Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería Mecatrónica

Diseño e implementación de rutinas de navegación y comunicación para un robot móvil utilizado en sistemas multirobot de enjambre

Informe de Proyecto de Graduación para optar por el título de Ingeniero en Mecatrónica con el grado académico de Licenciatura

Carlos Alonso Barrantes Jiménez

2014104538

Cartago, noviembre de 2019

INSTITUTO TECNOLÓGICO DE COSTA RICA

CARRERA DE INGENIERÍA MECATRÓNICA

PROYECTO DE GRADUACIÓN

ACTA DE APROBACIÓN

El Profesor Asesor, da fe de que el presente Proyecto de Graduación ha sido aprobado y cumple con las normas establecidas por la Carrera de Ingeniería Mecatrónica como requisito para optar por el título de Ingeniero en Mecatrónica, con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Estudiante: Carlos Alonso Barrantes Jiménez.

Nombre del Proyecto: Diseño e implementación de rutinas de navegación y comunicación para un robot móvil utilizado en sistemas multirobot de enjambre.

Ing. Juan Carlos Brenes Torres

Profesor Asesor

Cartago, 25 de noviembre del 2019.

Resumen

Se presenta el diseño e implementación de rutinas de navegación y comunicación para un robot móvil utilizado en sistemas multirobot de tipo enjambre para la digitalización de escenarios estáticos. Este trabajo forma parte del proyecto de investigación PROE E1F2, ejecutado por la escuela de matemática y el área académica de ingeniería mecatrónica del Instituto Tecnológico de Costa Rica.

Para solucionar el problema, se propuso un esquema de control que combina la cinemática directa e inversa del robot móvil, junto a controladores *PID* y de retroalimentación proporcional. Además, se implementó una unidad de medición inercial con un algoritmo de fusión de sensores para disminuir errores de odometría del tipo deslizamiento. Para el sistema de comunicación se utilizó la técnica acceso múltiple por división de tiempo, para la cual fue necesario un método de sincronización de relojes llamado sincronización de tiempo por medición de retardo.

El control automático presentó un error máximo de 2.16% en estado estacionario y con la integración de la unidad de medición inercial se redujo el error en la orientación del robot hasta en 10.70°. El sistema de comunicación logró manejar el flujo de información de cuatro nodos, recibiendo el 99.935% de los mensajes enviados a la base central.

Palabras clave: Robótica de enjambres, control de navegación, odometría, unidad de medición inercial, filtro de Madgwick, TDMA, DMTS.

Abstract

It’s shown the design and implementation of navigation and communication routines for a mobile robot used in a multirobot swarm system to digitize static scenarios. This work is part of the research project PROE E1F2, conducted by the mathematics school and the mechatronics engineer area of the Instituto Tecnológico de Costa Rica.

To solve the problem, a control scheme using direct and inverse kinematics in combination with a *PID* controller and proportional feedback controller was proposed. Also, an inertial measurement unit was implemented with a sensor fusion algorithm to diminish odometry errors of the slip type. For the communication system, the time division multiple access technique was used, for which was necessary a method to synchronize clocks called delay measurement time synchronization.

The automatic control presented a maximum error of 2.16% in the steady state and with the integration of the inertial measurement unit the robot orientation error was reduced up to 10.70°. The communication system was able to manage the information flow of four nodes, receiving 99.935% of the messages send to the central base.

Key words: Swarm robotics, navigation control, odometry, inertial measurement unit, Madgwick filter, TDMA, DMTS.

Agradecimientos

Le agradezco profundamente a toda mi familia: a mi madre Flor de Liz, a mi padre Carlos Gerardo, a mis dos hermanas Fernanda y Victoria y a mi abuela Aracely; por todo el amor, apoyo y comprensión que me dieron durante mi estadía en la universidad. Además, le agradezco a mi pareja Melissa, por todo su amor, tiempo y compañía a lo largo de este proceso.

También estoy agradecido con todas amistades y personas cercanas que de una u otra forma me ayudaron a lo largo de mi paso por la universidad y a llegar hasta aquí, permitiéndome tener la oportunidad de realizar este proyecto de graduación

Además, quiero agradecer a los y las profesores: Juan Carlos Brenes Torres, Cindy Calderón Arce y Rebeca Solís Ortega; quienes a finales del 2018 me dieron la oportunidad de ingresar al proyecto de investigación PROE E1F2 y que desde entonces han dedicado de su tiempo para enseñarme y ayudarme en lo necesario para hacer realidad este trabajo.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1 Introducción

1.1 Entorno del proyecto

Este proyecto de graduación se realizó como parte del proyecto de investigación PROE E1F2[1]: Implementación de un prototipo de enjambre de robots para la digitalización de escenarios estáticos y planificación de rutas óptimas. Dicho proyecto es coordinado por la escuela de matemática y desarrollado en conjunto con el área académica de ingeniería en mecatrónica; ambas del Instituto Tecnológico de Costa Rica (ITCR) y ubicadas en el Campus Tecnológico Central Cartago, ciudad de Cartago. La escuela de matemática contempla dentro de sus áreas de trabajo la matemática pura, matemática aplicada y matemática educativa. Mientras que el área académica de mecatrónica se focaliza en formar profesionales que proporcionen sistemas mecatrónicos integrales, inteligentes y flexibles a la industria nacional e internacional.

Entre el 2017 y el 2018, la escuela de matemática desarrolló el proyecto de investigación PROE E1F1: simulación computacional para la planificación de rutas óptimas de acceso y/o evacuación por medio de un enjambre centralizado en escenarios estáticos, utilizando técnicas de mapeo, procesamiento de datos y optimización multiobjetivo. Dados los resultados obtenidos en este primer proyecto, surgió la necesidad de desarrollar una implementación física de los algoritmos propuestos, con el fin de realizar una traducción y adaptación de estos para ser utilizados en un ambiente real, por este motivo se creó el proyecto PROE E1F2.

Debido a la complejidad ingenieril de esta segunda etapa, la escuela de matemática trabaja en conjunto con el área académica de ingeniería en mecatrónica, con el fin de cubrir las necesidades de desarrollo mecánico y electrónico del proyecto PROE E1F2. Actualmente, la investigación se encuentra a cargo de tres profesores del ITCR y un colaborador externo, los cuales son: M.Sc. Cindy Calderón Arce, M.Sc. Rebeca Solís Ortega, M.Sc. Juan Carlos Brenes Torres y el Dr. Tomas De Camino Beck, respectivamente. La coordinación del proyecto es llevada a cabo por la profesora Cindy Calderón Arce; sin embargo, la coordinación del desarrollo mecatrónico lo realiza el profesor Juan Carlos Brenes Torres.

[1] Las siglas PROE corresponden a “Planificación de Rutas Óptimas por medio de Enjambres”. El E1F2 representa segunda Fase de la primera Etapa.

1.2 Definición del problema

1.2.1 Diagrama Causa - Efecto (Ishikawa)

Los resultados de PROE E1F1 [1] indicaron que existe la necesidad de un método robusto de tipo multirobot para recolectar información veraz que posteriormente sea usada para la construcción del mapa de un escenario estático. La Figura 1.1 resume estas causas en un diagrama de Ishikawa.

Figura 1.1. Diagrama Causa – Efecto del proyecto.

Fuente: elaboración propia.

1.2.2 Generalidades

Para que un sistema sea considerado multirobot, este debe poseer 2 o más robots que interactúen entre sí para lograr alcanzar un objetivo común [2]. Un ejemplo de esto, son los robots en una línea de producción y ensamble, donde cooperan entre sí para manufacturar un producto; sin embargo, es común que en estos casos haya un centro de control para todos los robots y que, por lo tanto, la topología del sistema sea centralizada. Los enjambres de robots son un tipo específico de sistema multirobot, donde todos los robots comparten su información con el ambiente o con otros robots vecinos, creando así un flujo homogéneo de la información en el sistema multirobot y el ambiente en el que se desenvuelve.

El proyecto PROE E1F2 ha buscado desarrollar un sistema multirobot de enjambre que sea capaz de recopilar información sobre la ubicación de obstáculos en un escenario estático, para luego transmitir esta información a una unidad de procesamiento que construya un mapa del entorno donde se encuentran los robots. Para lograrlo, es necesario conocer con exactitud dónde está el objeto que el robot está detectando, ya sea un obstáculo, una pared o un agujero, para luego representarlo en el mapa.

En el diagrama de la Figura 1.1 se muestran algunas de las causas que crean el problema que ha tratado de resolver el proyecto PROE E1F2. Por ejemplo, en la causa de "Falta de soluciones a bajo costo" se puede mencionar que el robot Khepera IV, un robot móvil de alto desempeño desarrollado por el equipo K de Suiza, puede llegar a costar más de ₡ 2.200.000 [3], lo que refleja la necesidad de una solución de bajo costo. Además, estos robots no se encuentran en el país, lo que implica que para obtenerlos se debe realizar la compra en el extranjero y pagar impuestos de importación, aumentando más su precio y generando tiempos de espera. Otra de las limitaciones que presentan los sistemas actuales es que deben estar en ambientes muy controlados, pues dependen de un cuarto con cámaras para ubicar a los robots en el espacio. Y finalmente, muchos de estos robots ya existentes utilizan módulos de *WiFi* o Bluetooth, que limitan inmediatamente su aplicación a distancias de corto alcance o a zonas que cuenten con conexión a internet.

1.2.3 Justificación

Los proyectos PROE fueron diseñados con el objetivo final de desarrollar sistemas para planificar y obtener rutas óptimas en escenarios dinámicos. De esta manera se busca contribuir con el acceso y/o evacuación en situaciones de desastre, como lo son los derrumbes o incendios. Para lograr el objetivo final de PROE, es esencial controlar los movimientos de los robots que se utilizarán en el enjambre y que estos logren enviar toda la información de manera certera.

Un primer prototipo requiere de un diseño de controlador que sea capaz de dirigir correctamente los movimientos de todos los agentes (en la temática de enjambres de robots, agente es una manera de llamar a un robot). Es decir, es necesario desarrollar una estrategia de control que estime con exactitud la ubicación de los robots del enjambre, para así poder ubicar correctamente los objetos que ellos detectan. También es de suma importancia el desarrollo de una

plataforma que permita el flujo de información transmitida desde los agentes hasta la unidad de procesamiento, con miras a una escalabilidad en el sistema.

A pesar de esta necesidad latente, dentro del proyecto de investigación PROE no se ha realizado una justificación matemática debida para realizar un correcto diseño de los controladores de los robots, lo que ha causado implementaciones que dan resultados imprecisos. Tampoco existe un sistema de comunicación para los robots. Por lo tanto, es necesaria la creación y validación de rutinas de navegación y comunicación en un prototipo básico de un sistema multirobot de tipo enjambre.

1.2.4 Síntesis del problema

Hay una carencia de modelos matemáticos de los robots que se usarán en PROE E1F2, y, por lo tanto, una inexistente justificación de los controladores implementados para el control de los diferentes movimientos de los agentes móviles. Además, es necesario desarrollar un sistema de comunicación para manejar el flujo de datos desde el enjambre hacia la "colmena" o unidad de procesamiento.

1.3 Enfoque de la solución

Primero se debió hacer un estudio profundo en la temática de robótica y un análisis del estado del arte de sistemas multiagente y enjambres de robots para la exploración de escenarios y planificación de rutas. Lo anterior con el objetivo de definir técnicas para conocer el posicionamiento de los robots móviles. La técnica de odometría era de especial interés para el equipo de investigación, dada la experiencia y resultados obtenidos con este tipo de posicionamiento en el proyecto de investigación PROE E1F2. Esto conllevó a un análisis de la cinemática detrás del movimiento de un robot móvil, y de las diferentes maneras de manipular esta información para su fácil procesamiento en un computador.

Luego, se debió estudiar el comportamiento de la planta a controlar, para así obtener el modelo matemático que la represente mejor. Con base en esto, se definió el tipo de controlador que se adaptara mejor a la situación, ya fuera: PID, realimentación de estados, control por atraso/adelanto o algún otro tipo de control.

Además, se tuvo que realizar un estudio de los diferentes sensores y actuadores que se utilizaron en la construcción del robot móvil, para realizar una efectiva unión entre técnicas de

posicionamiento y control. Se debió analizar: codificadores, unidades de medición inercial (*IMU*) y motores CD.

Paralelo a lo anterior, fue necesario determinar la manera de comunicar inalámbricamente el robot con una unidad de procesamiento. Para esto, se valoró opciones como: internet de las cosas, bluetooth, radio frecuencia (RF), entre otras. Indiferentemente de la tecnología de comunicación escogida, también se tuvo que hacer un análisis de la eficacia de la implementación, ya que se debió valorar las diferentes topologías disponibles para realizar la comunicación, asegurando la escalabilidad del sistema.

Además, se debió crear un programa en la unidad de procesamiento que fuera capaz de recibir, interpretar y alistar los datos enviados por los nodos de comunicación para su posterior procesamiento, tomando en cuenta que es posible que haya más de un nodo tratando de comunicarse con la unidad de procesamiento a la vez. Por lo que el programa tuvo que incluir un método de acceso al medio.

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar las rutinas de navegación y comunicación necesarias para los robots móviles de un enjambre.

1.4.2 Objetivos específicos

1. Diseñar los modelos matemáticos y sus respectivos controladores para las rutinas de navegación del robot. **Indicador:** obtención de un error en estado estacionario menor a 1.5%, y un sobre impulso menor a 5% ante los cambios de referencia para los diferentes controladores que se diseñen.

2. Integrar al menos una de las funciones de una *IMU* para corregir errores de odometría. **Indicador:** lograr una reducción del error de posición u orientación del robot móvil ante errores de odometría de tipo deslizamiento.

3. Desarrollar el sistema de comunicación entre los robots y la unidad de procesamiento. **Indicador:** recepción de >90% de los datos enviados a la unidad central, para una cantidad máxima de emisores disponibles.

2 Marco teórico

2.1 Estado del Arte: sistemas multirobot y robótica de enjambres

La robótica de enjambres es la rama de la robótica que estudia cómo hacer que robots simples puedan colaborar conjuntamente como una única entidad compleja para realizar una tarea específica, que sería imposible o muy difícil de realizar para los robots de manera individual [4]. Esta rama de estudio es inspirada en la naturaleza [5], donde se puede encontrar fácilmente diferentes especies de animales donde cada individuo colabora como parte de un grupo con un objetivo común. En la Figura 2.1 se observan cuatro ejemplos: las abejas acuerdan todas seguir rutas diferentes para así maximizar su alcance en la búsqueda de flora; las hormigas exploran solas o en conjunto, y cuando encuentra algo de interés, guían a toda la colmena al lugar; algunos pájaros vuelan en forma de V para facilitarle el vuelo a los más viejos del grupo; y otro grupo de aves coordina vuelos en conjunto.

Figura 2.1. Agrupaciones de animales que colaboran en conjunto para alcanzar un objetivo común.

Fuente: [4].

Como ya se mencionó en el capítulo anterior, para que un sistema robótico sea considerado del tipo multirobot, este debe poseer 2 o más robots que interactúen entre sí para lograr alcanzar un objetivo común [2]. Existen diferentes razones por lo cual estos sistemas son necesarios: la complejidad de una tarea es muy alta para un solo robot, existen tareas que tienen una naturaleza

distribuida, usar múltiples robots aumenta la robustez del sistema agregando redundancia [6]. Los enjambres de robots son un tipo específico de sistema multirobot, donde se desea alcanzar un comportamiento colectivo que emerja al darse la interacción entre cada robot del enjambre con sus homólogos y con el ambiente, creando así un flujo homogéneo de la información entre el enjambre y el entorno en el que se desempeñan [7]. Todo enjambre es un sistema multirobot, pero no todo sistema multirobot es del tipo enjambre. En la literatura existen algunas reglas que determinan si un grupo de robots es o no un enjambre, Şahin en [7], Navarro y Matía en [5], y Khaldi y Cherif en [8] denotan cinco características en específico:

1. Los robots de un enjambre deben ser autónomos e interactuar con un ambiente físico. Este último aspecto es de importancia, pues por ejemplo, tener una gran cantidad de sensores recopilando información y comunicándola a alguna base central no debería tomarse como enjambre de robots, pues no posee capacidades de actuación sobre su ambiente.
2. El número de robots en el sistema debe ser alto[2], o al menos aspirar en un futuro a ser escalable.
3. Los robots deben ser homogéneos. El enjambre debe estar constituido por relativamente pocos grupos de robots homogéneos, y, como ya se dijo, cada grupo debe contener una alta cantidad de agentes.
4. Los robots deben ser incapaces de realizar la tarea por sí solos, pues se espera que haya trabajo colaborativo entre varios agentes para lograr la tarea.
5. Los robots solamente tienen capacidades locales de comunicación y monitorización, para así asegurar una distribución coordinada del enjambre.

Se espera que un sistema multirobot de tipo enjambre tenga un control no solo autónomo, sino también descentralizado; esto significa que no hay un líder en específico dentro del enjambre, sino que la información se encuentra de manera distribuida. Por último, se espera también que un enjambre sea altamente adaptable y escalable. La Tabla 2.1 resume las características anteriores y contrasta las diferencias entre un sistema multirobot y uno del tipo enjambre.

[2] Tanto Şahin en [5], como Hamann en [4], expresan que colocar un límite inferior para determinar a partir de cuándo se considera como enjambre un grupo de robots es difícil de justificar. Por ende, una solución que propone Hamann es que la cantidad mínima en realidad dependerá de la tarea que desempeñe el enjambre, sin embargo, no da un número concreto por tarea.

Tabla 2.1. Resumen de las diferencias más comunes entre sistemas multirobot y enjambres de robots.

Fuente: adaptada de [8].

Característica	Enjambre de robots	Sistema multirobot
Tamaño del sistema	Variación en gran rango	Normalmente pocos agentes
Control	Descentralizado y autónomo	Centralizado y remoto
Homogeneidad	Homogéneo	Heterogéneo
Adaptabilidad	Alta	Baja
Escalabilidad	Sí	No
Ambiente	Conocido/Desconocido	Conocido/Desconocido
Movilidad	Sí	Sí

Con las características de un enjambre de robots ya definidas, el lector se puede comenzar a hacer una idea de cómo estos se pueden desempeñar en un ambiente real. De manera concreta, un enjambre de robots puede ser utilizado para un gran número de tareas, que se descomponen en comportamientos básicos y primitivos de los robots. Parker menciona en [6] algunas de las tareas: búsqueda, vigilancia, evasión, exploración, mapeos, seguimiento de objetivos, entre otros. Dichas tareas se pueden clasificar de acuerdo con sus comportamientos, tal y como se detalla a continuación.

1. **Comportamiento relativo a otros robots:** formaciones, "pastoreo", acomodo.
2. **Comportamiento relativo al ambiente:** búsqueda, exploración, mapeo.
3. **Comportamiento relativo a agentes externos:** persecución, depredador-presa, seguimiento de objetivos.
4. **Comportamiento relativo a otros robots y el ambiente:** vigilancia de perímetro, rodeo de objetivos.
5. **Comportamiento relativo a otros robots, al ambiente y a agentes externos:** evasión, jugar fútbol.

Parker también explica algunos hitos de suma relevancia para esta rama, donde quizás el más impresionante es el caso de un enjambre de 108 robots que realizó una tarea de localización en una escuela vacía de 300 m^2. Su meta era esparcirse hasta encontrar un objetivo de interés específico y luego guiar un humano a este [6].

También existen diferentes laboratorios a nivel global que han construido sus propios prototipos de enjambres. Algunos de los modelos de robots más conocidos son los siguientes: Khepera IV, Colias, Kilobot, Pheeno, Mona, Thymio II, e-Puck. Cada modelo ha sido diseñado considerando diferentes aspectos para desempeñarse mejor en una u otra tarea específica, por ejemplo, el Kilobot se desempeña muy bien en tareas que requieran escalabilidad, llegando a enjambres de hasta miles de robots [9]. Debido a esta capacidad del Kilobot, se buscó un diseño económico, alcanzado un precio de $14 por unidad; sin embargo, lo anterior llevó a sacrificar en aspectos como locomoción y comunicación.

