstáculos y tomar decisiones propias en el desplazamiento de la superficie a la cual se desplaza. El comportamiento del robot está marcado por las señales y los datos que se adquieren en el entorno durante el desplazamiento del mismo, interactuando la acción y percepción, y así formar un sistema autónomo, bajo determinada conducta y posteriormente describir la inteligencia de los sistemas y exponer la robótica como una tecnología que requiere tanto del movimiento como de la inteligencia para poder cumplir o realizar una actividad determinada.

Para poder realizar el diseño y construcción del prototipo del robot móvil y para el transporte de mensajería, es necesario que se inicie con la recopilación de la información para el diseño estructural, la correcta selección de los motores y el hardware electrónico que controlará los mismos.

Para dotarle de inteligencia artificial al robot y éste pueda cumplir con el objetivo de moverse de un lugar a otro reconociendo ambientes y obstáculos, es necesario trabajar con sistemas de adquisición de datos que permitan realizar un mapeamiento adecuado del ambiente y a su vez reconocer o identificar posibles obstáculos, para esto se cuenta con el apoyo de algunos softwares (ROS, Rtabmap, Python, Arduino) en los que se elaborarán los diferentes algoritmos para cumplir con el objetivo trazado.

Una vez que el prototipo esté construido, es necesario establecer y ejecutar una serie de pruebas para verificar el correcto funcionamiento del mismo sometido a diferentes escenarios y en base a los resultados obtenidos, ir

haciendo las calibraciones y ajustes necesarios hasta llegar a un funcionamiento con el menor error posible.

ANTECEDENTES DEL PROBLEMA DE ESTUDIO

La navegación autónoma de robots en entornos no controlados es un desafío ya requiere un conjunto de subsistemas para trabajar. Requiere construir un mapa del entorno, localizar el robot en ese mapa, hacer un plan de movimiento de acuerdo con el mapa, ejecutar ese plan con un controlador y otras tareas; todo al mismo tiempo. Por otro lado, el problema de la navegación autónoma es muy importante para el futuro de la robótica.

Hay muchas aplicaciones que requieren que se resuelva este problema, como entrega de paquetes, limpieza, agricultura, vigilancia, búsqueda y rescate, construcción y transporte. Todas estas aplicaciones se producen en entornos no controlados. En este proyecto de titulación, intentamos resolver algunos problemas específicos relacionados con la navegación autónoma en entornos no controlados.

JUSTIFICACIÓN (IMPORTANCIA Y ALCANCES)

La sociedad moderna está haciendo de la robótica algo cotidiano en el día a día, en tareas como: entrega de paquetes, ordenamiento de inventario, entre las más importantes, muchos de ellos enfocados a lo que actualmente se conoce como industria 4.0.

Los robots autónomos han cobrado un renovado interés y sus aplicaciones en diversos campos son muy variadas, lo que ha hecho que se desarrollen dos tipos de robots: unos estáticos como los manipuladores, que son muy utilizados en la industria manufacturera y un segundo grupo de robots móviles aplicados generalmente al transporte, clasificación de objetos o paquetes y exploración en lugares en donde la presencia del ser humano no se puede dar.

La navegación autónoma de los robots móviles juega un papel importante en el desplazamiento del robot, ya que se requiere la aplicación de conocimientos de diferentes áreas como: modelado de sistemas, control no lineal, levantamiento de mapas, dinámica de los robots, diseño estructural, diseño electrónico, electrónica de potencia, microcontroladores y microprocesadores, electrónica básica, etc.

Por todo lo mencionado en los párrafos anteriores consideramos que estamos en la capacidad de desarrollar con tecnología propia del país robots móviles que cumplan con el objetivo de transportar documentación o cualquier otro tipo de carga de un lugar a otro.

OBJETIVOS

OBJETIVO GENERAL

Diseñar e implementar un robot autónomo, proporcionado de componentes necesarios para realizar el mapeo de una habitación o recinto, sin necesidad de ser guiado ni corregido por acciones externas.

OBJETIVOS ESPECÍFICOS

1. Estudiar el control del robot móvil de tracción diferencial
2. Diseñar el modelo estructural del robot móvil de tracción diferencial.
3. Aplicar el modelo cinemático extendido para implementar tareas básicas de control.
4. Implementar los algoritmos necesarios para el levantamiento de mapas desde la Secretaría de la carrera hasta la Dirección de Carrera de Electrónica y Automatización.

CAPÍTULO 1

Estado del Arte de la Robótica Móvil

1.1 Introducción a la Robótica Móvil
- 1.1.1 Introducción General
- 1.1.2 Robots Móviles
- 1.1.3 Robots Móviles y Trayectorias
  - 1.1.3.1 Trayectorias

1.2 Modelado Clásico de un Robot Móvil de Tracción Diferencial.
- 1.2.1 Modelado Clásico
  - 1.2.1.1 Parámetros Involucrados en el Modelo.
- 1.2.2 Modelado Cinemático Extendido.
- 1.2.3 Similitudes y Diferencias de los Modelos Matemáticos para el robot móvil de tracción diferencial.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA O ESTADO DEL ARTE

1.1 Introducción a la Robótica Móvil

1.1.1 Introducción General

El universo de la robótica tiene un crecimiento importante debido al avance tecnológico en cuanto a mecánica, computación, hardware y software. Los robots están revolucionando los procedimientos que se emplean en los campos de: industria, ganadera, minería, medicina, etc.

En la industria, se ha experimentado un importante desarrollo en los robots manipuladores, desde la década de los 70 en donde se comenzó a conocer de este tipo de robots, cobrando importancia de los 80s y 90s; y una mayor aplicación desde el año 2011 con la concepción del desarrollo de la producción inteligente, basada en la comunicación y el control telemático.

Para los robots manipuladores y para la robótica en general, es importante las bases fundamentales para que los robots trabajen de manera auténtica, las cuales son: el modelado cinemático y dinámico, el control, planificación y el reconocimiento del entorno.

1.1.2 Robots Móviles

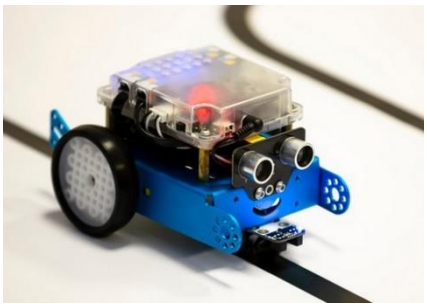
Los robots móviles están formados principalmente por una estructura (armazón) mecánica que contiene el sistema de locomoción que es capaz de explorar un determinado espacio o ambiente de trabajo con un importante nivel de autonomía. [1] [2]

Las aplicaciones son diversas, y se relacionan en sí con tareas que son nocivas o peligrosas para el ser humano; ejemplo de ellos son: traslado de materiales, mantenimiento en plantas nucleares, manipulación de materiales explosivos, exploración de zonas en incendios, etc. [1] [2]

La autonomía de un robot móvil se basa en la capacidad para poder modelar, planificar, percibir y actuar en el cumplimiento de los objetivos planteados, con una intervención parcial o sin la presencia de una persona, debido a que el robot móvil sería capaz de desenvolverse en ambientes estructurados o no estructurados, total o parcialmente conocidos. [1] [2]

Dentro de los robots móviles, tenemos los robots autónomos y los robots guiados, tal como se muestra en la figura 1a y 1b.

(a)



(b)

Figura 1. Robots Móviles: (a). Robot Móvil Autónomo, (b). Robot Móvil Guiado. [2] [3]

1.1.3 Robots Móviles y Trayectorias.

La operación o el funcionamiento de los robots móviles en ambientes no establecidos deben afrontar significativos problemas en cuanto a posición e identificación de objetos. En efecto, dichos problemas, se relacionan con la crítica tarea del traslado del robot desde un punto A hasta un punto B. [1] [5]

El robot móvil autónomo está caracterizado por su interacción lógica entre la percepción y acción, modelando el comportamiento del robot permitiéndolo cumplir las tareas programadas sobre ambientes no estructurados. Su grado de independencia se define de la manera que tiene el robot para manejarse sobre su entorno y procesar dicha información en órdenes, las cuales aplicadas al sistema de actuadores puedan trasladarse de un sitio a otro, tal como se muestra en la figura 2. [1] [5]

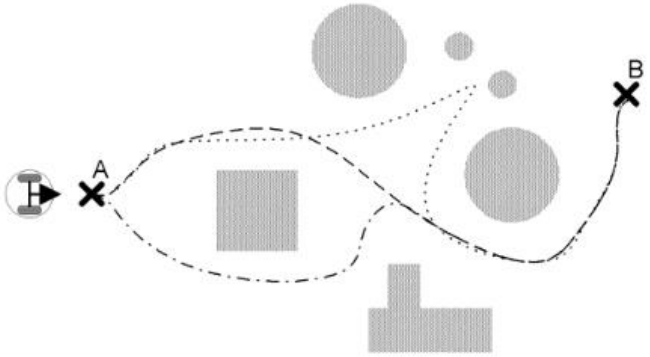


Figura 2. Trayectorias posibles que sigue el robot móvil. [1]

1.1.3.1 Trayectorias.

Las características que distinguen a un robot móvil de cualquier otro tipo de vehículo son:

Percepción: La capacidad del robot móvil para interactuar con el entorno que explora mediante su sistema sensorial; sintetiza dicha información que proporciona los sensores con el fin de generar mapas del entorno. [1] [5]

Razonamiento: La capacidad del robot móvil para procesar y actuar ante tareas definidas en base al estado del robot y el ambiente en el que se encuentra; planifica trayectorias globales y se adapta al entorno cuando se presentan obstáculos inesperados. [1] [5]

Dentro de la característica del razonamiento, tenemos:

- **Generador Global de Trayectorias (GGT)**: La información usada en este nivel jerárquico se genera en base a criterios predefinidos e información proporcionada por los sistemas sensoriales para el conocimiento previo del entorno. [1] [5]

- **Generador Local de Trayectorias (GLT)**: Es el encargado de simular la función de operador del robot móvil. Se conecta directamente con el sistema sensorial proporcionando la toma de decisiones y entregando valores para el registro local del sistema de locomoción, tal como se muestra en la figura 3. [1] [5]

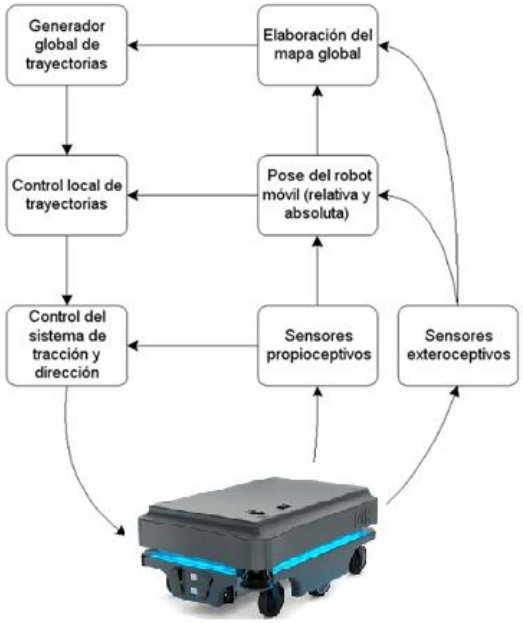


Figura 3. Esquema generalizado del control de un robot móvil. [1]

1.2 Modelado Cinemático de un Robot Móvil de Tracción Diferencial.

Para el estudio y análisis de un Robot Móvil de Tracción Diferencial, se aplica el Modelado Clásico o el Modelado Clásico Extendido.

Los Robots Móviles se construyen en base a diferentes tipos de mecanismos, los cuales se distinguen por el sistema de tracción. Las más comunes son las que incorporan un sistema de tracción diferencial, incorporando motores de forma independiente para cada una de las ruedas situadas en un mismo eje de acción; y para estabilidad, se coloca por lo general, ruedas locas (caster crazy wheel) a la plataforma. [4]

1.2.1 Modelado Clásico

La cinemática, centraliza el estudio del movimiento del robot móvil en base a la geometría del mismo. En el análisis se pueden obtener los siguientes parámetros: un modelo matemático a partir del diseño del controlador, el comportamiento cinemático a partir de una o varias simulaciones y la obtención de las ecuaciones odométricas. [4]

1.2.1.1 Parámetros Involucrados en el Modelo.

En la figura 4 se observan los parámetros para el modelado del robot móvil de tracción diferencial de dos ruedas.

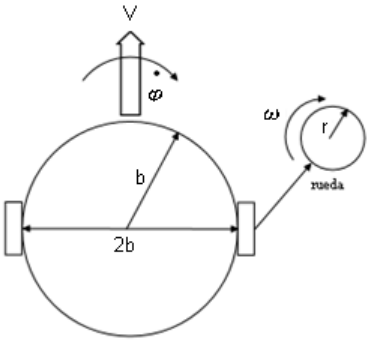


Figura 4. Robót Móvil de dos Ruedas. [4]

Los parámetros físicos del robot son los siguientes:

$2b$: $Longitud\ del\ eje\ o\ distancia\ entre\ ruedas.$
r: $radio\ de\ cada\ rueda.$

Las variables que interactúan con el movimiento del robot móvil son:

V: $Velocidad\ lineal\ del\ robot.$
$\dot{\varphi}$: $Velocidad\ angular\ del\ robot.$
$\omega_1, \omega_2 = \dot{\theta}_1, \dot{\theta}_2$: $Velocidad\ angular\ de\ la\ rueda\ derecha\ e\ izquierda.$

Entre las variables dinámicas se puede obtener la posición absoluta del robot en el espacio, limitadas en sí por las coordenadas bidimensionales del centro de masa (x, y) y el ángulo formado entre la dirección del robot y el eje cartesiano x, denominado φ, tal como se muestra en la figura 5. [4]

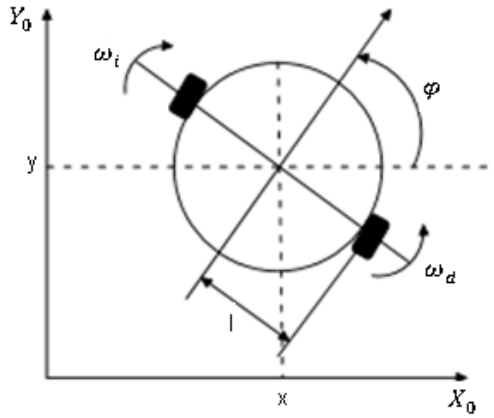


Figura 5. Localización del robot móvil en el plano cartesiano. [4].

Las ecuaciones cinemáticas relacionan la velocidad de las ruedas con las variables de posición del robot (x, y, φ).

Teniendo al robot como un cuerpo rígido, la velocidad lineal del centro de masa se consigue con el promedio de las velocidades lineales de cada una de sus ruedas; las cuales se obtienen con el producto de la velocidad angular y el radio de la rueda; la velocidad del centro de masa se especifica mediante la ecuación (1). [4].

$$V = \frac{r\left(\dot{\theta}_1 + \dot{\theta}_2\right)}{2} \qquad (1)$$

El ángulo de giro se obtiene a partir de las relaciones geométricas del movimiento de cada rueda del robot móvil, tal como se muestra en la figura 6.

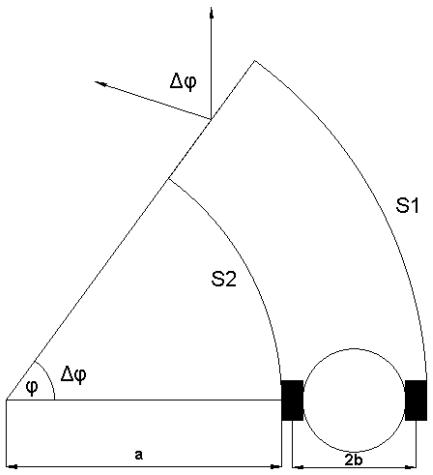


Figura 6. Relación Geométrica para el Ángulo de Giro. [4]

En la figura 6, el ángulo de giro del robot móvil es igual al del arco mantenido por la trayectoria [4]. El ángulo de dirección (φ) se incrementa en sentido contrario a las manecillas del reloj haciendo que el ángulo de dirección aumente Δφ. La rueda izquierda inscribe un arco de radio "a", por lo que la distancia recorrida por esa rueda está dada por la ecuación (2).

$$S_2 = r\Delta\theta_2 = a\Delta\varphi \qquad (2)$$

La rueda derecha al encontrarse más lejos del centro recorre una distancia mayor en el mismo instante de tiempo, definida por la ecuación (3).

$$S_1 = r\Delta\theta_1 = (a + 2b)\Delta\varphi \qquad (3)$$

Definiendo $\Delta\varphi$, como la diferencia del recorrido de ambas ruedas, tenemos:

$$\Delta\varphi = S_1 - S_2 \qquad (4)$$

En la ecuación (5), se muestra la relación entre la velocidad de giro y la velocidad de cada una de las ruedas.

$$\dot{\varphi} = \lim_{\Delta t \to 0} \frac{\Delta\varphi}{\Delta t} = \frac{r\left(\dot{\theta}_1 - \dot{\theta}_2\right)}{2b} \qquad (5)$$

$\Delta\varphi$: $Diferencia\ del\ recorrido\ de\ ambas\ ruedas.$

Δt: $tiempo\ transcurrido\ del\ recorrido\ realizado\ por\ las\ ruedas.$

Las coordenadas de la posición del centro de masa se obtienen con la descomposición de la velocidad lineal del robot en velocidades que se asocien a cada eje del plano XY, definido en la ecuación (6).

Las coordenadas de posición se obtienen a partir de la posición del centro

$$\begin{aligned} \dot{x} &= V cos(\varphi) \\ \dot{y} &= V sen(\varphi) \end{aligned} \qquad (6)$$

1.2.2 Modelado Cinemático Extendido

"El modelo cinemático extendido de un robot móvil de tracción diferencial, surge como una idea de facilitar el diseño de controladores basados en cinemática inversa"[1].

[1] J. C. Montesdeoca, M. C. P. Santos, M. Monllor and D. Herrera, "Trajectory tracking controller for differential-drive mobile robots," 2017 XVII Workshop on Information Processing and Control (RPIC), Mar del Plata, Argentina, 2017, pp. 1-4, doi: 10.23919/RPIC.2017.8211621.

En la figura 7, se muestra el modelo, en donde (x^{ω}, y^{ω}) corresponden al marco de referencia global fijo, mientras que (x, y) determinan las coordenadas del centro de masa del robot móvil [7].

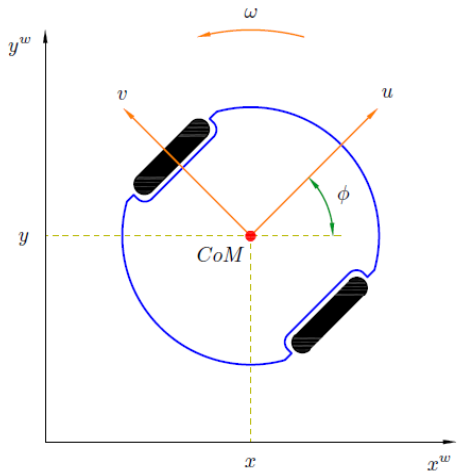


Figura 7. Disposición en el Plano Cartesiano del Robot Móvil de Tracción Diferencial [7].

