UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO

DE INGENIERO EN MECATRÓNICA

TEMA:

"NODO DE ODOMETRÍA DE UN ROBOT MÓVIL A TRAVÉS DEL SISTEMA OPERATIVO DE ROBOTS (ROS)"

AUTOR:

RAMOS ROMERO SEBASTIAN JESÚS

DIRECTOR:

ING. CARLOS XAVIER ROSERO CHANDI, MSc.

IBARRA, 2023

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	216342584		
APELLIDOS Y NOMBRES:	Ramos Romero Sebastián Jesús		
DIRECCIÓN:	Tabacundo		
EMAIL:	sebasraroecuspe@gmail.com		
TELÉFONO FIJO:		**TELÉFONO MÓVIL:**	0994802344

DATOS DE LA OBRA	
TÍTULO:	"NODO DE ODOMETRÍA DE UN ROBOT MÓVIL A TRAVÉS DEL SISTEMA OPERATIVO DE ROBOTS (ROS)"
AUTOR (ES):	Ramos Romero Sebastián Jesús
FECHA DE APROBACIÓN: DD/MM/AAAA	31/07/2023
PROGRAMA:	[X] **PREGRADO** [ ] **POSGRADO**
TITULO POR EL QUE OPTA:	**Ing. Mecatronico**
ASESOR /DIRECTOR:	**ING. CARLOS XAVIER ROSERO, MSc.**

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 31 días del mes de julio de 2023

EL AUTOR:

.....

Nombre: Ramos Sebastián

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

En calidad de director del trabajo de grado **"NODO DE ODOMETRÍA DE UN ROBOT MÓVIL A TRAVÉS DEL SISTEMA OPERATIVO DE ROBOTS (ROS)",** presentado por el estudiante Ramos Romero Sebastián Jesús, para optar por el titulo de Ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, a los 31 días de julio de 2023

Ing. Carlos Xavier Rosero, Msc.

DIRECTOR DE TESIS

AGRADECIEMIENTOS

En primer lugar, quiero agradecer a Dios, quien ha sido mi fortaleza y guía a lo largo de esta travesía académica. Su presencia constante y su apoyo inquebrantable me han dado la fuerza para enfrentar los desafíos y superar los obstáculos.

A mis amados padres, Juan y María, les agradezco infinitamente por el esfuerzo incansable que han dedicado a mi formación. Sin su sacrificio y dedicación, este logro no hubiera sido posible.

A mi querida novia, Génesis, mi mayor motivación y compañera de vida, gracias por creer en mí y alentarme en cada paso del camino. Tu amor y apoyo incondicional han sido el motor que me impulsó a seguir adelante incluso en los momentos más desafiantes.

A mis suegros, quienes siempre me han brindado su apoyo y cariño, agradezco sinceramente su presencia y consejos sabios que han sido de gran valor para mí durante esta etapa.

A mi tutor, cuya dedicación y paciencia han sido fundamentales en el desarrollo de esta investigación. Gracias por compartir su conocimiento y por sacar tiempo de donde no lo había para orientarme y guiarme en el proceso.

A mi amigo Wilson, gracias por estar siempre presente y brindarme tu apoyo sin condiciones y a mis mejores amigos, Marcos, Mishell y Brayan, gracias por ser esas amistades que siempre me impulsaron a ser mejor.

Sebastián J Ramos R

DEDICATORIA

Dedico este trabajo a las personas más importantes en mi vida, mis Padres. Ustedes, quienes me dieron la vida, han sido mi mayor inspiración y apoyo incondicional en cada paso que he dado. Su sabiduría y amor infinito han sido la luz que ha guiado mi camino y me ha dado fuerzas para enfrentar cada desafío.

Gracias por estar siempre a mi lado, por brindarme su apoyo y por ser mi guía en los momentos de incertidumbre. Sus palabras de aliento han sido el motor que me ha impulsado a superarme y alcanzar mis metas.

Y a ti, mi querida novia, Génesis, agradezco por ser la motivación que necesitaba en cada etapa de este camino. Tu presencia y amor incondicional han sido mi mayor fortaleza. Tu apoyo y aliento constante han hecho que cada obstáculo se vuelva más llevadero y cada triunfo sea más significativo.

Con amor y gratitud

Sebastián J Ramos R

RESUMEN

En este trabajo de investigación se presenta la implementación y evaluación de un sistema de odometría para un robot móvil con el objetivo de mejorar la estimación de su posición y orientación durante el desplazamiento. El robot móvil está equipado con cuatro motores DC, cada uno con un encoder para medir la posición y velocidad de rotación. También cuenta con un sensor IMU que registra la orientación, aceleración y rotación en tiempo real.

La implementación del nodo de odometría se llevó a cabo utilizando el sistema operativo ROS (Robot Operating System). Se creó un paquete en ROS llamado "Robot_diff" que incluye dependencias necesarias para el funcionamiento adecuado del nodo. Se emplearon dos técnicas de cálculo odométrico: una basada en el proceso diferencial del movimiento de las ruedas y otra utilizando el filtro de Kalman para fusionar el sensor IMU para mejorar la precisión del control de posición.

El sistema fue verificado y validado mediante pruebas en diferentes entornos, uno con suelo liso y otro con suelo rugoso, para evaluar su desempeño en situaciones de movimiento. Se realizaron pruebas de desplazamiento en línea recta y en un mismo punto para evaluar la precisión y capacidad de respuesta del robot. Además, se llevó a cabo una prueba de consumo de voltaje de los motores para evaluar el rendimiento energético del sistema.

Los resultados obtenidos demuestran que el sistema de odometría implementado proporciona una estimación precisa de la posición y orientación del robot durante el desplazamiento en ambos tipos de superficies. Se logró un control eficiente del robot, permitiendo un movimiento suave y preciso hacia los puntos de destino. Además, se verificó la correcta comunicación entre los sistemas ROS y Arduino, lo que asegura una transmisión de datos confiable para el procesamiento y control del robot.

En conclusión, la implementación del sistema de odometría ha sido exitosa y ha contribuido significativamente a mejorar el rendimiento del robot móvil. Los resultados obtenidos en las pruebas demuestran la viabilidad y eficacia del sistema propuesto. Como posibles mejoras para futuras investigaciones, se sugiere explorar la incorporación de sensores adicionales para una estimación más robusta y realizar pruebas en entornos más complejos y dinámicos para evaluar el sistema en escenarios reales de navegación.

ABSTRACT

In this research work, the implementation and evaluation of an odometry system for a mobile robot aimed at improving the estimation of its position and orientation during movement. The mobile robot is equipped with four DC motors, each with an encoder to measure position and rotational speed. Additionally, it features an IMU sensor that records real-time orientation, acceleration, and rotation.

The odometry node implementation was carried out using the Robot Operating System (ROS). A ROS package called "Robot_diff" was created, including necessary dependencies for proper node functioning. Two odometric calculation techniques were employed: one based on differential wheel movements and another using the Kalman filter to fuse data from different sensors and enhance position control precision.

The system was verified and validated through testing in different environments, one with a smooth floor and another with a rough floor, to evaluate its performance in various movement scenarios. Tests were conducted for straight-line displacement and rotation at a fixed point to assess robot accuracy and responsiveness. Additionally, a motor voltage consumption test was performed to evaluate the system's energy efficiency.

The obtained results demonstrate that the implemented odometry system provides accurate estimation of the robot's position and orientation during movement on both types of surfaces. Efficient robot control was achieved, enabling smooth and precise motion towards the desired points. Furthermore, proper communication between ROS and Arduino systems was confirmed, ensuring reliable data transmission for robot processing and control.

In conclusion, the implementation of the odometry system has been successful and significantly improved the performance of the mobile robot. The results obtained from the tests showcase the feasibility and effectiveness of the proposed system. As potential improvements for future research, it is suggested to explore the integration of additional sensors for a more robust estimation and to conduct tests in more complex and dynamic environments to evaluate the system's performance in real navigation scenarios.

INDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE EECUACIONES

CAPÍTULO I
INTRODUCCIÓN

1.1.PLANTEAMIENTO DEL PROBLEMA

En la Universidad Técnica del Norte se creó un robot móvil para experimentos en base a hardware y software de National Instuments [1]. Hasta el momento el robot posee telemetría, control de velocidad en lazo cerrado y teleoperación. Se desea evolucionarlo hasta conseguir su movimiento autónomo. Si bien la funcionalidad del robot está acorde con los objetivos planteados en el momento de su desarrollo, su avance ha sido afectado debido a la falta de compatibilidad de su hardware y software con el Sistema Operativo de Robots (ROS, Robot Operating System).

ROS es una colección de frameworks para el desarrollo de software de robots. Este es un metasistema operativo que ofrece los servicios estándar de un sistema operativo tales como: la abstracción del hardware, el control de dispositivos de bajo nivel, la implementación de funcionalidad de uso común, el paso de mensajes entre procesos y el mantenimiento de paquetes [2].

La compatibilidad entre el robot móvil mencionado y ROS permitirá la implementación de algoritmos del estado del arte para manejo de odometría, control en lazo cerrado de posición/velocidad angular de las ruedas, planificación y control de trayectorias, manejo de visión artificial, entre otros.

1.2. OBJETIVOS

1.2.1. OBJETIVO GENERAL

Desarrollar el nodo de odometría de un robot móvil dentro de ROS

1.2.2. OBJETIVOS ESPECÍFICOS

- Caracterizar el hardware que permita la implementación de ROS sobre el robot móvil.
- Implementar el nodo de odometría sobre el robot móvil.
- Verificar el desempeño del nodo bajo un escenario específico de trabajo.

1.3. JUSTIFICACIÓN

En la actualidad, existe un interés constante por el desarrollo de sistemas que integren bibliotecas de algoritmos [3]. Dentro de este contexto, ROS es bastante atractivo porque proporciona una serie de servicios y librerías que simplifican considerablemente el desarrollo de aplicaciones complejas para robots [4]. Su uso se amplifica al soportar lenguajes como Python, C++, Java, entre otros.

Este trabajo permitirá que el robot móvil se convierta en un sistema versátil en el que se puedan implementar algoritmos adicionales relacionados con modelos cinemáticos, dinámicos, visión artificial, planificación de rutas, entre otros., dentro de la robótica móvil. En particular, el nodo de odometría, se convertirá en la base para que los algoritmos de control y optimización de trayectorias funcionen.