La literatura en este ámbito es amplia, y varios autores han hecho comparaciones de los diferentes modelos y sus características. Khaldi, Cherif y Hilder et al. han realizado comparaciones generales de modelos como el Kilobot, Khepera III, Colias, Jasmine Pi Swarm, explicando el principal objetivo que se buscaba conseguir con el desarrollo del robot [8], [10]. Navarro, Matía y Wilson et al. han comparado especificaciones técnicas a nivel detallado de nueve modelos diferentes, donde se pueden ver por ejemplo, las capacidades computacionales. También es interesante notar que ocho de estos nueve modelos utilizan ruedas para desplazarse [5], [11]. Y finalmente, Arvin et al. en [3] exponen los precios de los diferentes modelos, si estos se usan para educación o investigación, y si se comercializan al público o no. Aquí es interesante notar que nueve de los diez modelos comparados son de código abierto, solamente el Khepera IV no lo es.

Finalmente, un aspecto técnico de suma importancia que destaca Navarro y Matía en [5], es que seis de estos nueve modelos que compararon utilizan un posicionamiento basado en tecnología infrarroja o cámaras. Esto significa que los robots no son viables fuera de sus ambientes controlados, reforzando aún más la clara existencia de la necesidad de un robot para enjambres que sea capaz de desempeñarse eficazmente en ambientes no controlados. Los otros tres modelos no reportan un sistema de posicionamiento relativo, lo cual es aún peor.

2.2 Robótica móvil terrestre

Un aspecto de suma importancia para la robótica es lograr que un robot tenga la capacidad de desplazarse, pues esto abre puertas para utilizar los robots en tareas donde la movilidad es esencial, como la exploración espacial. La movilidad que se desea en un robot restringirá en gran medida la construcción mecánica que tendrá, el tipo de control que se usará para regular sus movimientos, y los requerimientos energéticos que tendrá el robot, por lo que es necesario un estudio profundo del alcance que se espera de un robot.

Cuando se habla de movilidad terrestre, la naturaleza tiene un amplio rango de posibilidades que se destacan por lograr la locomoción de especies a través de múltiples terrenos. Estas han servido de inspiración para los humanos; sin embargo, la naturaleza supera nuestros sistemas ingenieriles, pues posee tiempos de respuesta mejores en los movimientos, sistemas de locomoción más robustos y alta eficiencia en la transmisión del movimiento [12]. La Figura 2.2 presenta un resumen de los tipos de movimiento encontrados en la naturaleza, la resistencia que encuentran los movimientos y la cinemática básica que los describe.

Type of motion	Resistance to motion	Basic kinematics of motion
Flow in a Channel	Hydrodynamic forces	Eddies
Crawl	Friction forces	Longitudinal vibration
Sliding	Friction forces	Transverse vibration
Running	Loss of kinetic energy	Oscillatory movement of a multi-link pendulum
Jumping	Loss of kinetic energy	Oscillatory movement of a multi-link pendulum
Walking	Gravitational forces	Rolling of a polygon

Figura 2.2. Sistemas de locomoción naturales.

Fuente: [12].

Existen robots que han imitado estos movimientos con éxito, pero a pesar de esto, el uso de ruedas sigue siendo el método de locomoción más utilizado por los humanos para dotar de movilidad a un robot terrestre. Dentro de este tipo de generación del movimiento existen diferentes posibilidades, Siegwart y Nourbakhsh en [12] exponen 17 configuraciones diferentes utilizando ruedas, donde las configuraciones usan desde 2 ruedas hasta 6 ruedas para generar el movimiento, la estabilidad, y el control deseado para el robot. Las configuraciones no se muestran aquí porque no es de interés profundizar en todos los tipos existentes, sino en una en especial, la locomoción diferencial con un punto de apoyo. El proyecto PROE E1F2 seleccionó esta por ser la más común en implementarse, permitiendo una rápida iteración en los prototipos construidos y por construir.

2.2.1 Direccionamiento diferencial en robótica móvil

El robot diferencial es probablemente el más sencillo de los robots movilizados por ruedas [13]. Este robot consta de dos ruedas independientes de radio *r* que están motorizadas y controlan su velocidad y dirección de giro, además cuentan con una o más (una para el caso de PROE E1F2) ruedas de balín o similar que permiten tener un punto de apoyo de baja fricción. Estos sistemas de movimiento normalmente tienen algunas características de interés, como por ejemplo: que el punto de referencia *(x,y)* esté ubicado en el centro de las dos ruedas y a lo largo del eje de movimiento, y que existe una orientación θ para la dirección en la que apuntan las ruedas, como lo representa la Figura 2.3.

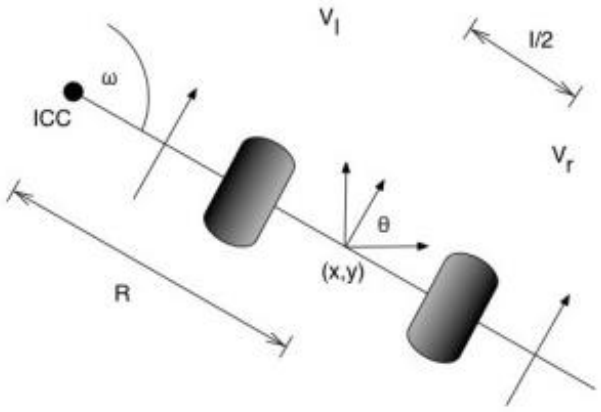

Figura 2.3. Centro instantáneo de rotación de una configuración diferencial terrestre.

Fuente: [14].

Esta configuración de ruedas diferenciales genera que el robot sea del tipo no holonómico [13], lo que significa que el movimiento del robot tiene una o más restricciones no holonómicas, las cuales son aquellas restricciones cinemáticas sobre las ruedas del robot que requieren incorporar las derivadas de las posiciones para poder ser expresadas en términos matemáticos [12]. Esto tiene implicaciones en el espacio de trabajo del robot y en cómo se controla el mismo, por ejemplo, ¿qué sucede si se desea mover el robot diferencial 1 cm a la izquierda o derecha a lo largo del eje de giro de las ruedas? Debido a las restricciones cinemáticas de las ruedas esto es imposible a no ser que se modifique el ángulo de orientación del robot. Lo anterior comienza a reflejar una relación implícita entre los grados de libertad de las ruedas (en este caso solamente uno por rueda, girar alrededor del eje de giro) y los grados de libertad del robot (tres para un robot diferencial: *x*, *y* y Θ).

Hay una estrecha relación entre las velocidades angulares de las ruedas del robot móvil diferencial y la velocidad lineal y angular del chasís del robot. Estas se relacionan así:

$$u_L = \frac{v - \omega d}{r} \tag{2.1}$$

$$u_R = \frac{v + \omega d}{r} \tag{2.2}$$

Donde u_L y u_R representan las velocidades angulares de las ruedas izquierda y derecha respectivamente, v la velocidad lineal del robot, ω la velocidad angular del robot, r el radio de las ruedas y d la distancia que hay entre una rueda y el centro del eje de giro de estas, como lo muestra la Figura 2.4.

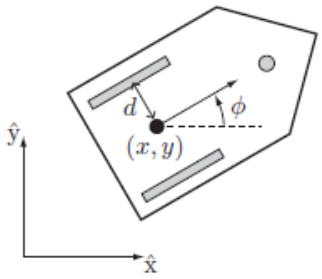

Figura 2.4. Robot diferencial de dos ruedas y un punto de apoyo con su marco de referencia global.

Fuente: [13].

En las ecuaciones (2.1) y (2.2) se ve cómo sabiendo la velocidad lineal y angular del robot, se puede obtener las velocidades angulares de las ruedas; y si se aplican las ecuaciones pero a la inversa, conociendo las velocidades angulares de las ruedas se puede estimar la velocidad lineal y angular que lleva el robot. Estas ecuaciones son parte de un concepto general aún mayor, que es la cinemática directa y cinemática inversa de robots móviles.

La cinemática directa de un robot móvil es el conjunto de ecuaciones cinemáticas (no solo de velocidad, sino también de posición) que determinan la pose[3] del robot con base en las velocidades presentes en su sistema de locomoción. Mientras que la cinemática inversa es el conjunto de ecuaciones que, dada una pose deseada para el robot, generan las velocidades que debería tener la locomoción de un robot móvil para alcanzar esa pose [15]. Para el caso de un robot móvil diferencial como los descritos en esta sección, las velocidades del sistema de locomoción son las velocidades angulares de las ruedas diferenciales.

2.2.2 Odometría

La odometría es un método matemático utilizado en la robótica móvil que, conociendo las ecuaciones cinemáticas de un cuerpo, estima la posición y orientación de este usando mediciones del movimiento de su sistema de locomoción; comúnmente las ruedas. Esta es una técnica de posicionamiento relativo, es decir, se puede llegar a conocer la posición y orientación del robot a partir de una posición y orientación inicial, pero esta condición inicial es desconocida, por lo que no se puede calcular la condición final absoluta.

En esta técnica de estimación de posición y orientación, existen dos tipos de errores: sistemáticos y no sistemáticos [16]. Los primeros son invariantes en el tiempo y se deben principalmente a errores mecánicos en la construcción del robot. Mientras que los errores no sistemáticos son impredecibles y son aquellos causados por la interacción del robot con el ambiente, como lo son: pisos irregulares, grietas, golpes, inclinaciones y demás.

Los dos errores sistemáticos que más afectan en la odometría son el error causado por un diámetro de ruedas diferentes y el error causado por la incertidumbre de la distancia entre los puntos de contacto de las dos ruedas (la distancia $2d$ en la Figura 2.4). Una diferencia en los diámetros causará un giro implícito al querer seguir una línea recta, mientras que no conocer con

[3] La pose de un objeto es aquello que engloba tanto su posición como su orientación en un sistema coordenado.

exactitud la distancia entre las ruedas afectará al desplazamiento angular del robot cuando este realice un giro. Sin embargo, gracias a que los errores son sistemáticos, pueden ser estudiados y eliminados hasta cierto punto.

Respecto a los errores no sistemáticos, uno de los más importantes es el error de tipo deslizamiento. Este error se da cuando parte del sistema de locomoción del robot móvil interactúa con el ambiente de forma tal que da falsos positivos al sistema de medición cuando en realidad no ha habido actuación. Para el caso específico de un robot móvil diferencial, esto se da cuando una o ambas ruedas pasan sobre obstáculos, dígase una piedra, un cable, o un desnivel. Esto causa que la rueda gire una cantidad mayor que el verdadero desplazamiento en el plano 2D del robot.

2.3 Sistemas de control automático

El control automático es una rama de la ingeniería que estudia cómo lograr que un proceso o sistema cumpla su objetivo con la menor intervención humana posible. Aquí, un sistema se interpreta como cualquier proceso que posea una o varias entradas y salidas, y que estas estén relacionadas entre sí. A diario se puede observar fácilmente ejemplos de estos, como un aire acondicionado que regula la temperatura y humedad de una habitación según los parámetros de entrada de temperatura y humedad deseada.

Un sistema básico de control tiene tres componentes, el objetivo que se desea, el sistema de control en sí y los resultados que se obtienen. Su relación se observa en la Figura 2.5, aquí se ve cómo los objetivos son entradas para el sistema, y el resultado es lo que se obtiene en su salida. Este control es llamado de lazo abierto, pues no se tiene ninguna noción del verdadero valor de la salida, es decir ¿fue este mayor, menor o igual a lo deseado?

Figura 2.5. Esquema básico de un sistema de control.

Fuente: [17].

Lo anterior se soluciona haciendo un control de lazo cerrado, donde la salida se retroalimenta para tener una percepción del resultado final y ajustar la entrada del sistema

acordemente. Este lazo se cierra normalmente con un sensor eléctrico. En la Figura 2.6 se muestra un diagrama de control de lazo cerrado mucho más robusto que el anterior. Se observa que la entrada del sistema es una velocidad angular deseada ω_r, la cual se compara con la salida gracias a un sensor de velocidad. La comparación de las señales genera una diferencia o error ω_e, que es lo que ingresa directamente al sistema de control. El control genera una salida hacia la planta, que es aquello que se desea controlar, en este caso un motor. Se pueden tener además perturbaciones externas al sistema que afecten el comportamiento de la planta, como lo es en este caso el torque externo T_L. Finalmente, se produce la salida ω, que será de nuevo comparada con la velocidad angular deseada, y así sucesivamente.

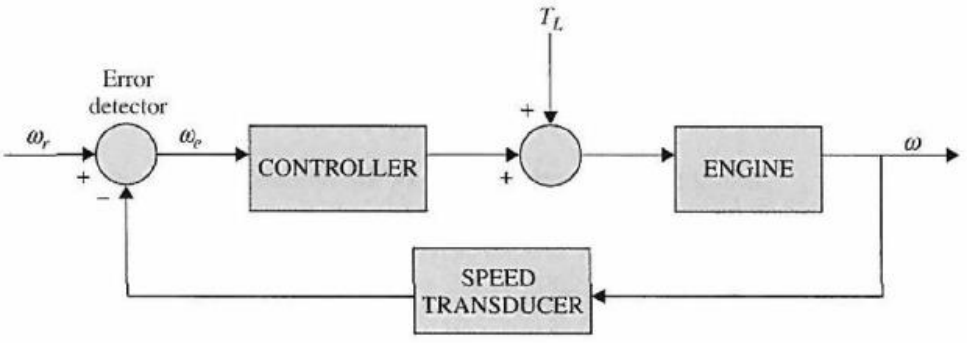

Figura 2.6. Lazo de control cerrado para la velocidad de un motor.

Fuente: [17].

2.3.1 Control *PID*

El control *PID* es un tipo específico de controlador. Es decir, el cuadro denominado "controller", o controlador por su nombre en español, de la Figura 2.6, podría ser del tipo *PID*. Este control se caracteriza por utilizar el error de tres maneras diferentes: una manera proporcional que solamente escala el error, una integral que integra el error a lo largo del tiempo, y una derivativa que analiza la manera en la que el error cambia a lo largo del tiempo; de ahí su nombre de *PID*. Todas estas componentes se suman y así se obtiene la señal final del control. La Figura 2.7 muestra un diagrama similar al de la Figura 2.6, pero esta vez el controlador es específicamente del tipo *PID*. Cabe resaltar que este diagrama está en el dominio *S* y no en el dominio del tiempo, por eso la parte integral del control se denota como una fracción de denominador *s* y la parte derivativa como una multiplicación de *s*.

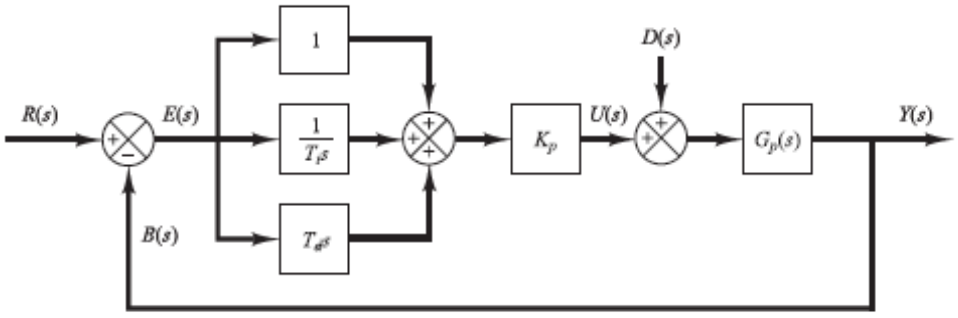

Figura 2.7. Diagrama de bloques de un sistema de control con un controlador *PID*.

Fuente: [18].

Los controladores *PID* tienen principalmente dos grandes ventajas: 1. Computacionalmente hablando, el control requiere poco procesamiento; 2. Es muy sencillo de incluirlo en una rutina de microcontrolador, pues incluso ya existen bibliotecas que realizan la parte de programación del control. Por lo tanto, es una técnica muy utilizada para controlar diversos tipos de sistemas.

Para la determinación de las constantes K_P, T_I y T_D, o bien K_P, K_I y K_D, los métodos que se utilizan son muy variados. Algunos de estos métodos son: sintonización Ziegler-Nichols, lugar de las raíces y herramientas matemáticas como MATLAB. El primero es una serie de reglas para obtener primeras aproximaciones de valores que harán que la salida del sistema se comporte aceptablemente, pero está lejos de ser un procedimiento rigurosamente matemático, sino más uno empírico y de prueba y error. El segundo se basa en conocer a profundidad la teoría de control que hay detrás de los controladores, y con base en el comportamiento que se desea de la planta, obtener los valores de las constantes. El último es de gran utilidad para iterar rápidamente en la implementación, pues permite observar diferentes comportamientos de una manera gráfica muy sencilla.

Es importante mencionar que, indiferentemente del método que se utilice, cada constante aporta algo en específico al control y esto siempre se debe tener presente. La contante de proporcionalidad K_P afecta la rapidez con la que el control actúa sobre la planta; la contante K_I contribuye a eliminar el error en estado estacionario en la respuesta del sistema; y finalmente, la contante K_D mejora el transitorio de la respuesta de la planta ante el control. Por lo tanto, siempre existe la posibilidad de realizar ajustes en las constantes de manera experimental utilizando los

criterios anteriores para mejorar un aspecto en específico que se desee de la respuesta de la planta, ya sea su transitorio, la rapidez de respuesta o el error en estado estacionario.

2.4 Electrónica en el control de velocidad de robots móviles terrestres

Para lograr implementar las técnicas que se han explicado en los apartados de odometría y direccionamiento diferencial es necesario poder controlar con exactitud la velocidad de las ruedas del robot. Muchas veces esto se logra con controles PID, explicados anteriormente, pero es necesario conocer un poco más a profundidad la manera en la que se manipula la electrónica que conlleva este control. Los motores CD son los más comunes de encontrar en estos robots, por lo que conocer sus principios básicos es fundamental. El *PWM* es la técnica más utilizada para modificar la velocidad de giro de un motor conectado a un microcontrolador; y los codificadores son los sensores que se utilizan para cerrar el lazo de control de velocidad o posición de un motor.

2.4.1 Motor CD

Existen diferentes tipos de motor CD, pero los que se usan principalmente para aplicaciones de robótica móvil de bajo costo, como es la solución que se desarrolló en este proyecto de graduación, suelen ser motores CD de imán permanente y con escobillas. En la Figura 2.8 se muestra un esquema simplificado de la manera que funciona un motor CD. Aquí, una fuente de voltaje externa energiza una bobina por medio de las escobillas eléctricas. Estas escobillas están colocadas de manera tal que cuando la bobina conmute de lado, la corriente fluya en la dirección que seguirá causando movimiento en la misma dirección que el motor ya giraba.

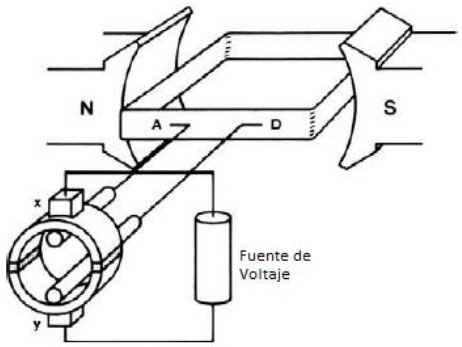

Figura 2.8. Representación simplificada de un motor CD con escobillas.

Fuente: [19].

En el estator del motor (su chasis) es donde se encuentran los imanes permanentes que se encargan de generar un campo magnético continuo, que al interactuar con las bobinas por las que fluye corriente, generan una fuerza electromotriz que causa un torque en el rotor del motor (la parte que gira) y de esta manera se produce el giro constante en el eje a la salida del motor [19]. Al variar el voltaje en las terminales del motor, la fuerza electromotriz aumenta, y consecuentemente hace girar más rápido al motor.

2.4.2 *PWM*

La modulación por ancho de pulso (*PWM*, por sus siglas en inglés) es un método para incrementar o disminuir la tensión percibida por una carga al apagar y prender continuamente una fuente fija de voltaje. La Figura 2.9 muestra un gráfico en el tiempo de esto.