Los parámetros relacionados con el movimiento del robot móvil son:

$\emptyset$: *orientación del robot respecto al eje* x^{ω}

u: *velocidad lineal del robot*

v: *velocidad de deslizamiento lateral*

ω: *velocidad de rotación del cuerpo del robot*

Coordenadas generalizadas del robot están dadas por:

$$q = [x \quad y \quad \emptyset]^T \qquad (7)$$

El modelo cinemático extendido para un robot móvil de tracción diferencial se define mediante la ecuación (8).

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\emptyset}\end{bmatrix} = \begin{bmatrix}cos\emptyset & -sin\emptyset & 0\\ sin\emptyset & cos\emptyset & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}u\\ v\\ \omega\end{bmatrix} \quad (8)$$

Tomando la ecuación (8), definimos que:

$$\dot{q} = A(q)u \qquad (9)$$

Donde:

$\dot{q}: corresponde\ a\ los\ estados\ generalizados$

$u: vector\ de\ acciones\ de\ control$

$A(q): matriz\ cinemática$

Al aplicar la cinemática inversa el modelo en (9), se obtiene otro sistema de rango completo, que se define por la ecuación (10).

$$\begin{bmatrix}u\\ v\\ \omega\end{bmatrix} = \begin{bmatrix}cos\emptyset & sin\emptyset & 0\\ -sin\emptyset & cos\emptyset & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\emptyset}\end{bmatrix} \quad (10)$$

Tomando como base la ecuación (10), se propone un controlador basado en cinemática inversa, en donde su representación está dada por la ecuación (11).

$$u_c = A^{-1}(q)\eta \quad (11)$$

Donde:

$u_c: vector\ seudo - entradas\ del\ control$

$\eta: vector\ de\ la\ ley\ de\ control\ auxiliar$

Por lo tanto, el controlador que se define se puede interpretar de forma matricial o como ecuaciones lineales:

$$\begin{bmatrix} u_c \\ v \\ \omega_c \end{bmatrix} = \begin{bmatrix} cos\emptyset & sin\emptyset & 0 \\ -sin\emptyset & cos\emptyset & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \eta_x \\ \eta_y \\ \eta_\emptyset \end{bmatrix} \quad (12)$$

$$u_c = \eta_x cos\emptyset + \eta_y sin\emptyset, \quad (13)$$

$$v = -\eta_x sin\emptyset + \eta_y cos\emptyset, \quad (14)$$

$$\omega_c = \eta_\emptyset, \quad (15)$$

1.2.3 Similitudes y Diferencias de los Modelos Matemáticos para el Robot Móvil de Tracción Diferencial.

Una vez definido y explicado cada uno de los modelos, se procede a comparar cada uno de ellos y definir el modelo a usar en el presente proyecto.

Similitudes	Diferencias
• Ambos modelos se basan en la cinemática del robot. • Se obtienen ecuaciones lineales en base a las velocidades del robot. • Ambos usan relaciones geométricas.	• Modelo Clásico se usan estructuras complejas para el controlador, mientras que el Clásico Extendido usa directamente la cinemática inversa como estrategia de control.

Tabla 1. Similitudes y Diferencias de los Modelos Matemáticos para el diseño del Robot Móvil de Tracción Diferencial.

El modelo matemático definido para el diseño e implementación del robot móvil en este proyecto es el Modelo Cinemático Extendido, ya que dicho modelo nos permite proponer un controlador sencillo en el cual se tiene en cuenta directamente los objetivos de control de la tarea a cumplir.

CAPÍTULO 2

Diseño y Construcción del Robot Móvil

2.1 Diseño Estructural del Robot.
- 2.1.1 Base del Robot.
- 2.1.2 Carcasa del Robot.

2.2 Proceso de Construcción del Robot.

2.3 Diseño e Implementación de la Electrónica del Robot.
- 2.3.1 Descripción del Hardware del Robot.
- 2.3.2 Descripción del Software del Robot.
  - 2.3.2.1 Implementación del Software Embebido.
  - 2.3.2.2 Implementación del framework ROS.

2.4 Diseño PCB.

CAPÍTULO 2. MARCO METODOLÓGICO

El diseño y la fabricación del robot móvil se conforma por dos etapas. La primera etapa se presenta el diseño y construcción estructural del robot y en la segunda etapa se muestra el diseño electrónico usado en la implementación del robot.

Para el desarrollo de la estructura del robot se considera la funcionalidad y la aplicabilidad que va a influenciar en el desempeño del robot móvil. Así, por ejemplo:

- El robot debe mantener una resistencia mecánica correcta para soportar tanto el peso en vacío como con carga
- La estructura del debe ser de componentes ligeros para que su movilidad sea eficiente y facilite el trabajo deseado
- El material de las ruedas debe ser de caucho debido a que con esto ayuda a tener mayor adherencia y la trayectoria del robot sea la correcta
- La dimensión de la estructura dependerá de los elementos que lleve abordo el robot tanto en el interior como en el exterior
- Los motores deben tener un torque elevado para que rompa la inercia del robot móvil

2.1 Diseño Estructural del Robot.

El sistema estructural debe ser capaz de equilibrar fuerzas a las que se le va someter.

En el diseño de la estructura, se necesitó la ayuda de un Ingeniero Mecánico; el diseño se realizó en base a las cargas y las condiciones de entorno a las que el robot estará sometido.

Se diseñó una plataforma compacta la para que la estructura pueda sostener tanto a los motores brushless como a la carcasa o cuerpo del robot.

2.1.1 Base del Robot.

La base del robot está diseñada de manera que tenga un correcto balance y pueda soportar las cargas que va a transportar, para ello se utilizó tubo estructural de ¾ plg. El diseño de la estructura se realizó en el software Inventor en donde se pudo obtener las respectivas simulaciones de pruebas de resistencia.

Los cálculos se realizaron en base a las cargas que el robot está sometido y a la disposición de la estructura.

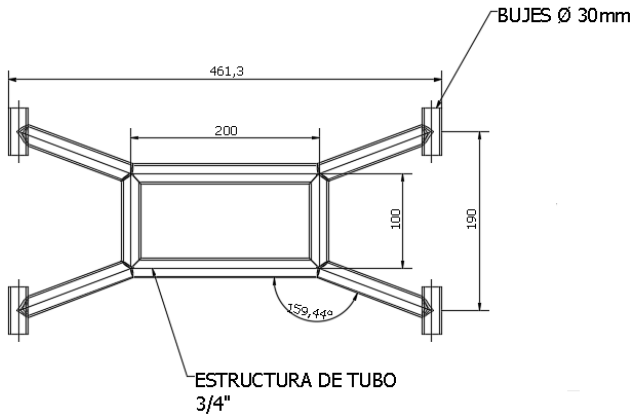


Figura 8. Dimensiones de la base del Robot.

Figura 9. Base del robot integrado los motores Brushless.

Pruebas de Resistencia Mecánica de la base del Robot Móvil.

Las pruebas mecánicas se dividen en dos partes. La primera parte, corresponde a las pruebas de resistencia mediante el software de Inventor, y la segunda parte son pruebas mecánicas realizadas a la estructura del robot móvil de tracción diferencial.

En las pruebas de resistencia se realizan comprobaciones en cuanto a la estabilidad, magnitud de desplazamientos para que estas variables no afecten al comportamiento de la estructura.

Para la realización de las pruebas mecánicas de la estructura (base) del robot, se usa el análisis de elementos finitos, el cual se ejecutó con la ayuda del Software de Inventor.

Dentro del software, con la estructura diseñada y previamente calculada, se hace uso de la herramienta “Stress Analysis” la cual nos permite hacer el análisis de elementos finitos. Una vez realizada la selección del tipo de material, se procede a la selección de los soportes en donde se indica al programa en dónde van ubicados los puntos de apoyo fijo, en nuestro caso, los puntos serían los 4 bujes en donde se alojan las ruedas. Para ello, se hace uso del comando “Fixed Constraint”, tal como se muestra en la figura 10.

Finalizado la selección de los puntos de apoyo fijo, se le indica al programa, en dónde va ubicada la fuerza gravitacional, aquí es necesario ubicar un punto de la estructura en donde el software pueda determinar el centroide del elemento.

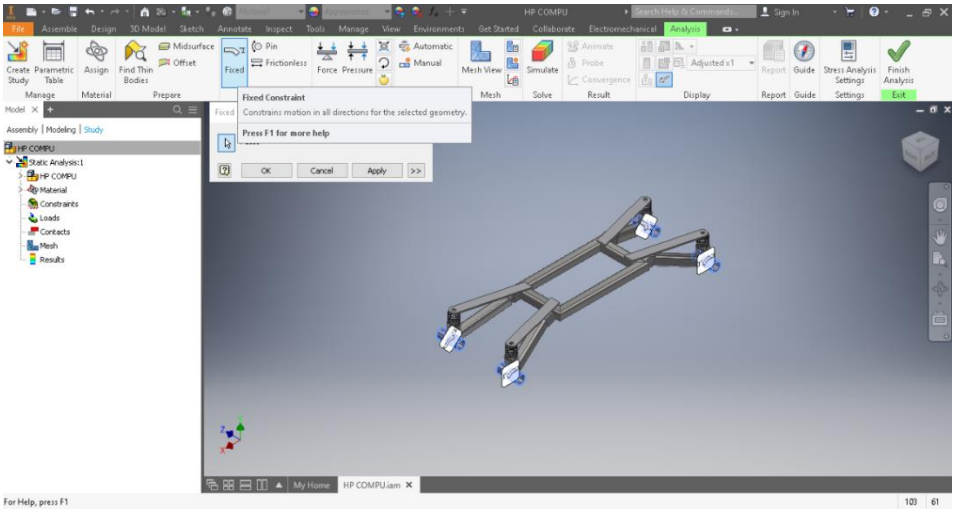


Figura 10. Selección de los puntos de apoyo fijo, con la ayuda del comando Fixed Constraint.

Una vez ubicado el centro de gravedad, se realiza la selección de los puntos o el punto en donde van ubicadas las cargas, en este caso, como a la base estructural se le acopla una estructura plástica de tipo PLA en donde se ubica todos los elementos electrónicos.

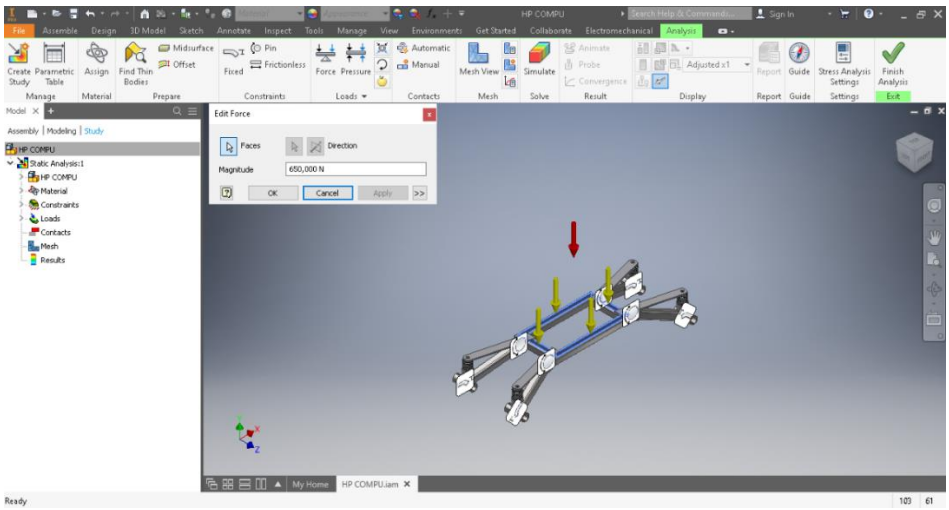


Figura 11. Verifiación de la resistencia de la estructura con 600N en los puntos de apoyo.

Por lo tanto, se demuestra por software que la estructura soporta un peso máximo aproximado de 61.18kgf, por lo que se realizan las pruebas experimentales para que el sustento teórico quede debidamente justificado.

2.1.2 Carcasa del Robot

Dentro de la carcasa se alberga toda la parte electrónica de control del robot móvil. La carcasa fue desarrollada usando una impresora 3D, además esta es desmontable de tal manera que sea desmontable, esto con el fin de realizar mantenimientos y futuras mejoras al sistema.

Al igual que la base del robot, la carcasa se diseñó en el software Inventor, integrando toda la parte de control del robot móvil, como se visualizan en las figuras 12 y 13.

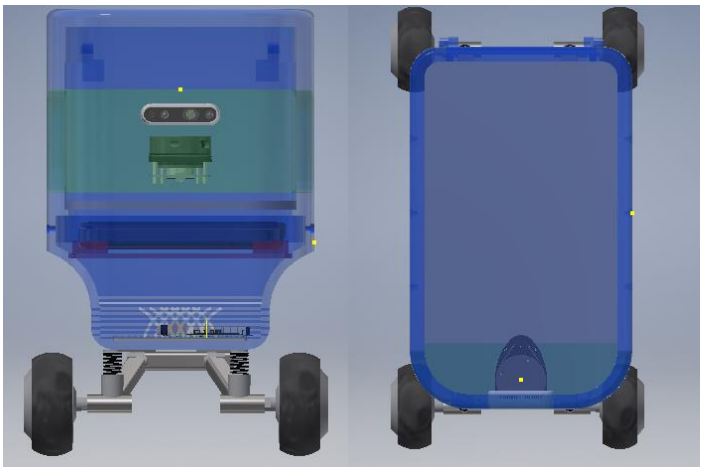


Figura 12. Diseño completo de la Carcasa del Robot Móvil

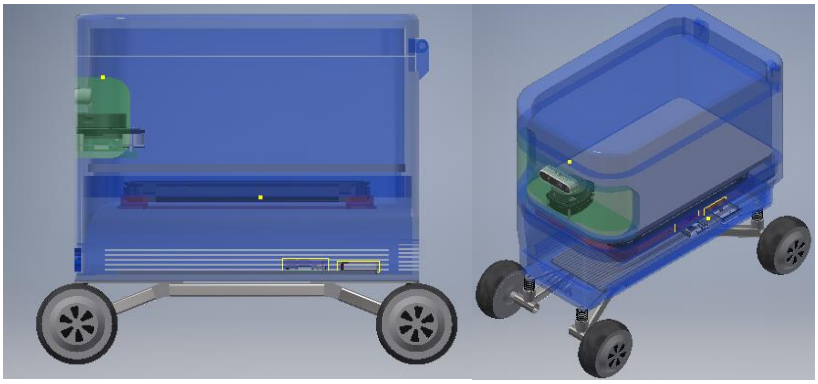


Figura 13. Carcasa del Robot de Tracción diferencial completa.

2.2 Proceso de Construcción del Robot Móvil

El proceso de construcción se divide en tres etapas:

- En la primera parte se realiza la construcción de la estructura mecánica.
- En la segunda etapa se ensambla el cuerpo del robot y se adecua las tarjetas electrónicas.
- En la tercera etapa se realizó la integración de la parte electrónica con los sensores y actuadores del robot.

A continuación, se describe las etapas antes indicadas:

Se visualiza en la figura 14 la base del robot junto con los actuadores (motores brushless). En apéndice 3 se puede ver el proceso de construcción a detalle.

Figura 14. Construcción de la base estrcutural del robot móvil.

La construcción de la carcasa se realizó mediante el uso de la impresión 3D, tal como se puede observar en la figura 15. La elección del material en base a las características mencionadas en el apartado 2.1.2.

Figura 15. Impresión 3D de la carcasa del robot móvil.

2.3 Diseño e Implementación de la Electrónica del Robot.

Se divide en dos partes:

- La primera en donde detallamos todo el hardware del robot móvil.
- La segunda que trata sobre el software implementado en el robot móvil.

2.3.1 Descripción del Hardware del Robot.

El hardware del robot es basado en un sistema de tracción conformado de cuatro ruedas, las mismas que son controladas independientemente por un driver, y en conjunto comandado por la tarjeta madre.

Se visualiza en la figura 16, el esquema electrónico general del hardware del robot móvil de tracción diferencial.

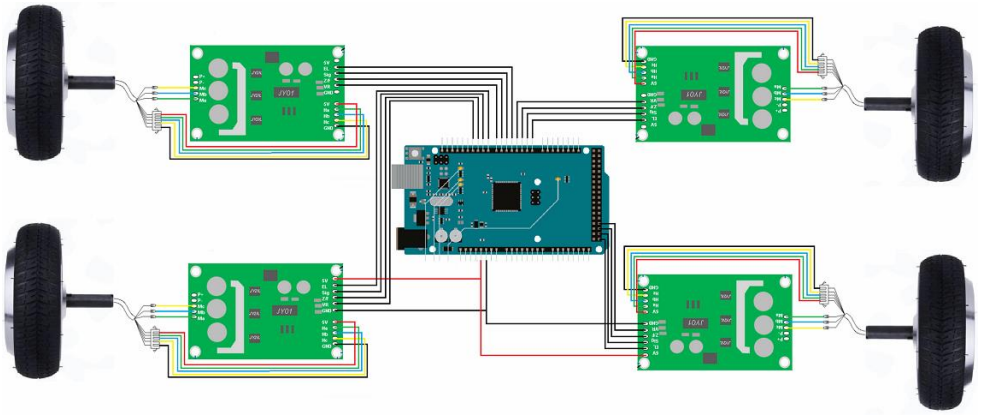


Figura 16. Esquema Electrónico General del Robot Móvil.

- **Placa Electrónica Principal**

La placa principal está integrada por un microcontrolador Arduino Mega, la cual se encarga de controlar a los sensores (encoders) y actuadores (motores).

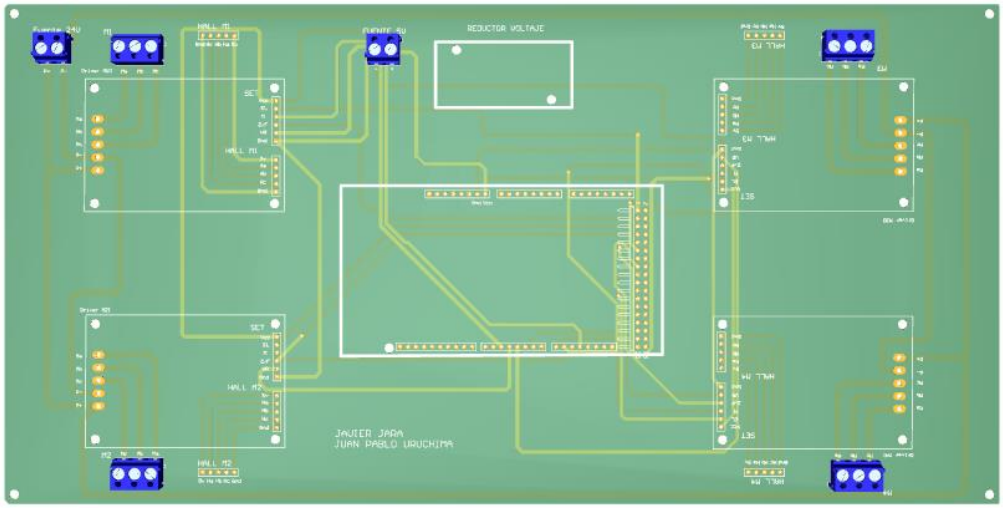


Figura 17. Placa Electrónica Principal.