1.4. ALCANCE

Los componentes desarrollados hasta el momento en el robot móvil en [1] son telemetría, control de velocidad en lazo cerrado y teleoperación. En el presente trabajo se escalará este desarrollo a través de la implementación de un nodo de odometría que funcione dentro de ROS.

Al implementar ROS se reescribirá el código de la odometría para trabajar dentro del paradigma publicador/suscriptor. Los datos que se manejarán corresponden a los de la unidad de medición inercial (acelerómetro y compás), los sensores de velocidad en cada rueda y el sensor de distancia.

CAPITULO II
REVISIÓN LITERARIA

En este apartado se procurará describir cada uno de los conceptos relacionados con el ámbito de odometría usados en ROS, además se resaltará conceptos importantes que permitan fortalecer el conocimiento para el trabajo que se sustentará posteriormente.

2.1. ESTADO DEL ARTE

En [1] se desarrolla el robot móvil sujeto del presente trabajo, con el objeto de que en futuros trabajos se pueda aumentar su funcionalidad. Posteriormente, se implementará el control de velocidad de este robot móvil basado en el estudio de su movimiento diferencial. En la presente investigación se diseña un controlador en lazo cerrado usando el método de Cecil-Smith, que permite seguir referencias de velocidad para cada rueda, además que mejora su estabilidad.

En [5] se presenta la planificación unificada de tareas y movimientos de robots con un planificador ampliado mediante el simulador ROS. Este documento presenta un enfoque híbrido de planificación de rutas, denominado RobMAP (Robotic Motion and Action Planning) para lograr una navegación eficiente realizando las tareas deseadas en una secuencia y, por lo tanto, generar la ruta óptima en un entorno interior a gran escala. Para lograr esto, la planificación de acciones se realiza utilizando el planificador del lenguaje de definición de dominio de planificación (PDDL) a través del marco ROSPlan y la ruta libre de colisiones, se genera utilizando el algoritmo de árbol aleatorio de exploración rápida (RRT) basado en muestreo.

En [6] se presenta el robot TraxBot y su integración completa con ROS. El objetivo de este trabajo es reducir drásticamente el tiempo de desarrollo, proporcionando abstracción de hardware y modos de operación intuitiva, permitiendo a los investigadores centrarse en sus motivaciones principales de investigación, por ejemplo, la búsqueda y rescate con múltiples robots o robótica de enjambres. Se describen las potencialidades del TraxBot, que, combinado con un controlador de ROS específicamente desarrollado, facilita el uso de varias herramientas para el análisis de datos y la interacción entre múltiples robots, sensores y dispositivos de tele operación.

En [7] se resuelve el problema de seguimiento de trayectoria para un robot móvil basado en su modelo cinemático y propone una solución mediante el diseño de una

estrategia de control que a priori toma en cuenta las cotas máximas permitidas de la señal de control i.e. la velocidad lineal y angular máximas que puede alcanzar el robot móvil. El objetivo es maximizar el uso de los actuadores sin poner en riesgo la estabilidad del sistema. La ley de control no lineal resultante se compone de un compensador no lineal basado en el modelo cinemático y de funciones de saturación anidadas. Esta ley de control contiene parámetros de sintonización que permiten que las trayectorias de la dinámica del error ingresen a una vecindad del origen, en un tiempo finito y se mantengan de ahí en adelante.

En [8] se desarrolla un sistema de posicionamiento indoor basado en el concepto de espacios inteligentes. Para ello, se propone un sistema de detección y seguimiento que, en complemento con un sistema de conversión de espacios 2D a 3D, permite obtener la posición de un objeto en el espacio a través de un marcador. El sistema desarrollado abre el camino a estudios biomecánicos empleando captura de movimiento, desarrollo de sistemas de control de trayectorias, reconstrucción geomorfológica, escaneo de piezas mecánicas o partes biomédicas, etc.

2.2. TECNOLOGÍAS IMPLICADAS EN LA SOLUCIÓN

2.2.1. ODOMETRÍA

Se puede definir como la técnica o función que permite la estimación relativa de la posición de un vehículo móvil basándose en el desplazamiento de sus ruedas (dentro de las coordenadas (X, Y) y el ángulo de orientación θ). Se considera relativa porque se dirige mediante un marco de referencia, donde se ve definido el origen de acuerdo con la situación real, así como se muestra en la Fig. 2.1[9].

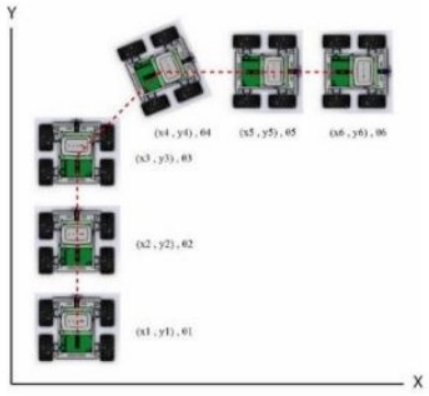


Fig. 2.1. Trayectoria de un Robot móvil [9].

La odometría permite la recopilación de datos provenientes de actuadores y encoder para poder determinar la posición y la velocidad de desempeño del robot desde su posición inicial. Y como se mencionó anteriormente, cada nueva posición se basa en una anterior. El cálculo de la nueva posición se puede obtener fácilmente mediante la siguiente ecuación.

$$d = vt \tag{1}$$

Se debe considerar que la odometría no determina de forma general y exacta la posición del robot, sino que la estima a partir de datos que se obtienen de la lectura de codificadores, de esta manera la odometría permite el surgimiento de errores acumulativos en la medición del desplazamiento.

2.2.2. LOCALIZACIÓN DEL ROBOT

Dentro de todos los métodos usados para poder ejercer una correcta localización del robot se encuentra el odométrico, este sistema permite integrar la trayectoria mediante la estimación de la posición y la orientación basado en el movimiento de las ruedas motrices.

Sin embargo, se debe considerar los errores que se obtienen del actuar de las distintas fuentes, unas están asociadas a sistemas mecánicos y las otras provocadas por deslizamientos. Estos errores pueden provocar incertidumbre en la obtención de datos en cuanto posición y orientación, lo cual provoca que los datos del sistema odométrico no estén acorde con la realidad [9].

2.2.3. CINEMÁTICA DEL ROBOT

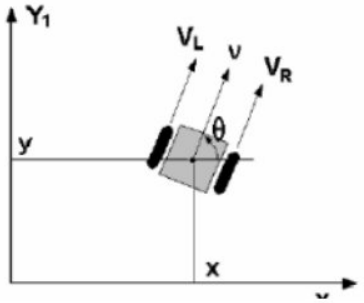
La localización del robot se obtiene desde el modelo cinemático del sistema de tracción, este sistema le permite al robot moverse en un determinado entorno. En la configuración diferencial la posición del robot se puede estimar mediante ecuaciones geométricas que surgen de la relación entre componentes del sistema de tracción y de la información de los codificadores rotativos posicionados en las ruedas.

En la Fig. 2.2 , la localización del robot en el punto (X, Y) donde V es la velocidad lineal del robot móvil y V_L, V_R la velocidad tangencial de las ruedas, brindando la posición del robot expresada en la ecuación

$$Vl = wl.r \quad (2)$$

$$Vr=wr.r \quad (3)$$

Donde wl y wr son velocidades angulares de cada rueda[9]



(5)

Fig. 2.2. Interpretación de movimiento del robot [9]

En cuanto al movimiento en la superficie plana del robot se puede demostrar que es una matriz que representa las relaciones de las posiciones lineales que combina las velocidades de las coordenadas (x, y) expresada en la ecuación 4.

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix} = \begin{pmatrix}-sin\theta & 0\\ cos\theta & 0\\ 0 & 1\end{pmatrix} x \begin{pmatrix}v\\ w\end{pmatrix} \quad (4)$$

2.2.4. CINEMÁTICA DE ACCIONAMIENTO

Para obtener la trayectoria actual del robot se debe monitorear constantemente los codificadores de las ruedas. La distancia recorrida por el robot diferencial [9], se describe por

$$SL = \frac{2\pi \text{ x r x pulsosL}}{\text{PPR}} \quad (6)$$

$$SR = \frac{2\pi \text{ x r x pulsosR}}{\text{PPR}} \quad (6)$$

donde se sabe que *r* es el radio de la rueda, *d* representa la distancia entre ruedas motrices, *PPR* es el número de pulsos.

- r = radio de la rueda
- d = distancia entre ruedas motrices.
- PPR = número de pulsos del codificador en una revolución completa.
- pulsosL = número de pulsos durante la medición izquierda.
- pulsosR = número de pulsos durante la medición del codificador derecho.

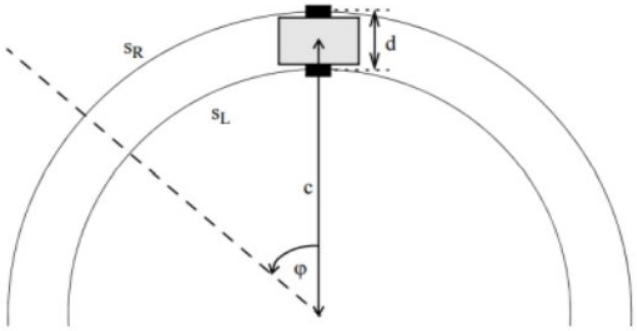


Fig. 2.3. Cálculo de movimiento utilizando los pulsos [9]

La Fig. 2.3 muestra que el proceso a desarrollar es determinar el valor de SL y SR expresado en metros, siendo estos las distancias recorridas por las ruedas tanto izquierda y derecha. La división de los pulsos medidos entre el número de pulsos por revolución nos brinda el número de revoluciones de las ruedas, que multiplicado por la circunferencia de estas nos da la distancia recorrida en metros.

Se puede considerar la siguiente ecuación para el cálculo del recorrido de rotación del robot basado en el trayecto de la rueda izquierda y la rueda derecha, dividido por la distancia transcurrida frente a la posición inicial, donde el ángulo de giro será expresado por:

$$\varphi = \frac{\mathrm{SR} - \mathrm{SL}}{\mathrm{d}} \qquad (7)$$

2.2.5. ESPACIOS INTELIGENTES

Se puede considerar espacios inteligentes a los ambientes o espacios que tienen la capacidad de percibir y entender posibles eventos que ocurran en el interior. Cada uno de

estos espacios se encuentran equipados con sensores, cámaras, ultrasonidos, micrófonos, etc. Siendo los encargados de la percepción de eventos.