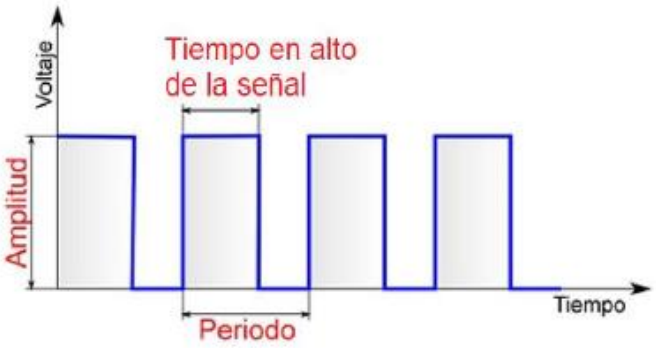

Figura 2.9. Representación gráfica en el tiempo de un *PWM*.

Fuente: [14].

Básicamente lo que se realiza es generar un tren de pulsos cuadrados que tienen una amplitud constante, pero se varía el tiempo que ellos permanecen apagados dentro de un periodo fijo. A mayor tiempo en alto, mayor tensión promedio habrá en la salida del *PWM*. Cabe destacar que la mayor tensión que se puede entregar es igual a la amplitud del tren de pulsos, y en este caso el tiempo en el que están apagados es nulo, por lo que se estaría entregando tensión de manera continua a la carga. Finalmente, el *PWM* es de gran utilidad para variar de manera sencilla con un microcontrolador la tensión que percibe un motor CD, esto se logra utilizando un número dentro del microcontrolador que va de 0 a 255, siendo 255 el máximo valor de tensión.

2.4.3 Codificadores

Los codificadores son sensores que se utilizan para medir desplazamiento angular en un eje. Pueden ser utilizados para medir desplazamiento lineal también, pero estos son muy poco comunes. Estos sensores aplican principios ópticos o magnéticos para detectar cuándo un punto específico de una rueda pasa por un punto específico del sensor, y así logran cuantificar lo que ha girado la rueda. Si estos pulsos se cuentan en el tiempo, se puede saber la posición de la rueda en determinado momento, o bien, saber cuánto se mueve por unidad de tiempo, o sea su velocidad.

Existen dos tipos de codificadores, los absolutos y los incrementales, siendo los últimos los más comunes de encontrar a bajo costo y los que se explicarán en este apartado. En la Figura 2.10 se observa a la izquierda la "plantilla" que usa un codificador óptico angular e incremental. Las rejillas que están en negro bloquean un haz de luz, lo que hace que el sensor detecte una señal baja de voltaje. El tamaño de cada rejilla dictará la resolución del codificador, pues a más delgada la rejilla, más pulsos hay en un giro total.

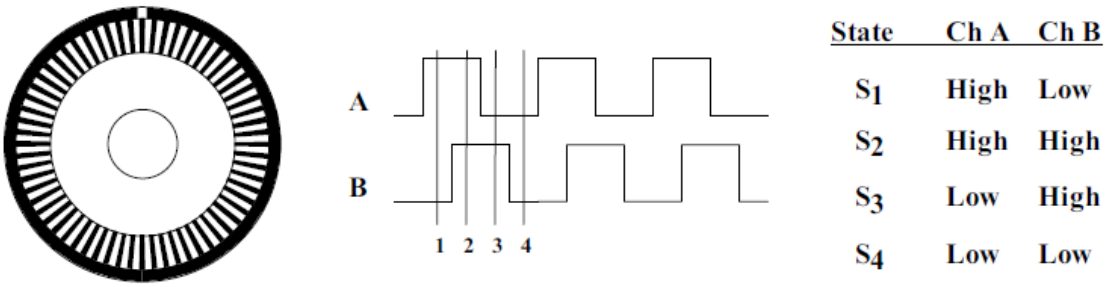

Figura 2.10. Salida eléctrica de un codificador óptico incremental angular y sus respectivos estados lógicos.

Fuente: [20].

Estos sensores suelen tener dos canales, o sea, dos haces de luz, donde estos están desfasados entre sí a como lo muestra la Figura 2.10. La función que cumple este desfase es el de saber la dirección en la que gira la rueda y así cuantificar no solo la magnitud, sino también la dirección el movimiento.

2.4.4 Microcontrolador

Es imposible pensar en el control de un robot móvil sin cuestionar ¿qué se encargará de leer las señales de los sensores, interpretarlas y realizar cálculos con base en ellas para luego enviarle acciones de control a los actuadores del robot? Esto lo realiza un microcontrolador. Williams define en [21] un microcontrolador como un sistema electrónico que integra diferentes subcomponentes como: un CPU, relojes, contadores, convertidores analógicos/digitales, entre otros; y que además se ha diseñado para ser utilizado en un sistema embebido.

Un microcontrolador se puede encontrar en arquitecturas de 8, 16 y 32 bits. Esto lo que significa es que la memoria interna del microcontrolador estará estructurada utilizando registros[4] de esa cantidad de bits, permitiendo guardar 1, 2 o 4 bytes de información. Esto normalmente afecta al resto de componentes electrónicos del microcontrolador, y en general a toda su arquitectura, pues todo el sistema de control interno de datos debe estar en capacidad de manejar la cantidad de bits de los registros.

En este proyecto se utilizó la placa Feather M0, que posee un microcontrolador de 32 bits SAMD21 M0 con arquitectura ARM. Posee una gran cantidad de periféricos, entre ellos un contador de tiempo real (*RTC*, por sus siglas en inglés). Este contador posee diferentes modos de operación, permitiendo funcionar como un contador de 32 bits que utiliza un reloj externo al que usa el resto del microcontrolador, llegando a una frecuencia máxima de 32.76 kHz y facilitando al microcontrolador operar en modo sueño. Además, permite activar interrupciones en diferentes circunstancias de conteo.

2.5 Unidad de medición inercial (*IMU*)

Las unidades de medición inercial son sensores que se utilizan para medir la pose del objeto que las contenga. Esto lo logran integrando un acelerómetro y un giroscopio, y en algunos casos, un magnetómetro. Ellas funcionan dando información relevante sobre los tres ejes que las componen: X, Y y Z; dependiendo del sensor, así será la información que brindan.

Con el acelerómetro es posible detectar cambios de gravedad en cualquiera de los tres ejes. Utilizando el giroscopio se puede medir la velocidad angular de los tres diferentes ejes. Y

[4] Un registro es un componente electrónico utilizado en las memorias que se encarga de guardar información en forma de 1 o 0.

finalmente, con el magnetómetro se puede medir el cambio del campo magnético en los tres ejes, que, por consiguiente, se puede traducir a un desplazamiento angular en los tres ejes, pues el campo magnético terrestre tiene una orientación invariante para una locación dada. Toda esta información es recolectada e integrada (cuando lo amerita) para obtener el desplazamiento en un espacio 3D de un cuerpo móvil.

2.5.1 Errores en las *IMU*

Como cualquier otro sensor, una *IMU* presentará errores, y para poder entenderlos, es necesario estudiar individualmente cada una de las partes que componen la *IMU*. Además, por cada sensor, los errores se deben catalogar en diferentes tipos.

Iniciando con el magnetómetro, es esencial entender que este componente es sensible a perturbaciones magnéticas externas que afectan el valor del campo magnético medido. Estas perturbaciones pueden ser divididas en dos: hierro duro e hierro suave [22], por su traducción del inglés. La primera es causada por imanes o materiales férreos que generan un campo magnético fijo o que varía levemente con el tiempo; esto generará un corrimiento fijo en la medición del sensor. Los errores de hierro suave son los más importantes, pues muchas veces afectan mucho la medición. Estos se dan por fluctuaciones magnéticas producidas por materiales ferromagnéticos en la vecindad del sensor, o incluso se pueden producir por el mismo sensor. De manera más completa, el modelo del error de un magnetómetro es el siguiente:

$$R = M_m \cdot S \cdot SI \cdot (M + O + n) \tag{2.3}$$

Aquí M_m representa un error de alineamiento entre los ejes del sensor y los ejes del cuerpo móvil, S es un error de escalamiento en cada eje, SI el error de hierro suave, O el error de hierro duro y n el ruido general del sensor. M es la medición cruda del sensor y R la medición corregida. Este modelo es matricial, pues abarca los tres ejes simultáneamente. Por lo tanto M_m, S y SI son matrices 3x3 y R, M, O y n son vectores de tres componentes. Es común ignorar los errores de escalamiento y alineamiento, pues con lograr eliminar el error de hierro suave y duro es suficiente para obtener una buena medición del sensor. En otras ocasiones se unen M_m, S y SI en una sola matriz y se obtiene una única matriz con factores de escalamiento.

Para los acelerómetros y giroscopios el modelo es igual, pero se simplifica. Estos dos sensores no poseen errores del tipo hierro, si no que solamente se ven afectados por el escalamiento

y el corrimiento que afectan comúnmente a los sensores [23], por lo que SI desaparece de la ecuación (2.3), y el factor O tiene otra interpretación física. También, al igual que el magnetómetro, se ven afectados por un factor de alineamiento; esto porque los tres ejes del sensor pueden no calzar a la perfección con los del cuerpo móvil. Lo anterior se refleja en la Figura 2.11, donde se ve que los ejes sensitivos del sensor (magnetómetro, acelerómetro o giroscopio) no necesariamente coinciden con los ejes coordenados del cuerpo móvil debido a imperfecciones en el montaje del sensor.

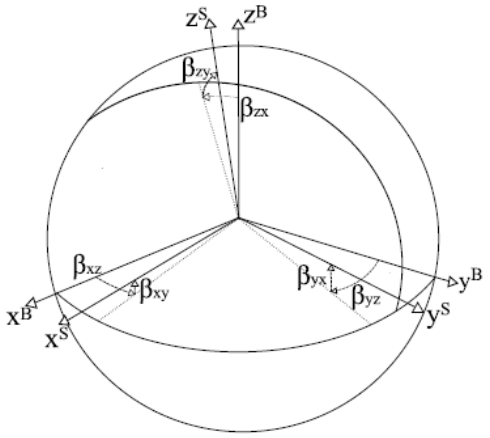

Figura 2.11. Diferencia entre los ejes sensitivos del sensor x^S, y^S, z^S, y ejes coordenados del cuerpo x^B, y^B, z^B.

Fuente: [23].

2.5.2 Calibración de las *IMU*

Cada uno de los tres sensores de la *IMU* debe pasar por una calibración diferente. En lo que respecta a los magnetómetros, se debe corregir los errores de hierro suave y dulce. Debido a la naturaleza del sensor, es sencillo entender su calibración mediante la representación gráfica, por eso en la Figura 2.12 se muestra la información sin calibrar en rojo, la información calibrada en azul, y una esfera unitaria como referencia. Si se grafica la medición calibrada de un magnetómetro en un espacio 3D, esta debería representar una espera perfecta, centrada en (0,0,0) y de radio igual al campo magnético[5] terrestre del lugar donde está el magnetómetro. Sin embargo, el error de

[5] En caso de dividir las mediciones entre el campo magnético del lugar, entonces se obtendrá una esfera de radio unitario como la de la Figura 2.12.

hierro suave deforma las mediciones en un elipsoide, y el error de hierro duro traslada el centro de las mediciones a otro punto coordenado.

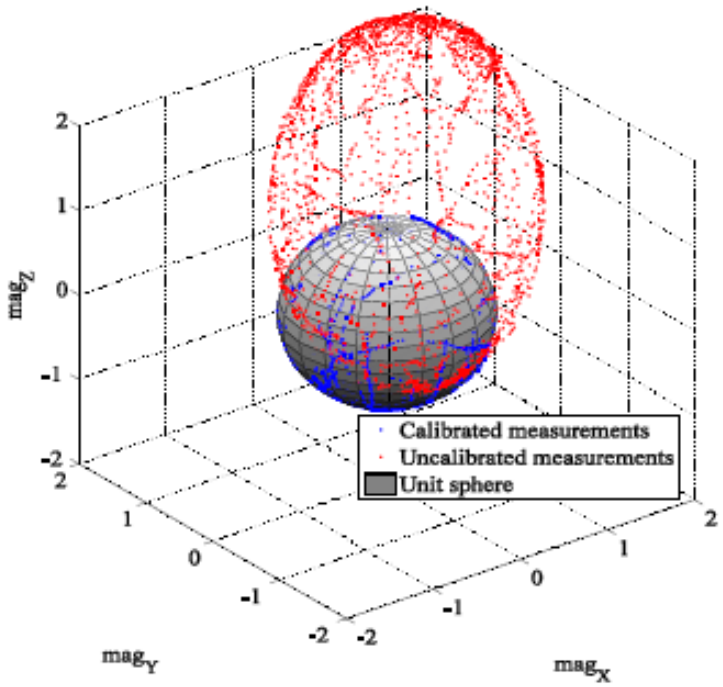

Figura 2.12. Efecto de la calibración de un magnetómetro sobre la representación gráfica de la información medida.

Fuente: [24].

Es común encontrar dos métodos para calibrar una *IMU*, el manual y el automático. En el primero el usuario debe aplicar métodos matemáticos que utilizan aproximamientos elipsoidales[6], para así calcular paso a paso el elipsoide que más se ajusta a los datos crudos. El segundo es básicamente lo mismo, pero por medio de programas de computadora que toman las mediciones crudas y arrojan las nueve constantes de escalamiento y las tres constantes de desplazamiento necesarias para calibrar las mediciones.

Tedaldi y Salehi et al. muestran métodos complejos para calibrar un magnetómetro en cualquiera de sus tres ejes [23], [24]. De manera interesante, Skvortzov et al. explican en [25] otro

[6] Estas aproximaciones representan matemáticamente las elipses que se generan al proyectar el elipsoide en los tres planos de un espacio coordenado 3D. Es usual que utilicen la representación general de las elipses como cuadráticas: $AX^2 + BXY + CY^2 + DX + EY + F = 0$

método matemático que simplifica el proceso. En este reporte científico se apoyan en el hecho de que el magnetómetro se utilizará solamente como compás para un robot móvil, por lo que solo es necesario calibrar el eje vertical del sensor; de esta manera la calibración pasa de ser con un elipsoide 3D a ser con una elipse en el plano XY.

Para calibrar un giroscopio y un acelerómetro es común utilizar equipo mecánico complejo, pues se necesita una mesa de tres ejes que permita rotar con exactitud y precisión los sensores alrededor de sus ejes X, Y y Z, debido a esto puede ser costoso lograr calibrarlos. Al igual que el magnetómetro, también existen métodos computacionales que se vuelven independientes de equipo físico, como el algoritmo que explica Tedaldi en [23], pero estos pueden exigir el uso de métodos matemáticos complejos. Por otro lado, existen aproximaciones mucho más sencillas como la explicada por Looney en [26], que permite calibrar los sensores pero con cierta incertidumbre involucrada.

2.5.3 Filtro de Madgwick

El filtro de Madgwick es un filtro de orientación aplicado a *IMUs* que contengan giroscopios, acelerómetros y magnetómetros. Este algoritmo se basa en las mediciones de los tres sensores para generar una percepción común de la rotación del cuerpo móvil sobre sus tres ejes, es decir, unifica la medición de los tres sensores en una única estimación del ángulo de giro del cuerpo móvil en X, Y y Z [27]. Esto genera un aumento significativo en la exactitud de la medición, pues aprovecha las ventajas de cada sensor para subsanar las debilidades de estos.

Este filtro presenta dos ventajas en específico: (1) hace uso de aritmética escalar, lo que disminuye en gran medida el costo computacional que tiene el algoritmo; (2) el filtro ha sido diseñado para que en situaciones de baja frecuencia de muestreo su rendimiento no se vea afectado. Estas dos características hacen que el filtro sea excelente para implementarse en microcontroladores, ya que normalmente poseen bajos recursos computacionales. El filtro ha sido extensamente probado contra otros filtros importantes, como el de Kalman, y ha demostrado incluso ser superior a este en las mediciones que obtiene.

2.6 Tecnologías de comunicación inalámbrica

Una manera amplia y general de definir las comunicaciones inalámbricas es la de toda comunicación entre dispositivos, móviles o no, que utilizan el espectro electromagnético para

transferir información [28]. Este tipo de comunicación involucra muchas tecnologías, pues incluye desde la comunicación bluetooth entre un celular y un reloj inteligente, hasta la conexión que hay entre las naves espaciales y la tierra; por lo que es un campo muy amplio.

Las tecnologías se pueden agrupar en tres grandes secciones:

- Redes inalámbricas de área personal (*WPAN*, por sus siglas en inglés).
- Redes inalámbricas de área local (*WLAN*, por sus siglas en inglés).
- Redes inalámbricas de área extendida (*WWAN*, por sus siglas en inglés).

Conforme se avanza en los tres grupos, la tecnología es más compleja y robusta, tiene mucho mayor alcance y posee más ancho de banda. La primera categoría contempla el Bluetooth, ZigBee, módulos de RF, NFC, IrDA, entre otras. La *WLAN* abarca principalmente el protocolo IEEE 802.11 y todas sus variaciones. Y finalmente, la *WWAN* abarca todos los protocolos robustos para telefonía móvil como el GSM, GPRS, 4G/LTE; además de las satelitales y los radioenlace [28].

2.6.1 Topologías en redes de comunicación

La topología de una red hace énfasis a la configuración de los enlaces entre los nodos de la red. Estos nodos pueden ser cualquier sistema electrónico que tenga la capacidad de enviar y/o recibir información [29]. Existen diversas formas de acomodar una red, cada una tiene ventajas y desventajas. La Figura 2.13 muestra tres tipos de topología: la básica punto a punto, en la cual solo existen dos nodos que transfieren la información; la topología tipo bus, donde todas las computadoras están conectadas por una única línea de información; y la de aro, donde todas están conectadas de manera tal que se genera un círculo.

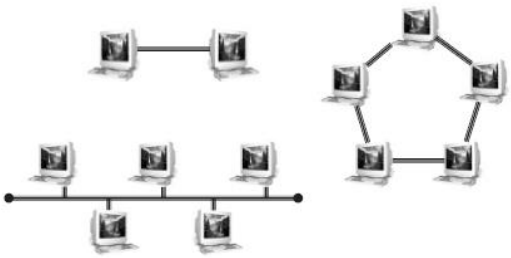

Figura 2.13. Topologías en redes de computadores: punto a punto, bus y aro.

Fuente: [29].

Para grandes cantidades de nodos es más común ver otros tipos de topologías, como la estrella, la de árbol y la de tipo malla. La topología estrella es del tipo punto a multipunto, donde un nodo maestro tiene acceso a *n* nodos esclavos, y todos se comunican solamente con él. Un buen ejemplo de esto es el router de internet de un hogar, donde todos los dispositivos (computadoras, celulares, TV y demás) se conectan a un único dispositivo, el router. La topología árbol es una extensión de la estrella, donde los nodos maestros ahora pueden hablar con otros nodos de comunicación. Esto último lo ejemplifica la Figura 2.14.

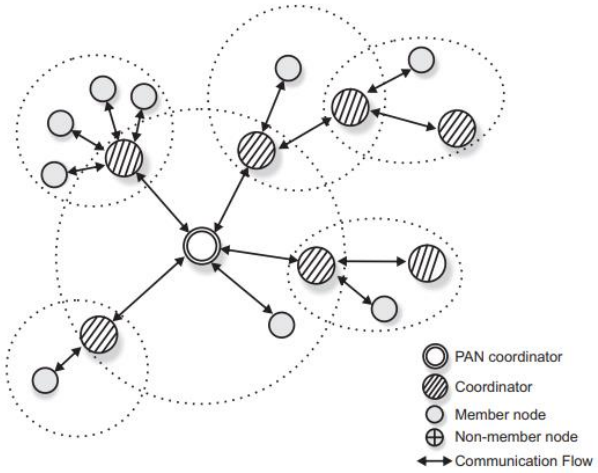

Figura 2.14. Topología de comunicación tipo árbol con agrupaciones internas.