En la figura 18 se observa el montaje de la placa electrónica la cual comanda a los actuadores del robot móvil y además toma las lecturas de los sensores.

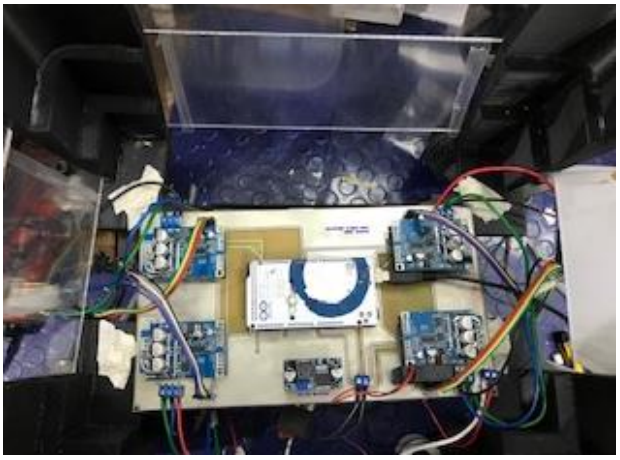


Figura 18. Montaje de la tarjeta electrónica en la carcasa del robot móvil.

- **Actuadores del Robot.**

Para el robot móvil diseñado, se escoge motores trifásicos sin escobillas (BLDC), los cuales mediante un driver pueden ser controlados en velocidad

a través de una señal PWM. Estos motores tienen integrado un sensor de efecto Hall que nos permite la obtención de la velocidad de rotación de los motores [8].

Para el robot móvil se escogieron los motores mencionados por las siguientes características:

- Facilidad de control de velocidad
- Alta eficiencia.
- Elevado par motor con respecto a otros motores de corriente continua.

Figura 19. Motor Brushless BDLC [9] .

- **Drivers**

El módulo JYQD v6.3 sirve para controlar los motores del robot móvil. Maneja una potencia de hasta 250W, control de velocidad mediante una señal PWM, además posee una señal de retroalimentación mediante pulsos emitido por efecto hall desde el motor BDLC.

Figura 20. Módulo JYQD, driver motor brushless [10].

Descripción de los pines del módulo JYQD v6.3:

- Posee un voltaje de salida interno de la placa controladora de 5V.
- EL, este puerto se activa al conectarse a 5Vpermitiendo el giro de las ruedas del robot.
- Z/F son puertos de control de dirección.
- VR es el puerto de control de velocidad.
- Los puertos de alimentación al motor trifásico son: MA, MB y MC.
- La placa se alimenta con los pines VCC, GND [11].

2.3.2 Descripción del Software del Robot.

La implementación del software se desarrolló usando el framework ROS (Robot Operating System), tal como se muestra en la figura 21.

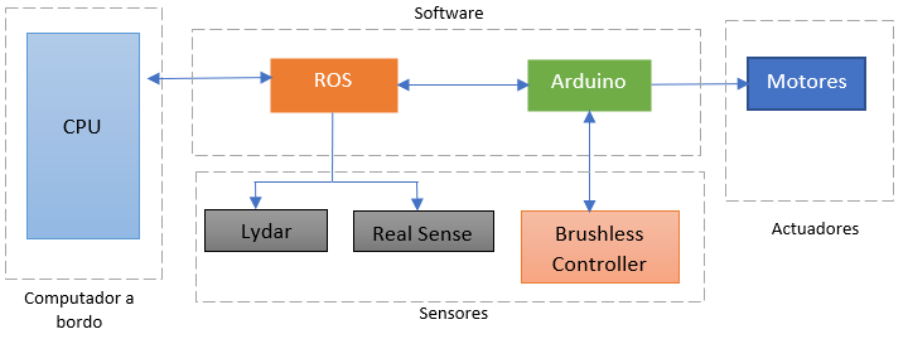


Figura 21. Esquema de Control del Robot Móvil.

El software embebido está conformado de tal manera que se pueda obtener datos de sensores y actuadores para activar o desactivar los puertos que manejan los algoritmos de bajo nivel.

El framework ROS por otra parte, establece la comunicación con los diferentes dispositivos conectados a la raspberry Pi 4.

2.3.2.1 Implementación del Software Embebido.

El software embebido se basa en el control de los motores BLDC mediante un controlador PID; el cual está programado en una tarjeta Arduino, que también permite la lectura de los sensores de retroalimentación de velocidad.

A continuación, se describe el modelado matemático del motor y el controlador PID aplicado.

Modelo Matemático del Motor

El modelado que se plantea de los motores BLDC se visualiza en la figura 22.

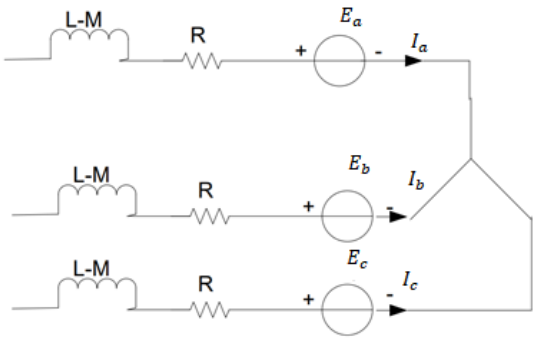


Figura 22. Diagrama del circuito de devanado del estator [8].

Las ecuaciones del circuito del motor pueden describirse mediante las siguientes ecuaciones:

$$Va = I_a R + L\frac{di_a}{dt} + E_a \qquad (16)$$

$$Vb = I_b R + L\frac{di_b}{dt} + E_b \qquad (17)$$

$$Vc = I_c R + L\frac{di_c}{dt} + E_c \qquad (18)$$

Por lo tanto, la funcion de tranferencia se obtiene utilzando la relación del radio y velocidad angular con la fuente de voltaje:

$$G(s) = \frac{\omega_m}{V_s} = \frac{\frac{1}{K_e}}{\tau_m \tau_e s^2 + \tau_m \cdot s + 1} \qquad (19)$$

$$\tau_m = \frac{RJ}{K_e K_t} \qquad (20)$$

$$\tau_e = \frac{L}{3xR} \qquad (21)$$

$$K_e = \frac{3R_\emptyset J}{K_t \tau_m} \qquad (22)$$

Las variables relacionadas con la ecuación son:

$\tau_m : variable\ mecanica\ (constante\ en\ el\ tiempo)$

$\tau_{e,} : par\ de\ endtrada\ del\ devanado$

$R = R_\emptyset\ es\ la\ resitencia\ \ de\ fase\ a\ fase$

$J\ es\ la\ incercia\ del\ motor$

$L\ es\ la\ inductacia\ de\ armadura$

$K_t\ es\ la\ constante\ par$

$P\ es\ le\ numero\ de\ polos$

$B\ es\ el\ coeficiente\ calculado\ a\ partir\ del\ momento\ de\ incercia$

Los parámetros del motor que se utilizaron son los mostrados en la Tabla 2.

No	Parámetro	Valor
1	Resistencia de armadura (Ra)	0.94Ω
2	Inductancia (La)	$1.19x10^{-3}H$
3	Momento de inercia (J)	$5.1x10^{-3}Kgm^2$
4	Constante Par (Kt)	4.17Nm/A
5	Constante de tiempo (mecánica) (Tm)	0.00491s

Tabla 2. Parámetros del Motor BLDC.

Estos valores se reemplazan en las ecuaciones 19, 20, 21 y 22; obteniéndose:

$$\tau_e = \frac{L}{3xR} = \frac{1.19x10^{-3}}{3x0.94} = 422\text{x}10^{-6}$$

$$K_e = \frac{3R_\emptyset J}{K_t\,\tau_m} = \frac{3x0.94x5.1x10^{-6}}{0.426x0.00491} = 6.87\text{x}10^{-3}V\cdot s/rad$$

$$G(s) = \frac{\omega_m}{V_s} = \frac{\frac{1}{6.87\text{x}10^{-3}}}{0.00491\,x\,422\,x10^{-6}s^2 + 0.00491.\text{s} + 1}$$

Dando como resultado:

$$G(s) = \frac{145.5}{2.072\,x10^{-6}s^2 + 0.00491.\text{s} + 1} \quad (23)$$

Después de obtención de la función de transferencia se modela la misma mediante simulink tal como se visualiza en la figura 23.

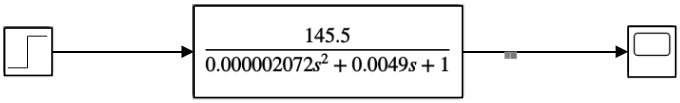


Figura 23. Diagrama de Bloques de la Planta

Se simula con una respuesta al escalón para conocer como responde la planta mostrado en en la figura 24.

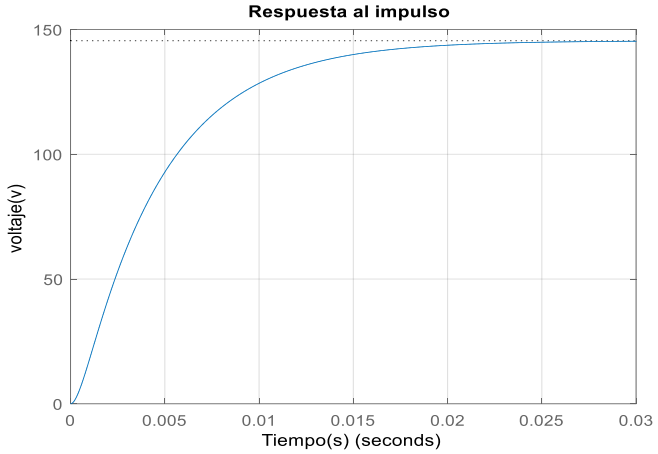


Figura 24. Respuesta al Escalón.

Diseño de PID

Para el diseño del algoritmo que se va utilizar para el control, se requiere conocer el comportamiento de la planta que se va a controlar, a partir de la figura 24 se obtiene el modelo del motor BLDC [8].

Para la planta antes mostrada se aplica un controlador digital PID de velocidad, como se puede visualizar en la figura 25.

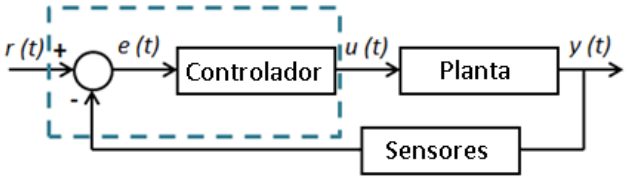


Figura 25.Diagrama del control de velocidad para el motor BLDC.

Para la sintonización del controlador PID se utilizó el método de Ziegler Nichols, el cual permite la definicion de ganancias tanto en proporcional, integrativa y derivativa segun la respuesta del sistema. Este método es adaptativo a los sistemas que son estables en lazo abierto y tienen un tiempo de retardo desde se recibe la señal de control hasta que se comienza a actuar en el sistema [12].

Para determinar la respuesta al escalón que se tiene de la planta, se retira el controlador PID y se sustituye por una señal escalón que se aplica en la entrada del sistema.

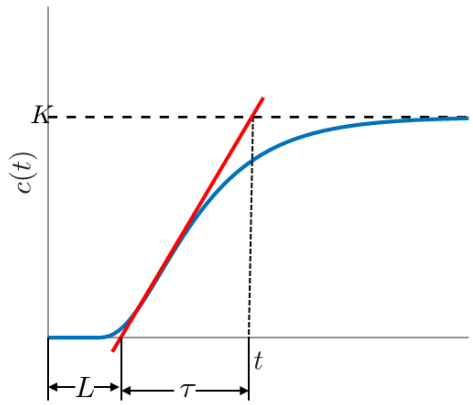


Figura 26. Sintonización de parámetros según Ziegler –Nicholls [12].

Esta respuesta en lazo abierto se caracteriza por dos parámetros, el tiempo de retardo (L) y la constante de tiempo (T) como se ve en la figura 26. Estos parametros se hallan dibujando las tangentes en la respuesta del sistema y

en los puntos de inflexión y a si eliminando los ejes tanto vertiales como horizontales, los parámetros del controlador se visualizan en la Tabla 3 [12].

TIPO DE PID	$\boldsymbol{Kp}$	$\boldsymbol{TI = \frac{Kp}{Ki}}$	$\boldsymbol{TD = \frac{KD}{Kp}}$
P	$\frac{T}{L}$	∞	0
PI	$0.9x\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2x\frac{T}{L}$	$2xL$	$0.5xL$

Tabla 3. Parámetros del método Ziegler -Nicholls [12].

Tomando en cuenta estos parámetros se procede a simular en el software Simulink para poder evaluar los mismos, como se puede ver en la figura 27.

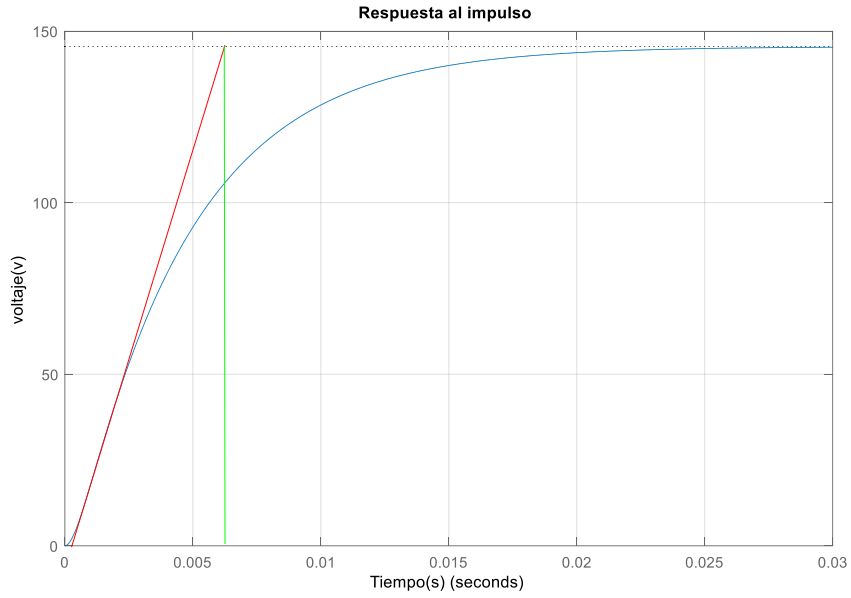


Figura 27. Parámetros de la respuesta al escalón de la planta.

Según la gráfica se pudo obtener los siguientes valores:

$$L = 1.86x10^{-3}\ T = 6.16x10^{-3}K = 145$$

Dados los valores se procede a reemplazarlos en la siguiente tabla:

TIPO DE PID	Kp	$TI = \frac{Kp}{Ki}$	$TD = \frac{KD}{Kp}$
P	3.31	∞	0
PI	2.98	0.0062	0
PID	3.97	0.0037	0.0093

Tabla 4. Parámetros obtenidos del controlado medinte Ziegler –Nicholls

Hallados los parámetros del controlador se lo asocia al sistema propuesto en lazo cerrado como se observa en la figura 28.

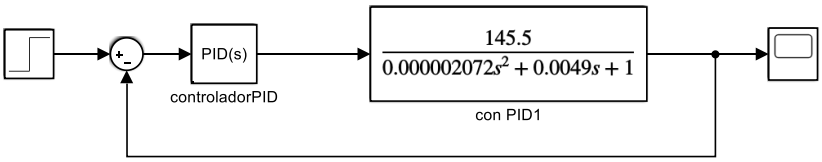


Figura 28. Diagrama de la planta + control PID.

Se realiza la simulación del sistema haciendo una comparativa entre el control PID y el sistema sin control PID como se ve a continuacion en la figura 29.

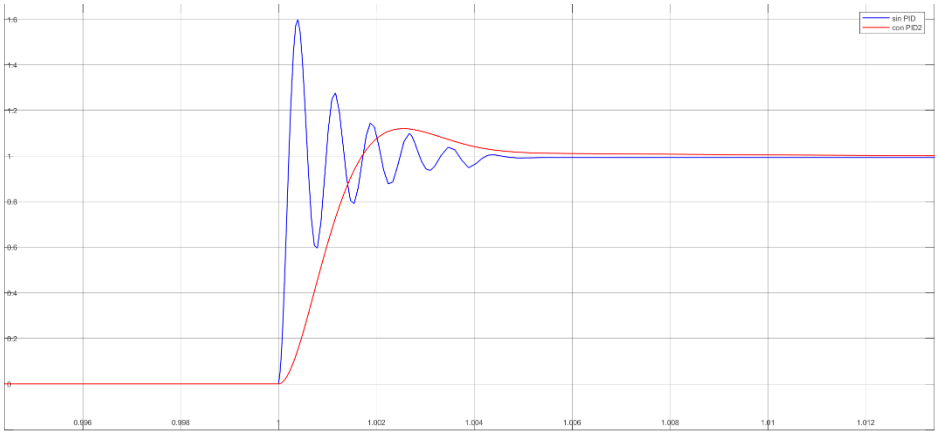


Figura 29. Respuesta del sistema en lazo cerrado con control PID (rojo) , respuesta del sistema en lazo cerrado sin control PID (azul).

Se observa que la respuesta de la planta mejora considerablente en cuanto a los siguientes criterios : sobreimpulso, tiempo de establecimiento y de subida. Con esto se puede controlar de manera precisa la señal del PWM que varia el voltaje promedio a la entrada del motor .

- **Control Digital PID**

Tomando en cuenta el diseño del controlador PID, el procesamiento digital del mismo se realiza mediante las librerías de Arduino.

En la figura 30 se muestra la asignación de pines para la lectura de los sensores y control PWM para los motores.

```
const byte interruptA = 18; //Rueda atras izquierda
const byte interruptB = 19; // Rueda atras derecha
const byte interruptC = 20; // Rueda delantera derecha
const byte interruptD = 21; // Rueda delantera izquierda

const byte pinbreakA = 23; //Rueda atras izquierda
const byte pinbreakB = 25; // Rueda atras derecha
const byte pinbreakC = 27; // Rueda delantera derecha
const byte pinbreakD = 29; // Rueda delantera izquierda

const int pinPWMA = 2; //Rueda atras izquierda
const int pinPWMB = 3; // Rueda atras derecha
const int pinPWMC = 4; // Rueda delantera derecha
const int pinPWMD = 5; // Rueda delantera izquierda

const int pindirectA = 24; //Rueda atras izquierda
const int pindirectB = 26;// Rueda atras derecha
const int pindirectC = 32;// Rueda delantera derecha
const int pindirectD = 34;// Rueda delantera izquierda
```

Figura 30. Captura de pantalla de asignacion de variables.