Los eventos de cada percepción que realiza el robot son analizados y provocan una reacción del espacio inteligente ante ellos, la cual es comunicada a través de una placa de control que envía tareas para cumplir al robot, y esto es posible gracias al código fuente que controla e interpreta las señales de entrada y da una respuesta como se ve manifestada en la Fig. 2.4 donde se observa como los componentes del robot adquieren información del entorno.[8]

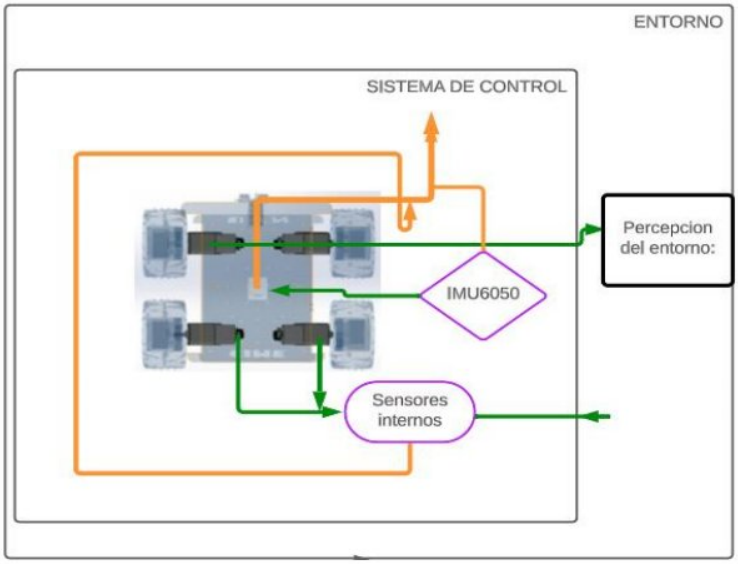


Fig. 2.4 Interpretación de adquisición de datos del entorno

La percepción es dotar al robot de un sistema sensorial capaz de proporcionar suficiente información del entorno para que pueda determinar su localización de manera autónoma, esto se desarrolla dentro del ámbito del espacio inteligente, fortaleciendo la odometría del robot.

2.2.6. SISTEMA DE POSICIONAMIENTO BASADO EN ENCODERS E IMU

Un sistema de posicionamiento basado en encoders e IMU (Unidad de Medición Inercial) combina dos tecnologías para obtener información precisa sobre la posición y el movimiento de un objeto o vehículo. Los encoders proporcionan datos sobre la velocidad y el desplazamiento lineal, mientras que la IMU ofrece datos de la orientación y la aceleración angular. Al fusionar esta información, se obtiene una solución más robusta y precisa para el seguimiento y la localización en aplicaciones como sistemas de navegación, robots móviles y vehículos autónomos[8].

2.2.7. CINEMÁTICA DEL SISTEMA

Existe algunas consideraciones que se debe tener en cuenta en el momento de querer describir el comportamiento del robot, estas son:

- El desplazamiento del robot se desarrolla en superficies planas idealmente sin rozamiento.
- Los ejes de cada rueda están ubicados de forma perpendicular con referencia al suelo por donde se desplaza.
- El robot deberá ser movido solo por la fuerza ejercida por el movimiento rotacional de las ruedas

2.3. ANÁLISIS

A lo largo de esta revisión se ha demostrado que existen estructuras tecnológicas desarrolladas en ROS que mediante placas de control permiten introducir esta tecnología en el mercado industrial y de esa forma, se logra fortalecer cada uno de los medios tecnológicos. Así como en [1] el robot móvil basa su estudio o composición del movimiento diferencial para poder desarrollar el control de velocidad conforme a obtención de datos dirigidos por la velocidad generada por los motores, la dificultad de todos estos estudios es el costo y la no disponibilidad de materiales en el mercado ecuatoriano y como importarlo resulta costoso, esta es la principal razón para frenar investigaciones dentro del ámbito educativo.

Entonces, dado el elevado costo del proceso de investigación comparado con las capacidades económicas de las universidades ecuatorianas, lo que se busca realizar es una ingeniería inversa para poder desarrollar tecnología y poder replicarla en los centros de estudios de Ecuador.

La poca inversión del estado y las universidades impiden desarrollar espacios de investigación, y también impide aprovechar cada conocimiento de los docentes que cuentan con capacidades superiores en temas relacionados con tecnología. Es de esta forma que la investigación y los estudios deben apuntar cada vez más a la capacidad de reducir espacios y costos para el correcto desarrollo en Latinoamérica. Así como en el desarrollo basado en el robot TraxBot y su integración completa con ROS, (ya mencionado antes) que busca el rescate de otros robots mediante una comunicación de

red entre robots, y la interpretación de señales otorgadas por una serie de sensores que permite realizar un complejo análisis de datos.

Todo esto se desarrolla de forma completa y con la inversión de mucho tiempo que consume recursos económicos para poder lograr resultados específicos, razón por la cual se desarrolla la investigación de inserción de ROS en una plataforma móvil ya existente que cuenta con una placa de la marca National Instrument, para realizar el control de movimiento y planeación del mismo, se procura reemplazar esta placa que en la actualidad es un poco costosa y no compatible con muchos sistemas operativos, por una placa Rasperry Pi 4, que nos permitirá realizar la misma acción, reducir los costos y también reducir el espacio de trabajo de la plataforma.

La planeación de cada robot se vuelve fundamental para poder desarrollarse autónomamente, por ello se trabajará en el proceso de implementación del sistema ROS, realizando una biblioteca que permita describir y procesar la edometría del robot, utilizando placas de control que le den mejor autonomía y a menor precio y sobre todo existentes en nuestro mercado tecnológico como es la Rasperry Pi 4 y el Arduino Mega.

El concepto de espacios inteligentes parte de la conversión de los espacios que inicialmente se encuentran en 2D pasarlos a 3D permitiendo que el objeto de estudio sea accesible para el sistema de reproducción de señales odométricas del robot.

CAPITULO III
MARCO METODOLÓGICO
REQUERIMIENTOS Y REQUISITOS DELSISTEMA

En este capítulo se describen cada uno de los componentes mecánicos y eléctricos de la plataforma robótica. Además, se procede a describir los pasos del diseño del código en ROS para el correcto funcionamiento del sistema, explicando cada una de las partes necesarias para programar el código.

3.1. DESCRIPCIÓN GENERAL DEL SISTEMA

El sistema de operación robótico deberá tener la capacidad de reconocer su posición inicial para luego identificar mediante los sensores su entorno de desarrollo, y así planificar trayectos de forma autónoma. Además, la implementación de este sistema deberá permitir a la plataforma móvil leer señales externas, luego generar una determinada acción basada en el control de velocidad para identificar su trayecto y dirección.

3.2. REQUERIMIENTOS DEL SISTEMA

Los requerimientos del sistema se especifican teniendo como base la plataforma móvil ya desarrollada, se incluyen cada uno de los requerimientos específicos del sistema a implementar.

- La estructura del robot debe ser liviana y simétrico entre todos los elementos que le conforman.
- El tamaño de la estructura se reducirá al reemplazar placas de control de tamaño relativamente grande por unos de tamaño considerable.
- Los componentes del robot deberán ser asequibles para su sustitución y adquisición.
- Los motores deben tener el torque necesario para mover el robot.
- El robot desarrollara su funcionamiento en espacios abiertos planos y de preferencia sin irregulares para evitar interrupciones de adquisición de señales.
- El nodo de odometria tendrá la capacidad de controlar al robot mediante planificación.
- El robot deberá en lo posible ser autónomo, por lo que contará con un algoritmo de control específico y baterías indicadas para lograrlo.

3.3. HARDWARE DEL ROBOT

Se vuelve importante el reconocer los componentes físicos que se encuentran en el robot, para poder comprender el comportamiento y la capacidad de desenvolvimiento con el que se podrá desarrollar la plataforma móvil. Se tiene en cuenta cada uno de los componentes desarrollados previamente en [9] donde se los describe detalladamente, en este apartado se expresa de forma general los artículos de composición física del robot.

- Chasis
- Codificadores
- Puente H
- Sensor IMU
- Motores

3.3.1. CHASIS

El chasis fue realizado en material acrílico de un espesor de 5mm mediante práctica de corte en láser como se muestra en la Fig. 3.1 donde consta de una base que permite la sujeción de los motores y baterías y una separación de la base mediante tubos de poste de 25mm y 40mm espacio que permite el acople de los componentes eléctricos y de control[9].

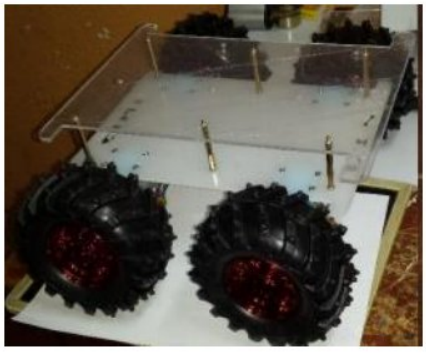


Fig. 3.1. Chasis de la plataforma móvil [9]

3.3.2. CODIFICADORES

Los codificadores se encuentran ubicados en cada motor como se muestra en la Fig. 3.2 donde se manifiesta que cuenta con dos salidas A y B que son ondas cuadradas de 0V a 5Vcc aproximadamente con un desfase de 90º. La frecuencia de las transiciones

muestra la velocidad del motor y el orden de las transiciones indica la dirección. En la Tabla 3.3.2 se da de manifiesto las especificaciones técnicas. [9]

Tabla 3.3.2: Especificaciones técnicas del codificador [9]

CODIFICADOR			
Especificaciones Eléctricas	Tensión de alimentación		5 V
	Corriente		10 mA
Especificaciones Electrónicas	Encoder	Resolución eje del Motor	64 CPR
		Resolución salida caja de cambios	4480 CPR
Especificaciones Físicas	Tamaño		37 D x 70Lmm
	Peso		225 g
	Diámetro del eje		6 mm

Fig. 3.2: Codificador [9]

3.3.3. PUENTE H

En la Fig. 3.3 se muestra un controlador que contine dos canales con modo de trabajo en puente H utiliza el L298N como chip principal con una capacidad de manejar un motor paso a paso de 2 fases, o dos motores con corriente continua que es el caso de esta plataforma móvil, el suministro de energía es un total de 5V (Detallado en la Tabla 3.3.3 donde se manifiesta las especificaciones técnicas.