Fuente: [30].

Finalmente, la topología malla busca conectar a todos los miembros de una red de manera que la información pueda fluir lo más homogéneo posible. Existen dos tipos de malla: la parcial, donde no necesariamente todos los nodos están conectados a todos los nodos; y la completa, donde todos los nodos están conectados explícitamente a todos los demás nodos. Además, una topología del tipo malla puede ser estructurada, donde ya existen nodos de control predeterminados; o bien, puede ser del tipo "ad hoc", que es una malla descentralizada, donde no existe una estructura previamente definida.

2.6.2 Acceso múltiple por división de tiempo

Las redes de comunicación inalámbricas necesitan un tipo de sincronización que ordene el envío y recibimiento de los paquetes de información. Para lograr esto, es común aplicar técnicas

de multiplexación, que no solo sirven para que varios nodos accedan al mismo medio sin interferencia, sino que también pueden permitir mayores tasas de envío de datos.

Existen varios métodos de control de acceso al medio (*MAC*, por sus siglas en inglés) para redes inalámbricas, entre esos el acceso múltiple por división de frecuencia (*FDMA*, por sus siglas en inglés), el acceso múltiple por división de espacio, el acceso múltiple por división de código (*CDMA*, por sus siglas en inglés), y el acceso múltiple por división de tiempo (*TDMA*, por sus siglas en inglés) [29]; en esta sección se explicará el último.

La técnica de *TDMA* permite a varios dispositivos comunicarse entre sí a través de una única frecuencia sin hacer colisionar la información, esto lo logra a través de un acomodo en el tiempo de los nodos, permitiendo a un único nodo hablar a la vez [29]. La Figura 2.15 muestra cómo cada dispositivo tiene un periodo de tiempo T_S para emitir información, y una vez que todos los dispositivos tuvieron su tiempo para transmitir información, el ciclo se reinicia.

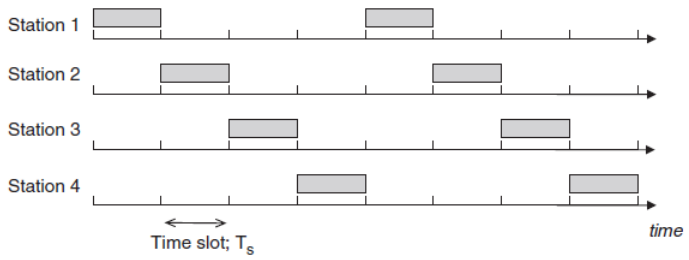

Figura 2.15. Acomodo en el tiempo de diferentes nodos de comunicación gracias a la técnica *TDMA*.

Fuente: [29].

El *TDMA* es extensamente utilizado en redes de sensores inalámbricos (*WSN*, por sus siglas en inglés). Una *WSN* consiste en una gran cantidad de pequeños sensores de bajo costo que son alimentados por baterías, y que poseen sistemas electrónicos de comunicación para transmitir la información que recolectan [31]. Pueden enviar los datos con cualquier tecnología de comunicación, ya sea *WiFi*, Bluetooth u otras, pero sí es esencial que la transmitan automáticamente.

Algunos ejemplos concretos de *TDMA* en *WSN* y otros campos son los siguientes: Bhandari y Gautam lo utilizaron en [32] para comunicar un pequeño sistema multirobot; Gong et al. en [31]

modificó el *TDMA* para hacerlo más versátil, extendiendo sus aplicaciones a redes de sensores donde estos no siempre están monitorizando, sino que lo hacen solo cuando hay ciertos eventos específicos; y Che et al. lo utilizaron en [33] para crear una red de sensores submarinos, dejando clara la enorme versatilidad de esta técnica.

2.6.3 Sincronización de tiempo por medición del retardo

Para poder utilizar la técnica de *TDMA*, los nodos deben ser capaces de sincronizarse de manera inalámbrica en el tiempo, para que así todos puedan respetar los periodos de tiempo para enviar información. Lograr esto requiere el uso de algoritmos de sincronización de reloj. Estos algoritmos deben de tomar en cuenta las cuatro etapas de retardo que existen al enviar un mensaje inalámbricamente: el procesamiento en el transmisor, el acceso al emisor, la propagación del mensaje, y el procesamiento en el receptor. La Figura 2.16 muestra esto de una manera secuencial.

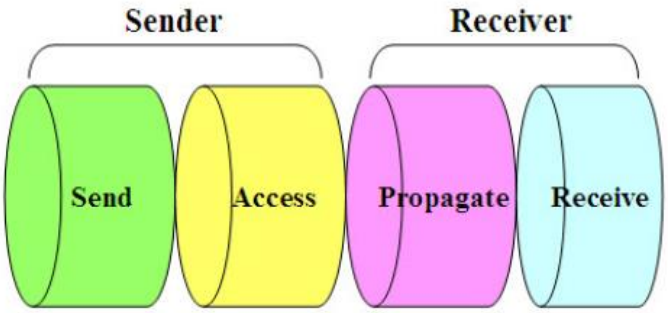

Figura 2.16. Descomposición del retardo de un mensaje inalámbrico en el tiempo.

Fuente: [34].

Existen diferentes algoritmos para sincronizar relojes, Rhee et al. explican en [34] varios de ellos, como: sincronización de referencia por broadcast[7], protocolo de sincronización de tiempo para sensores, sincronización de tiempo por medición de retardo (*DMTS*, por sus siglas en inglés), servicios de sincronización de reloj probabilístico y otros. El *DMTS* es un excelente protocolo para dispositivos de baja capacidad de procesamiento, pues este requiere solamente una transmisión de

[7] Un mensaje de tipo broadcast es aquel en el que un nodo envía el mensaje para todos los demás nodos que puedan escucharlo, sin importar el ID identificador de los nodos. Análogamente, es como gritar para que muchas personas escuchen un mensaje al mismo tiempo, en vez de ir persona por persona diciéndolo.

broadcast para sincronizar toda la red, además de que el algoritmo no requiere de aritmética compleja para ser implementado.

El *DMTS* consiste en lo siguiente: el líder de la comunicación envía un mensaje tipo broadcast que contiene el valor de su reloj, todos los demás nodos reciben el mensaje y lo procesan, cada nodo utiliza un cálculo matemático para reajustar su reloj de forma tal que ahora el nuevo reloj sea lo más cercano posible al del líder [35]. La ecuación que define el nuevo reloj de los nodos es la siguiente:

$$t_R = t + n \cdot \tau + (t_2 - t_1) \tag{2.4}$$

En la ecuación (2.4), t_R representa el nuevo reloj del nodo esclavo, t es el valor del reloj del líder, n es la cantidad de bits a transmitir en el mensaje y τ la velocidad de transmisión del medio y t_2 y t_1 son estampas de tiempo que utiliza el receptor. t_1 es colocado justo cuando el receptor se percata de que tiene un mensaje para recibir, y t_2 se coloca una vez que el mensaje ha sido procesado, de esta manera se compensa el tiempo de procesamiento en el receptor. El término $n \cdot \tau$ puede ser determinado empíricamente, pues no siempre es tan sencillo de calcular con la descripción dada anteriormente.

3 Desarrollo de la solución

3.1 Consideraciones de diseño

En la búsqueda de soluciones que lograran satisfacer no solo los objetivos del presente proyecto de graduación, si no también lo esperado por la coordinación del proyecto de investigación PROE E1F2, fue necesario plantear una serie de requerimientos de diseño que delimitaran la manera en la que se desarrollaría el control automático de la navegación y el sistema de comunicación del enjambre. A continuación, se presenta la lista de consideraciones que se utilizaron para plantear las soluciones.

1. El proyecto PROE E1F2 posee un diseño de prototipo de robot móvil, el cual tiene definido tanto la estructura mecánica como los componentes electrónicos. Por lo tanto, la solución propuesta se debía apegar al uso de este sistema ya desarrollado. La Figura 3.1 muestra el robot con el que se trabajó, este posee: dos motores CD modelos N20 con sus respectivos codificadores y ruedas, un puente H modelo TB6612FNG, un convertidor de voltaje CD-CD modelo PowerBoost 500, el microcontrolador Feather M0 con un subsistema de comunicación RF con el chip RFM69HCW y un circuito integrado de *IMU* que posee el chip LSM303 (acelerómetro y magnetómetro) y el L3GD20H (giroscopio). Entonces, se debe aclarar que en este proyecto de graduación no se diseñó ni el sistema electrónico ni mecánico del prototipo, solamente se utilizó el establecido en el proyecto de investigación.

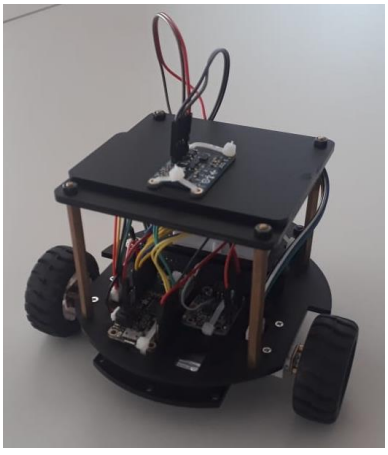

Figura 3.1. Prototipo de robot móvil del proyecto PROE E1F2.

Fuente: elaboración propia.

2. Según las necesidades del proyecto de investigación, era necesario diseñar un control que se ajustara fácilmente a nuevas versiones y modificaciones del prototipo del robot móvil y que no fuese un diseño de control específico para el prototipo existente.

3. No era necesario que el control automático que se propusiera contemplara las pequeñas variaciones mecánicas como peso, distancias entre ruedas y el diámetro de las ruedas, que pueden existir entre la construcción de un robot y otro.

4. Era deseable que, en la medida de lo posible, los componentes que se integraran de la *IMU* se calibraran de manera automatizada e in situ. Lo anterior porque se debe tener presente que un enjambre de robots puede poseer decenas de agentes y, por cada uno de ellos, se deberá calibrar la *IMU*.

5. A pesar de que es ideal que la topología de comunicación de un enjambre de robots permita que la información fluya de manera homogénea, se aceptarían propuestas que no necesariamente cumplieran esto, siempre y cuando las soluciones planteadas fueran escalables a topologías más complejas y dinámicas que sí alcancen este objetivo.

6. El diseño del sistema de comunicación se centró en las rutinas de control y de manejo de la información. Es decir, en este proyecto no se consideraron aspectos de la potencia del sistema de comunicación, ni del diseño de la antena de este.

7. El fin último del sistema de comunicación era que un nodo de la red funcionara como base central, y estuviese conectado a una computadora portátil (*PC*, por sus siglas en inglés) y de alguna manera transfiera la información recopilada a esta. Por lo tanto, era necesario un método que transmitiera la información o datos recolectados a una computadora.

8. El diseño del sistema de comunicación debía contemplar que PROE E1F2 pretende construir un enjambre de aproximadamente 15 robots para el año 2020.

3.2 Desarrollo del control automático de la navegación

Para empezar, se realizó un estudio y análisis de las metodologías de control automático para la navegación de robots móviles, enfocándose principalmente en aspectos como: entorno, objetivos y fundamentos teóricos utilizados. Con el objetivo de conocer y comprender a detalle las implementaciones o propuestas actuales relacionadas con esta temática.

En los reportes científicos fue común encontrar investigaciones avanzadas en la teoría de control automático para robots móviles, por ejemplo: Aneesh en [36] presenta un controlador no lineal y un controlador adaptativo; Anushree y Prasad en [37], Kanavama et al. en [38] y Sandeep et al. en [39] proponen el uso de un controlador de Lyapunov; y He en [40] presenta nuevamente un control no lineal, pero esta vez linealizado por retroalimentación de estados.

Por otro lado, Kanavama et al. en [38] y Díaz y Kelly en [41] proponen un controlador para seguir trayectoria e indican que existe una estrecha relación entre controlar la orientación del robot y el seguir una trayectoria. Además, Sandeep et al. en [39] y He en [40] presentan diagramas sobre el esquema general del control de posición y orientación utilizado, dejando una clara idea de cuál puede ser una metodología a utilizar; en la Figura 3.2 se muestra una de ellas.

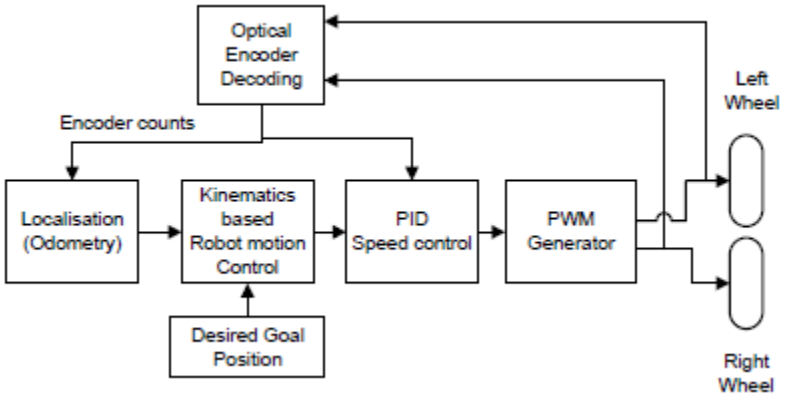

Figura 3.2. Diagrama de control de posición implementado en un robot móvil diferencial.

Fuente: [39].

En la Figura 3.2 se puede observar cómo mediante un modelo cinemático del robot móvil se controla la posición de este y, que de alguna manera, este control de posición se relaciona con un control *PID* de velocidad que envía pulsos *PWM* a los motores. Luego, mediante codificadores,

se retroalimenta información al control de velocidad y a un subsistema de localización por odometría, que posiblemente luego envíe un error al control de posición. También se puede observar en el diagrama que se establece una posición deseada. Indiferentemente del tipo de controlador que se utilice, esta es una metodología robusta que engloba tanto el control de la velocidad, como el control de la pose del robot.

Por otro lado, Siegwart y Nourbakhsh en [12] plantean una relación entre las derivadas de la pose del robot, o sea el vector $\left[\frac{dx}{dt}, \frac{dy}{dt}, \frac{d\theta}{dt}\right]^T$, y la velocidad lineal y angular del chasis del robot móvil, y luego realizan una transformación de coordenadas cartesianas a coordenadas polares. Finalmente, Lynch y Park en [13] muestran la cinemática usada en el control de varios tipos de robots móviles, entre esos el robot móvil diferencial. Coincidentemente, esta bibliografía utiliza la misma metodología que el esquema de la Figura 3.2. Combina el uso de modelos de cinemática inversa y directa junto a teoría de control automático, y enlistan las limitaciones que tiene el control que proponen.

Una vez revisadas las bibliografías ya mencionadas, se tomó la decisión de utilizar la teoría planteada por Lynch y Park en [13]. Esto por varias razones: 1. El control automático a utilizarse en una primera versión del prototipo de robot móvil debería ser una configuración conocida y ampliamente investigada en la literatura para que la primera iteración sea exitosa, aspecto que cumple lo presentado allí; 2. La teoría se simuló en MATALAB, y su desempeño fue exitoso; 3. Esta metodología permite realizar el control de posición y velocidad a la vez; 4. La manera en que se plantea la estrategia de control hace que sea factible a futuro adaptarla a otras configuraciones de robots móviles, aspecto de interés para los investigadores de PROE E1F2. A continuación se explicará a detalle el método.

Primeramente, es necesario especificar que, para el control de un robot móvil no holonómico, existen tres casos de control diferentes:

1. **Estabilización de una pose:** dada una pose del robot deseada p_d, llevar el error de $p_d - p(t)$ a cero conforme el tiempo avanza.

2. **Seguimiento de una trayectoria:** dada una trayectoria deseada $p_d(t)$ para el robot, llevar el error de $p_d(t) - p(t)$ a cero conforme el tiempo avanza. Cabe resaltar que,

como se denota, una trayectoria depende del tiempo. Por lo que controlar la trayectoria del robot no solo implica que el robot siga una serie de puntos dados, sino que lo haga en un tiempo específico.

3. **Seguimiento de un camino o ruta:** dado un camino deseado $p_d(s)$, hacer que el robot lo siga. La diferencia con la trayectoria es que acá no importa cuánto durará el robot haciéndolo, sino que más bien se le dice la velocidad lineal y angular que se desea que mantenga durante el recorrido.

Dada la naturaleza del problema que se buscó resolver en este proyecto de graduación, el caso de control que se debe atacar es el primero. Sin embargo, Lynch y Park presentan en [13] un teorema de control que estipula que si un sistema de control automático posee menos vectores de control que variables a controlar, este sistema no es controlable por un control lineal e invariante en el tiempo como el que ellos presentan. Debido a que el robot móvil posee solamente dos entradas de control (dos ruedas diferenciales) y tres variables a controlar (la pose), es imposible mediante un control clásico atacar este problema y se debió buscar una lógica diferente para controlar el robot.

Ahora, según lo dicho en párrafos anteriores, gracias a la investigación previa que se hizo, se notó que existe una estrecha relación entre el camino a seguir por el robot y la orientación final que tendrá, pues si por ejemplo se le indica al robot que la ruta a seguir es avanzar en línea recta, teóricamente no debería existir un cambio en la orientación durante el movimiento. Dado este principio se decidió implementar el tercer caso de control y planificar las rutas de forma tal que la orientación se controle implícitamente.

Dicho esto, entonces lo primero que se debe hacer es pasar de controlar la pose del robot, a controlar solamente la posición de él. Esto se logró escogiendo un punto *P* dentro del chasis del robot para así tener una coordenada (x_P, y_P) que controlar. La única restricción es que el punto *P* no puede estar dentro del eje de rotación de las ruedas debido a las ecuaciones cinemáticas que gobiernan al robot diferencial que se expondrán más adelante.

La Figura 3.3 muestra un diagrama con diferentes variables de interés del robot. Existe el ya mencionado punto *P* que posee coordenadas globales (x_P, y_P), pero que, dentro del marco de referencia local del robot denotado por $\hat{x}_b$ y $\hat{y}_b$, posee coordenadas (x_r, y_r). Además, se tiene el

punto coordenado (x, y) que corresponde al centro del eje de giro de las ruedas (la Figura 2.4 muestra la relación entre las ruedas y este punto) y por lo tanto, a la ubicación del eje de giro del robot. Este punto está representado en el marco de referencia global. Finalmente, se observa la orientación del marco de referencia local respecto al marco de referencia global (ver Figura 2.4) denotado mediante $\emptyset$, que como se había dicho anteriormente, representa también la orientación del robot en el espacio.

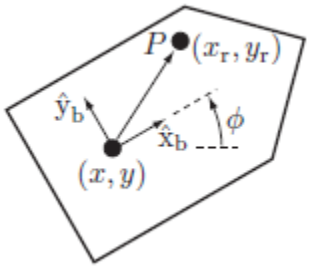

Figura 3.3. Variables de interés del robot móvil para el control automático.

Fuente: [13].