Se asigna las variables para la matriz de transformación.

```
volatile float u=0;
volatile float w=0;
float r = 0.08;
float d = 0.28;
volatile float w_r ;
volatile float w_l;
```

Figura 31. Variables para la matriz de transformación.

Se usa el timer 4 debido a que presenta una mejor aproximación para el tiempo de muestreo y no interfiere con los pines de interrupción de los sensores. Se crea interrupciones cada 100ms para que actúe el PID.

```
// Configuración de Lectura

attachInterrupt(digitalPinToInterrupt(interruptA), encoderA, CHANGE);
attachInterrupt(digitalPinToInterrupt(interruptB), encoderB, CHANGE);
attachInterrupt(digitalPinToInterrupt(interruptC), encoderC, CHANGE);
attachInterrupt(digitalPinToInterrupt(interruptD), encoderD, CHANGE);

noInterrupts(); // disable all interrupts
 //set timer4 interrupt at 10Hz
 TCCR4A = 0;// set entire TCCR1A register to 0
 TCCR4B = 0;// same for TCCR1B
 TCNT4  = 0;//initialize counter value to 0
 // set compare match register for 1hz increments
 //OCR4A = 15624/1;// = (16*10^6) / (1*1024) - 1 (must be <65536)
 // set compare match register for 10hz increments
 OCR4A = 1561;// = (16*10^6) / (10*1024) - 1 (must be <65536)
 // turn on CTC mode
 TCCR4B |= (1 << WGM12);
 // Set CS12 and CS10 bits for 1024 prescaler
 TCCR4B |= (1 << CS12) | (1 << CS10);
 // enable timer compare interrupt
 TIMSK4 |= (1 << OCIE4A);
interrupts(); // enable all interrupts

}
```

Figura 32. Configuración del Timer 4.

Se usa la función callback que está asociada a la interrupción para obtener la lectura de la velocidad angular de las ruedas independientemente en rad/s.

```
ISR(TIMER4_COMPA_vect){

 velA = (countA/(0.1*resolucion))*(2*pi);//rad/s
 velB = (countB/(0.1*resolucion))*(2*pi);//rad/s
 velC = (countC/(0.1*resolucion))*(2*pi);//rad/s
 velD = (countD/(0.1*resolucion))*(2*pi);//rad/s

  //Serial.println(velA);
  //Serial.println(velB);
  //Serial.println(velC);
  //Serial.println(velD);

  countA = 0;
  countB = 0;
  countC = 0;
  countD = 0;

 }
```

Figura 33. Lectura de las velocidades angulares de cada una de las ruedas.

Se usa un control PID en tiempo discreto definiendo la ecuación tal que:

$$u(k) = u(k-1) + q_0 e(k) + q_1 e(k-1) + q_2 e(k-2)$$

Para la parte del controlador digital, se usa la librería de arduino PID_V2 y se asigna las constantes del controlador obtenidas teóricamente y ajustadas en la parte práctica.

```
//Constantes PID

double Kp =0.95, Ki = 0.9, Kd = 0.065;

PID_v2 myPID_A(Kp, Ki, Kd, PID::Direct);
PID_v2 myPID_B(Kp, Ki, Kd, PID::Direct);
PID_v2 myPID_C(Kp, Ki, Kd, PID::Direct);
PID_v2 myPID_D(Kp, Ki, Kd, PID::Direct);
```

Figura 34. Librería PID_V2 y cosntantes del controlador.

Se define el ingreso, salida y setpoint para el controlador.

```
//PID
myPID_A.Start(analogRead(velA), pinPWMA, 0);
myPID_B.Start(analogRead(velB), pinPWMB, 0);
myPID_C.Start(analogRead(velC), pinPWMC, 0);
myPID_D.Start(analogRead(velD), pinPWMD, 0);
```

Figura 35. Parámetros del controlador.

Se configura la dirección de acuerdo a la lectura que toman las varibales de velocidad linel y velocidad angular.

```
            direccion();

            myPID_A.Setpoint(w_l);
            myPID_B.Setpoint(w_r);
            myPID_C.Setpoint(w_r);
            myPID_D.Setpoint(w_l);


            PWMA = myPID_A.Run(velA);
            PWMB = myPID_B.Run(velB);
            PWMC = myPID_C.Run(velC);
            PWMD = myPID_D.Run(velD);

            analogWrite(pinPWMA, PWMA);
            analogWrite(pinPWMB, PWMB);
            analogWrite(pinPWMC, PWMC);
            analogWrite(pinPWMD, PWMD);

          }
```

Figura 36. Definimos el estado para la dirección de las ruedas.

```
void direccion(){

 if(w_r<0 && w_l >0){

      digitalWrite(pindirectA, HIGH );// Rueda atras izquierda 24
      digitalWrite(pindirectB, HIGH); // Rueda atras derecha 26
      digitalWrite(pindirectC, HIGH); // Rueda delantera derecha 32
      digitalWrite(pindirectD, HIGH); //Rueda delantera izquierda  34

}

 else if (w_r>0 && w_l<0){

      digitalWrite(pindirectA, LOW);// Rueda atras izquierda 24
      digitalWrite(pindirectB, LOW); // Rueda atras derecha 26
      digitalWrite(pindirectC, LOW); // Rueda delantera derecha 32
      digitalWrite(pindirectD, LOW); //Rueda delantera izquierda  34

 }

 else if (w_r<0 && w_l<0){

  digitalWrite(pindirectA, LOW);// Rueda atras izquierda 24
  digitalWrite(pindirectB, HIGH); // Rueda atras derecha 26
  digitalWrite(pindirectC, HIGH); // Rueda delantera derecha 32
  digitalWrite(pindirectD, LOW); //Rueda delantera izquierda  34
```

```
else {

 digitalWrite(pindirectA, HIGH );// Rueda atras izquierda 24
 digitalWrite(pindirectB, LOW); // Rueda atras derecha 26
 digitalWrite(pindirectC, LOW); // Rueda delantera derecha 32
 digitalWrite(pindirectD, HIGH); //Rueda delantera izquierda  34
}

w_r=abs(w_r);
w_l=abs(w_l);

}
```

Figura 37. Configuración de la dirección de las ruedas.

2.3.2.2 Implementación del framework ROS.

El framework ROS (Robot Operating System) es la plataforma encargada del procesamiento de la información que se recibe desde la cámara Intel Real Sense D435 y el sensor YDLIDAR. El objetivo principal del sistema operativo es permitir que el robot móvil integre diferentes algoritmos de manera sencilla en una sola plataforma, en la figura 38 se muestra los periféricos que están conectados [13].

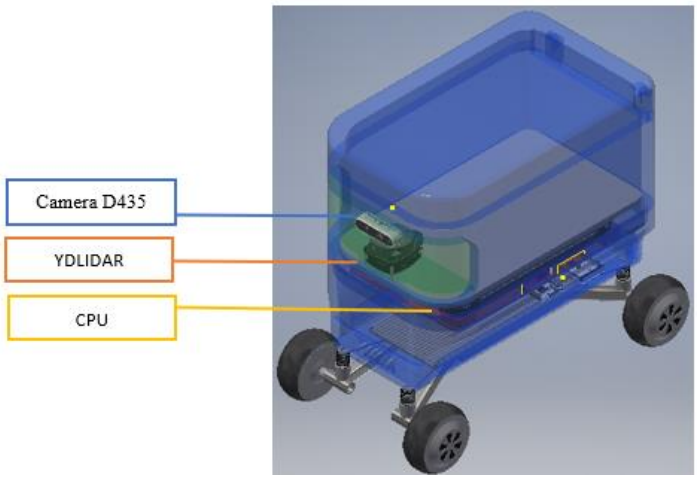


Figura 38. Dispositivos periféricos conectados al robot móvil.

El procesamiento de la información en el framework ROS se realiza mediante nodos. Un nodo maestro permite al resto de los nodos que se encuentren para

comunicarse. Cada nodo está encargado de realizar una tarea específica de una orden que se le dé al robot[13] [14].

En la figura 39, se muestran las entidades y los medios necesarios para la comunicación de los nodos

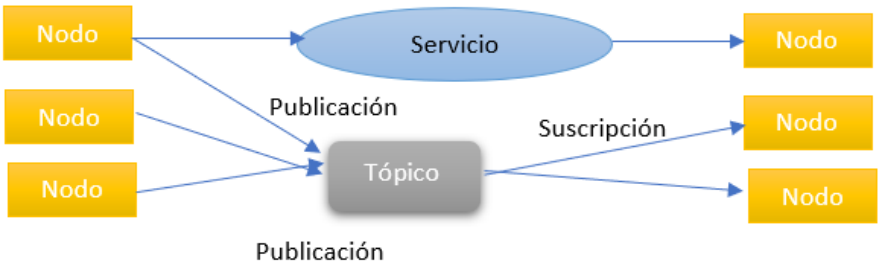


Figura 39. Arquitectura de ROS.

La composición de Ros consta de dos partes:

- Sistema operativo base.
- ROS-pkg, que consta de una agrupación de paquetes con aporte por diferentes usuarios [13].

En el robot móvil se utiliza el SLAM, basado en conjuntos tales como percepción, localización, mapeo, evasión de obstáculos. ROS tiene diferentes algoritmos para la implementación del SLAM, el utilizado en este caso RTAB-Map; el mismo que genera una nube de puntos a partir de información que se obtiene desde el dispositivo Real Sense para la construcción de un mapa, que se utilizara para la odometría del robot móvil [13] [14].

Real Sense

En el robot móvil se utiliza la cámara de visión estéreo “Intel Real Sense Depth Camera D435”, que se integra al software de ROS, ésta captura escenas a analizar por medio de las librerías proporcionadas por Intel [19].

Esta adquisición es mediante una nube de puntos texturizada conjuntamente con la imagen que es de profundidad ; para capturar profundidad, la cámara

funciona de manera simultánea con el sensor infrarrojo YDLIDAR teniendo así la imagen y los valores intrínsecos del sensor [20].

La nube de puntos también puede ser obtenida mediante puntos con tres dimensiones *(X, Y, Z)* en donde la profundidad está representada por medio de la coordenada como se muestra en la figura 40, haciendo la de proyección de la imagen de profundidad [20].

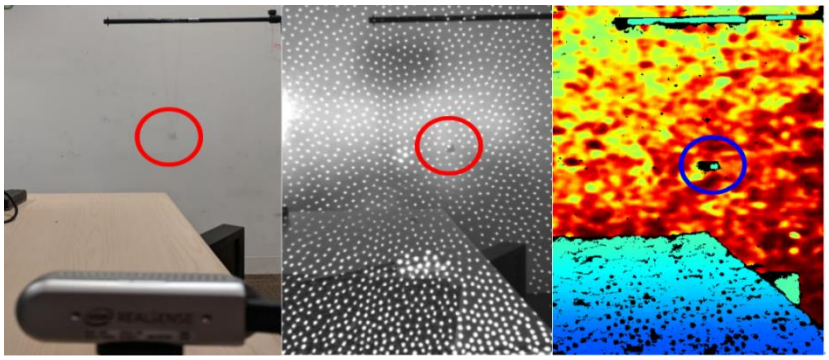


Figura 40. Muestra un objeto pequeño de 20 mm suspendido en el aire, como se indica con un círculo. Izquierda: Imagen de la cámara D435 apuntando a un objeto pequeño. Centro: imagen infrarroja del objeto. Derecha: mapa de profundidad.

YDLIDAR

El sensor YDLIDAR capta una nube de puntos del lugar en donde está establecido , haciendo que el CPU procese una imagen bidimensional en tiempo real , esto lo logra mediante la unión que proporciona el RViz que es un programa alternativo de ROS que permite la unión de sensores para obtener un mapa, mediante el cual se visualiza una serie de puntos rojos que se obtienen del láser como podemos ver en la figura 41, estas representaciones son de manera simultánea de los objetos que existen en el entorno actualizándose en tiempo real [21] [22] [23].

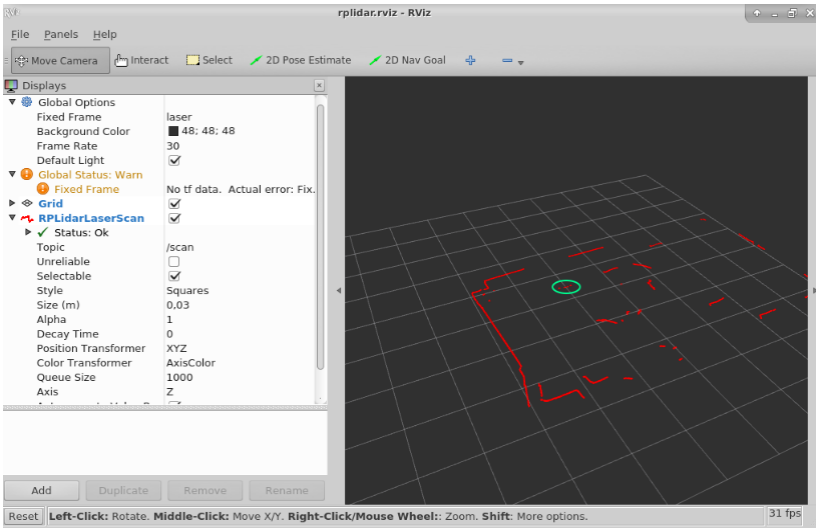


Figura 41. Visualización RVIZ [21].

Cabe recalcar que lo fundamental de la nube de puntos es precisar la posición en el espacio y la distancia hasta él; de esta manera evita que el robot en el transcurso de las trayectorias que realice impacte con algún objeto [21] [22].

RTAB Map

El algoritmo RTAB-Map desarrolla el SLAM sin limitaciones en cuanto al tiempo de sistematización en recorridos que son extensos. El mismo que divide la memoria de ROS en una memoria que es de trabajo (WM), una memoria a largo plazo (LTM), una memoria a corto plazo (STM) y en una memoria sensorial (SM). La función principal de este fraccionamiento es que la memoria de trabajo (WM) se mantengan las localizaciones recientes y de manera frecuente; por otro lado, se puede reducir los cálculos y comparaciones mediante el respaldo de las localizaciones en la memoria a largo plazo. En la figura 42, se observa el diagrama de flujo de información de cada imagen representada en el algoritmo RTAB-Map [15] [16] [17].

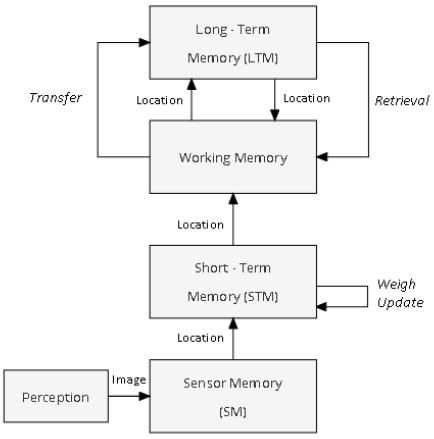


Figura 42. Esquema de memoria de RTAB -Map [15].

La capacidad para reconocer la localización anterior es uno de los puntos esenciales de las técnicas de SLAM, [15] [16]. En la figura 43 se puede visualizar la Detección de bucle cerrado" o "(Loop closure detection), proceso mediante el cual se determina si la localización es nueva o actual. [17].

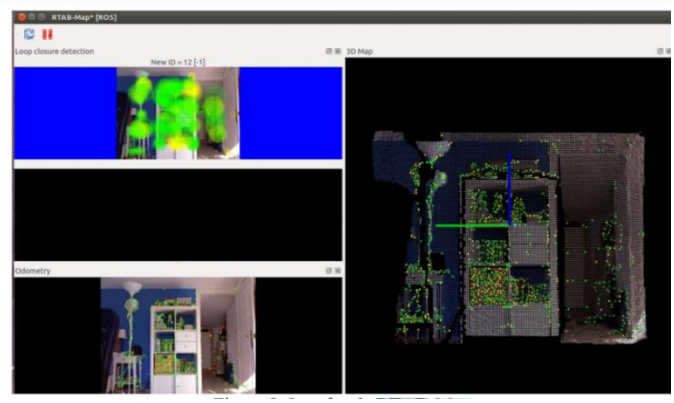


Figura 43. Detección de bucle cerrado (loop closure detection) RTAB-Map [18].

2.5 Diseño PCB

El diseño se realizó con la integración de los drivers para los motores y el Arduino que es para la parte de mando y control de los mismos.

Los criterios de diseño se basaron generalmente en la parte de la cantidad de corriente que va a pasar por los componentes, en este caso los drivers.

Como nuestra placa maneja corrientes altas, se realizó el cálculo del ancho de pistas para un correcto funcionamiento de nuestro PCB.

$$Ancho = \frac{\left\{\left[\frac{l}{(k1 * \Delta T k2)}\right] * \left[\frac{1}{k3}\right]\right\}}{(L * 1.378)}$$

Donde I representa la corriente máxima; k1 es una constante determinada por el estándar que se aplica y su valor es de 0,0150 cuando la pista es interna, es decir con más de dos capas, y de 0,0647 cuando es externa; k2 es también una constante que su valor es de 0,5453 cuando la pista es interna y de 0,4281 cuando la pista es externa y k3 es una constante más que su valor corresponde a 0,7349 cuando la pista es interna y de 0,6732 cuando es externa; y finalmente, L que simboliza el grosor de la pista; ΔT variación de la temperatura.