Fig. 3.3: Controlador de motores [16]

Tabla 3.3.3 Especificaciones técnicas de DRIVER L298D [9]

Driver L298D		
Especificaciones Eléctricas	Voltaje lógico	5 V
	Corriente lógica	0 mA - 36 m A
	Temperatura	-20 ºC +135 ºC
	Modo de operación	puente - H
	Corriente operación	2 A
	Voltaje operación	5 V - 35 V
	Potencia máxima	25 W
Especificaciones Físicas	Dimensiones	43x43x27mm
	Peso	24 g

3.3.4. PLACA RAPERRY PI 4

Contiene un procesador ARM con cuatro núcleos capaz de desarrollarse hasta los 1.5 Hz, cuenta con una GPU con la capacidad de decodificar un video con capacidad 4K hasta 60 fotogramas por segundo. Complementándose con una conexión de wifi dual de 2.0, además con una conexión de bluetooth 5.0. Esta versión necesita una conexión de memoria de almacenamiento externa debido a que la placa no cuenta con una, pero si resalta la memoria RAM de 4gb que viene incluida las especificaciones técnicas se desarrollan de manera completa en la Tabla 3.3.4.[10]

Tabla 3.3.4 Especificaciones técnicas de placa Rasperry PI 4 [10]

Componente	Especificaciones
Voltaje de funcionamiento	5v
Procesador	ARM Cortex-A72
Frecuencia de reloj	1,5 GHz
GPU	VideoCore VI (con soporte para OpenGL ES 3.x)
Memoria	4 GB LPDDR4 SDRAM
Conectividad	Bluetooth 5.0, Wi-Fi 802.11ac, Gigabit Ethernet
Puertos	GPIO 40 pines, 2 x micro HDMI, 2 x USB 2.0, 2 x USB 3.0, CSI (cámara Raspberry Pi), DSI (pantalla táctil), Micro SD, Conector de audio jack, USB-C (alimentación)

3.3.5. ARDUINO MEGA

La placa Arduino mega es conocida por su gran cantidad de pines digitales y pines de entrada analógica, lo que la hace adecuada para proyectos que requieren una amplia capacidad de E/S. Cuenta con una memoria flash considerablemente buena con un reloj de 16MHz. En la Tabla 3.3.5 se expresa las características principales que sobresalen de esta placa y la razón por la cual se le considera para este proyecto [11].

Tabla 3.3.5 Especificaciones técnicas Arduino Mega [11]

Componente	Especificaciones
Microcontrolador	ATmega2560
Pines digitales	54 (15 de ellos con salidas PWM)
Pines de entrada analógica	16
Memoria flash	256KB
SRAM	8KB
EEPROM	4KB
Velocidad del reloj	16MHz
Voltaje de funcionamiento	5V
Tamaño	101.52 mm x 53.3 mm

3.3.6. SENSOR IMU

En la Fig. 3.4 se muestra el MPU6050 que es un módulo de sensores inerciales (IMU) que combina giroscopio y acelerómetro de triple eje. Proporciona seis grados de movimiento e inercia con interfaz I2C, resolución ajustable, procesador de movimiento digital, bajo consumo de energía y uso en robótica, drones, realidad virtual, entre otros. Versátil y preciso[12]

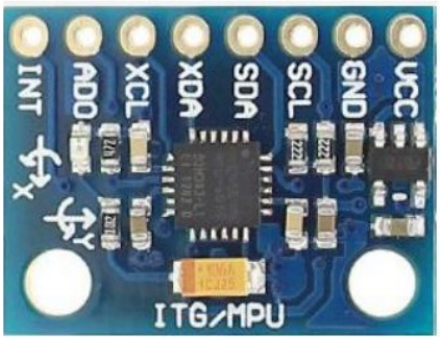


Fig. 3.4. MPU 6050 [13]

En un 6DOF, el acelerómetro mide la aceleración lineal en tres ejes (X, Y, Z), mientras que el giroscopio mide la velocidad angular en esos mismos tres ejes. Estos seis grados de libertad permiten obtener información sobre el movimiento y la orientación del dispositivo en el espacio tridimensional. En la Tabla 3.3.6 se detalla las especificaciones. [13]

Tabla 3.3.6 Especificaciones técnicas MPU 6050[13]

UNIDAD DE MEDICION INERCIAL IMU			
Especificaciones Eléctricas	Tensión de alimentación		3.3 V
	Giroscopio	ITG - 3200	3 ejes
	Acelerómetro	ADXL345	3 ejes
	Magnetómetro	No incluido	-
	Comunicación		Serial 232
	Procesamiento		ATmega328
	Frecuencia de trabajo		400 kHz
	Velocidad de transmisión de datos		400 kbps
Especificaciones Físicas	Dimensiones		15x21 mm

3.3.7. MOTORES

Se ha seleccionado un motorreductor potente de escobillas de 12V DC con una relación de 70:1 en la caja de engranajes de metal. Este motorreductor cuenta con un codificador de cuadratura integrado que proporciona una resolución de 64 pulsos por revolución del eje del motor, lo que corresponde a 4480 pulsos por revolución del eje de salida de la caja de cambios. [9]

La selección de estos motores se consideró funcional, ya que previamente se realizó un proceso de selección utilizando la técnica heurística. Este estudio previo se considera válido y confiable, por lo que se ha decidido utilizar los motores seleccionados en este trabajo de investigación en la Tabla 3.3.7 se muestra las especificaciones técnicas necesarias.

Tabla 3.3.7 Especificaciones técnicas motorreductor [9]

MOTORREDUCTOR POLOLU 70:1		
Especificaciones Eléctricas	Tensión de alimentación	12 V
	Corriente sin carga	300 mA
	Corriente máxima	5000 mA

Especificaciones Electrónicas	Encoder	Resolución eje del motor	64 CPR
		Resolución salida caja de cambios	4480 CPR
Especificaciones Mecánicas	Relación de transmisión		70;1
	Velocidad		150 RPM
	Torque		200 oz in
	Peso		225 g
	Diámetro del eje		6 mm

3.4. SUBSISTEMAS DEL ROBOT

Los subsistemas de un robot son componentes o conjuntos de componentes que desempeñan funciones específicas y se interrelacionan entre sí para lograr el funcionamiento global del robot. Cada subsistema cumple un rol crucial en el sistema completo y contribuye a su desempeño general. En la Fig. 3.5 se presenta las divisiones básicas del proyecto.[14]

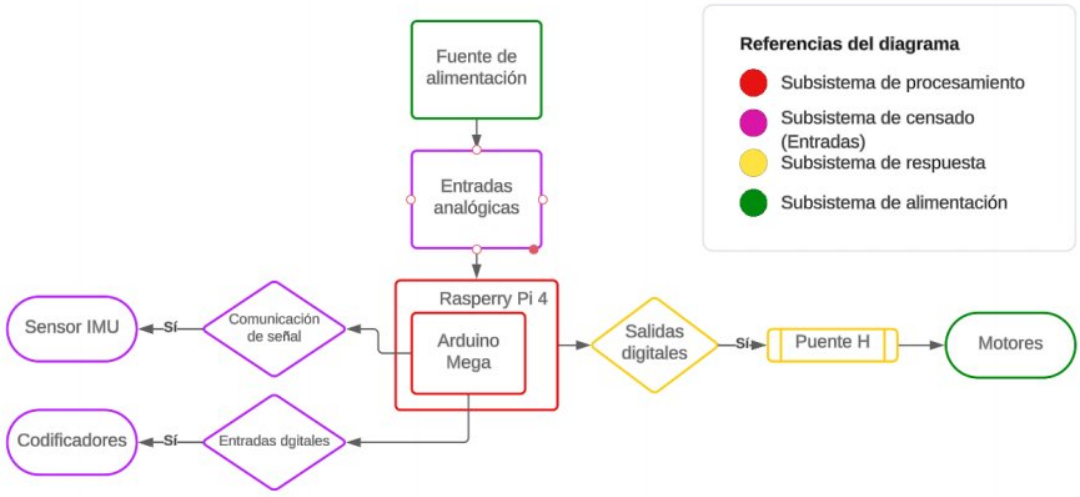


Fig. 3.5. Descripción de sistema y subsistema de la plataforma móvil

3.4.1. SUBSISTEMA DE PROCESAMIENTO

Las placas de control son responsables de procesar y controlar en general todo el sistema. Estas placas ejecutan el software y algoritmos necesarios para el funcionamiento del robot o del sistema. En este caso, se fusiona el funcionamiento de dos placas: el Arduino Mega y la Raspberry Pi 4.

El Arduino Mega tiene la capacidad de recibir señales y enviar comandos a otros componentes del sistema. Actúa como una interfaz entre los diferentes dispositivos y

sensores del robot, recopilando información del entorno y enviándola a la Raspberry Pi 4.

Por su parte, la Raspberry Pi 4 actúa como la placa de control principal y la unidad de procesamiento central. Se encarga de almacenar y ejecutar el software principal del sistema utilizando el marco de trabajo ROS (Robot Operating System). La Raspberry Pi 4 realiza el procesamiento de datos, toma decisiones y coordina las acciones del robot en base a la información recibida del Arduino Mega y otros dispositivos.

3.4.2. SUBSISTEMA DE CENSADO

En este apartado, podemos identificar las entradas analógicas encargadas de recopilar datos o señales analógicas del entorno. Entre estas entradas se encuentran los codificadores y el sensor IMU (Unidad de Medición Inercial) [15].

Estos sensores analógicos son fundamentales para obtener información precisa sobre el entorno y el estado del robot. Los datos recopilados por los codificadores y el sensor IMU son utilizados por el subsistema de procesamiento para tomar decisiones y controlar el comportamiento del robot en función de la posición, el movimiento y las fuerzas externas.

3.4.3. SUBSISTEMA DE RESPUESTA

Las salidas digitales del sistema permiten la comunicación de información y la ejecución de acciones físicas en respuesta a eventos o decisiones del sistema de procesamiento. El puente H es el componente encargado de controlar y direccionar la corriente hacia los motores, lo cual habilita al sistema de respuesta para controlar y ajustar el movimiento de los motores según sea requerido.