La estrategia de control a seguir es la de un controlador con retroalimentación proporcional del error, donde el error se obtiene entre comparar el par coordenado que se desea para el punto *P* y el que actualmente posee el robot. Por lo tanto, se obtiene la siguiente ecuación de control:

$$\begin{bmatrix} \dot{x}_P \\ \dot{y}_P \end{bmatrix} = \begin{bmatrix} k_P(x_{Pd} - x_P) \\ k_P(y_{Pd} - y_P) \end{bmatrix} = \begin{bmatrix} k_P x_{Pe} \\ k_P y_{Pe} \end{bmatrix} \tag{3.1}$$

Esta ecuación escala proporcionalmente el error dado en las coordenadas del punto *P* y define que esto es igual a la derivada de cada una de esas coordenadas. Ahora se debe definir matemáticamente estas coordenadas, para así poder analizar qué es lo que este término representa. Por lo que usando la Figura 3.3 se tiene que:

$$\begin{bmatrix} x_P \\ y_P \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \cos\emptyset & -\operatorname{sen}\emptyset \\ \operatorname{sen}\emptyset & \cos\emptyset \end{bmatrix} \begin{bmatrix} x_r \\ y_r \end{bmatrix} \tag{3.2}$$

Ahora, si se deriva la ecuación (3.2) se obtiene entonces:

$$\begin{bmatrix} \dot{x_P} \\ \dot{y_P} \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} + \dot{\emptyset} \begin{bmatrix} -\operatorname{sen} \emptyset & -\cos \emptyset \\ \cos \emptyset & -\operatorname{sen} \emptyset \end{bmatrix} \begin{bmatrix} x_r \\ y_r \end{bmatrix} \tag{3.3}$$

Y sabiendo que: la velocidad angular ω del robot es igual a $\dot{\emptyset}$ y que la velocidad lineal del robot se expresa como $(\dot{x}, \dot{y}) = (v \cos \emptyset\,, v \operatorname{sen} \emptyset)$. Se obtiene de la ecuación (3.3) entonces:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \frac{1}{x_r} \begin{bmatrix} x_r \cos \emptyset - y_r \operatorname{sen} \emptyset & x_r \operatorname{sen} \emptyset + y_r \cos \emptyset \\ -\operatorname{sen} \emptyset & \cos \emptyset \end{bmatrix} \begin{bmatrix} \dot{x_P} \\ \dot{y_P} \end{bmatrix} \tag{3.4}$$

Nótese el término x_r en el denominador, es por esto que anteriormente se dijo que el punto *P* no puede estar sobre el eje de giro de las ruedas, pues el término x_r valdría cero. Finalmente, para mantener la representación matricial, las ecuaciones (2.1) y (2.2) se convierten en:

$$\begin{bmatrix} u_L \\ u_R \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & \frac{-d}{r} \\ \frac{1}{r} & \frac{d}{r} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3.5}$$

Por lo tanto, utilizando el control de retroalimentación proporcional de la ecuación (3.1), y mediante la cinemática inversa del robot móvil diferencial presente en las ecuaciones (3.4) y (3.5), es posible obtener la velocidad angular necesaria en las ruedas del robot para lograr llevar a cero el error existente en el control. La referencia por seguir (x_{Pd}, y_{Pd}) variará con el tiempo, pero es indicada al robot; esto quiere decir que no se calcula, solo se le indica al sistema de control. Pero las coordenadas actuales (x_P, y_P) del punto *P* sí deben estimarse de alguna manera.

Dicho lo anterior es donde entra entonces el modelo cinemático directo del robot, el cual se compone de la siguiente forma. Se reitera que la velocidad angular ω del robot es igual a $\dot{\emptyset}$ y la velocidad lineal del robot se expresa como $(\dot{x}, \dot{y}) = (v \cos \emptyset\,, v \sin \emptyset)$. Entonces, sustituyendo esto en la ecuación (3.5) y reacomodando, se obtiene que:

$$\begin{bmatrix} \dot{\emptyset} \\ \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} -\frac{r}{2d} & \frac{r}{2d} \\ \frac{r}{2} \cos \emptyset & \frac{r}{2} \cos \emptyset \\ \frac{r}{2} \operatorname{sen} \emptyset & \frac{r}{2} \operatorname{sen} \emptyset \end{bmatrix} \begin{bmatrix} u_L \\ u_R \end{bmatrix} \tag{3.6}$$

Si se integra en el tiempo el término $\dot{\emptyset}$ de la ecuación (3.6) se obtiene que:

$$\emptyset = \int_{t_1}^{t_2} \dot{\emptyset}\, dt \tag{3.7}$$

Se puede ahora ingresar los valores a la izquierda del igual de las ecuaciones (3.6) y (3.7) en la ecuación (3.3) e integrar el lado izquierdo de la ecuación (3.3) para así finalmente obtener las coordenadas actuales del punto (x_P, y_P) usando las velocidades angulares u_L y u_R de los motores. De esta manera se cierra el lazo de control, generando con las ecuaciones anteriores la retroalimentación de las coordenadas del punto *P*. La Figura 3.4 resume lo que se explicó anteriormente, pero hay que hacer énfasis en que la orientación no se controla, sino que se escoge el camino seguido por (x_d, y_d) de forma tal que se alcance el $\emptyset_d$. Además, se aclara que en esta imagen (w_d, w_i) es lo mismo que (u_R, u_L).

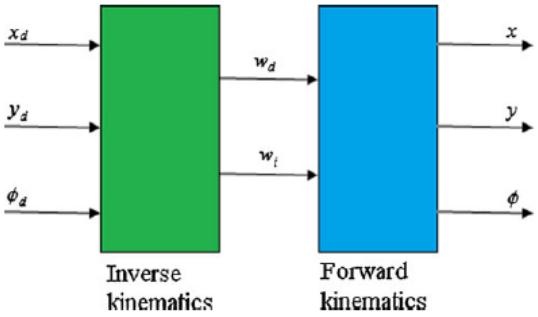

Figura 3.4. Representación gráfica del flujo de información dentro del control.

Fuente: [15].

Como se dijo anteriormente, este control se simuló en MATLAB, y los resultados fueron positivos. En la Figura 3.5 se puede observar el diagrama de bloques que se realizó en *Simulink*. En el siguiente apartado se hablará con detalle sobre los resultados obtenidos de esta simulación, pero aquí se aclara que gracias a esos resultados fue que se vio la viabilidad de esta solución.

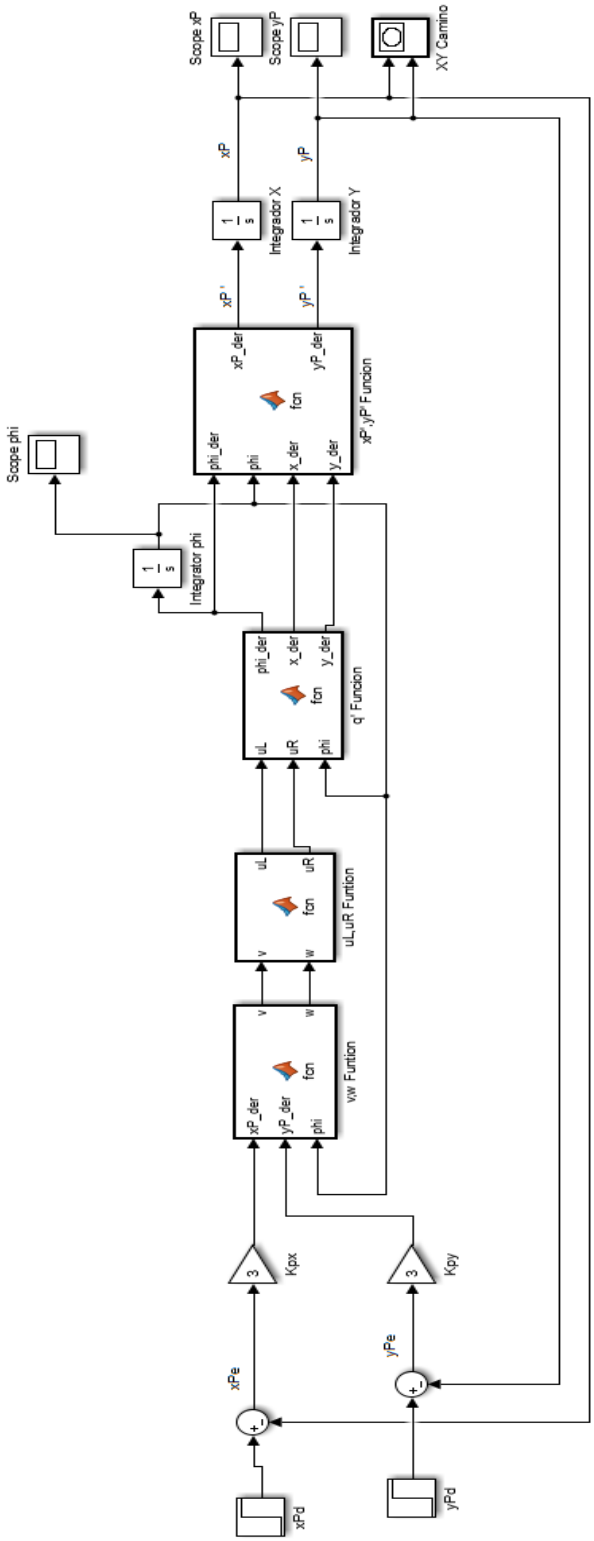

Figura 3.5. Simulación del control automático propuesto en *Simulink*.

Fuente: elaboración propia.

Según lo ya explicado, se puede notar que nada asegura que las ruedas del robot giren exactamente a las velocidades angulares que se calcularon con la cinemática inversa, por lo que la cinemática directa simplemente no puede utilizar el mismo valor que arroja el bloque anterior para obtener la posición actual. Si no que en medio de esas dos componentes del control entra un subsistema: un controlador que se encarga de hacer que las ruedas sigan lo mejor posible la velocidad angular que dicta la cinemática inversa; y codificadores en cada motor, que realimentan la velocidad angular real de los motores al sistema de control de velocidad y a la cinemática directa del robot.

Por lo tanto, también se debió implementar un controlador *PID* para finalizar el diseño del control automático del robot móvil. Se escogió este tipo de controlador debido a que regular la velocidad de giro de un motor CD no es una tarea compleja y es un controlador sencillo de implementar porque existen bibliotecas para microcontroladores que facilitan su integración. El control final del sistema es el mostrado en la Figura 3.6, donde se puede observar que se hace una clara distinción entre U_L, U_R; u_L, u_R; y ω_L, ω_R.

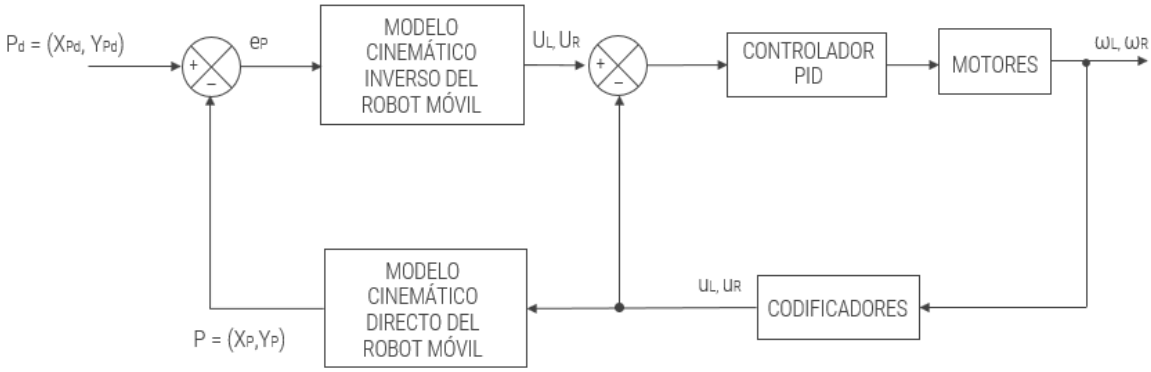

Figura 3.6. Esquema general del control propuesto.

Fuente: elaboración propia.

Aquí, U_L y U_R representan las velocidades angulares teóricas que debe seguir el robot para eliminar el error, y se comparan con la verdadera velocidad angular que mide el codificador para generar las señales de referencia que entran al controlador *PID*. Luego, ω_L y ω_R representan las velocidades reales que tienen los motores CD. Y finalmente, u_L y u_R son las velocidades angulares que los codificadores miden. Las tres se denotan como variables diferentes porque en todo proceso

de control y medición de una señal existirá un error asociado, por lo que no es posible asegurar que los motores seguirán a la perfección la señal de referencia; o bien, que los codificadores medirán sin ningún error la velocidad angular de las ruedas.

Para poder desarrollar el controlador *PID* se realizaron experimentos con una planta, específicamente los motores del prototipo de robot móvil de PROE E1F2 que se muestra en la Figura 3.1. Se midió la respuesta de velocidad angular de ambos motores ante una entrada de *PWM* con valor de 255, de esta manera se simuló la respuesta a una entrada de escalón unitario.

Con la herramienta *System Identification* de MATLAB se procesaron los datos obtenidos del experimento para obtener un modelo discretizado y representativo de cada motor. Una vez obtenidas las plantas a controlar, se utilizó la herramienta *PID Tuner* de MATLAB para obtener las constantes k_P, k_I y k_D del controlador que mejor se ajustaron al comportamiento deseado para los motores del robot.

Según lo descrito anteriormente, se observa que el control del robot móvil en sí ya estaba solucionado, pero aún se debía crear la lógica que construyera la ruta o camino que el robot seguiría, para controlar no solo la posición, si no también implícitamente la orientación. La estrategia que se desarrolló inicialmente fue indicar al control un par ordenado (x_{Pd}, y_{Pd}) lo suficientemente lejos del punto *P* actual y que obligara al robot a perseguir una orientación ortogonal a la que poseía originalmente. Cabe resaltar además que el punto P que se decidió controlar del robot fue la de un punto ubicado en el "para choques" de la estructura, lo más alejado posible del centro del robot. La Figura 3.7 muestra una representación gráfica del punto que se controló del robot y la ruta que se esperaría para un caso hipotético donde se le indicó que tuviera una nueva orientación de 90° en sentido horario.

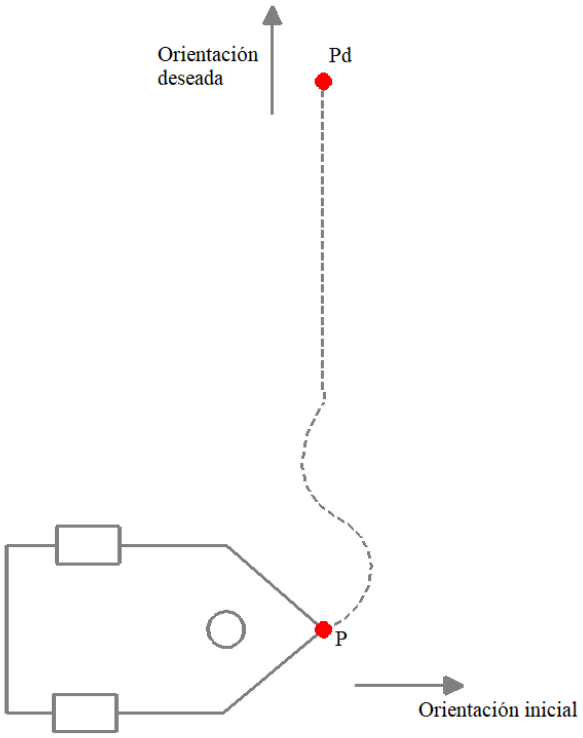

Figura 3.7. Primera estrategia de control implícito para la orientación del robot móvil.

Fuente: elaboración propia.

Se escogió que los puntos estuvieran colocados de esa manera porque el proyecto PROE E1F2 utiliza un algoritmo de caminata al azar, donde el robot escoge entre izquierda, hacia adelante o derecha, de manera aleatoria. Esto quiere decir que el robot solo tendrá que dar giros de ± 90° o seguir directo. Por lo que la primera estrategia se planteó tratando de unificar dos movimientos en uno: girar y avanzar, en ese orden. Por ejemplo, como ya se dijo, con la Figura 3.7 se pretendía un giro a la izquierda del robot de 90° en conjunto con un movimiento de avanzar.

Se esperaba que el punto deseado, al estar lejos del actual, permitiera al robot seguir un camino tal que lo llevara a alinearse con la nueva orientación deseada, sin embargo, esta estrategia tenía dos problemas: 1. Obligaba al robot a hacer dos movimientos en uno, sin permitir una escogencia aleatoria del segundo, 2. No necesariamente el robot seguiría ese camino, pues en realidad su orientación no se controlaba con nada, y podría simplemente no terminar con la orientación que se esperaba, y de hecho no lo hizo. Por lo tanto, no fue la solución óptima, no

obstante, permitió verificar que los controles de posición y velocidad funcionaron correctamente, y, por ende, conllevó a implementar una nueva metodología basada en estos resultados.

Para solventarlo, se crearon dos rutinas en el microcontrolador que ayudarían al control a cumplir su función más eficientemente:

1. **Generador de rutas:** cumplió la función de que dado un punto deseado (x_{Pd}, y_{Pd}), generaba una serie de puntos entre el inicio y el final para tener un movimiento más preciso; es decir, el robot no daría un único gran "paso", si no que daría varios y pequeños hasta llegar a su destino. Permitió también variar la resolución del camino generado desde el código fuente, relacionando la velocidad que se quiso que el robot siguiera con la cantidad de puntos a trazar en la ruta.
2. **Vigilante de la orientación final:** mantuvo un registro de la orientación teórica que debía poseer el robot según su historial de movimientos (0°, 90°, 180° o 270°; dado el algoritmo de exploración) y lo comparaba con la orientación final que arrojaba el control automático. En caso de existir un error entre la orientación que tenía el robot y la que debería tener, llamaba al generador de rutas para que construyera un arco que se encargara de eliminar ese error.

El generador de rutas funcionó creando dos clases de rutas diferentes: arcos y líneas. El arco se generó usando la cantidad de grados que se quiso que el robot girara, y se obligó al punto *P* a seguir una circunferencia con radio igual a la distancia entre el centro de giro del robot (el punto (x, y) de la Figura 2.4) y el punto *P*. Las líneas se crearon partiendo de la orientación inicial del robot, generando puntos que se mantuvieran sobre una línea recta en el plano 2D que mantuvieran la misma orientación inicial del robot.

La Figura 3.8 y la Figura 3.9 ejemplifican ambos casos. En la Figura 3.8 se representa un arco de 90°, pero la rutina puede generar arcos con cualquier ángulo de apertura, que es lo que se hace con la rutina de vigilancia de la orientación final. Cabe también recordar que, gracias a la rutina de generación de caminos, en realidad el punto *P* deseado no es solo el final, sino que es un punto que cambia con el tiempo y pasa por todo el camino que se desea que el robot siga. No obstante, en la Figura 3.8 y la Figura 3.9 solo se muestra el punto *P* deseado final.

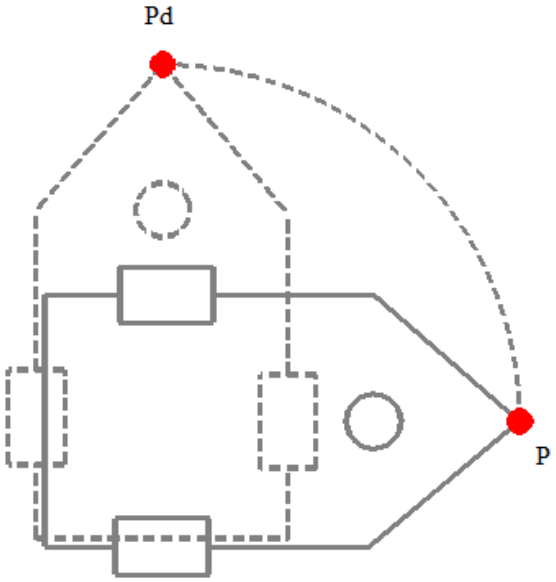

Figura 3.8. Generación de un camino tipo arco para el robot móvil diferencial.

Fuente: elaboración propia.

Figura 3.9. Generación de un camino tipo línea para el robot móvil diferencial.

Fuente: elaboración propia.

La estrategia de control completa que se implementó finalmente en el microcontrolador del robot móvil es la mostrada en la Figura 3.10. Cabe resaltar que el bloque llamado "control del robot móvil" resume todo el diagrama mostrado en la Figura 3.6. Aquí, al subsistema de generación de rutas se le indica un punto final deseado y este crea una serie de puntos que le entrega al control

automático. El control automático sigue estos puntos hasta llegar al par coordenado final deseado, y una vez allí, el subsistema de vigilancia de orientación final verifica el error que hay entre la orientación que teóricamente debería tener el robot y con la que terminó realmente el robot luego de la acción del control automático. Una vez obtenido el error en la orientación, el subsistema de vigilancia determina si debe o no crear un nuevo punto final deseado *P* que estará lo más cerca posible del punto *P* final deseado anterior y eliminará el error que hay en la orientación. En caso de que el error sea sobrepase un umbral, entonces este nuevo punto se pasa otra vez al subsistema generador de rutas para que calcule el nuevo movimiento y así se elimina el error.