Para el grosor de la pista se usó medidas estándar de 1, 2 y 3 onz por pie cuadrado, dicho de otra manera 35, 70 y 105 micras respectivamente.

$$Ancho = \frac{\left\{\left[\frac{10A}{(0.0647 * (50 - 25)(0.4281))}\right] * \left[\frac{1}{(0.6732)}\right]\right\}}{(2 * 1.378)}$$

$$Ancho = 7.78\ th$$

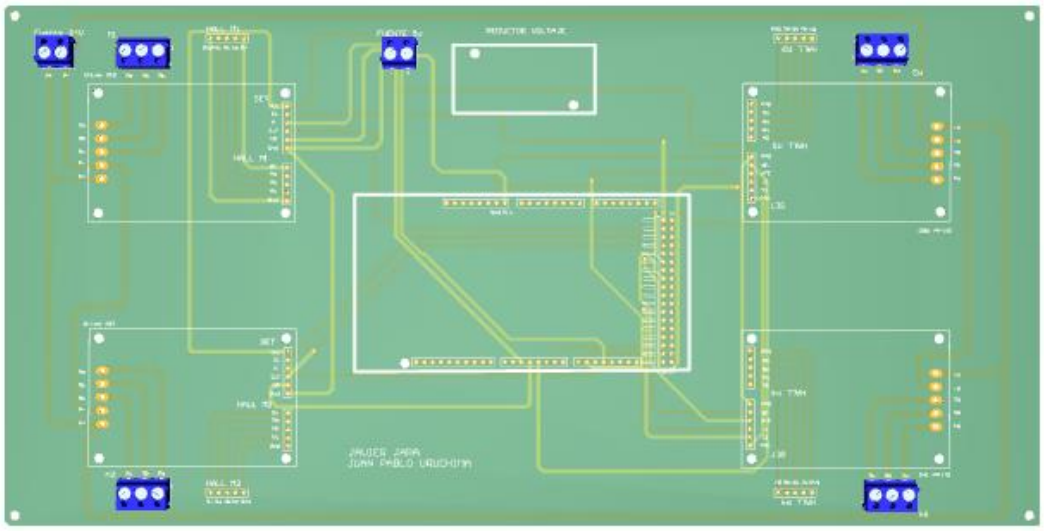


Figura 44. Diseño PCB para integración de drivers y arduino mega.

CAPÍTULO 3

Pruebas de Funcionamiento y Experimentación del Robot

3.1 Pruebas de desempeño del controlador de bajo nivel.

En esta sección se presentan las pruebas del controlador de bajo nivel (PID), en donde se proponen que el robot se desplace a una velocidad de 0.25m/s sometido a diferentes cargas (en vacío, con 5, 20 y 35kg, ver figura 45). En las figuras 47, 49, 51 y 53 se compara la velocidad deseada con la velocidad real del robot, se observa que el controlador PID tiene una muy buena respuesta y la velocidad real del robot se apega a la velocidad deseada incluso cuando está sometido a diferentes cargas.

En vacío

Carga de 5kg

Carga de 20kg

Carga de 35kg

Figura 45. Cargas a las que fue sometido el robot en las pruebas de campo.

Cuando el robot está sin carga

En la figura 46 se observa la trayectoria que recorre el robot cuando no tiene ninguna carga (en vacío), se observa que al recorrer aproximadamente 1.58m el robot se alinea a la posición deseada, le toma 1.8s esta acción de alinearse a la trayectoria deseada.

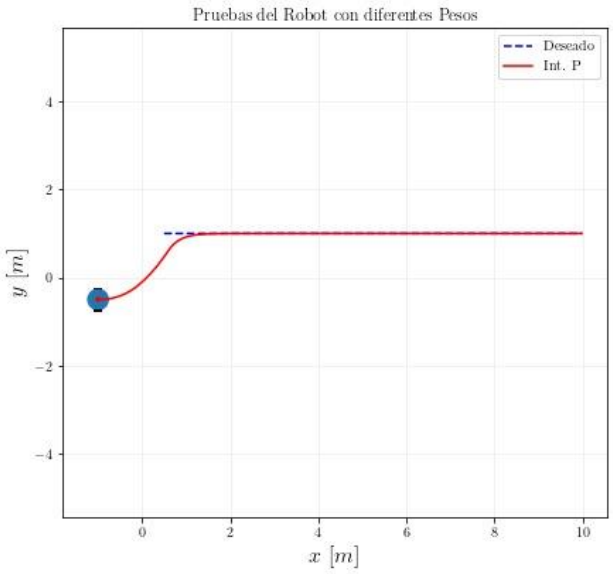


Figura 46. Trayectoria del robot sin carga.

En la figura 47 se puede observar errores de seguimiento y acciones de control del robot móvil cuando está sin carga, la velocidad lineal que establecemos para la prueba es de 0.25m/s; conforme a las pruebas realizadas observamos un píco inicial de 1.3m/s y que converge a 0.25m/s conforme a los requerimientos impuestos de velocidad en un tiempo 3.4s.

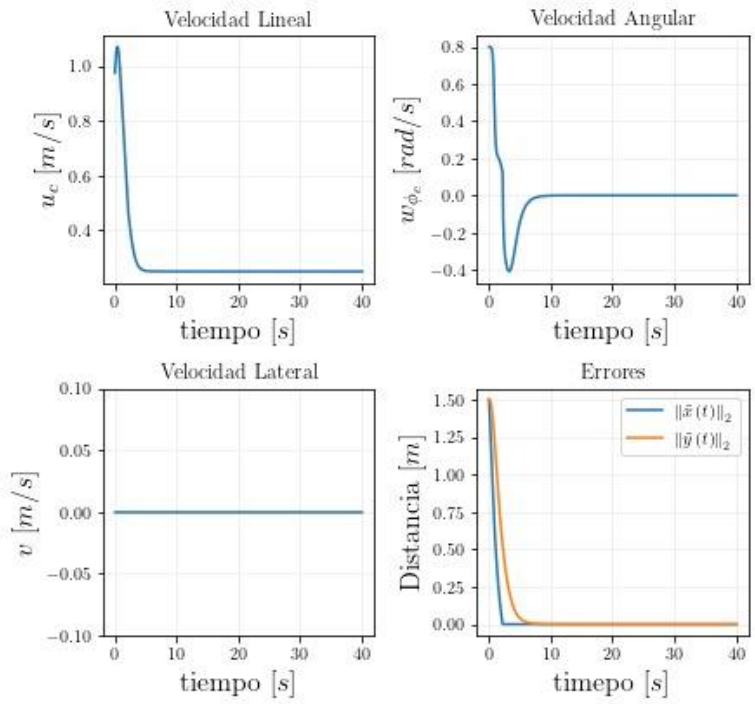


Figura 47. Comparación de las velocidades cuando el robot está sin carga.

Con carga de 5kg

En la figura 48 se observa la trayectoria que recorre el robot cuando tiene carga de 5kg, se observa que al recorrer aproximadamente 1.42m el robot se alinea a la posición deseada, le toma 1.2s esta acción de alinearse a la trayectoria deseada.

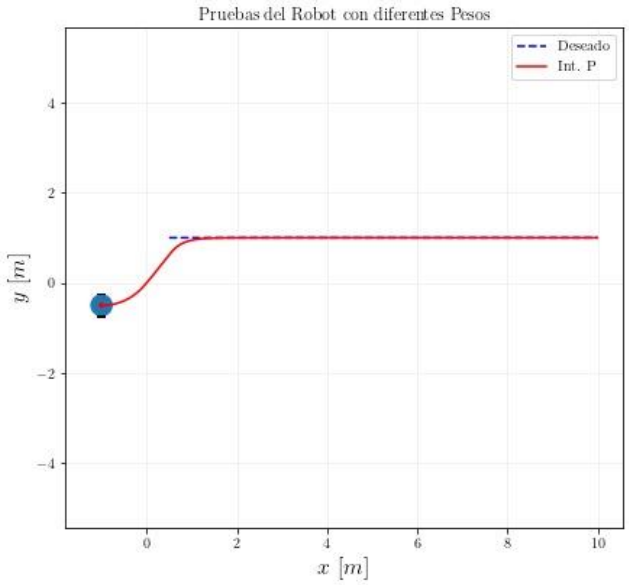


Figura 48. Trayectoria del robot con una carga de 5kg.

En la figura 49 se observa las acciones de control y los errores de seguimiento del robot móvil cuando está con una carga de 5kg, la velocidad lineal converge a 0.25m/s conforme a los requerimientos establecidos. A diferencia de un robot sin carga, el pico que se genera en la velocidad lineal es menor debido al torque que genera los motores, la velocidad lineal converge y se estabiliza a los 2.8s

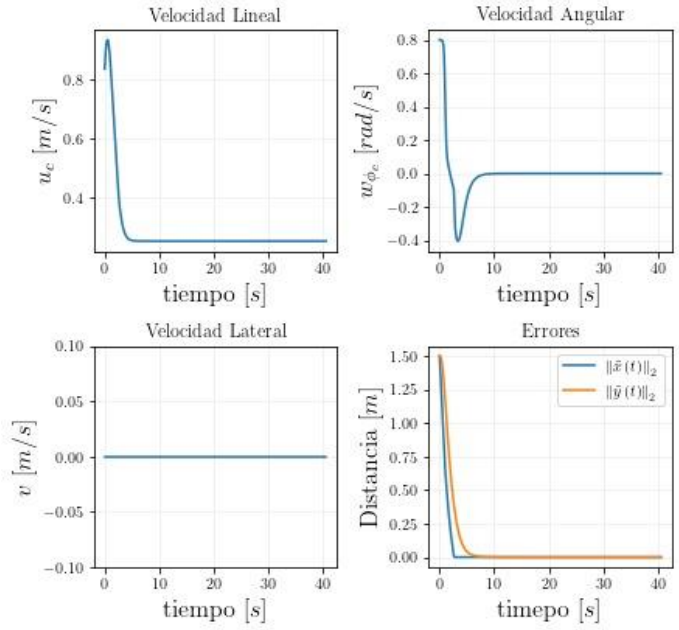


Figura 49. Comparación de las velocidades cuando el robot está con una carga 5kg.

Con carga de 20kg

En la figura 50 se observa la trayectoria que recorre el robot cuando tiene carga de 20kg, se observa que al recorrer aproximadamente 0.65m el robot se alinea a la posición deseada, le toma 1.08s esta acción de alinearse a la trayectoria deseada.

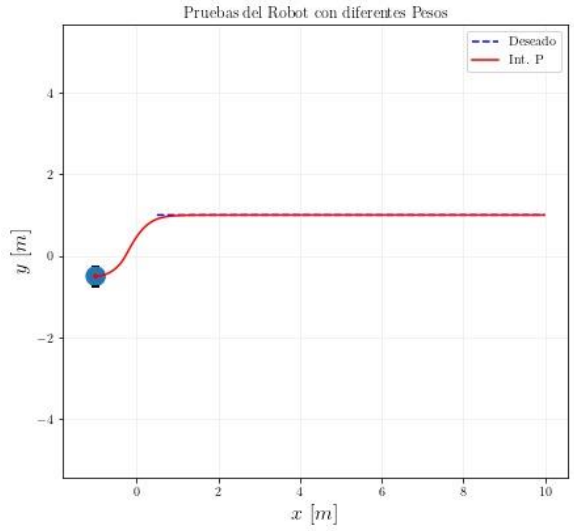


Figura 50. Trayectoria del robot con una carga de 20kg.

En la figura 51 se puede observar los errores de seguimiento y acciones de control del robot móvil cuando está con una carga de 20kg, la velocidad lineal converge a 0.25m/s conforme a los requerimientos establecidos. El pico que se genera en la velocidad lineal cada vez disminuye, eso debido a que se genera un mayor esfuerzo el romper la inercia con un mayor peso, la velocidad lineal converge y se estabiliza a los 2.03s.

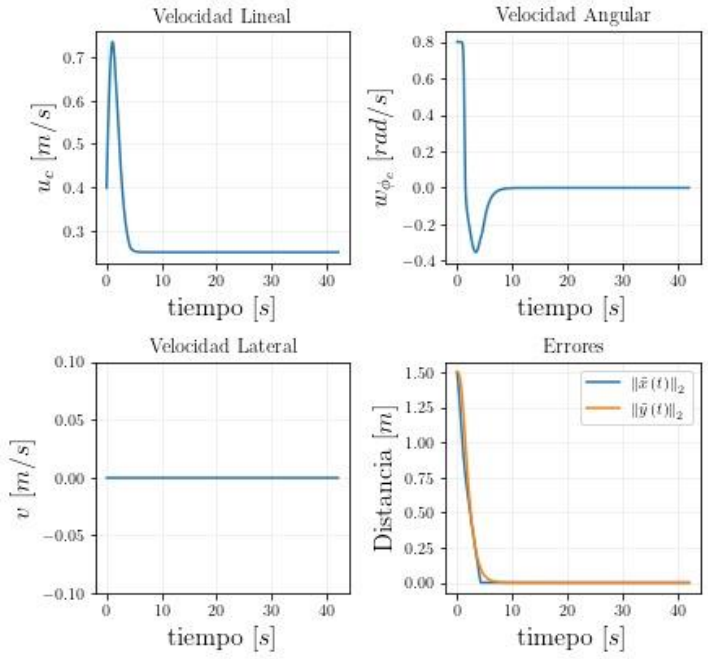


Figura 51. Comparación de las velocidades cuando el robot está con una carga 20kg.

Con carga de 35kg

En la figura 52 se observa la trayectoria que recorre el robot cuando tiene carga de 35kg, se observa que al recorrer aproximadamente 3.75m el robot se alinea a la posición deseada, le toma 3.4s esta acción de alinearse a la trayectoria deseada.

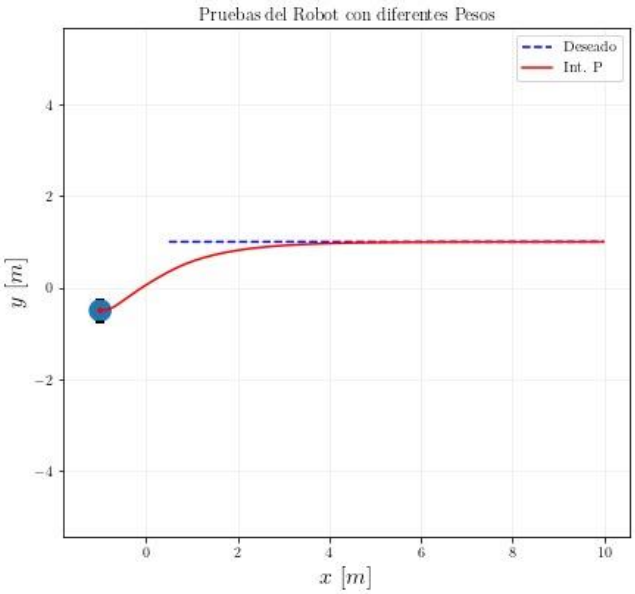


Figura 52. Trayectoria del robot con una carga de 35kg.

En la figura 53 se observa las acciones de control y los errores de seguimiento del robot móvil cuando está con una vcarga de 35kg, la velocidad lineal converge a 0.25m/s conforme a los requerimientos establecidos. El pico de velocidad lineal cada vez es menor, esto se debe a que cada vez que se coloque una carga mayor, el robot realiza mayor torque por lo tanto la velocidad se reduce.

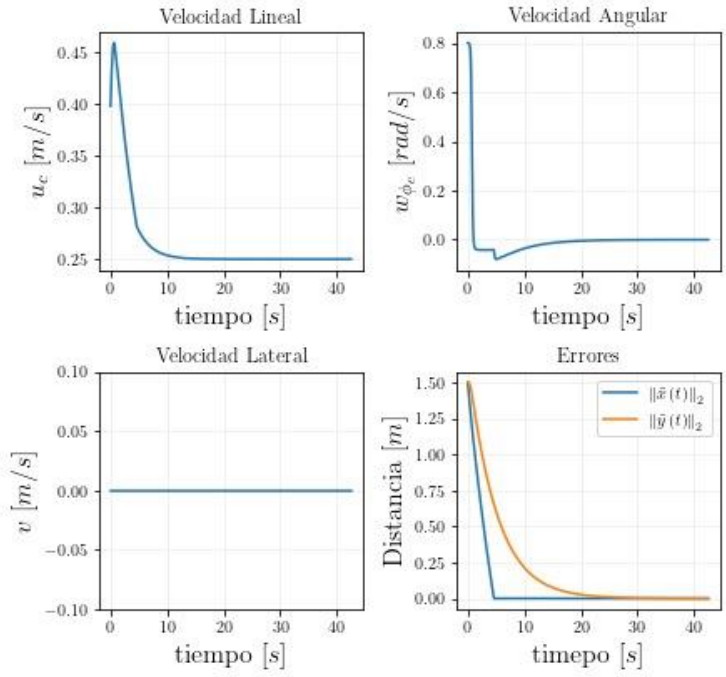


Figura 53. Comparación de las velocidades cuando el robot está con una carga 35kg.

En la tabla 5 se exponen los valores de las diferente, cargas empleadas, la velocidad del robot en estado estacionario, la velocidad de referencia, distancia a la que converge y el tiempo que se demora en converger. A medida que se incrementa el peso en el robot, los picos de velocidad disminuyen, esto debido a que, mientras más peso de le cargue al robot, a éste se le va a hacer más difícil romper la inercia y por lo tanto disminuye la velocidad.

Peso	**Pico de Velocidad Lineal**	**Velocidad de Referencia**	**Distancia a la que converge**	**Tiempo de convergencia**
Sin carga	1.12m/s	0.25m/s	1.58m	1.8s
Peso de 5kg	0.94m/s	0.25m/s	1.42s	1.2s

Peso de 20kg	0.76m/s	0.25m/s	0.65m	1.08s
Peso de 35kg	0.453m/s	0.25m/s	3.75m	3.4s

Tabla 5. Valores de velocidad, distancia y tiempo de convergencia de acuerdo a las cargas proporcionadas al robot móvil.

3.2 Navegación Autónoma del Robot.

Para la navegación autónoma del robot se realizó un algoritmo de seguimiento de trayectoria en donde el robot parte de una posición específica y debe alcanzar la trayectoria deseada lo más pronto posible. En la figura 54, se observa que el robot móvil sigue la trayectoria elíptica establecida a la que converge en un tiempo de 2.35s.

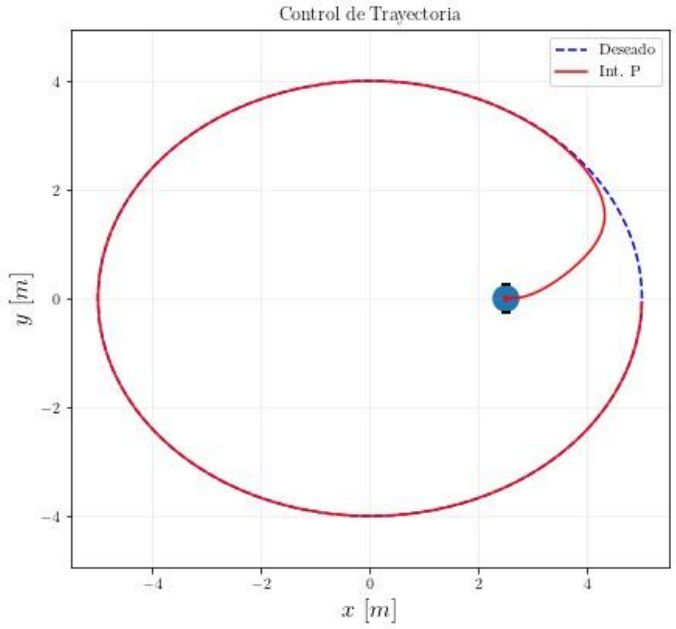


Figura 54. Trayectoria elíptica obtenida por el robot móvil.