3.4.4. SUBSISTEMA DE ALIMENTACIÓN

El subsistema de alimentación proporciona la energía necesaria para el funcionamiento del sistema. Incluye fuentes de alimentación como baterías o adaptadores de corriente. Los motores, que forman parte de este subsistema, son responsables de generar el movimiento físico del sistema en respuesta a las instrucciones del subsistema de procesamiento y la retroalimentación del subsistema de censado.

3.5.PAQUETES NECESARIOS DE ROS

En el desarrollo de sistemas robóticos utilizando ROS (Robot Operating System), se utilizan varios paquetes que proporcionan funcionalidades específicas. Estos paquetes son módulos de software que contienen conjuntos de bibliotecas, controladores y herramientas para facilitar el desarrollo y la comunicación entre los componentes del sistema robótico.[2]

Los paquetes de ROS abarcan una amplia gama de áreas, desde la comunicación y la percepción hasta el control y la planificación de movimientos. Algunos de los paquetes usados dentro del proyecto se describen a continuación.

1. **rospy:** Este paquete proporciona una biblioteca cliente para Python que permite interactuar con el sistema ROS. Proporciona funciones para crear nodos, publicar y suscribirse a tópicos, llamar a servicios y realizar otras operaciones básicas en ROS.
2. **roslib:** Este paquete proporciona funciones y clases adicionales para trabajar con ROS en Python. Incluye herramientas para cargar y descubrir paquetes, acceder a mensajes y servicios, y realizar otras operaciones relacionadas con la biblioteca ROS.
3. **nav_msgs:** Este paquete contiene mensajes relacionados con la navegación de robots. Algunos de los mensajes utilizados incluyen 'Odometry', que proporciona información de la odometría del robot, y otros mensajes relacionados con mapas, trayectorias y planificación de ruta.
4. **geometry_msgs:** Este paquete contiene mensajes para describir geometría y transformaciones en el espacio tridimensional. Los mensajes utilizados incluyen 'Twist', que representa las velocidades lineales y angulares, y 'Pose2D', que representa una posición en el plano XY con una orientación en el eje Z.
5. **std_msgs:** Este paquete contiene mensajes estándar utilizados en ROS. Algunos de los mensajes utilizados incluyen 'Int16', que se utiliza para representar números enteros de 16 bits, y otros mensajes como 'String', 'Float32' y 'Bool'.
6. **tf:** Este paquete proporciona herramientas para trabajar con transformaciones entre marcos de referencia en ROS. Incluye funciones para realizar conversiones entre cuaterniones y ángulos de Euler, así como para publicar y recibir transformaciones entre diferentes marcos de referencia.

Los paquetes de ROS son componentes esenciales en el desarrollo de sistemas robóticos y proporcionan las herramientas necesarias para construir aplicaciones sofisticadas y funcionales. Su versatilidad y flexibilidad los convierten en una opción poderosa para la implementación de una amplia gama de proyectos robóticos, desde robots móviles simples hasta sistemas complejos de navegación autónoma y manipulación de objetos.

CAPITULO IV

RESULTADOS Y ANALISIS

En este capítulo, se presentarán los resultados detallados de cada uno de los objetivos específicos, incluyendo la caracterización del hardware, la descripción de la implementación del nodo de odometría y el análisis del desempeño del nodo bajo el escenario de prueba establecido. Además, se discutirán los hallazgos y se ofrecerán conclusiones sobre el cumplimiento de los objetivos establecidos, destacando las contribuciones y posibles mejoras para futuras investigaciones.

4.1. CARACTERIZACIÓN DEL HARDWARE

El robot móvil está equipado con cuatro motores DC, cada uno de ellos está conectado a un encoder que permite medir la posición y velocidad de rotación de los motores. Estos encoders proporcionan información precisa sobre el desplazamiento del robot, lo cual es esencial para la estimación de la odometría. Además, el robot cuenta con un sensor IMU, que combina acelerómetros, giroscopios y magnetómetros para medir y registrar la orientación, aceleración y rotación del robot en tiempo real.

La placa Raspberry Pi 4 se utiliza como el cerebro del robot, ejecutando el sistema operativo ROS y controlando las operaciones y comunicaciones del robot. Por otro lado, el Arduino Mega actúa como un controlador para los motores y recopila la información proveniente de los encoders. Estas dos placas trabajan en conjunto para asegurar un control preciso y eficiente del robot. Como se describe de forma gráfica todo el sistema en la Fig 4.1 donde mediante bloques se trata de explicar la lógica antes mencionada.

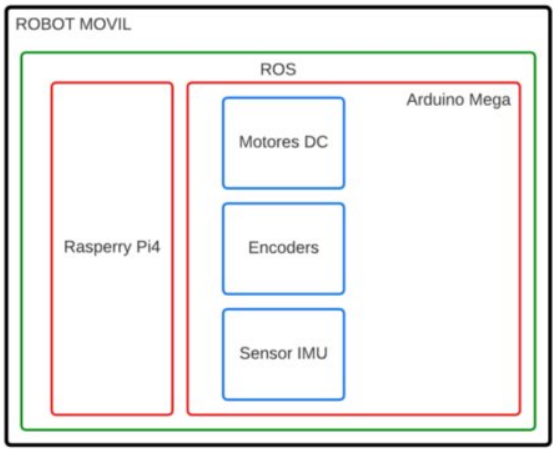


Fig. 4.1: Interpretación de la caracterización del software

4.2. IMPLEMENTACIÓN DEL NODO DE ODOMETRÍA

En esta sección, se proporcionará una descripción detallada de la implementación del nodo de odometría sobre el robot móvil. Se explicarán los pasos y procesos llevados a cabo para calcular la odometría, asegurando un funcionamiento preciso y confiable del sistema.

4.2.1. DESCRIPCIÓN DETALLADA DE LA IMPLEMENTACIÓN DEL NODO DE ODOMETRÍA

A) Condiciones iniciales

Para asegurar una ejecución exitosa del nodo de odometría, se verificaron las compatibilidades de los sistemas utilizados. Se decidió trabajar con ROS Noetic y Ubuntu 20.04 LTS, garantizando una configuración estable. Se siguieron los procedimientos recomendados por la wiki de ROS para instalar el sistema y sus paquetes relacionados.

B) Creación del ambiente de trabajo ROS

Se procede a implementar un entorno de trabajo en ROS para poder gestionar el correcto funcionamiento de los paquetes, nodos, mensajes y servicios del proyecto robótico. El proceso para crear el entorno de trabajo incluyó:

1. Creación de una carpeta denominada **"Entorno_robot"** y una subcarpeta llamada **"src"** para poder incluir el paquete del proyecto.
2. Ejecución del comando **"catkin_make"** en la terminal para compilar y construir el espacio de trabajo.
3. Para la correcta configuración del archivo se ejecutó ". bashrc" esto para que el este entorno de trabajo esté disponible en todos los terminales.

C) Creación del paquete en ROS

En la carpeta **"src"**, se creó un paquete en ROS llamado **"Robot_diff"** para el desarrollo del nodo de odometría. Este paquete incluye las dependencias necesarias para su funcionamiento adecuado, como bibliotecas de ROS y paquetes relacionados con la navegación y estimación de estado.

La selección y configuración de estas dependencias se realizó basados en las recomendaciones y guías de toda la comunidad de ROS presente en la web. Estas

dependencias incluyen bibliotecas estándar como **"std_msgs"** para la comunicación entre nodos, y bibliotecas específicas como "rospy" para la escritura de nodos en Python y "roscpp" para la escritura de nodos en C++.

Además, se incluyeron paquetes relacionados con la navegación y la estimación de estado, como **"move_base_msgs"** y **"nav_msgs"**, que proporcionan mensajes y servicios utilizados en la planificación de rutas y el control del movimiento del robot. También se utilizó el paquete **"robot_pose_ekf"** para implementar un filtro de estimación de estado extendido (EKF) que fusiona la información de la odometría y el sensor IMU.

D) Técnicas de cálculo y control de la odometría en ROS

En esta etapa se consideró dos técnicas de cálculo odometrico como el de odometria basado en el proceso diferencial del actuar del robot y el filtro de Kalman siendo este un algoritmo de estimación que permite fusionar datos con mejor precisión provenientes de distintos sensores sobre el control de posición.

La odometria basado en el proceso diferencial describe el movimiento de las ruedas para poder estimar el desplazamiento y la orientación de la plataforma robótica, para lograr esto se usa el enconder ubicados en cada rueda que es quien brinda los datos de posición. Esto es posible calcular mediante el número de rotaciones que da la rueda multiplicado por la circunferencia de esta.

$$Desplazamiento = RX * CX \quad (8)$$

RX = Numero de rotaciones de la rueda

CX = Circunferencia de la rueda

El algoritmo de Kalman está basado en dos etapas principales: la etapa de predicción y la etapa de actualización. En la primera, se utiliza el modelo matemático del sistema y las mediciones previas para estimar el estado futuro del sistema. Esto es posible mediante la propagación del estado inicial y la matriz de covarianza a través de un modelo de transición.

Mientras que, en la segunda etapa de actualización, se incorporan los datos actuales del sensor para corregir y fortalecer la estimación del estado. Este trabajo se

desarrolla mediante el cálculo de la innovación, que basa su análisis en la diferencia de la medición real y la predicción del estado.

Estos datos dan paso al cálculo de la expresión ganancia de Kalman, que determina qué tan confiable es la medición en comparación con la predicción. Finalmente, se actualiza el estado estimado y la matriz de covarianza utilizando la ganancia de Kalman y la innovación.

4.3. DIAGRAMAS DE CONTROL DEL NODO

Cada diagrama permite controlar un aspecto del nodo de odometria desde la obtención de datos hasta la publicación de la respuesta que debe dar el sistema frente a un entorno determinado. Estos diagramas muestran la interacción de diferentes mensajes y suscripciones que se presenta en ROS.