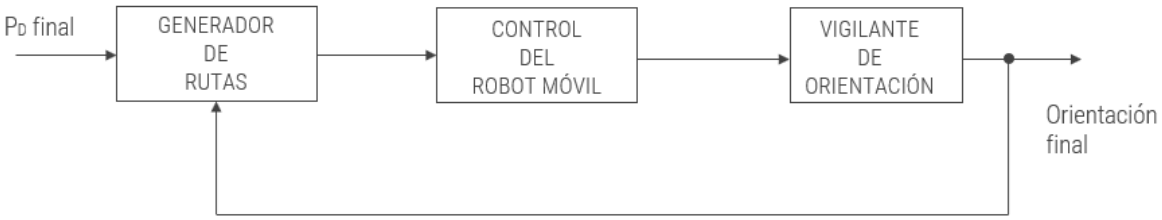

Figura 3.10. Diagrama del control de la orientación.

Fuente: elaboración propia.

Se comenta además que el control automático posee una rutina interna que le permite saber cuándo alcanzó el estado estacionario. Esta rutina lo que realiza es que una vez que se indicó al control automático el último punto deseado (o sea, el punto final), aumenta un contador en software cada vez que la señal de referencia hacia los motores es igual a 0 o cada vez que la posición actual del robot está dentro de un umbral de error previamente determinado en comparación con el punto final deseado. Para los movimientos de desplazamiento en línea recta, el margen de error en la coordenada x_P y y_P es de 0.5 cm, mientras que para los movimientos en arco es de 5°.

3.3 Integración de funcionalidades de la *IMU* para mejorar el error de la orientación

Lo que se buscó al incluir una *IMU* como un extra para el control de la navegación, era poder tener un segundo método de supervisión del error en la posición u orientación, pues como se vio en el apartado del marco teórico, la odometría está sujeta a errores que pueden afectar la

estimación de cualquiera de estas dos variables con el tiempo. Como ya se ha mencionado, la orientación es la variable más crítica en el sistema por su falta de control, por lo tanto, la *IMU* se implementó con miras a mejorar el error final de control en la orientación para casos donde existan errores del tipo deslizamiento.

En un inicio se planteó una solución que buscaba satisfacer las necesidades de diseño de manera sencilla. Se propuso calibrar solamente el magnetómetro, y hacerlo solo en el plano 2D sobre el que se desplaza el robot, el *XY*. De esta manera, se buscó crear una brújula digital que estimara la orientación de la *IMU* usando el campo magnético terrestre. Para esto se debió idear un método de calibración que tomara en cuenta los errores en magnetómetros explicados en la sección de Errores en las *IMU* y que además tuviera posibilidad de ser automatizable.

Para cumplir lo anterior, se investigaron diferentes métodos de calibración de magnetómetros y se decidió implementar una combinación de las estrategias explicadas por Skvortzov et al. en [25] y por Halir y Flusser en [42], las cuales se apoyan en la teoría matemática explicada por Fitzgibbon et al. en [43]. Se desarrolló un algoritmo que hacía al robot realizar un giro de 360° sobre su propio eje mientras recolectaba una gran cantidad de mediciones del magnetómetro y luego las ponderaba para hacer un grupo final de 20 datos promediados. Este grupo de puntos contendría los errores de hierro suave y duro, representando entonces una elipse en el plano *XY*. Utilizando el algoritmo descrito por Halir y Flusser, se procesaron en MATLAB los datos ponderados para obtener la información de la elipse que más se ajustara a ellos, y luego en el microcontrolador se utilizaba esta información para calibrar el resto de las mediciones del magnetómetro.

Esta primera implementación funcionó bastante bien y facilitaba lograr una automatización en el proceso, pero era muy susceptible al ruido magnético, como lo advierten Lee y Cai en [44]. Por lo tanto, funcionaba solamente en el lugar puntual donde se realizó la calibración, y si el robot se desplazaba a otra ubicación, el magnetómetro comenzaba a arrojar datos erróneos. Esto demostró que sí se podía obtener buena precisión y exactitud con el sensor de bajo costo que poseía el proyecto PROE E1F2, pero que utilizar solamente el magnetómetro no era viable.

Para eliminar la susceptibilidad al ruido magnético en una *IMU* existen varias técnicas, una de las más usadas es aplicar algoritmos de fusión de sensores. Con esto se busca apaciguar la debilidad de un sensor con la fortaleza de otro y viceversa. En este caso se unieron las mediciones

del acelerómetro, giroscopio y magnetómetro de la *IMU* mediante el filtro de Madgwick para crear una única medición de la orientación del robot móvil. La ventaja de esto es que al utilizar la medición del magnetómetro junto a la del giroscopio se obtiene un ángulo de giro bastante exacto que puede ser comparado con la orientación del robot móvil calculada por la cinemática directa. Además, el acelerómetro dota al sistema de medición de inmunidad ante inclinaciones en el plano de movimiento.

Los dos principales motivos por los cuales se escogió el filtro de Madgwick es porque: 1. No compromete al microcontrolador en cuanto a poder de procesamiento, pues sus operaciones matemáticas se pueden hacer con aritmética básica, 2. Su implementación es sencilla ya que el autor del filtro permitió su implementación en código abierto, por lo que existe una biblioteca ya lista para microcontroladores que permite una rápida integración del filtro.

Para implementar el filtro correctamente primero se calibraron los sensores. La biblioteca del filtro utiliza un método de calibración específico para el magnetómetro, que facilita el procesamiento interno en el algoritmo; este lo realiza mediante un programa de código abierto para *PC* llamado Motion Cal. Utilizando el programa nada más se debe rotar la *IMU* alrededor de todos los ejes hasta que la calibración se estabiliza y genera una matriz 3x3 y un vector 3x1, que corresponden a los términos $M_m \cdot S \cdot SI$ y a O,respectivamente, de la teoría vista en el capítulo anterior.

Para calibrar el giroscopio y el acelerómetro se creó un código que extrajo 1600 datos crudos de cada uno de los tres ejes de cada sensor mientras estos permanecían sin ningún estímulo en su entrada durante 2 minutos. El fin de lo anterior fue obtener el corrimiento de la medición en cada eje. Los 1600 datos se ponderaron en uno único dato por eje y por sensor, por lo que al final se obtuvieron los dos vectores O_a y O_g del respectivo modelo de error del giroscopio y acelerómetro. Debido a que la obtención de las constantes de escalamiento para estos dos sensores implica equipo complejo con el cual no se dispone, o algoritmos avanzados que se debían desarrollar, se decidió no obtener esas constantes de calibración y calibrar el giroscopio y el acelerómetro solamente con los dos vectores O_a y O_g. Cabe resaltar que para para obtener el corrimiento del eje Z del acelerómetro, se utilizó el valor teórico de la gravedad en Cartago, 9.74 m/s^2,

Luego de la calibración de los sensores, se debió corroborar que la frecuencia de muestreo que el microcontrolador era capaz de dar para el filtro era válida y permitía un correcto funcionamiento de este. Se realizaron mediciones de tiempo dentro del código y se determinó que, debido al procesamiento de las rutinas de control automático, el microcontrolador lograba llamar al filtro a una frecuencia de 36 Hz. El algoritmo está diseñado para funcionar a frecuencias de muestreo bajas, incluso a 10 Hz, por lo que tener una frecuencia de muestreo de 36 Hz no sería mayor problema. Una vez implementado, el algoritmo de Madgwick resultó tener una alta exactitud, llegando incluso a los 0.5° de error promedio si se tomaban las mediciones justo en el punto donde se calibró la *IMU*, y solamente presentó leves desfases en la medición al desplazar el sensor a otras ubicaciones (ver resultados en sección 4).

Con la *IMU* ya calibrada y funcionando en el microcontrolador, se desarrolló la lógica que ayudaría al control automático a mejorar la estimación de la orientación. Específicamente para reducir los errores no sistemáticos del tipo deslizamiento, como lo indica el objetivo específico número 2 de este proyecto. Para solventar esto se propuso replicar la rutina de vigilancia de la orientación final, pero ahora en vez de comparar la orientación que estima el control y la orientación teórica que debería tener el robot, se compararía la medición de la *IMU* contra la orientación teórica.

En la Figura 3.11 se observa el diagrama de flujo de la rutina de inicialización de la *IMU*.

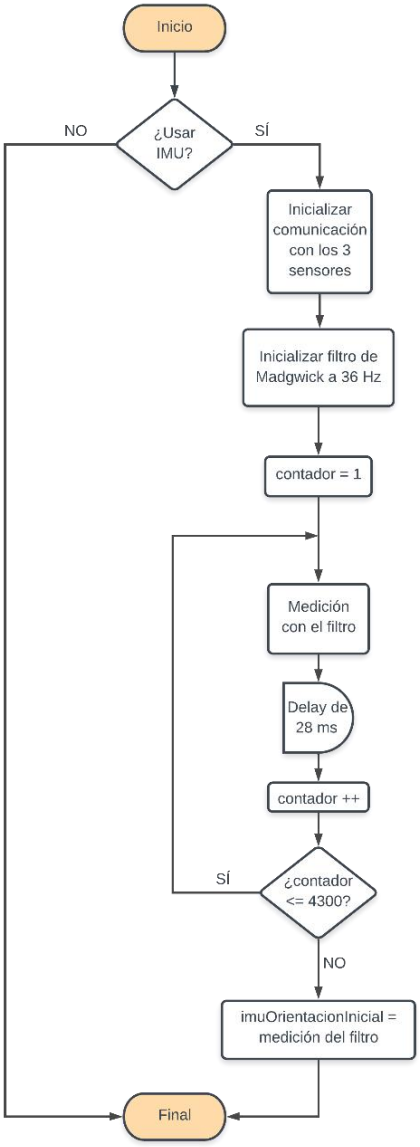

Figura 3.11. Diagrama de flujo para la rutina de inicialización de la *IMU*.

Fuente: elaboración propia.

Esta pequeña rutina lo que hace es poner en funcionamiento el filtro de Madgwick durante: $28\ \text{ms} \cdot 4300\ \text{repeticiones} = 120.4\ \text{s} \approx 2\ \text{minutos}$, que es lo que necesita el filtro para poder estabilizarse y comenzar a funcionar correctamente. Según la frecuencia de muestreo del filtro, así será la rapidez de convergencia inicial de este, pero en este caso se decidió utilizar la misma frecuencia de muestreo que la que se utilizaría en el código del robot, por lo que el tiempo de estabilización de la *IMU* al energizar el robot fue entonces de 2 minutos. Una vez que han pasado

los 2 minutos, la rutina almacena el último valor del filtro en una variable global, y así se tiene entonces la referencia inicial de la orientación medida por la *IMU* para ser comparada más adelante. Lo anterior se hace porque los sensores junto al filtro miden el ángulo final de manera puntual, y no un desplazamiento angular. Pero al tener el dato inicial se puede luego obtener cuál ha sido el desplazamiento angular del robot.

Una vez ya inicializado el sensor se pasa al código principal del microcontrolador donde se muestrea con el filtro a 36 Hz el valor de la orientación de la *IMU*, y una vez que se termina la acción de control, se llama a la rutina que corrige la orientación con el valor de la *IMU*. El diagrama de flujo de esta rutina se muestra a continuación en la Figura 3.12.

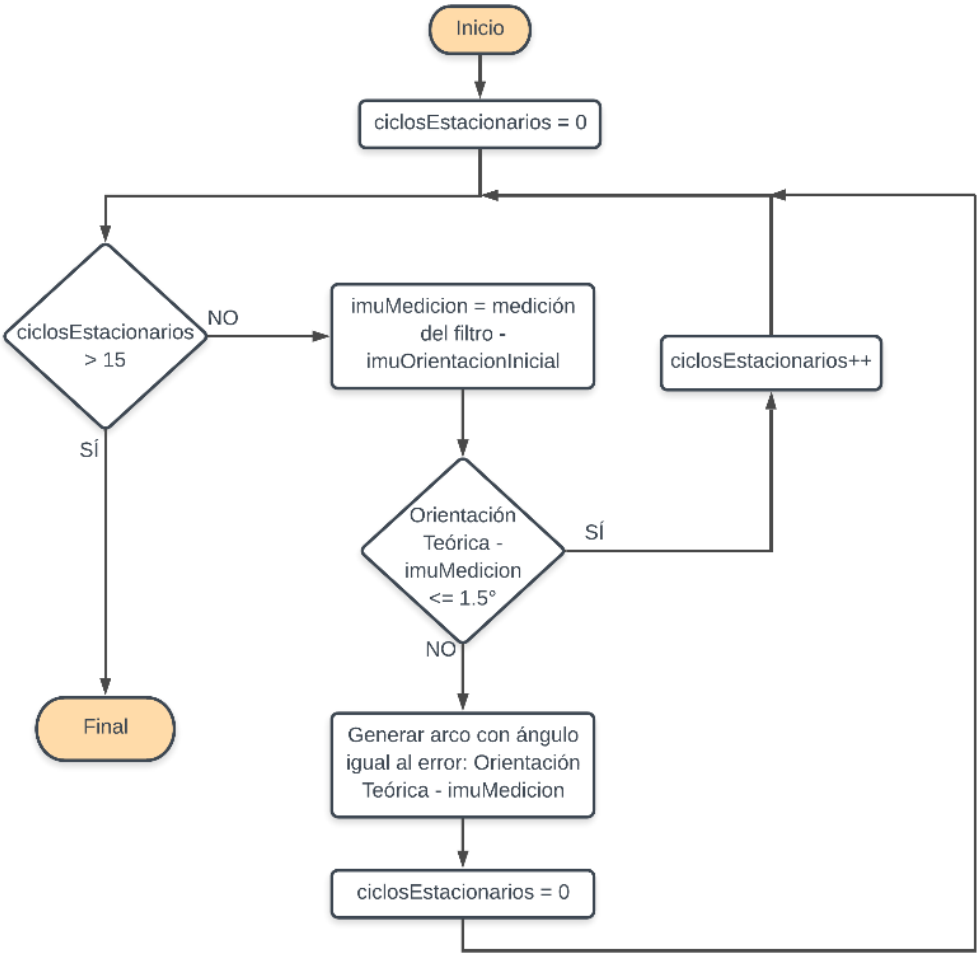

Figura 3.12. Diagrama de flujo del proceso de vigilancia de la orientación con la *IMU*.

Fuente: elaboración propia.

Este diagrama es básicamente un ciclo *while* que espera a que un contador sea mayor a 15 para finalizar la ejecución de la rutina. Internamente en el ciclo while se suma uno al contador cada vez que la medición del desplazamiento angular dado por la *IMU* y la orientación teórica que debería tener el robot tienen una diferencia menor o igual a 1.5°. Si esto no se cumple, se llama a la rutina de generación de un arco con el valor del error en la orientación, para que el control mueva al robot de manera tal que se elimine al máximo el error. Si esto sucede, se reinicia el valor del contador para que así se vuelva a comenzar la cuenta. Es importante mencionar que se intentó más de un método de integración de los datos de la *IMU* al control, pero el explicado aquí fue el que dio mejores resultados.

Por último, cabe resaltar que la *IMU* estuvo montada mecánicamente en el robot móvil utilizando separadores metálicos y una pequeña plataforma cuadrada de acrílico (ver Figura 3.1).

3.4 Sistema de comunicación

La solución escogida para el sistema de comunicación del robot abarcó tres niveles específicos: el nivel básico, la electrónica a utilizar para enviar la información; un nivel intermedio, la topología para el acomodo de los nodos en la red; y el nivel más complejo, el protocolo de ordenamiento del flujo de la información, pues al haber muchos emisores comunicándose al mismo tiempo, se debía crear una estrategia eficiente para que no existieran colisiones en los mensajes. Para proponer una solución, se tomó inspiración en la temática de redes de sensores inalámbricos (*WSN*, por sus siglas en inglés). Se hizo esto porque análogamente los robots son un conjunto de sensores que se desplazan en el espacio y actúan sobre el ambiente en el que se desenvuelven.

Como ya se habló anteriormente, en una *WSN* se puede utilizar diferentes tecnologías de comunicación, por lo que la primera decisión que se tomó fue utilizar la tecnología de RF. Esto por la naturaleza final del proyecto de investigación PROE E1F2, pues en un futuro se desea que el enjambre de robots se pueda utilizar para explorar escenarios de desastres o zonas de difícil acceso, y estos panoramas presentan limitaciones para algunas tecnologías de comunicación.

Las limitaciones anteriormente mencionadas que presentan las tecnologías son: es poco probable encontrar una conexión *WiFi* en zonas como las ya dichas, o en general, en zonas exteriores; también, esta tecnología suele consumir mucha potencia debido a su alta velocidad de transmisión, algo crítico para un sistema alimentado por batería. Por otro lado, la tecnología de

Bluetooth permite conexiones a distancias máximas de 10 m aproximadamente, y es común que permita solo una topología punto a punto, donde existe un máximo de dos nodos, lo que no es viable para un enjambre. Finalmente, la tecnología de radio frecuencia presenta tres ventajas claras: 1. Permite crear topologías variadas, permitiendo la adaptabilidad que un enjambre necesita para la transferencia de datos; 2. Puede alcanzar distancias de hasta 500 m con el horizonte despejado, o lograr comunicación en distancias de aproximadamente 50 m atravesando paredes de concreto, beneficioso para zonas con edificios; 3. El proyecto PROE E1F2 ya había experimentado con esta tecnología, y los microcontroladores que poseen tienen integrado a la placa del mismo un transmisor/receptor de RF modelo RFM69HCW.

El enjambre de robots requiere un protocolo *MAC* para organizar el envío y recibimiento de la información en la red. En este proyecto se escogió el *TDMA* por encima del *FDMA* y el *CDMA* porque este no requiere gran ancho de banda, como el *FDMA*, ni procesamiento computacional extra, como el *CDMA*. Además, como ya se vio, la técnica de *TDMA* ha tenido ya varias implementaciones exitosas en *WSN*. Sin embargo, hay que prestar atención al diseño de la red conforme esta escala, pues Gong et al. advierten en [31] que para ser escalable, se debe hacer mejoras al simple algoritmo de *TDMA*.

Como ya se vio en el capítulo 2 de este documento, para poder implementar el protocolo *TDMA* es necesario desarrollar una estrategia de sincronización de relojes entre los nodos de la red de comunicación. En este proyecto se utilizó la técnica de *DMTS* porque es la técnica que consume menos poder de procesamiento y es la más sencilla de implementar [34], además de que su desempeño en una topología centralizada multisalto de dos niveles ha sido probado con éxito [35].

La sincronización de la técnica *DMTS* se llevó a cabo usando el *RTC* periférico del microcontrolador del robot móvil. Este componente electrónico está presente en todos los microcontroladores del proyecto PROE E1F2, lo que permite ahorrar dinero y tiempo al no tener que implementar un *RTC* externo. Específicamente se escogió un *RTC* y no un Timer para poder dotar al robot de la posibilidad de entrar en modo sueño sin perder la sincronización con la red, esto en caso de que se llegue a necesitar esta opción más adelante.

Para lograr desarrollar efectivamente el protocolo *TDMA*, también fue necesario que existiera una manera sencilla de enviar y recibir mensajes, y no tener que construir las capas de más bajo nivel del protocolo de comunicación desde un inicio, sino solamente integrar rutinas ya

creadas. Para esto se decidió utilizar la biblioteca Radio Head Packet Radio Library [45], que ya posee integrado todos los programas para manipular fácilmente el RFM69HCW del microcontrolador del robot móvil. Esta biblioteca tiene rutinas que permiten protocolos tan complejos como se desee, pues tiene opciones incluso para implementar una topología tipo malla.