La siguiente figura refleja la velocidad lineal y velocidad angular que el robot presenta al alcanzar la trayectoria propuesta en la figura anterior. Se probó el robot con una carga de 5kg, se puede observar un pico de velocidad lineal de 0.88m/s y velocidad angular de 0.88 rad/s, la velocidad lineal oscila entre 0.4 y 0.55m/s debido a la geometría de la trayectoria. La velocidad angular tiene una oscilación entre 0.1rad/s y 0.15rad/s.

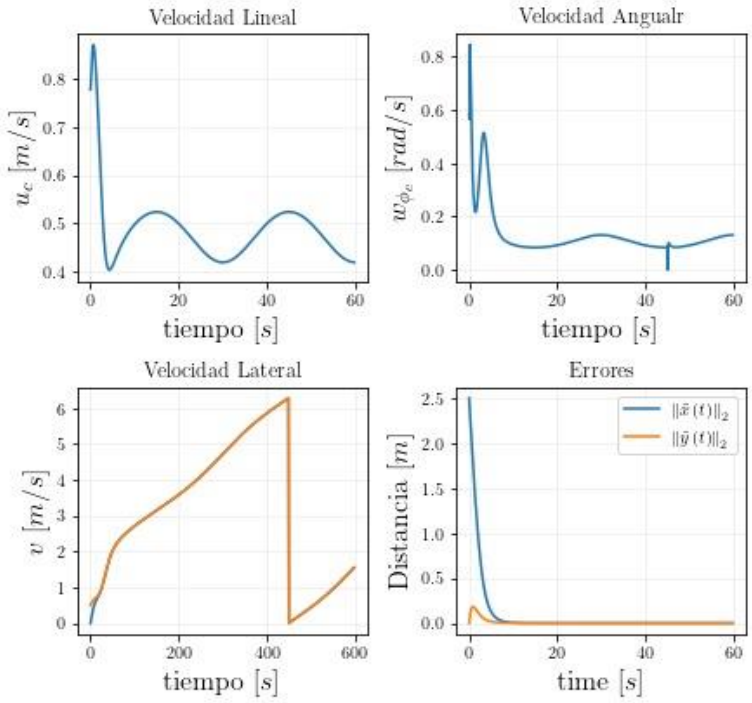


Figura 55. Comparación de las velocidades cuando el robot recorre una trayectoria elíptica.

En la figura 56, se observa que el robot ahora sigue una trayectoria de tipo lemniscata y también presenta un muy buen desempeño. En esta trayectoria se observa que la velocidad cambia con respecto al tiempo esto debido a la geometría de la trayectoria; el robot se demora 2.5s en alinearse a la trayectoria.

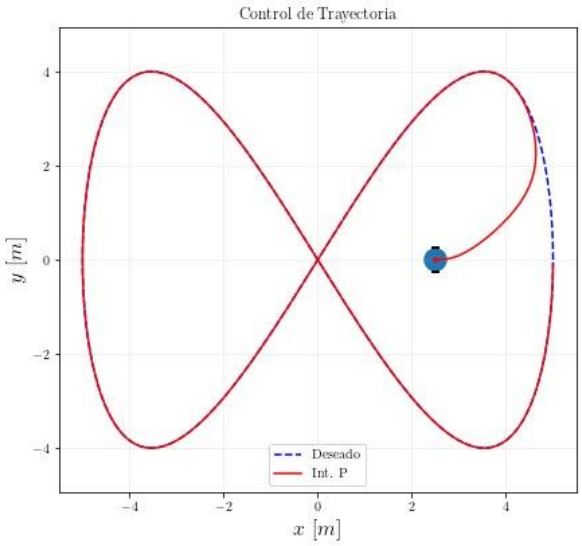


Figura 56. Trayectoria lemniscata obtenida por el robot móvil.

En la figura 57 se observa la velocidad lineal y la velocidad angular que el robot presenta al alcanzar la trayectoria lemniscata propuesta en la figura anterior; se puede observar oscilaciones de velocidad entre 0.2m/s y 0.6m/s esto debido a la geometría de la trayectoria, en cuanto a la velocidad angular, se presenta una oscilación entre 0.1rad/s y 0.3rad/s; y de -0.1rad/s y -0.090rad/s debido a los giros que realizan las ruedas para poder realizar la trayectoria.

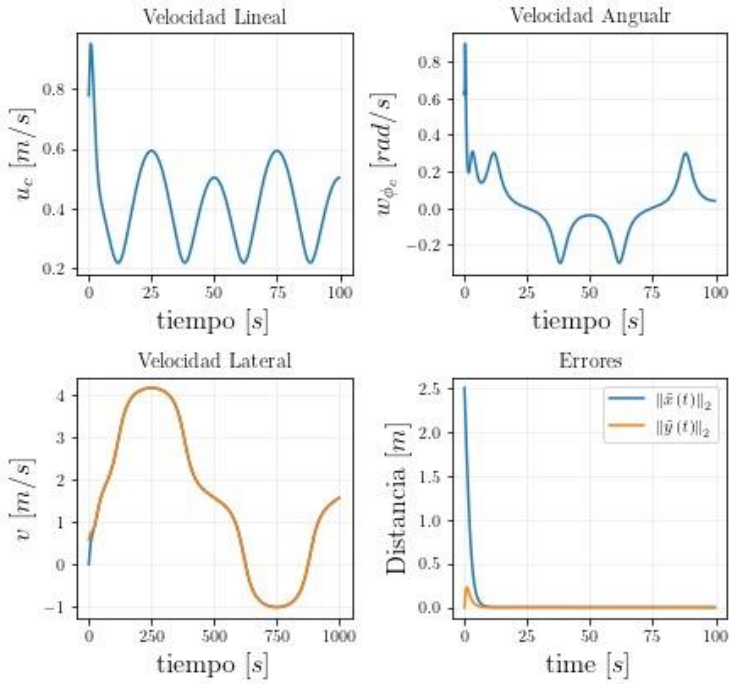


Figura 57. Comparación de las velocidades cuando el robot recorre una trayectoria lemniscata.

3.2.1 Levantamiento de Mapas del Ambiente de Navegación.

El objetivo del levantamiento de mapas es obtener el mapa del entorno en el cual navega el robot móvil, de tal forma que el mapa obtenido pueda ser usado por otras tareas de control.

Mediante una representación geométrica se obtiene las restricciones de espacio en dos dimensiones, como por ejemplo: paredes, puertas de acceso, pasillos, muebles, entre otros. Con esto es posible que le robot navegue de forma segura evitando posibles colisiones.

Al tener en cuenta la representación geométrica se puede distinguir entre zonas ocupadas y zonas aptas para la navegación, de esta manera se obtiene una descripción completa del espacio explorado, por consiguiente, se estima la posición del robot y la representación del mapa. Se puede elegir una

estrategia para conseguir el camino que deberá seguir el robot, o usar la teleoperación durante la exploración.

En la figura 58 se presenta el levantamiento de mapas de un ambiente interno. Para esta tarea se teleopera el robot mediante un joystick, la velocidad promedio de desplazamiento del robot es de 0.2m/s (velocidad impuesta). El levantamiento de mapa se realizó dentro de un domicilio en la planta baja y la proyección del mapa se realizó mediante el software Rtabmap.

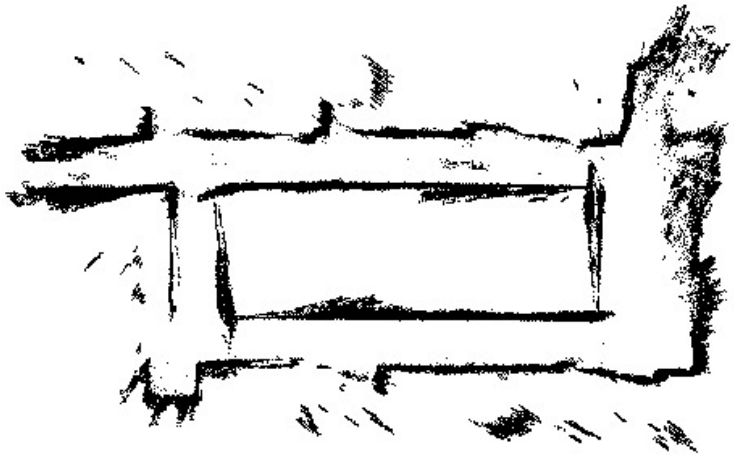


Figura 58. Levantamiento de mapa de un ambiente interno teleoperado con un joystick.

A continuación, se muestra el mapeo en tiempo real junto con los datos obtenidos del mapeo con el láser.

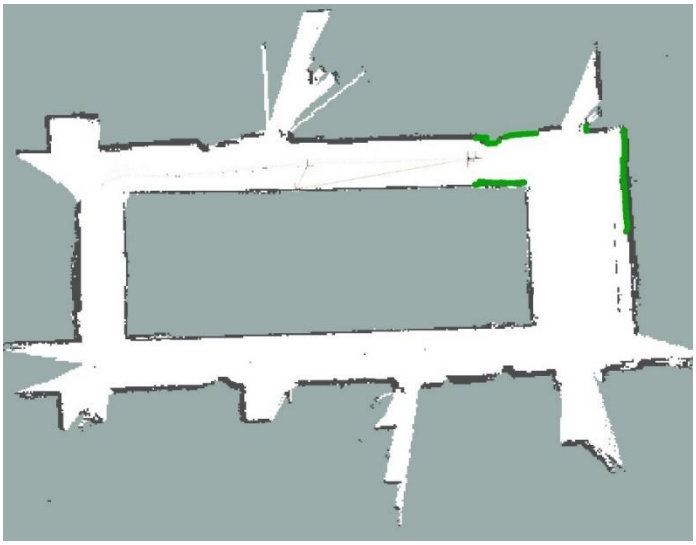


Figura 59. Levantamiento de mapa de un ambiente interno teleoperado con el láser.

En la figura 59, los puntos verdes que se exponen correspondes a los puntos del láser y se comprueba que el láser y la cámara están trabajando simultáneamente. Como se puede observar el mapeo obtenido por el robot es bastante confiable en comparación a las medidas reales, por ejemplo, el pasillo tiene una medida de 1.4m y en el mapeo se obtiene una medida de 1.395m. El error de medición en el área de mapeo con respecto al valor obtenido mediante el mapeo con el robot móvil es de 0.005m, corresponde a un 0.36% de error.

A continuación, se presenta otra zona mapeada en donde se puede visualizar el buen desempeño del algoritmo.

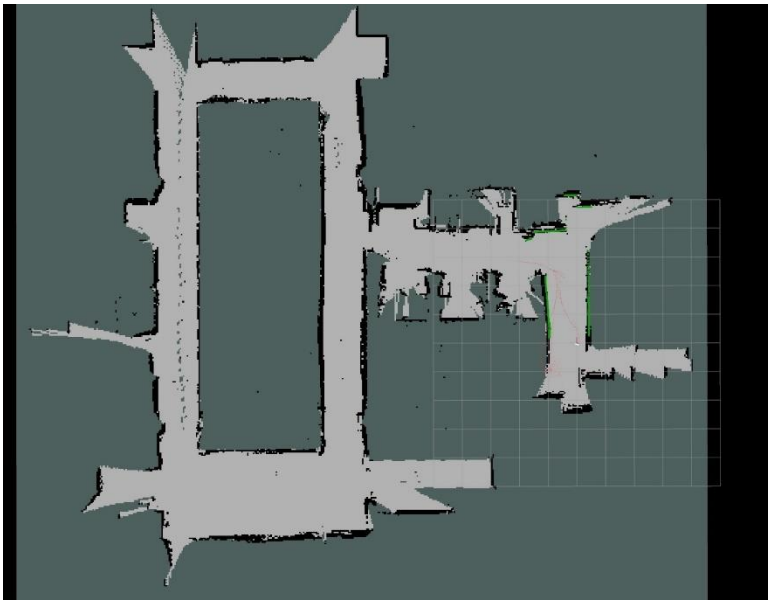


Figura 62. Levantamiento de mapa de un ambiente interno.

3.3 Lista de Equipos, Materiales y Presupuesto.

En la siguiente tabla se detalla los materiales usados y procesos de construcción realizados en el proyecto.

Material	Cantidad	Precio Unidad	Total
Motor Brushless	4	32.29	129.16
Driver Motor Brushless	12	15.84	190.08
Sensor RP Lydar	1	109.99	109.99
Cámara Intel Real Sense D435	1	285.06	285.06
Kit Raspberry Pi4	1	149.99	149.99

Tubo estructural ¾”	2	5	10
Soldado Estructura Robot	1	100	100
Colocación Sistema de Resortes	1	120	120
Impresión 3D Estructura	1	200	200
Impresión Placas Electrónicas	1	80	80
Batería 12V 7.5A	2	25	50
Materiales Electrónicos (borneras, peinetas, etc)	1	25	25
Total:			**1339,28**

Tabla 6. Lista de materiales, equipos y presupuesto.

CAPÍTULO 4

Conclusiones, Recomendaciones y Trabajo futuro

CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

Según diseño previo de la estructura del robot se pudo considerar aspectos tales como carga máxima, disposición de las ruedas, dimensiones de la carga a transportar y se logró una construcción óptima del robot. Algunas otras experiencias en cuanto a la construcción del robot hacen referencia a la distribución de pesos para la estabilidad y la importancia de la adherencia de las ruedas al piso, ya que al principio se modeló una estructura en donde al probar el robot se constató que todos los motores no tenían contacto con el suelo en terrenos irregulares, por lo que se diseñó y se montó un sistema de resortes (individuales para cada rueda) que permiten empujarle a los motores y tener mayor adherencia en el suelo al momento de presentarse terrenos irregulares.

En cuanto a la construcción del cuerpo del robot, es necesario considerar que la impresión 3D debe tener la suficiente robustez para que el robot pueda soportar el peso al que va a ser sometido. En nuestro caso, se consideró un relleno del 75% ya que con este valor se obtuvo la robustez deseada.

En el diseño de la placa electrónica en donde se integra los drivers de los motores y el Arduino, se debe tener especial cuidado con las conexiones de voltaje, ya que los drivers poseen una fuente interna de 5V que permite el comando directo de los motores. Así mismo, se debe realizar el conexionado de todas las tomas a tierra de los drivers, el Arduino y la fuente de alimentación, para evitar el reinicio forzado ya sea del Arduino o de los drivers.

Para la implementación de los controladores se usa un Arduino Mega, debido a que tiene la capacidad de manejar seis pines de interrupción, a diferencia de otros dispositivos. En nuestro caso usamos cuatro interrupciones para la lectura de los encoders de velocidad y una interrupción adicional para sincronizar el tiempo de muestreo.

Es necesario realizar una adecuada calibración de los controladores de bajo nivel, debido a que en los controladores de alto nivel se asumen un perfecto seguimiento de las velocidades de referencia que se envían hacia los motores.

CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES

En este trabajo se realizó una primera aproximación mediante métodos de sintonización gráfica y luego un ajuste fino de parámetros del controlador para lograr un mejor desempeño del robot móvil.

El robot puede ser teleoperado mediante un joystick, en donde es necesario vincular al Arduino con ROS para el envío y recepción de los comandos de control, para este propósito se usa Python debido a la versatilidad que presenta para integrarse con otros softwares.

El uso del modelo cinemático extendido permite proponer un controlador sencillo en el cual se tiene en cuenta directamente los objetivos de control, los cuales cumplen con los requerimientos de velocidad que necesita el robot móvil para completar la tarea asignada. Los valores de ganancia del controlador tuvieron un gran desempeño en las tareas propuestas, los valores usados fueron de $kx = 0.8\,, ky = 0.8, k\phi = 0.5$. Estos valores pueden ser modificados de acuerdo al desempeño del controlador de alto nivel.

En la instalación de los sensores Lidar y cámara RGB-D, se debe configurar los permisos de lectura y escritura de los puertos a los cuales se conectan, esto debido a que el robot está implementado en una arquitectura Linux.

Para el levantamiento de mapas es necesario verificar el correcto funcionamiento de los sensores e instalar adecuadamente las librerías y drivers necesarios. Además, se debe corroborar la sincronización del sensor Lidar con la cámara RGB-D para el adecuado funcionamiento del algoritmo de mapeo.

Durante la adquisición del mapa de un ambiente interno, se obtuvieron mediciones con errores del 0.36% en relación con el mapa real y el mapa obtenido con el algoritmo. Las áreas mapeadas coincidieron en un 100% con el espacio real al que fue inspeccionado por el robot móvil. No se pudo realizar el levantamiento de mapa del ambiente que corresponde desde la secretaria de la universidad hacia la dirección de carrera debido a las restricciones por la pandemia y los movimientos de oficinas que se dieron.

En la depuración del Software de control Ros como configuración o puesta a punto de los dispositivos, cámara Intel Real Sense y láser, es necesario la calibración del sensor Lidar para que este interactúe con la cámara de manera sincronizada para el sensando correcto en el Rtabmap permitiendo dar una correcta movilidad y accionamiento ante diferentes escenarios de navegación (odometría), para esto se utiliza la función "*rqt tf tree*" que brinda información detallada sobre la interacción que se tiene de los periféricos que están trabajando en Ros proporcionando estadísticas sobre la sincronización del láser con la cámara.

4.2 Recomendaciones

A lo largo de la elaboración de este proyecto, nos pudimos encontrar con una serie de problemas que nos brindaron la posibilidad de desarrollar nuevos aprendizajes y nuevas destrezas dentro del campo de la ingeniería, por tal motivo consideramos necesario proponer las siguientes recomendaciones que estamos seguros ayudarán a mejorar los procesos de diseño, construcción y puesta en funcionamiento del robot móvil:

- Es necesario considerar que el diseño de la estructura realizado debe ser de la misma distancia entre ejes (formando así una estructura cuadrada) por lo que al no ser de esa manera afecta en el desplazamiento ya que en velocidad tangencial el movimiento no es natural.
- En el diseño de la impresión 3D se debe tener en cuenta que los pilares principales en donde van alojadas las cargas, los ángulos que se forman en este diseño no deben ser perpendiculares ya que debido a esto las áreas de contacto son propensas a quebrarse; se recomienda diseñar las uniones en forma de arco para distribución de fuerzas en el área de sujeción.
- Para la impresión 3D de la estructura, utilizar un material más resistente que pueda soportar cargas mucho más grandes a las que se plantea en este proyecto; en nuestro caso el objetivo principal es llevar o transportar documentos de un lugar a otro por lo que no se vio la necesidad de implementar una estructura robusta.
- Para el diseño de las fuentes de alimentación es importante tomar en cuenta los valores de la corriente con la que funcionaran los

actuadores tales como los motores, ya que esta magnitud es fundamental para la parte de electrónica de potencia necesaria para el funcionamiento correcto de los drivers.