4.3.1. DIAGRAMA DE CONTROL EN ARDUINO

La secuencia de funcionamiento de los eventos comienza con la recepción de datos de velocidad lineal y angular alojados en la función **"cmd_vel_cb"** para luego permitir la actualización de valores. Dentro de un proceso de comprobación se verifica que el tiempo transcurrido para la primera acción sea de por lo menos 10ms desde su última actualización, si este es el caso se calcula la diferencia de los encoders y se actualizan las variables de control (**left_setpoint** y **right_setpoint**) dando luz verde para ejecutar los controladores PID (**leftPID.Compute**() y **rightPID.Compute**()) para luego aplicar las salidas a los motores (**left.rotate**(left_output) y **right.rotate**(right_output)).

Por último se llama a la función (**publishPos()**) para poder publicar la posición de los encoders esto permite que ROS pueda procesar los mensajes con la función (**nh.spinOnce()**).

Todos estos pasos de forma secuencial se ven detallados en la Fig. 4.2 donde se describe mediante un diagrama de bloques cada uno de los pasos que se ejecutan dentro del código para poder recibir las señales y enviarlas al cerebro central que actúa directamente con ROS.

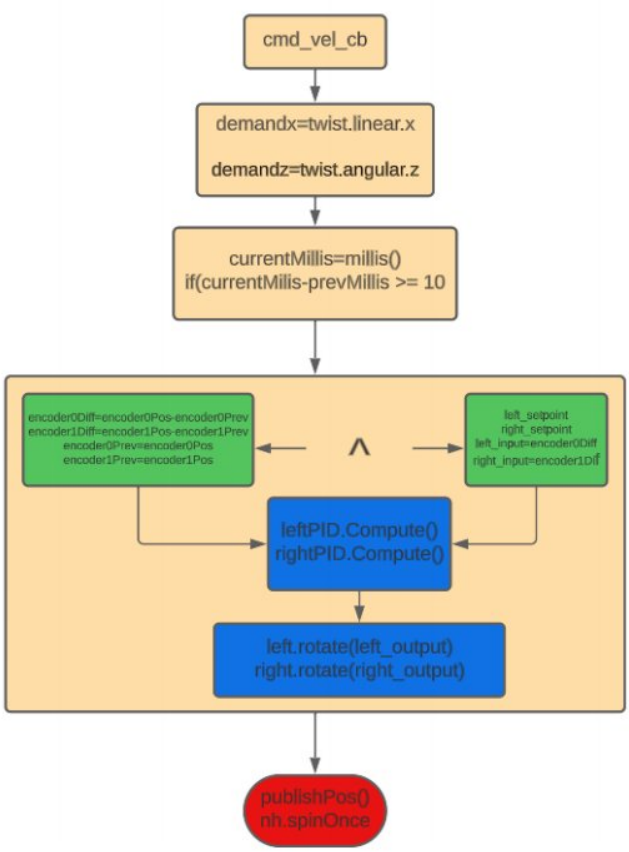


Fig. 4.2. Diagrama de flujo del código de Arduino

4.3.2. CONTROL DE CÁLCULO DE DISTANCIA LINEAL Y ANGULAR

En este diagrama se presenta la secuencia de llamados de los métodos en la clase **DiffTf.** Se inicia con el método (**spin** ()) que es la entrada principal del programa, esta llama al método (**update** ()) en la realización de un bucle hasta que se cierre el nodo de ROS. El método (**update** ()) es quien se encarga de actualizar la odometria y publicar los mensajes correspondientes.

Los métodos **lwheelCallback()** y **rwheelCallback()** son callbacks para las suscripciones realizadas en **"lwheel"** y **"rwheel"** respectivamente. Se usa en el código los quaternios para una representación tridimensional.

En este código se representan las distancias recorridas de las ruedas tanto izquierda como derecha y se calcula la diferencia actual y anterior de los encoders mediante la función (**self.ticks_meter**) que representa el número de ticks del encoder por metro recorrido. Luego calcula la distancia como promedio, para luego calcular las distancias recorridas en las direcciones X e Y utilizando el ángulo **th** que es el cambio de orientación. Estos valores se utilizan para actualizar las coordenadas **self.x** y **self.y** del

robot en el plano XY. Se puede reforzar la comprensión con la Fig. 4.3 donde se describe los principales métodos y la secuencia en que interactúan.

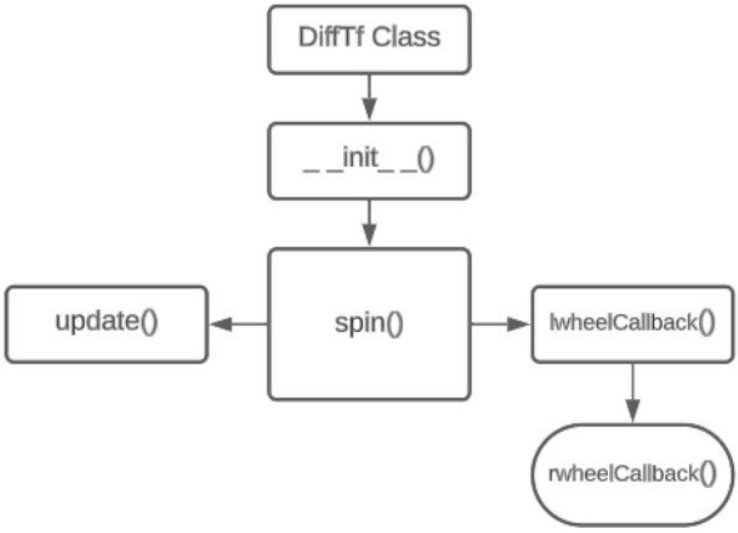


Fig. 4.3. Diagrama de flujo del código de control de distancias y publicación de la odometría.

4.3.3. CONTROL DE CÁLCULO DE VELOCIDAD LINEAL Y ANGULAR

Estando ubicados en la clase ControlBot, el método **move2goal()** es el punto de entrada principal del programa permitiendo al usuario ingresar coordenadas de destino para luego calcular la velocidad lineal y angular necesaria para alcanzar la meta. El objeto Twist permite publicar las velocidades en el tópico **/cmd_vel** y realizar el movimiento hacia la meta esto continúa funcionando hasta que llegue a la meta o se interrumpa la acción.

La función **"euclidean_distance"** calcula la distancia euclidiana entre la posición actual y la meta, mientras que **"steering_angle"** determina el ángulo de dirección hacia la meta. La función **"linear_vel"** utiliza la distancia euclidiana para calcular la velocidad lineal del robot, multiplicando la distancia por una constante ajustable. Del mismo modo, la función **"angular_vel"** calcula la velocidad angular del robot utilizando el ángulo de dirección y ajustándola según una constante. Estas constantes permiten ajustar la velocidad máxima lineal y angular del robot para adaptarse a los requisitos del sistema y las limitaciones del robot. Al ajustar adecuadamente estas constantes, se logra un movimiento suave y preciso del robot hacia su objetivo. En la Fig. 4.4 se da por manifiesto que al realizar el proceso principal se reinicia el proceso y vuele al estado inicial.

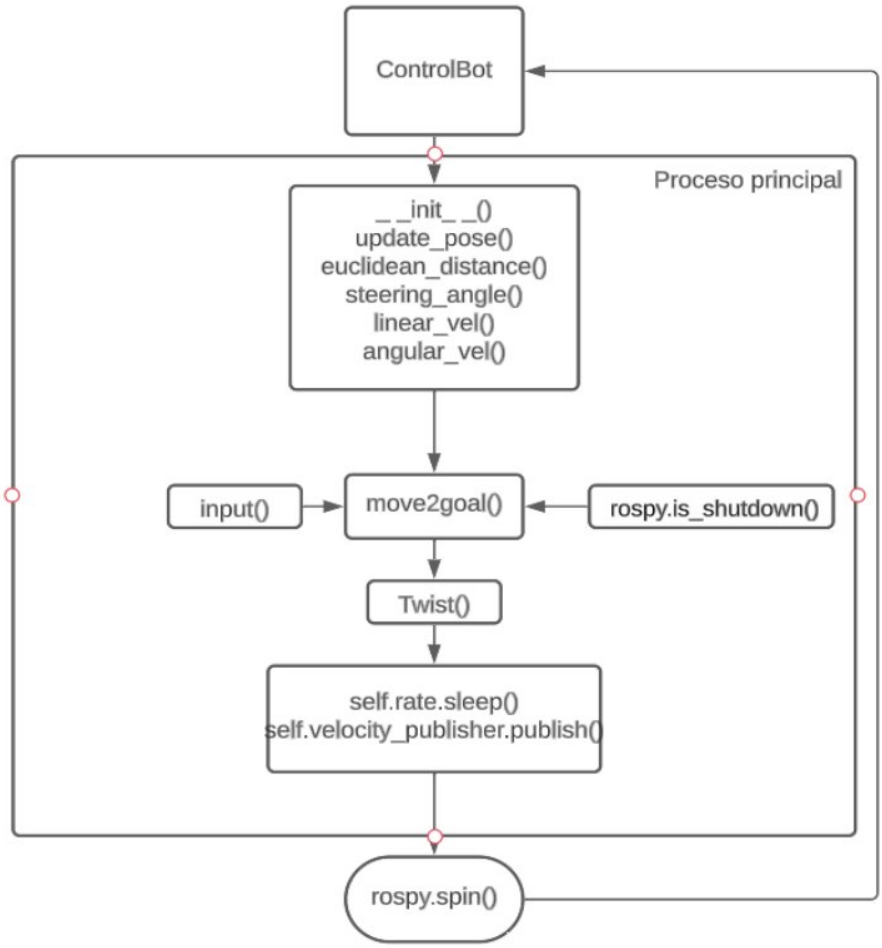


Fig. 4.4. Diagrama de flujo para control manual de distancias a un determinado punto.

4.4. VERIFICACIÓN DE COMUNICACIÓN

En esta sección, se presenta la verificación de la comunicación entre los sistemas implementados y se valida el correcto funcionamiento del diseño y lectura de los encoders y sensores. Se describe el proceso de verificación de la conexión SSH y la ejecución exitosa del código rosserial para la obtención de datos. Además, se comprueba la precisión de los encoders y la correcta lectura de los sensores, asegurando así la integridad de la información obtenida para su posterior procesamiento y control en el sistema robótico.