Se propuso una topología centralizada de un nivel, donde la base central tome el valor de su reloj *RTC* y lo envíe por mensaje broadcast a todos los robots de la red. Cada robot recibirá el tiempo del máster y ajustará su reloj *RTC* para estar sincronizado con el de la base central. Luego, los nodos usarán su ID para ubicarse en su ventana de tiempo respectiva, pues el nodo 1 irá en la ventana de tiempo 1 y así sucesivamente; la cantidad de ventanas es igual a la cantidad de nodos en la red sin contar la base y cada nodo ya posee esta información previamente. De ahí en adelante los nodos usarán su propio *RTC* para la sincronización, pues este se encargará de disparar una interrupción que le indique al nodo cuándo entró en su ventana de envío de mensajes y que, por lo tanto, debe transmitir su información. De esta manera se asegura entonces un orden en toda la red, solucionando así el problema.

Para lograr esta propuesta primero se debió configurar el *RTC* del microcontrolador SAMD21 utilizando los registros del sistema, esto con ayuda de la hoja de datos del microcontrolador [46]. Se debió hacer una configuración para el *RTC* de la base central que es receptora y otra configuración para los *RTC* de los nodos, que son transmisores.

Para los transmisores, la rutina lo que hace es: 1. Configura la potencia del reloj que se usará en el periférico; 2. Deshabilita el *RTC*; 3. Llama a la función que configura el reloj interno del *RTC* para utilizar una frecuencia de 1 kHz, se escogió esta frecuencia para tener una relación directa entre un conteo del reloj y 1 ms; 4. Se escribe directamente al registro de control del *RTC* para configurarlo al modo de operación correspondiente; 5. Se inicia la cuenta del reloj del *RTC* en cero; 6. Se escribe en el registro de comparación el valor contra el que se quiere comparar el reloj del *RTC* para más adelante generar una interrupción, inicialmente se pone un valor alto, pero no el deseado; 7. Se habilita la interrupción ante comparaciones positivas; 8. Se habilita la interrupción del periférico *RTC* dentro del controlador de vectores de interrupción del microcontrolador; 9. Se habilita de nuevo el *RTC*. La rutina para el receptor tiene los pasos 1, 2, 3, 4 y 9 ya descritos anteriormente.

Luego, se debió integrar la biblioteca Radio Head. De ella se utilizaron dos rutinas del código para el chip RFM69HCW, la *send()* y la *recv()*. Para más información de estas rutinas y la biblioteca en general, el lector puede acudir a [45]. Ahora bien, sí fue necesario modificar la biblioteca para agregar una rutina específica y modificar levemente la rutina *recv()*. Se creó la rutina *sendTimeStamp()* con base en la ya mencionada rutina *send()*. Esta rutina es básicamente un caso específico de *send()*, el caso específico donde el mensaje que se quiere enviar es la estampa de tiempo del máster.

Se debió hacer esta rutina específica por separado porque la idea de la metodología *DMTS* es que la estampa de tiempo del máster se cree lo más cerca posible de enviar el mensaje, para que así el tiempo que el microcontrolador tarda en ejecutar las instrucción siguientes afecte lo mínimo a la estimación del retardo de tiempo. Por esta misma razón fue que también se modificó la función *recv()* levemente, y es porque los receptores también deben colocar una estampa de tiempo lo más cerca posible a cuando recibieron el reloj del máster. Así que se le agregó un parámetro a los argumentos de la función: una variable tipo long que internamente la función puede modificar para guardar allí su reloj actual cuando recibió el mensaje.

Con esto implementado, ya se reciben y transmiten satisfactoriamente los valores t y t_1 de la ecuación (2.4), necesaria para completar el tercer paso de la implementación, sincronizar el protocolo. La estampa de tiempo t_2 la coloca el receptor una vez que decodificó el mensaje recibido por *recv()*, esto lo hace simplemente leyendo el registro que posee la cuenta actual de su *RTC*. Para determinar la parte final de la ecuación, el término $\tau \cdot n$, se realizó un experimento donde se envió el reloj del máster y se evalúo su sincronización con un nodo. Sin tener presente este término, el desfase entre los relojes siempre fue de 2 ms, lo que indicó experimentalmente que $\tau \cdot n = 2\ ms$. Esto también determinó que estos transmisores/receptores tardan ~2 ms enviando un mensaje de 8 bytes (4 bytes para la cabeza del mensaje y 4 bytes para el reloj del máster), o sea, una velocidad de 4 bytes/s.

Siguiente en el proceso de implementación, estuvo calcular la duración T_S de la ventana de tiempo (ver Figura 2.15) del protocolo *TDMA*. Para esto lo que se consideró fue: 1. La velocidad de transmisión es de aproximadamente 4 bytes/ms; 2. El chip RFM69HCW tiene una máxima capacidad de 60 bytes por mensaje. Entonces, se determinó que la ventana mínima de tiempo necesaria era de 25 ms. Un robot tendría máximo 15 ms para transmitir sus 60 bytes de

información, y habría 10 ms de tiempo entre el final de un nodo y el inicio de otro por aquello de que existieran desfases en los relojes. Estos 10 ms se determinaron experimentalmente, y en el siguiente capítulo se explicará su origen. Sin embargo, cuando el sistema se puso a prueba enviando la misma información que teóricamente enviaría un robot, se descubrió que la ventana de 25 ms no era suficiente, y podía ocasionar pérdida de información. Por lo que al final se decidió establecer la ventana de tiempo en 50 ms.

También se debió estructurar el acomodo del mensaje que cada nodo enviaría a la base central. Debido a que el nodo será un robot, y la intención del sistema de comunicación es ayudar a la transferencia de la información necesaria para digitalizar un mapa del escenario que los robots navegarán, la información que ellos deben enviar es: su ID, su posición y orientación, el tipo de obstáculo que encontraron y la ubicación de ese obstáculo relativo a ellos. El tipo de obstáculo va del 0 al 3: 0 para indicar que no hay obstáculo, 1 para indicar que es un obstáculo que impide avanzar, 2 para indicar que es un obstáculo de tipo temperatura y 3 para indicar que el obstáculo es un hueco frente a ellos. La ubicación del obstáculo se da en coordenadas polares como (distancia, ángulo), o bien (ρ, α); por lo que esto equivale a dos datos más. Así, el mensaje se vería como: $(ID, x_P, y_P, \emptyset, tipo, \rho, \alpha)$. Cabe resaltar que la posición y orientación del robot corresponden a la pose que este tenía cuando detectó el obstáculo.

Para poder enviar los datos anteriores, fue necesario crear la estrategia de composición y descomposición del mensaje. Primeramente, se debió definir el tipo de dato que tendría cada valor. La pose del robot y la ubicación del objeto deben ser flotantes, pues poseen decimales; mientras que el ID y el tipo de obstáculo podrían ser enteros de 1 byte. Dado que la mayoría son flotantes y estos poseen 4 bytes, se decidió enviar el ID y el tipo de obstáculo también como flotantes, para que el construir y deconstruir el mensaje fuera más sencillo para ambas partes. Así, tanto el receptor como el emisor deberían de segmentar el mensaje y extraer un dato cada 4 bytes siempre. La Figura 3.13 muestra la composición final del mensaje. Aquí se puede apreciar que se debe incluir al inicio del mensaje una cabeza de 4 bytes que va adherida debido a la biblioteca Radio Head, por lo que el mensaje final está constituido por un total de 32 bytes.

Protocolo RH	Información del robot				Información del obstáculo		
4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes	4 bytes
CABEZA	ID	x_P	y_P	$\emptyset$	$tipo$	ρ	α

Figura 3.13. Composición del mensaje enviado por los nodos.

Fuente: elaboración propia.

Finalmente, el último aspecto a mencionar es que se creó una interrupción en el código del transmisor que es la que se encargó de indicarle al nodo cuándo enviar la información. Esta rutina de interrupción se activaba cada vez que al transmisor le corresponde hablar y utiliza la rutina *send()* para enviar un mensaje, además actualiza la interrupción para la próxima ventana de tiempo del robot y se limpia la bandera de interrupción.

La Figura 3.14 muestra el diagrama de la lógica de comunicación de la base central, o bien, el receptor del sistema. Como se puede observar, el diagrama no tiene un final, esto es porque este diagrama no representa una subrutina, sino que es la rutina entera del microcontrolador. La base central primero se asegura de haber enviado su reloj 10 veces seguidas, solo si logra esto podrá pasar a la etapa de recibir mensajes. Una vez como receptor se quedará dentro de un ciclo *while* de manera permanente solo escuchando mensajes. Si un mensaje llega, lo guarda y lo imprime en pantalla, si no, sigue escuchando.

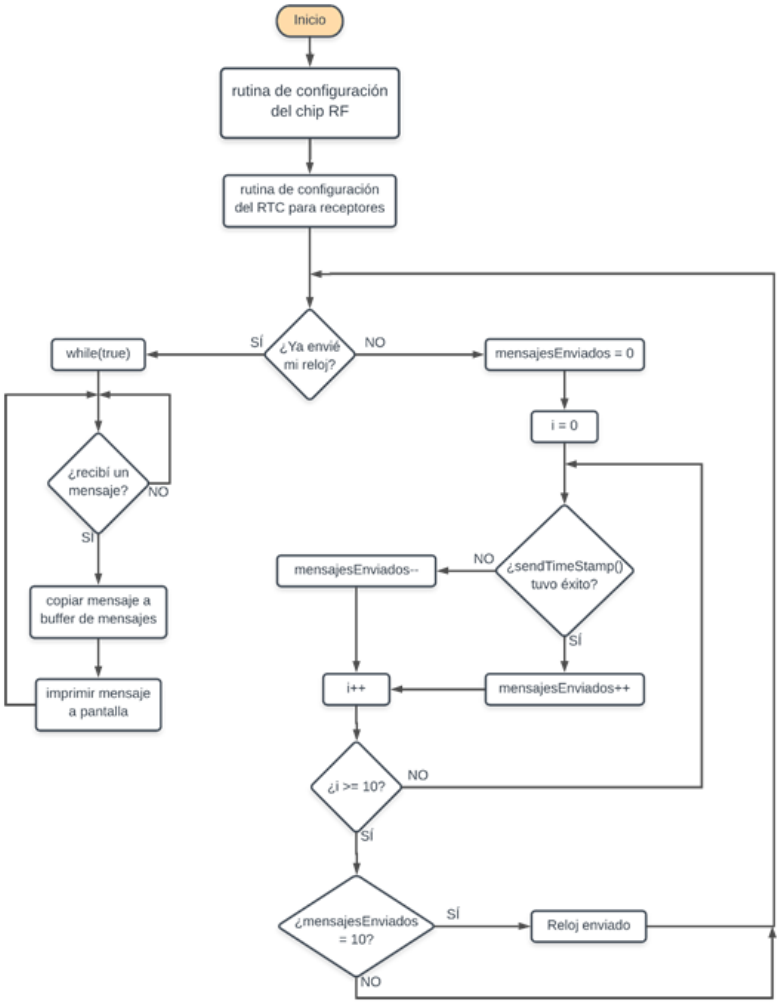

Figura 3.14. Diagrama de flujo de la base central en el sistema de comunicación.

Fuente: elaboración propia.

El diagrama de flujo de los emisores, o sea todos los demás nodos de la red, se puede observar en la Figura 3.15. Los transmisores inicialmente están en modo receptor esperando a recibir el reloj del máster. Una vez recibido el reloj, el transmisor se va a dedicar exclusivamente a crear mensajes, y cuando sea su turno de enviar información (durante la interrupción), lo hace.

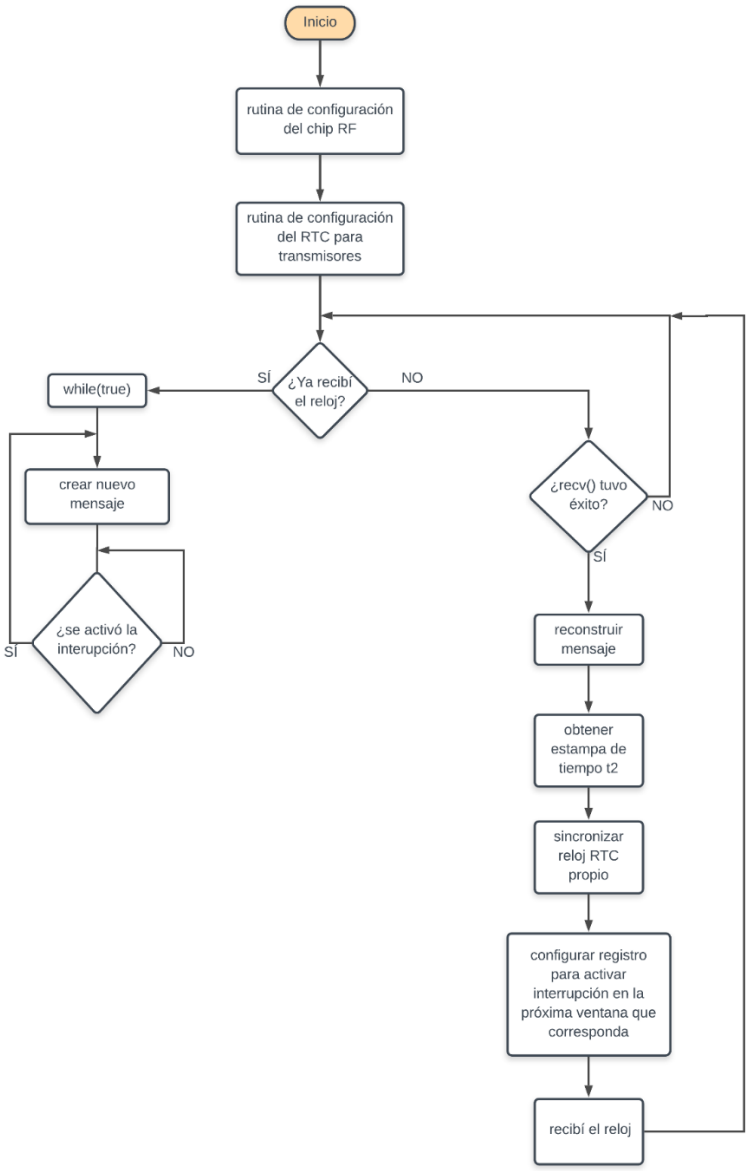

Figura 3.15. Diagrama de flujo para el sistema de comunicación de los nodos transmisores.

Fuente: elaboración propia.

Por aparte se comenta que para cumplir con el aspecto de que la base central debía dejar la información ordenada, se utilizó el programa de software libre Extra PuTTY. Este permite visualizad comunicaciones seriales entre un dispositivo externo y la *PC.* Tiene además la opción de crea un documento tipo .txt con todo lo imprimido en el puerto serial del *PC*, por lo que en realidad no es la base central la que enlista la información para luego ser procesada, si no que de esto se encarga el programa.

4 Experimentos y sus resultados

En este apartado se explicará cómo se llevó acabo los experimentos realizados para corroborar el funcionamiento de las soluciones desarrolladas en el proyecto de graduación y se mostrarán sus respectivos resultados.

4.1 Resultados del control automático de la navegación

El primer resultado concreto que se obtuvo con el control automático de la navegación fue haber logrado simular la teoría presentada por Lynch y Park en [13]. Se recuerda al lector que el diagrama de bloques de la simulación se puede ver en la Figura 3.5, y que este fue implementado y simulado utilizando la herramienta *Simulink* de MATLAB. En la Figura 4.1 se observan los resultados obtenidos de la simulación, en esta se le indicó al robot que se desplazara al punto P $(15, 15)$ cm.

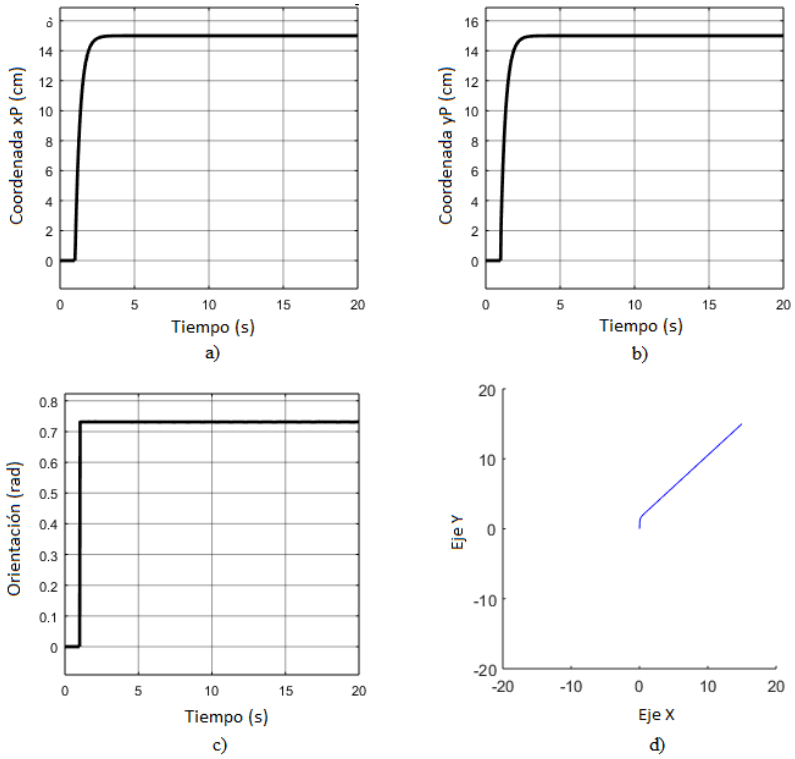

Figura 4.1. Resultados de la simulación realizada: a) Coordenada $\boldsymbol{x_P}$ respecto al tiempo; b) Coordenada $\boldsymbol{y_P}$ respecto al tiempo; c) Orientación $\emptyset$ respecto al tiempo; d) Camino recorrido por el robot en el plano 2D.

Fuente: elaboración propia.

Una vez evaluada la viabilidad de la teoría propuesta para el control de posición, se procedió entonces a construir el controlador *PID* con la herramienta *System Identification* y el *PID Tuner*. Para esto, se energizó intermitentemente solo uno de los dos motores, haciendo que el robot girara sobre la rueda no energizada. De manera paralela, se envió mediante el puerto serial del microcontrolador los valores de velocidad y *PWM* que poseía el motor que sí estaba energizado y girando. En la Figura 4.2 se observa la respuesta que presentó el motor izquierdo del robot. Los datos se obtuvieron con una periodo de muestreo de 0.01 s, y lo mostrado en el gráfico de velocidad pasó por un filtro de media móvil de promedio 10 para suavizar la salida. El motor derecho no se muestra en ninguna figura, pero tuvo un comportamiento muy similar.

Luego, procesando en MATLAB los datos recopilados de ambos motores, se obtuvieron los modelos de las plantas a controlar por el controlador *PID*. En la ecuación (4.1) se muestra el motor izquierdo discretizado, y en la (4.2) el motor derecho también discretizado.

$$M_{izquiedo}(z) = \frac{0.0031406 * z^{-1}}{(1 - 0.8962 * z^{-1})(1 - 0.8109 * z^{-1})} \tag{4.1}$$

$$M_{derecho}(z) = \frac{0.0026423 * z^{-1}}{(1 - 1.744 * z^{-1} + 0.7622 * z^{-2})} \tag{4.2}$$

Utilizando las ecuaciones (4.1) y (4.2) dentro de la herramienta *PID Tuner*, se obtuvieron las constantes ideales para el controlador *PID* de velocidad de cada uno de los motores. La Figura 4.3 muestra la respuesta teórica ante una referencia unitaria del motor izquierdo siendo controlado por un *PID* con las siguientes constantes: $k_P = 2.412; k_I = 16.68; k_D = 0.06774$. Estas constantes fueron obtenidas teniendo en cuenta los criterios de diseño impuestos para los controladores: menos de 5% de sobre impulso y menos de 1.5% de error en estado estacionario, y se fueron ajustando poco a poco hasta obtener la respuesta de la Figura 4.3. La respuesta del motor derecho es la misma, por lo que no se muestra; las constantes obtenidas para el motor derecho fueron: $k_P = 2.49; k_I = 17.96; k_D = 0.08631$.