- Tener en cuenta que cada elemento o dispositivo maneja su propia alimentación, por lo que es necesario unir las tierras de todos dispositivos para que tengan un nodo de referencia común; ya que, al no unir las tierras, las señales que reciben los drivers son de manera intermitente.
- Los drivers de los motores funcionan con 24V y el Arduino con un voltaje de 5 a 12V, por este motivo se tomó la decisión de alimentar a los drivers con una batería de 24V y al Arduino mediante el puerto serial proveniente de la Raspberry Pi. Esta decisión se tomó debido a que las pruebas con los reguladores de voltaje que proporcionaban los 5V para el Arduino no tenían la capacidad de absorber transitorios que se producían en el momento de accionamiento de los motores, trayendo como consecuencias en más de una ocasión el daño de los drivers.
- Para la lectura de los valores que proporcionan los encoders, se maneja interrupciones individuales para cada uno de los drivers ya que al manejar de manera conjunta existen conflictos con los puertos digitales.
- Se necesita la adecuada sincronización de la cámara RGB-D con el sensor Lidar, ya que al no estar estos elementos sincronizados no se puede obtener la profundidad correcta de los espacios a los que el robot móvil está mapeando.

4.3 Trabajo futuro

Como trabajo a futuro se propone la implementación de un controlador de alto nivel, hacer la respectiva comparación con el de bajo nivel ya implementado y dar a conocer las mejoras en cuanto al nuevo tipo de controlador.

Como el proyecto se basa en el levantamiento de mapas de ambientes para la navegación autónoma, quedaría pendiente realizar la navegación autónoma del robot sin el control con el joystick; con esto se puede lograr objetivos a nivel de aplicación como por ejemplo: la desinfección de áreas, entrega de paquetes o correspondencia (delivery), la exploración de áreas peligrosas, manipulación de materiales nocivos para la salud humana, etc.

Uno de los objetivos planteados al principio de la realización del proyecto fue la implementación de un brazo con 8 grados de libertad, esto para que el robot con el brazo pueda manipular o coger objetos para una tarea en concreto.

APÉNDICES

APÉNDICE A

Manual de Usuario

El robot móvil de tracción diferencial es un robot que utiliza la plataforma ROS y Arduino. El software y el hardware son de libre acceso. Al final del apéndice A, se encuentran los enlaces para descargar, con las respectivas especificaciones y aplicaciones manejadas en el proyecto y los códigos de los ejemplos expuestos.

Figura A. Robot Tracción Diferencial.

Como se muestra en la figura A, el robot consta de 4 ruedas que le permiten desplazarse. Para el control se utilizó la plataforma Arduino la cual permite controlar los drivers ya su vez estos controlan la velocidad de los motores la forma de conexión de los drivers hacia la tarjeta Arduino Mega, tal como se muestra en la figura B.

APÉNDICES

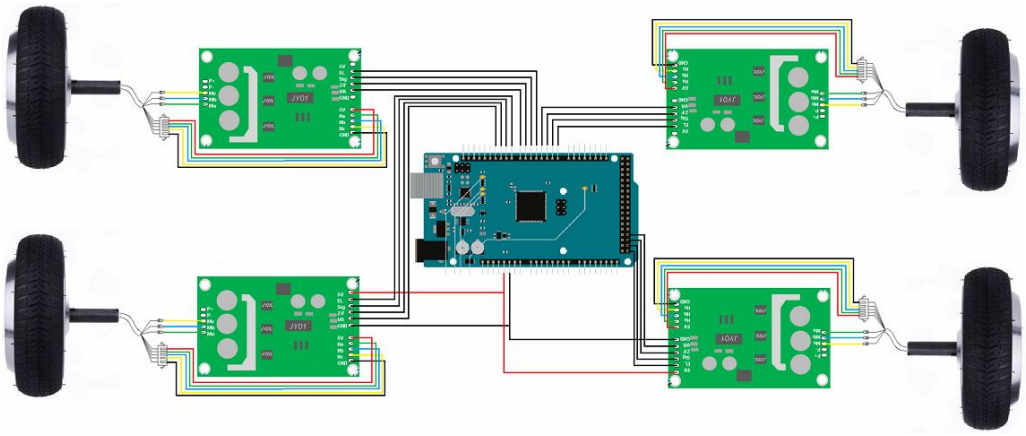


Figura B. Esquema de conexión Electrónico.

Para poder realizar intercambio de datos con el robot móvil, el módulo Arduino se debe conectar mediante un interfaz USB, permitiendo la comunicación con el ordenador en uso, tal como se expone en la figura C.

Figura C. Comunicación serial Arduino - PC.

Una vez realizada la comunicación de Arduino con la PC, en el programa se procede a instalar librerías de Arduino que son para la comunicación con el

framework Ros y utilización de librearías de PID la cuales se pueden apreciar a continuación en la Figura D.

```
#include <PID_v2.h>
#include "ros.h"
#include "ros/time.h"
#include "geometry_msgs/Twist.h"
void commandCallback(const geometry_msgs::Twist& cmd_msg)
```

Figura D. Instalación de las librerías en Arduino.

Se asigna los pines de interrupciones, enable de los motores, control de velocidad y dirección; estos estos pines que controlan de manera individual los drivers para cada motor.

```
const byte interruptA = 18; //Rueda atras izquierda
const byte interruptB = 19; // Rueda atras derecha
const byte interruptC = 20; // Rueda delantera derecha
const byte interruptD = 21; // Rueda delantera izquierda

const byte pinbreakA = 23; //Rueda atras izquierda
const byte pinbreakB = 25; // Rueda atras derecha
const byte pinbreakC = 27; // Rueda delantera derecha
const byte pinbreakD = 29; // Rueda delantera izquierda

const int pinPWMA = 2; //Rueda atras izquierda
const int pinPWMB = 3; // Rueda atras derecha
const int pinPWMC = 4; // Rueda delantera derecha
const int pinPWMD = 5; // Rueda delantera izquierda

const int pindirectA = 24; //Rueda atras izquierda
const int pindirectB = 26;// Rueda atras derecha
const int pindirectC = 32;// Rueda delantera derecha
const int pindirectD = 34;// Rueda delantera izquierda
```

Fig. E. Captura de pantalla de asignacion de variables.

Se asignan las variables para la matriz de transformación, para esto se toma el radio y la distancia entre ejes, tal como se muestran en la figura F y figura G.

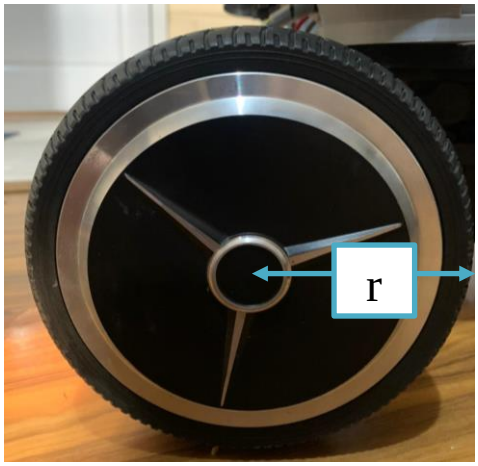


Figura F. Obtención del radio

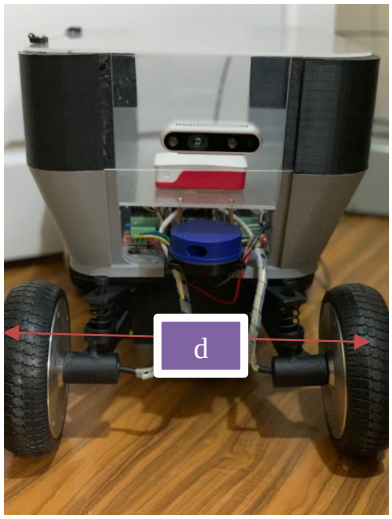


Figura G. Obtención de la distancia entre ejes.

Una vez obtenido el radio y la distancia entre ejes, se asigna en las variables del programa r y d correspondientemente.

```
volatile float u=0;
volatile float w=0;
float r = 0.08;
float d = 0.28;
volatile float w_r ;
volatile float w_l;
```

Figura H. Variables para la matriz de transformación.

Con la configuración de pines y la varibales fijadas, se completa la configuracion de los algortimos de bajo nivel , permitiendo el control mediante un controlador PID independiente para cada rueda controlando asi la velocidad y direccion de los motores se procede a la isntalacion de software ROS .

Instalacion de Software Ros

Para la instalación de ROS se necesita de los siguientes pasos :

1. Se configuran los repositorios de Ubuntu.
2. Se configura la computadora para admitir el software.

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

3. Configuración de llaves.

- sudo apt install curl

4. Se procede a instalar el sistema de Ros deseado.

- sudo apt instalar ros-noetic-desktop

5. Instalación de escritorio: todo en ROS-Base más herramientas como rqt y rviz

- sudo apt instalar ros-noetic-desktop

6. Instalación de configuración de entorno.

- fuente /opt/ros/noetic/setup.bash

7. Finalmente se compila las librerías.

- sudo apt install python3-rosdep

8. Se inicializa ROS.

- sudo rosdep init

Ya teniendo la recepción de datos desde el arduino, se procede a comunicarlo mediante la platafroma de ROS, la cual se ejecuta mediante el paquete rosserial_arduino en el que se puede usar ROS con IDE de Arudino.

"Rosserial" provee de un protocolo de comunicaciones en ROS para trabajar con Arduino, creando un nodo, para obtener parámetros del sistema ROS.

Instalación del software

Primordialmente se debe realizar la instalación de rosserial para sistema Arduino, y luego para descargo del paquete se debe tener en cuenta los siguientes puntos, en el siguiente orden.

1. Accedemos a un terminal y descargamos el paquete.

- sudo apt-get install ros-indigo-rosserial-arduino

2. Descargamos el paquete ejecutando.

- git clone https://github.com/ros-drivers/rosserial.git

3. Compilamos el paquete ejecutando.

- catkin_make install

4. Recompilamos las fuentes en ROS.

- source install/setup.bash

5. Finalmente compilamos las librerías de rosserial.

- rosrun rosserial_arduino make_libraries.py .

Una vez reiniciado el IDE, se busca en la lista:

:

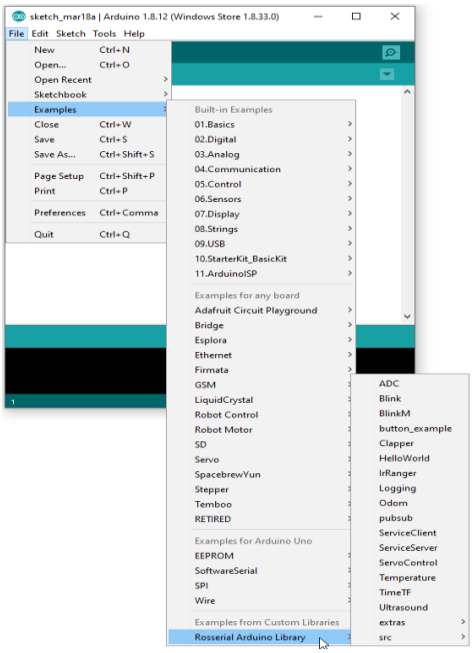


Figura I. Visualización de la librería de ROS en arduino.

Por otra parte, para combinar la odometría, el láser y la imagen RGB-D para la localización, se realiza con el código abierto llamado RTAB-Map. Este dispone de un amplio número de parámetros que se pueden modificar y

ajustar en base a las condiciones en las que se encuentra el robot; a continuación, se da los requerimientos para poder instalar el mismo.

Instalación de RTAB-Map.

Pasos a seguir para descarga del paquete:

1. Descargamos el paquete ejecutando

- sudo apt install ros-noetic-rtabmap-ros

2. Compilamos el paquete ejecutando.

- cd catkin_ws

3. Se descarga desde el repositorio.

- git clone -b indigo-devel https://github.com/introlab/rtabmap_ros.git src/rtabmap_ros

4. S compila el paquete.

- catkin_make -j1

5. Finalmente compilamos las librerías inicializando.

- rtabmap_ros

Ya instalado el RTAB-Map se procede a instalar los paquetes del Sensor YLIDAR y de la cámara Intel Real Sense.

Los paquetes para la instalación de la cámara son preconstruidos para el proceso de instalación se demuestra a continuación:

1. Se agrega el servidor.

- sudo add-apt-repository "deb https://librealsense.intel.com/Debian/apt-repo $(lsb_release -cs) main" -u

2. Se instala las bibliotecas.

- sudo apt-get install librealsense2-dkms

3. Para poder correr el programa se necesita que la cámara este instalada.

- realsense-viewer

Instalación de YDLIDAR SDK

El procedimiento de instalación es el siguiente:

1. Nos situamos, desde un terminal donde vayamos a descargar el paquete.

- sudo apt install cmake pkg-config

2. En el directorio YDLidar SDK, ejecute los siguientes comandos para compilar el proyecto:

```
git clone https://github.com/YDLIDAR/sdk.git
cd sdk/build
cmake ..
make
sudo make install
```

3. Se ejecuta el ydlidar_test para conectar la unidad

- ./ydlidar_test

4. Después de compilar el SDK de YDLidar como se muestra en la sección

- `ydlidar_test.exe`

5. Luego, puede ver que SDK inicializa la información de la siguiente manera:

```
YDLidar SDK initializing
YDLidar SDK has been initialized
[YDLIDAR]:SDK Version: 1.0.0
LiDAR successfully connected
```

Figura J. Inicialización del SDK.

Este sensor se puede conectar al host directamente por serie o a través de la placa adaptadora YDLidar. Cuando la unidad está conectada al host directamente por serial, el host establecerá comunicación con cada unidad individualmente. Y si la unidad se conecta al host a través de la placa adaptadora, entonces el host solo se comunica con la placa adaptadora YDLidar mientras que la placa adaptadora se comunica con cada unidad LiDAR.

Procedimiento para inicializar el Robot

A continuación, se describe el procedimiento para inicializar el robot y cumpla el objetivo preestablecido.

1) Encendido del robot.
   El encendido del robot se lleva a cabo mediante un pulsante que está ubicado en el costado derecho, este a su vez da paso a la alimentación general del robot poniendo listo para cualquier operación, tal como se muestra en la Figura K.

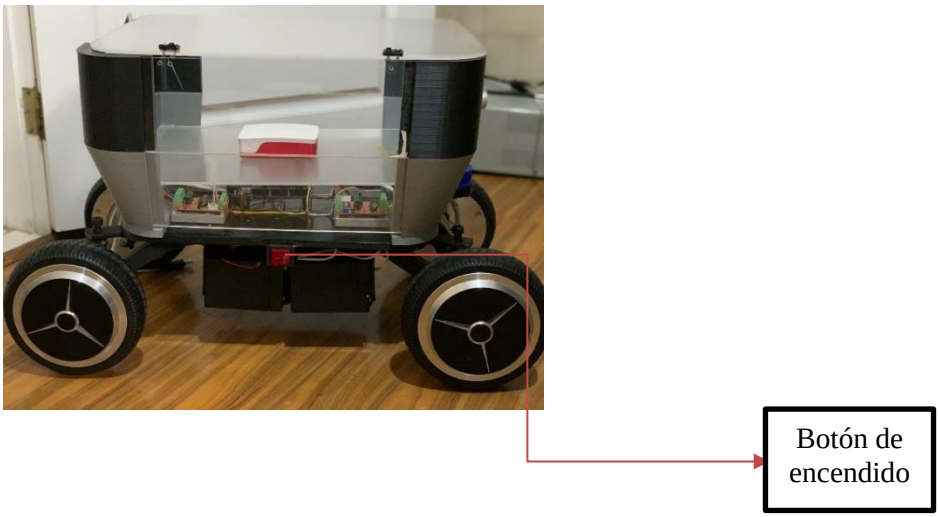


Figura K. Botón de encendido del robot.

2) Inicialización de la trayectoria.
   Como la trayectoria esta preestablecida se debe dar un aviso al robot mediante la inicialización de Ros, esto se lleva a cabo mediante un control haciendo que el robot inicie la trayectoria; aparte este control permite modificar la velocidad y también permite cambiar de modo autónomo a modo manual, para el inicio de trayectoria se acciona el botón de seguridad.

Figura L. Control general del robot.

3) Colocación de la mensajería.
   La colocación de mensajería es mediante una tapa superior la cual se levanta de manera manual para colocar los documentos u otro servicio de correspondencia que se desee transportar.
   Se debe recalcar el que peso máximo es de 35Kg y la dimensiones que tiene es de un formato A4.

APÉNDICES

Figura M. Compartimento de carga.

4) Punto de partida.
   El punto de partida es preestablecido por lo cual el operario coloca la carga en el compartimento e inicializa con el botón de seguridad para que inicie la navegación
5) Entrega de mensajería.
   El momento que llega a su punto de entrega la persona debe retirar la carga del compartimento y accionar nuevamente el botón de seguridad para que regrese al punto de partida.
6) Termino de entrega.
   Finalmente, cuando haya completado la trayectoria el robot, el programa automáticamente entra en modo reposo por lo cual está a la espera de una acción siguiente.

APÉNDICE B

Diseño Mecánico

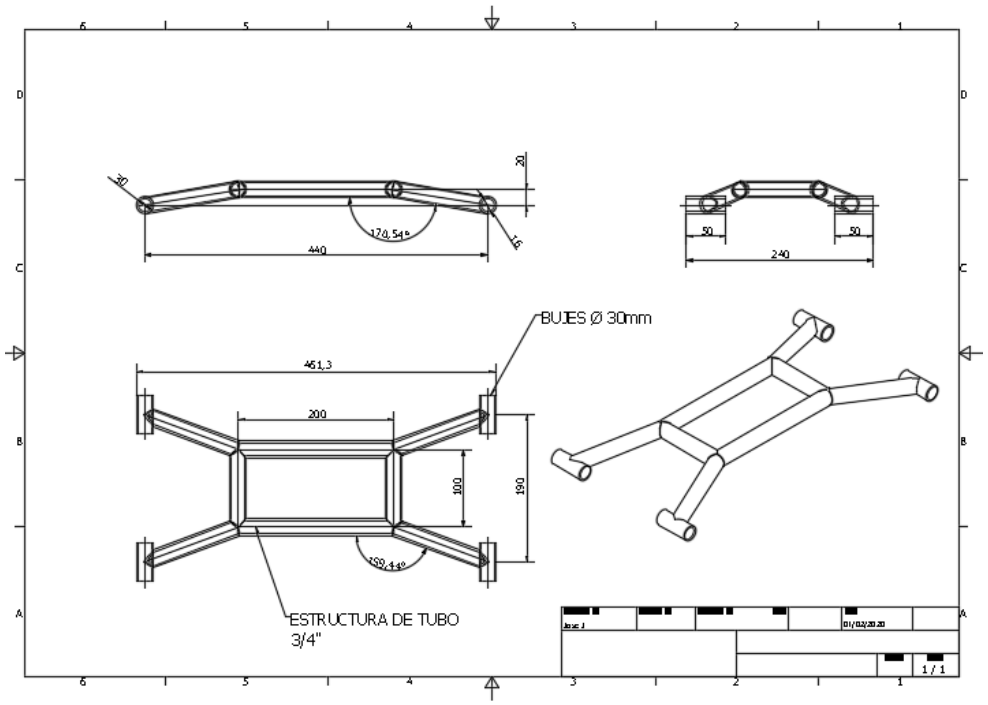


Figura I. Diseño de la Base del Robot para la fabricación.