4.4.1. PRUEBA DE LECTURA DE CODIFICADORES

En esta prueba, se verifica y valida el funcionamiento adecuado del algoritmo desarrollado en Arduino para controlar los pulsos positivos y negativos generados por los encoders. En primer lugar, se comprueba que el algoritmo permita la acumulación de pulsos cuando se realiza un giro en sentido horario, y la resta de estos cuando se efectúa

un giro en sentido antihorario. Esta verificación garantiza la correcta detección y procesamiento de los movimientos del sistema, como se muestra en la Fig. 4.5 donde se ve el número de pulsos de los encoders en sentido antihorario.

```
Numero de cuentas: 0: 0: 0: 0
Numero de cuentas: 0: 0: 0: 0
Numero de cuentas: -139: -132: -139: -123
Numero de cuentas: -453: -437: -427: -401
Numero de cuentas: -836: -819: -770: -745
Numero de cuentas: -1264: -1244: -1141: -1124
Numero de cuentas: -1709: -1688: -1526: -1517
Numero de cuentas: -2158: -2136: -1911: -1912
Numero de cuentas: -2616: -2593: -2303: -2315
Numero de cuentas: -3076: -3053: -2698: -2719
Numero de cuentas: -3532: -3509: -3087: -3119
Numero de cuentas: -3994: -3970: -3482: -3521
Numero de cuentas: -4457: -4432: -3876: -3924
Numero de cuentas: -4915: -4888: -4267: -4323
Numero de cuentas: -5376: -5349: -4662: -4727
Numero de cuentas: -5837: -5811: -5055: -5132
Numero de cuentas: -6293: -6267: -5447: -5533
Numero de cuentas: -6753: -6727: -5842: -5938
Numero de cuentas: -7214: -7189: -6236: -6345
```

Fig. 4.5 Prueba y calibración de numero de pulsos

4.4.2. PRUEBA DE CONSUMO DE VOLTAJE DE LOS MOTORES

Para verificar el consumo, se activa los cuatro motores y se realizó una medición utilizando un multímetro. Esta prueba permitió obtener lecturas precisas del consumo eléctrico en amperios generado por los motores en funcionamiento. Además, se registraron y analizaron los datos obtenidos para evaluar el rendimiento del sistema en términos de consumo energético. En la Fig. 4.6 se muestra el consumo en amperios.

Para calcular el consumo energético que puede soportar la batería de 2.6 Ah, se utiliza la siguiente fórmula:

$$TF = \frac{CB}{CT} \tag{9}$$

TF= Tiempo de funcionamiento en horas

CB=Capacidad de la batería (Ah)

CT=Consumo total(A)

$$TF = \frac{2.6Ah}{1.6A} = 1.625h$$

El resultado nos permite concluir que la batería nos da un tiempo de funcionamiento aproximado de una hora y media con el funcionamiento de los cuatro motores y las dos placas de control.

Fig. 4.6 Medición de amperaje de motores.

4.4.3. PRUEBA DE SENSOR IMU MPU6050

Durante la prueba del sensor IMU MPU6050 utilizando Arduino, se recopilaron datos relevantes para evaluar la estabilidad del sistema. Utilizando el sensor conectado al Arduino, se registraron mediciones de aceleración y velocidad angular en diversas condiciones y actividades. El análisis de los datos obtenidos permitió identificar patrones de movimiento, desviaciones y posibles ajustes necesarios para mejorar la estabilidad del sistema.

4.4.4. COMUNICACIÓN ENTRE ROS Y ARDUINO

Para establecer la comunicación entre dos sistemas, en este caso, un computador y una Raspberry Pi 4, se utiliza el protocolo de comunicación SSH (Secure Shell). Este proceso se realiza desde el computador hacia la Raspberry Pi 4, permitiendo acceder de forma segura a la Pi mediante una conexión encriptada.

Una vez establecida la conexión SSH, en el computador se abren dos terminales. En uno de ellos se encuentra activa la conexión SSH hacia la Raspberry Pi, mientras que en el otro terminal se ejecuta el código **rosserial**. Este código permite obtener los datos provenientes de Arduino en el nodo maestro de la Raspberry Pi 4, y se comunican de manera inmediata al computador a través de la conexión SSH previamente establecida. En la Fig. 4.7 se muestra que la comunicación con el Arduino se realizó de forma exitosa porque comienza a enviar datos a nuestra terminal.

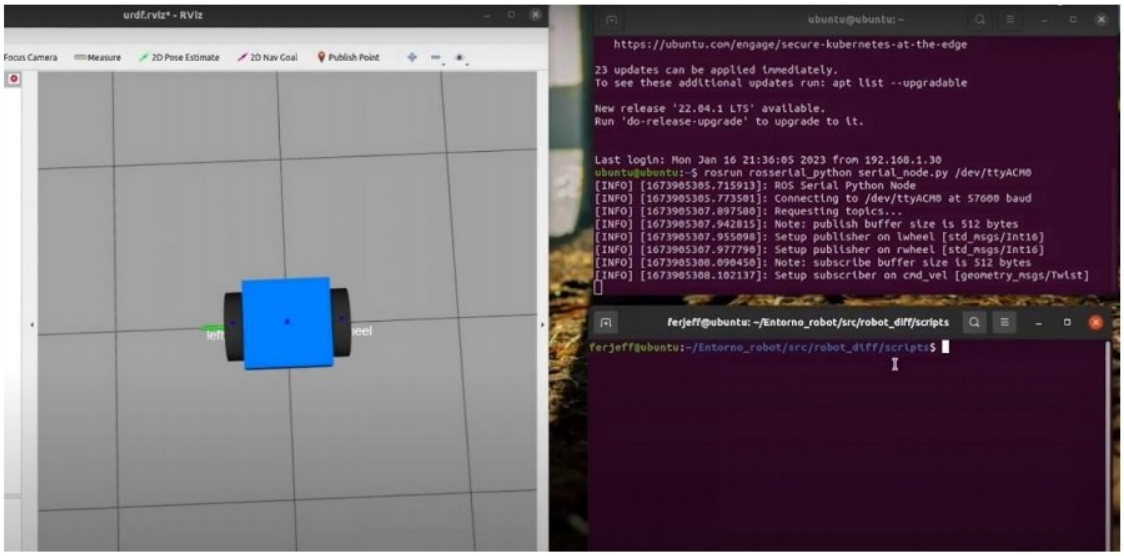


Fig. 4.7 Comunicación SSH y comunicación rosserial con el Arduino

Para verificar la correcta recepción de los datos, se realizan acciones que involucran el movimiento de los motores. De esta manera, se pueden visualizar los cambios correspondientes en RVIZ, una herramienta de visualización en ROS. De esta forma, se comprueba que la conexión entre los sistemas fue exitosa y los datos se están transmitiendo correctamente.

4.5. PRUEBAS DE DESEMPEÑO DEL NODO EN LA PLATAFORMA

La verificación del desarrollo se llevará a cabo en dos entornos de prueba: uno con suelo liso y otro con suelo rugoso. Estos entornos representan diferentes condiciones que pueden encontrar el robot durante su funcionamiento. Se realizarán dos tipos de pruebas en cada entorno: desplazamiento en línea recta y desplazamiento en un mismo punto. Estas pruebas permitirán evaluar el rendimiento y la capacidad de respuesta del robot en situaciones de movimiento tanto en superficies lisas como en superficies más irregulares.

4.5.1. PRUEBA DE ANÁLISIS ODOMETRICO DEL NODO

En esta prueba, se demuestra el funcionamiento del algoritmo de odometria del robot del nodo de ROS. El objetivo es obtener la distancia recorrida y la orientación angular del robot a medida que se desplaza 150cm. Para poder verificar la precisión del algoritmo, se realiza una medición con un decámetro para comparar la posición y orientación real del robot con los resultados obtenidos a través de las ecuaciones del análisis odometrico.

Los resultados obtenidos de estas pruebas se registran en la Tabla 4.5.1 donde se verifico la distancia, el tiempo y los grados de deviación del recorrido.

Tabla 4.5.1 Datos de odometria del robot móvil

Nº PRUEBA	DISTANCIA RRECORRIDA cm	TIEMPO TRANSCURRIDO	ANGULO DE DESVIACIÓN grados
1	40	4	2
2	45	4.5	2.3
3	55	5	2.7
4	65	6.5	3.1
5	70	7	3.4
6	80	8	3.7
7	90	9.3	4.8
8	100	10.4	5.4
9	120	12.5	6.7
10	150	15	7.8
PROMEDIO	81.5	8.22	4.19

La expresión de datos obtenidos nos permite demostrar que el ángulo de desviación en todas las pruebas realizadas es notablemente preciso, ya que arrojó un promedio de 4.19, lo que se aproxima significativamente a los cálculos odométricos. Estos resultados indican que la precisión de las mediciones es bastante buena.

4.5.2. PRUEBA EN SUPERFICIE LISA

4.5.2.1. PRUEBA EN TRAYECTORIA RECTA

Considerando que nuestra condición inicial en cuanto posición es de (0,0) en coordenadas x, y se procede a realizar la prueba en un trayecto rectilíneo y ver la respuesta del sistema.

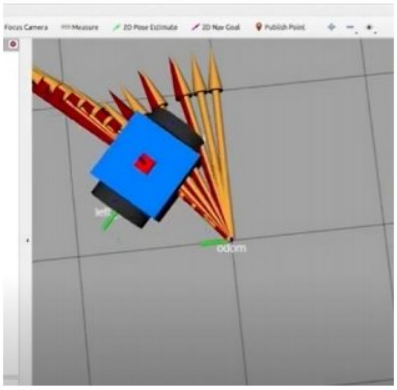


Fig. 4.8. Odometria grafica del robot en RVIZ

En la Fig. 4.8 se muestran los resultados generales del nodo de odometría, que realiza cálculos precisos utilizando información de los encoders. La línea amarilla refleja el trayecto realizado y simulado por RVIZ, indicando la dirección hacia la que el robot se dirige según los datos de los encoders.

Además, la línea roja en el gráfico representa la información proporcionada por el sensor IMU, encargado de medir la orientación y movimientos angulares del robot. Aunque tanto el nodo de odometría como el sensor IMU contribuyen a la localización y navegación precisa del robot, existe una pequeña diferencia en su tiempo de respuesta.

Durante el trayecto del robot y la simulación en RVIZ, se observó una correspondencia notable entre los datos proporcionados por la odometría y la realidad. Esto se evidencia en la Fig. 4.9 donde se representa la trayectoria realizada con precisión, reflejando adecuadamente la tendencia y apertura angular del robot.

Fig. 4.9. Trayecto en superficie lisa del robot

4.5.2.2.PRUEBA DE TRAYECTORIA EN UN MISMO PUNTO

Se ha observado que cuando el robot gira sobre su propio eje, la trayectoria descrita coincide en un solo punto. Sin embargo, al desarrollarse en una superficie lisa, se detecta un error debido al deslizamiento del robot, lo que conlleva a la pérdida del trayecto, como se muestra en la Fig. 4.10.

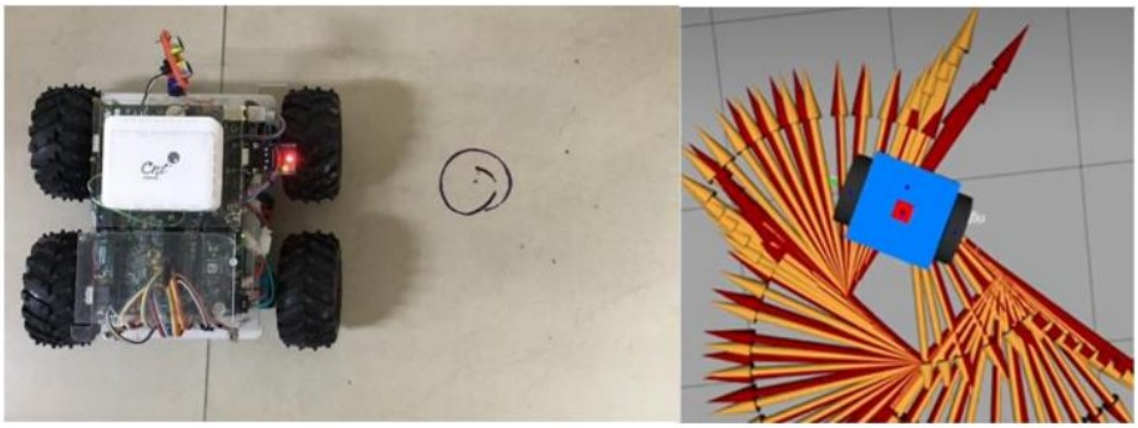


Fig. 4.10. Trayectoria en un mismo punto en superficie lisa

4.5.3. PRUEBA EN SUPERFICIE RUGOSA

4.5.3.1.PRUEBA EN TRAYECTORIA RECTA

Considerando las mismas condiciones de funcionamiento, el robot es desplazado a una distancia específica de 200 cm para observar su comportamiento en una superficie con mayor rugosidad.

En la Fig. 4.11 se muestra la prueba de campo del robot, donde se ha verificado la precisión de la información enviada por los sensores y el funcionamiento de la odometría. Los resultados registrados en la interfaz indican una distancia recorrida de 200cm con un ángulo de desviación de 18.7º, mientras que al medir con el decámetro se obtuvo una distancia de 212 cm con un ángulo de desviación de 19º.

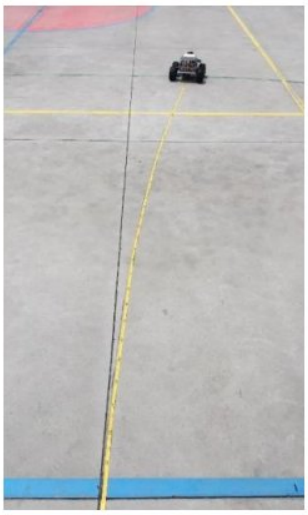


Fig. 4.11. Trayecto del robot en superficie rugosa

Para validar la precisión de los datos obtenidos en la interfaz, se realizó una prueba enviando al robot a un punto objetivo específico, en este caso, desde la posición inicial (0.0) hasta la posición deseada de (0.5, 0.5). Sin embargo, se pudo observar una variación considerable entre la posición ideal y la posición final alcanzada por el robot, como se muestra en la Fig. 4.12, que representa todos los datos recopilados por el nodo de odometría.

Esta variación puede atribuirse a diversos factores que afectan la precisión del sistema de odometría, como errores acumulativos en la medición de los encoders, incertidumbre en los datos del sensor IMU y posibles deslizamientos de las ruedas durante el desplazamiento.

```
---------------------------------------------------
v_lineal:  0.5155080495227479
v_angular:  0.27972399845526913
pose x:  0.4186
pose y:  0.368
pose obj x:  0.5
pose obj y:  0.5
angulo robot: 0.7639174324539381
angulo objetivo: 1.0182119765041828
distancia euclidiana: 0.15508049522747855
---------------------------------------------------
```

Fig. 4.12 Datos analizados y contemplados por el nodo de odometria.

4.5.3.2.PRUEBA DE TRAYECTORIA EN UN MISMO PUNTO

Durante esta prueba, se planteó de manera lógica que debido a la situación y al cambio de superficie, las ruedas del robot tendrían un mejor agarre, lo que se esperaba resultara en mediciones más precisas. Sin embargo, como se muestra en la Fig. 4,13, el

robot aún presenta una leve desviación de su eje de rotación, exhibiendo un resultado similar al observado en la superficie lisa.

Fig. 4.13. Trayectoria en un mismo punto del robot

CAPITULO V

CONCLUSIONES Y TRABAJO FUTURO

5.1.1. CONCLUSIONES

- La reducción en cuanto presupuesto de los materiales electrónicos y su disponibilidad dentro del país ha sido un factor clave en este trabajo de investigación. Esto ha permitido que se tenga la seguridad de que dentro de esta plataforma se puede realizar muchas mejoras e investigaciones en la robótica.
- La robustez del sistema ROS ha demostrado ser valiosa para llevar a cabo proyectos complejos. La capacidad de utilizar paquetes de software preexistentes ha agilizado el proceso de implementación de este proyecto y permitirá abordar proyectos más ambiciosos en el futuro.
- La precisión de la odometría ha mejorado significativamente mediante una calibración adecuada de los encoders y sensores. Además, la integración del sistema ROS ha permitido obtener datos más precisos y detallados.
- La combinación de placas de control como Raspberry Pi y Arduino ha resultado efectiva y accesible para el proyecto. Estos dispositivos son conocidos y ampliamente utilizados en la comunidad estudiantil y de robótica.
- La naturaleza en constante desarrollo de ROS lo convierte en un sistema atractivo y prometedor para el campo de la robótica. La amplia disponibilidad de paquetes y librerías hace que ROS sea una herramienta valiosa para futuras investigaciones y proyectos.
- La validación satisfactoria de la estructura del nodo odométrico durante las pruebas confirma su utilidad como base para futuros trabajos en el desarrollo de robots. Esto proporciona una plataforma estable y confiable para seguir construyendo y mejorando.

5.1.2. POSIBLES TRABAJOS FUTUROS

A partir de este trabajo de investigación se plantea algunas recomendaciones para desarrollar futuras trabajos o tesis:

- Explorar técnicas de SLAM para mejorar la localización y mapeo.
- Investigar la integración de sensores adicionales para mayor precisión.

- Desarrollar una interfaz de usuario intuitiva para facilitar la interacción con el robot.
- Implementar algoritmos de planificación de trayectorias para tareas más complejas.
- Realizar pruebas en diferentes entornos para validar el desempeño en escenarios variados.

BIBLIOGRAFÍA

[1] G. Andrés and F. Cruz, "'CONTROL DE VELOCIDAD DE UN ROBOT MÓVIL CON DIRECCIÓN DIFERENCIAL.'"

[2] A. García Cazorla Director and J. Luis Muñoz Lozano, "ROS: Robot Operating System."

[3] J. Riquelme -Pere Botella, E. Alba, J. G. Nieto, and F. Chicano, "ROS: SERVICIO DE OPTIMIZACIÓN REMOTA," 2006. [Online]. Available: http://neo.lcc.uma.es

[4] Siciliano B. Khatib O, "Springer Handbook of Robotics," 2016.

[5] G. Rajendran, U. V, and B. O'Brien, "Unified robot task and motion planning with extended planner using ROS simulator," *Journal of King Saud University - Computer and Information Sciences*, Oct. 2021, doi: 10.1016/j.jksuci.2021.07.002.

[6] A. Araújo, D. Portugal, M. S. Couceiro, J. Sales, and R. P. Rocha, "Desarrollo de un robot móvil compacto integrado en el middleware ROS," *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, vol. 11, no. 3, pp. 315–326, 2014, doi: 10.1016/j.riai.2014.02.009.

[7] J. F. Guerrero-Castellanos, M. G. Villarreal-Cervantes, J. P. Sánchez-Santana, and S. Ramírez-Martínez, "Trajectory tracking of a mobile robot (3,0) by means of bounded control.," *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, vol. 11, no. 4, pp. 426–434, Oct. 2014, doi: 10.1016/j.riai.2014.09.005.

[8] J. Paiva Mimbela, "Sistema de posicionamiento indoor para el guiado de robots móviles implementado en Robot Operating System(ROS)," 2019.

[9] W. A. Vaca Paredes, "UNIVERSIDAD TÉCNICA DEL NORTE TRABAJO DE GRADO PREVIO LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA 'MOBILE ROBOT FOR RESEARCH ON PATH PLANNING ALGORITHMS: ODOMETRY SYSTEM,'" 2017.

[10] J. Gutiérrez Pérez, "Introducción a ROS en Raspberry Pi," 2017.

[11] RobotShop, "Arduino Mega 2560 Datasheet." Accessed: Jul. 24, 2023. [Online]. Available:

https://repositorio.uisek.edu.ec/bitstream/123456789/2893/7/ArduinoMega2560Datasheet.pdf

[12] W. F. Lages, "Remote Teaching of Dynamics and Control of Robots Using ROS 2," *IFAC-PapersOnLine*, vol. 55, no. 17, pp. 279–284, 2022, doi: 10.1016/j.ifacol.2022.09.292.

[13] IvenseSense, "Sensor MPU6050. – Electrónica Práctica Aplicada." https://www.diarioelectronicohoy.com/blog/sensor-mpu6050 (accessed Jul. 23, 2023).

[14] O. Edwin, "A Primer on Odometry and Motor Control," 2004.

[15] C. Fairchild and T. L. Harman, *ROS robotics by example : learning to control wheeled, limbed, and flying robots using ROS Kinetic Kame*, Segunda. 2017.

[16] Components101, "L298N Motor Driver Module Pinout, Datasheet, Features & Specs." https://components101.com/modules/l293n-motor-driver-module (accessed Jul. 24, 2023).