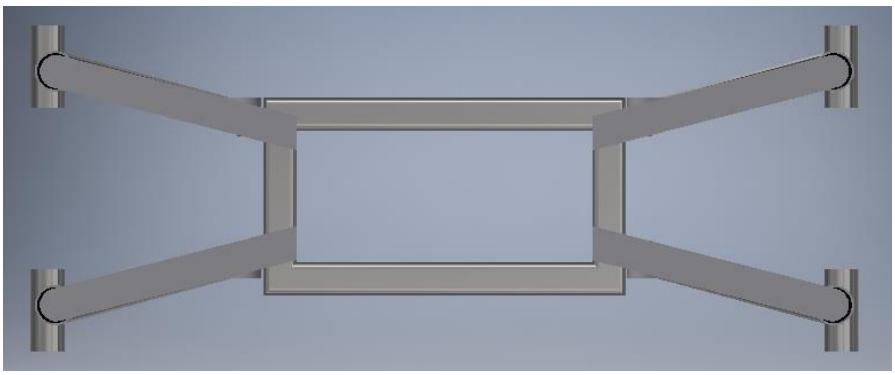


Figura II. Vista superior de la base del Robot.

APÉNDICES

Figura III. Vista lateral de la base del Robot.

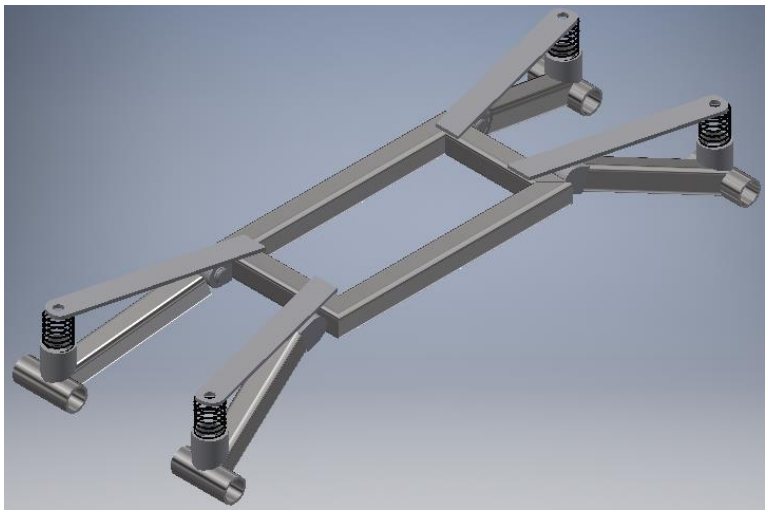


Figura IV. Perspectiva de la base del Robot.

Figura V. Plancha de aluminio para la disipación de calor en los elementos integrados.

APÉNDICES

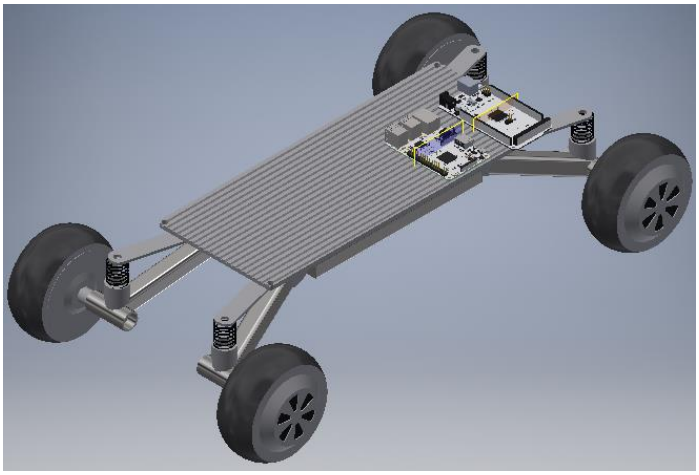


Figura VI. Perspectiva de la plancha de aluminio para la disipación de calor en los elementos integrados.

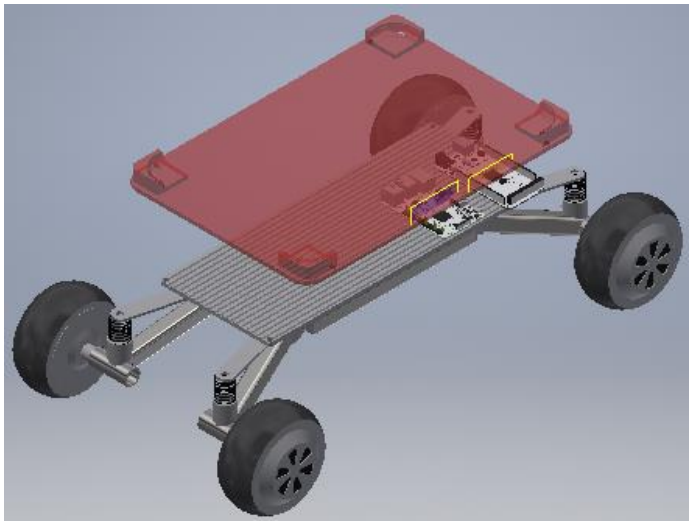


Figura VII. Perspectiva del primer piso falso en donde se alberga la PC.

APÉNDICES

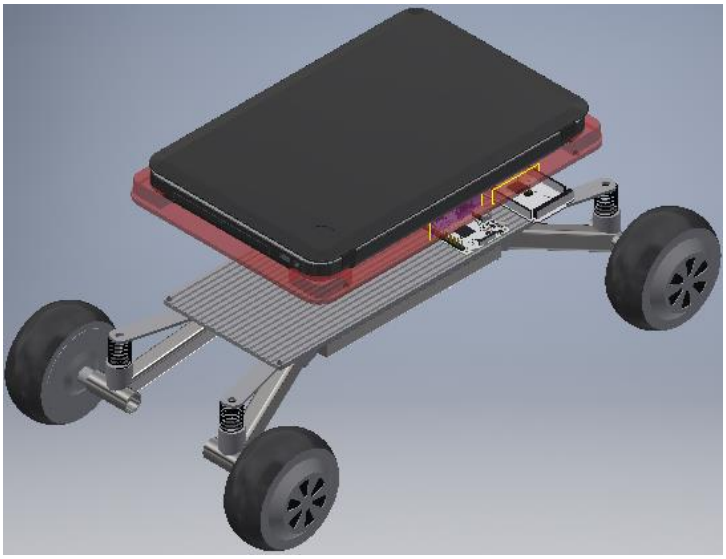


Figura VIII. Piso falso con la PC.

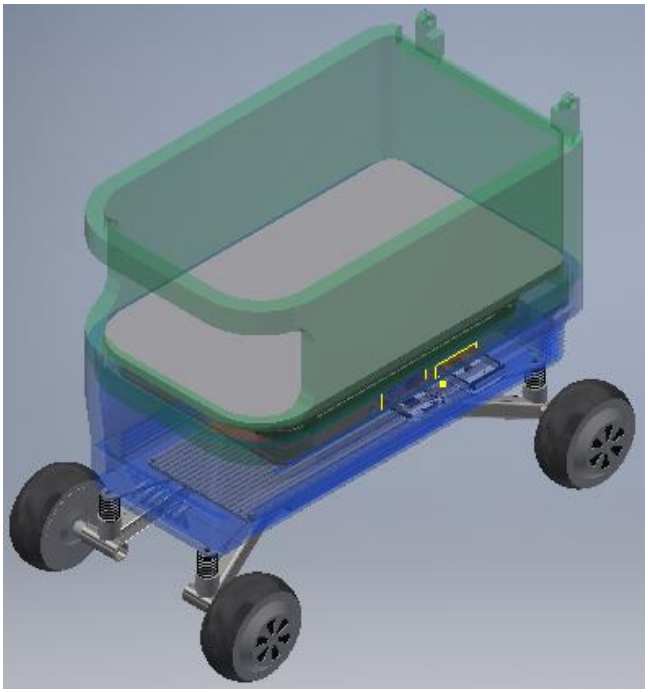


Figura IX. Espacio para transporta elementos.

APÉNDICE C

Proceso de Construcción

Figura X. Proceso de Construcción de la base estructural, bujes de sujeción.

Figura XI. Acople de bujes con las ruedas y la estrcutura.

APÉNDICES

Figura XII. Integración de sistema de amortiguación.

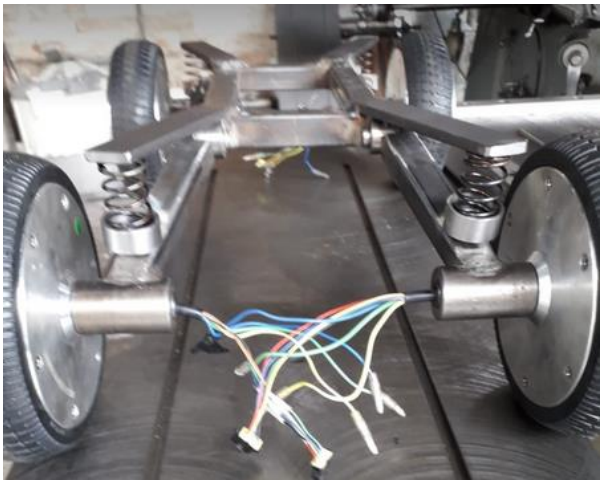


Figura XIII. Integración de sistema de amortiguación.

Figura XIV. Estructura terminada.

APÉNDICES

Figura XV. Impresión 3D y acople de la estructura con la carcasa.

Figura XVI. Impresión 3D de la carcasa.

APÉNDICES

Diseño PCB

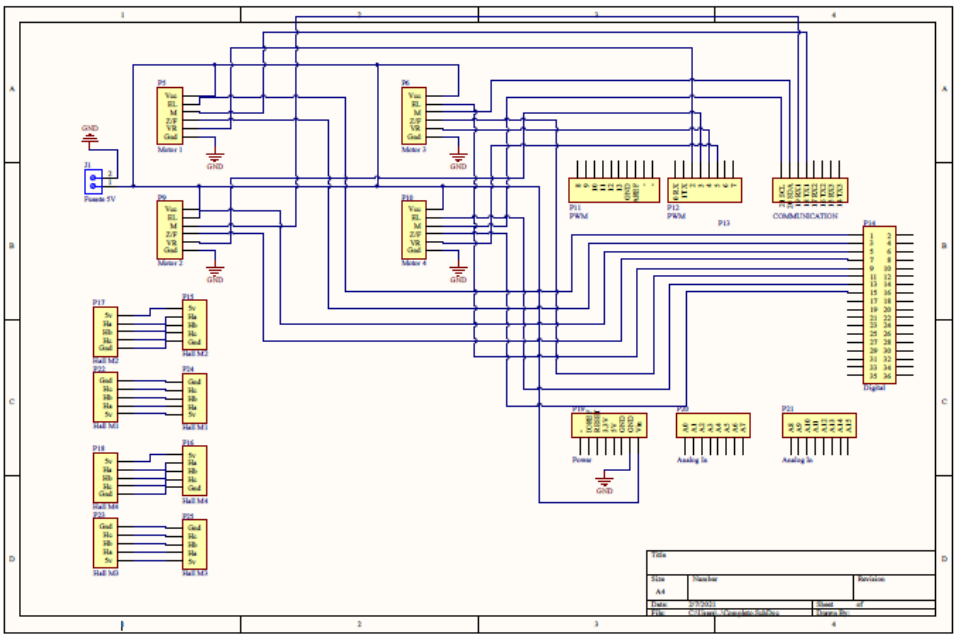


Figura XVII. Esquemático de la Integración de drivers y arduino mega.

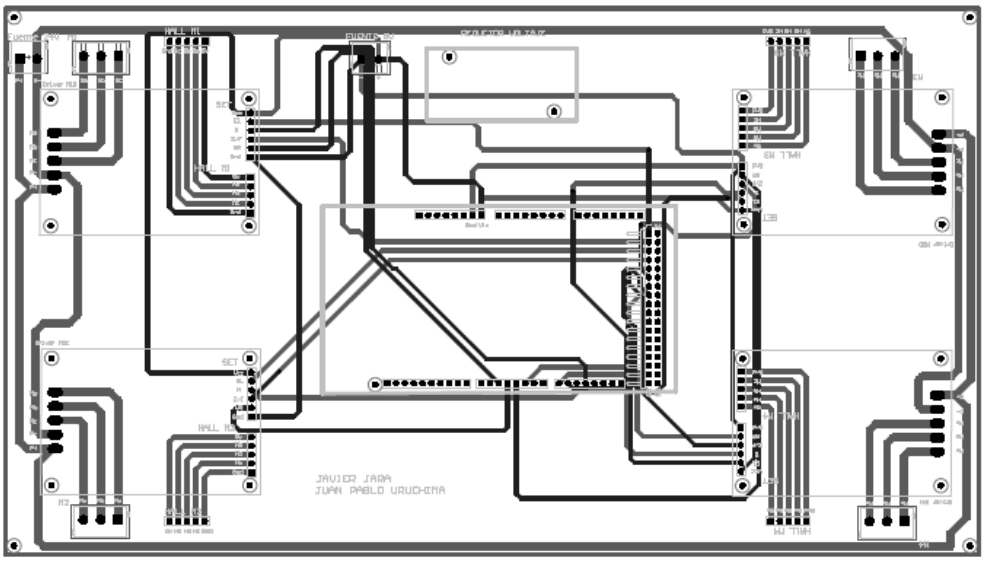


Figura XVIII. Diseño PCB a dos capas.

REFERENCIAS BIBLIOGRAFICAS

[1] I. Bambino, Una Introducción a los Robots Móviles, 2008.

[2] A. De Luca, G. Oriolo y C. Samson, Feedback Control of a Nonholonomic Car-like Robot, Roma, Italia: Springer, ISBN 3-540-76219-1, 1998.

[3] J. Penalva, «Xataca,» webedia, [En línea]. Available: https://www.xataka.com/robotica-e-ia/competencia-glovo-robot-autonomo-que-ira-acera-para-evitar-atascos. [Último acceso: 20 12 2019].

[4] E. Barenholtz y W. Hanh, «Infobae,» [En línea]. Available: https://www.infobae.com/salud/ciencia/2019/10/14/podran-los-robots-de-hoy-ser-las-nuevas-ratas-de-laboratorio-del-futuro/. [Último acceso: 21 12 2019].

[5] T. Mohamed Lamine, A. Ouahiba, H. Mohamed y B. Mohamed, «Mobile Robot Path Planning for Complex Dynamic Environments,» *Institute of Electrical and Electronics Engineers Inc.,* Vols. %1 de %2Saranli U.,Kalkan S., nº 978-146737509-2, p. 7, 2015.

[6] L. H. Rios y M. Bueno, «Modelo Matemático para un Robot Móvil,» Scientia Et Technica, Pereira, Colombia, 2008.

[7] J. C. Montesdeoca, J. M. Toibero y R. Carelli, «Person-Following Robot with Socially Appropriate Motion for Low Human User Velocities,» 10.23919/RPIC.2017.8211621, Mar del Plata, Argentina, 2017.

[8] J. D. Delgado y C. J. Boaños, CONTROL DE VELOCIDAD PARA MOTOR DCBRUSHLESS SIN SENSORES, Bogotá, Colombia:

PONTIFICIA UNIVERSIDAD JAVERIANA, FACULTAD DE INGENIERÍA, 2013.

[9] S. Sahota, «Fictiv,» 04 11 2016. [En línea]. Available: https://www.fictiv.com/teardowns/hoverboard-teardown. [Último acceso: 12 11 2020].

[10] ARDUINO, «FORUM ARDUINO,» [En línea]. Available: https://forum.arduino.cc/t/hoverboard-wheel-synch-with-bldc-controller/684146. [Último acceso: 15 11 2020].

[11] E. D. S. JUYI, «JUYITECH,» [En línea]. Available: http://www.bldcmotor-driver.com/sale-11463978-12v-36v-dc-sensorless-bldc-motor-driver-20-85-o-v-l-v-protection.html. [Último acceso: 18 11 2020].

[12] K. Ogata, Ingeniería de Control Moderna, Prentice Hall, 1970.

[13] C. G. Miguélez, I. O. Benítez, A. M. Rivera y V. M. , «Implementación de Sistema Operativo Robótico en una Plataforma de Robot Móvil,» Universidad Autónoma de Ciudad Juárez, Ciudad Juárez, 2020.

[14] J. Zhu y L. Xu , «Diseño e implementación de un sistema autónomo de navegación y posicionamiento de robots móviles basado en ROS,» IEEE , Wuhan, China, 2019.

[15] B. F. Hernández y J. E. Morocho, «Implementación de SLAM (Simultaneous Localization and Mapping) con un Robot Móvil,» Escuela Superior Politécnica del Litoral, Guayaquil, Ecuador, 2017.

[16] P. Sankalprajan , T. Sharma , H. D. Perur y P. S. Pagala , «Análisis comparativo de algoritmos SLAM 2D y 3D basados en ROS para vehículos terrestres autónomos,» IEEE, Belgaum, India, 2020.

[17] M. Filipenko y I. Afanasyev , «Comparación de varios sistemas SLAM para robots móviles en un entorno interior,» IEEE, Funchal, Portugal, 2019.

REFERENCIAS BIBLIOGRÁFICAS

[18] P. López, «Análisis de algoritmos para localización y mapeado simultáneo de objetos,» Escuela Técnica Superior de Ingeniería Universidad de Sevilla, Sevilla, España, 2016.

[19] Intel, «Intel Real Sense,» [En línea]. Available: https://www.intelrealsense.com/depth-camera-d435/. [Último acceso: 04 12 2020].

[20] J. Musuña, B. Zapata, L. Oñate y G. Composano, «Diseño y construcción de un robot móvil que permita la obtención de una nube de puntos del escaneo de habitaciones utilizando láser y webcams,» INGENIUS, Cuenca, Ecuador, 2014.

[21] M. Auncancela y W. López, «Implementación de Vehículo Autónomo en entorno de simulación usando ROS,» UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ, Quito, Ecuador, 2020.

[22] H. Deilamsalehy y T. C. Havens , «Sensor de localización tridimensional fusionada usando IMU, cámara y LiDAR,» IEEE Xplore, Orlando, Florida , 2016.

[23] R. Ramkumar , S. Adarsh y K. Ramachandran , «Fusión de sensores ultrasónicos y RP Lidar 360 utilizando ANFIS para la navegación de robots móviles,» IEEE Xplore, Coimbatore, India, 2018.

[24] J. Mario y A. F. Moreno, *Diseño de Control Robusto de Velocidad de Motores Brushless para Robótica Aérea*, Bogotá, Colombia, 2010.