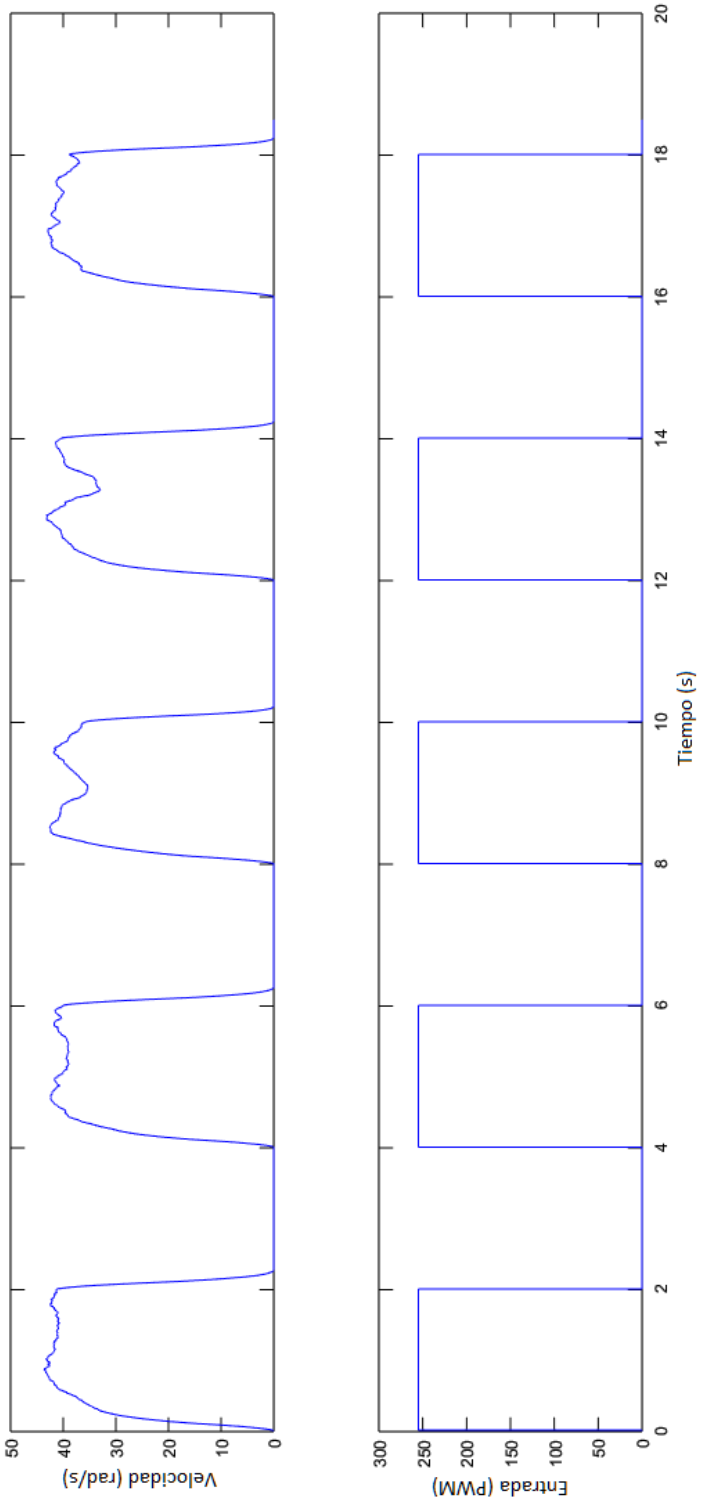

Figura 4.2. Velocidad del motor izquierdo ante un escalón unitario.

Fuente: elaboración propia.

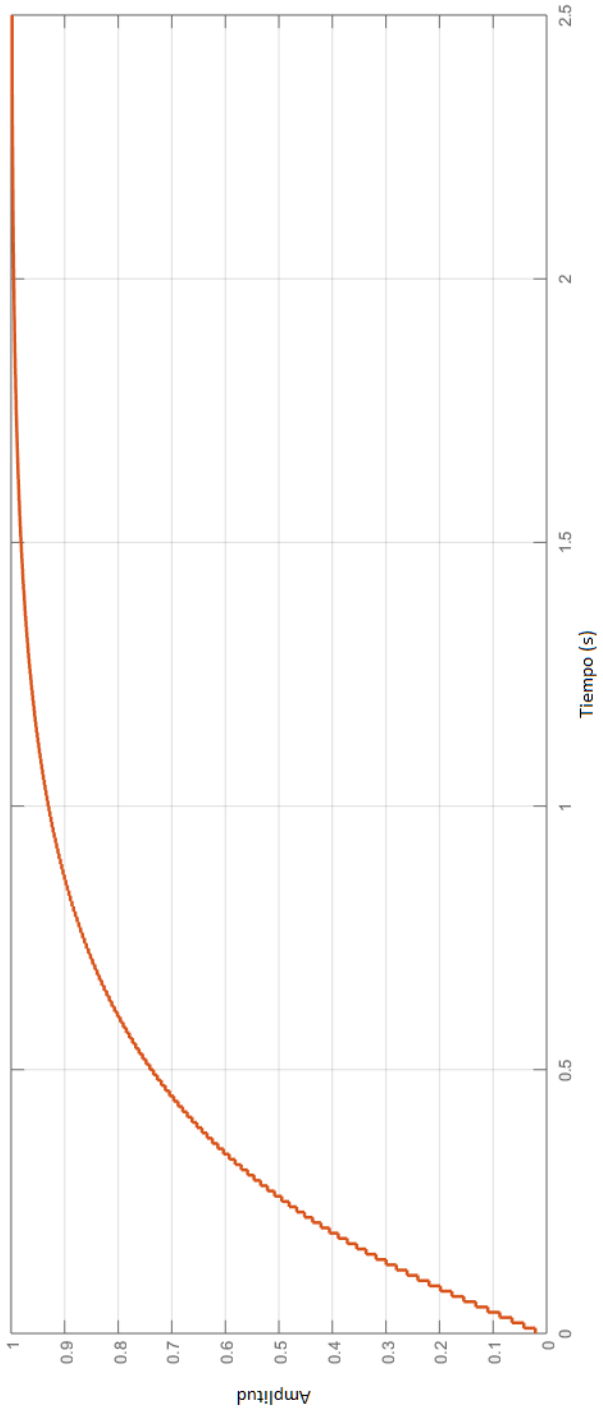

Figura 4.3. Seguimiento de una entrada de referencia unitaria para el motor izquierdo controlado.

Fuente: elaboración propia.

Como se obtuvieron dos grupos de constantes, al implementar el control de velocidad en el microcontrolador se debió desarrollar dos controladores *PID*, uno para cada motor del robot móvil. Se realizaron varios experimentos con los controladores para corroborar que cumplieran con los objetivos de diseño. La primera prueba fue observar la capacidad del control de velocidad de los motores de seguir referencias variantes en el tiempo, el resultado se observa en la Figura 4.4. En este experimento ambos motores siguieron una referencia inicialmente de 15 rad/s, que a los cinco segundos tuvo un cambio a 20 rad/s, y a los diez segundos bajó a 10 rad/s. Las gráficas en tono claro muestran las salidas de velocidad de los motores sin filtrar, la azul claro corresponde al motor derecho y la naranja claro al motor izquierdo. En tono oscuro se muestra la información de ambos motores filtrada con la técnica de media móvil con periodo igual a 10, de la misma manera el tono azul oscuro corresponde al motor derecho y el naranja oscuro al motor izquierdo. Cabe resaltar que estos resultados fueron obtenidos con los motores sin carga mecánica, pero el modelado sí se hizo con la carga del peso del robot.

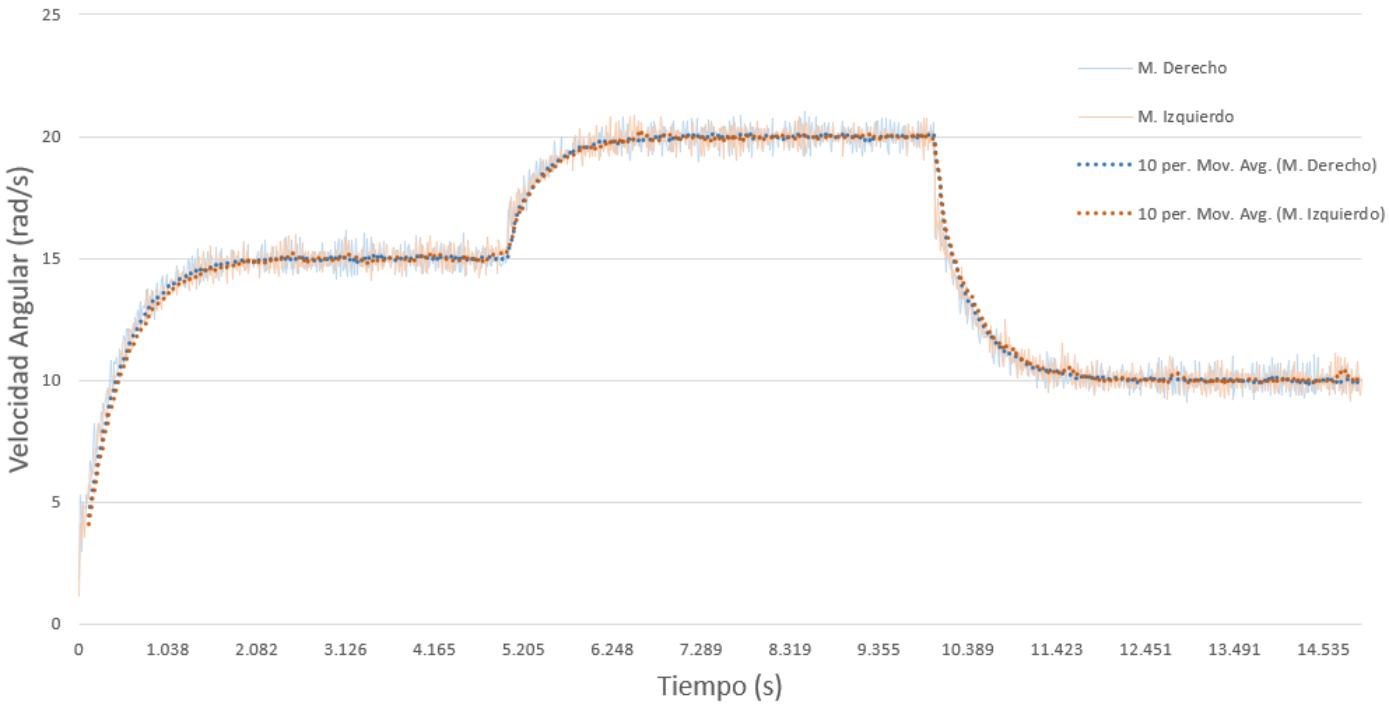

Figura 4.4. Respuesta de los controladores *PID* de los motores ante variaciones en la referencia de velocidad angular.

Fuente: elaboración propia.

En la Tabla 4.1 se ve un promedio de las velocidades obtenidas en los motores de este primer experimento. Cada dato se muestra de la manera *promedio* ± *desviación estándar*, y fue obtenido ponderando 1.5 s de las respuestas de los motores cuando habían alcanzado el estado estacionario ante el cambio de referencia. Además, se muestra su error ante la respectiva entrada que debían seguir los motores: 15, 20 y 10 rad/s; este error se obtuvo utilizando la fórmula general del error relativo, siendo el valor teórico la referencia.

Tabla 4.1. Respuesta promedio en la velocidad de los motores ante los cambios de referencia en el primer experimento.

Fuente: elaboración propia.

	Motor Izquierdo		**Motor Derecho**	
Referencia (rad/s)	**Respuesta (rad/s)**	**Error (%)**	**Respuesta (rad/s)**	**Error (%)**
15	14.961 ± 0.367	0.26	14.970 ± 0.463	0.20
20	19.962 ± 0.388	0.19	20.008 ± 0.430	-0.04
10	10.023 ± 0.371	-0.23	10.017 ± 0.403	-0.17

A pesar de que el objetivo de control no englobaba controladores inmunes a perturbaciones, se decidió medir la respuesta del sistema ante las mismas. Esto se observa en la Figura 4.5, donde se sigue la misma clave de colores que en el gráfico anterior: los tonos claros corresponden a la velocidad angular de los motores sin filtrar y los tonos oscuros a la velocidad filtrada por la técnica de media móvil con periodo igual a 10, el color azul es para el motor derecho y el naranja para el motor izquierdo. Esta prueba fue hecha sin carga, y la perturbación que se insertó al sistema fue frenar una de las rueda por completo durante un instante y luego soltarla, primeramente se frenó la rueda izquierda dos veces, y luego se frenó la rueda derecha en dos ocasiones más. La referencia que seguían los motores era de 15 rad/s.

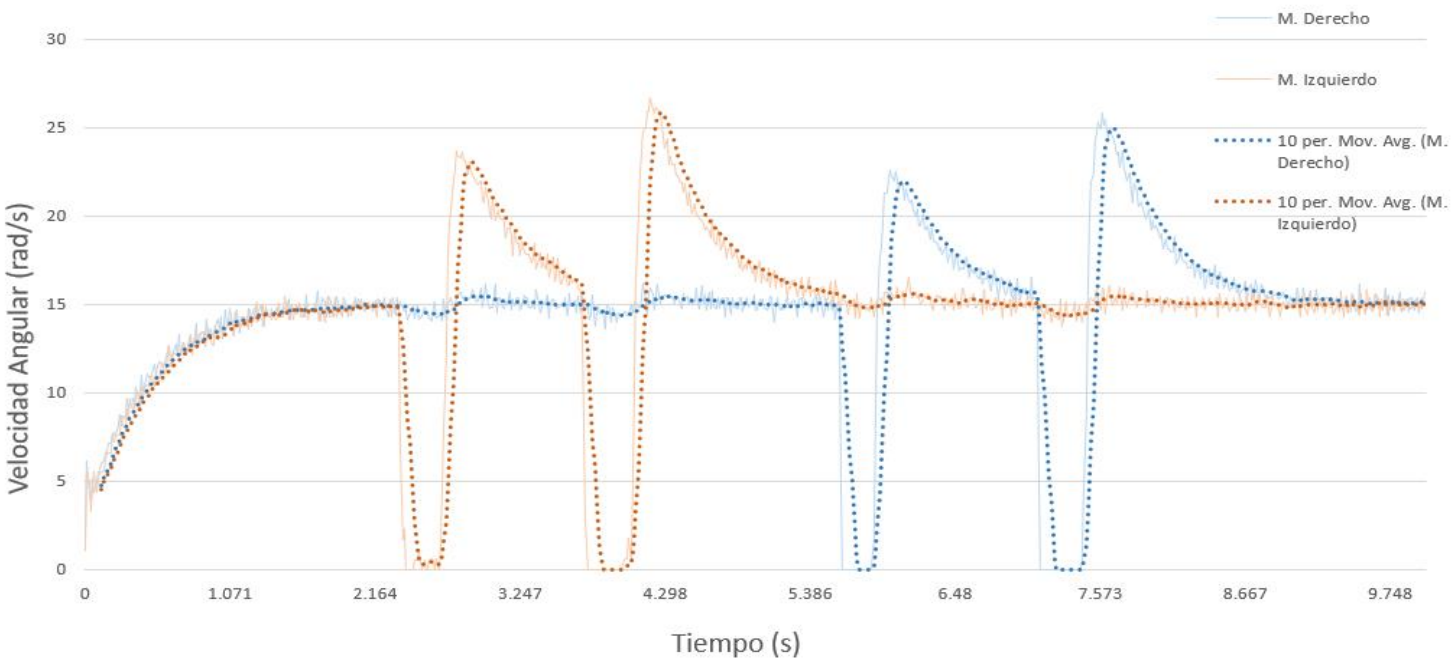

Figura 4.5. Rechazo de perturbaciones en los controladores *PID* de los motores.

Fuente: elaboración propia.

Finalmente, se probaron los controladores en su ambiente real, con la carga del peso del robot. La respuesta de ambos motores se observa en la Figura 4.6, aquí se indicó una velocidad de referencia igual 15 rad/s para los dos motores y se dejó al robot móvil avanzar en línea recta. El código de colores sigue siendo el mismo que en los dos gráficos anteriores, usando los tonos claro para la velocidad angular de los motores sin filtrar, y los tonos oscuro para la velocidad filtrada por un media móvil de promedio igual a 10.

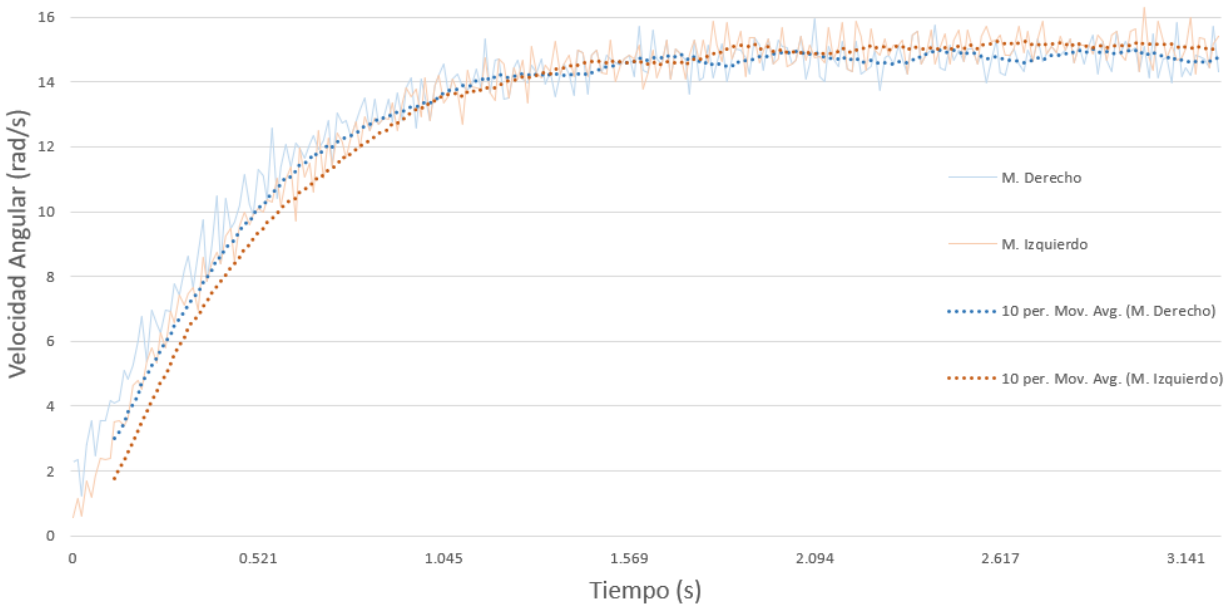

Figura 4.6. Comportamiento de los controladores *PID* de los motores ante carga mecánica.

Fuente: elaboración propia.

En la Tabla 4.2 se aprecian las velocidades promedio y el error del control de velocidad en el segundo y tercer experimento, donde se seguía una referencia de 15 rad/s. Estos datos fueron obtenidos de la misma manera que en la Tabla 4.1, se promediaron las mediciones en un periodo de 1.5 s durante el estado estacionario de cada motor.

Tabla 4.2. Respuesta promedio en la velocidad de los motores durante el segundo y tercer experimento de los controladores *PID*.

Fuente: elaboración propia.

	Motor Izquierdo		**Motor Derecho**	
Experimento	**Respuesta (rad/s)**	**Error (%)**	**Respuesta (rad/s)**	**Error (%)**
Perturbación (15 rad/s)	15.009 ± 0.500	-0.06	15.010 ± 0.418	-0.08
Carga (15 rad/s)	15.087 ± 0.418	-0.58	14.804 ± 0.437	1.31

Teniendo ya los controladores de velocidad diseñados e implementados en el microcontrolador de manera exitosa, se procedió a medir la exactitud y precisión del control de posición, para lo que se utilizó un sistema de visión que posee el proyecto PROE E1F2. El sistema de visión fue diseñado, implementado y validado fuera del presente proyecto, por lo que en este apartado no se entrará en detalles específicos del mismo; sin embargo, es importante recalcar que el sistema de visión posee un error relativo igual a ±0.37% al medir distancias, por lo que sus mediciones son bastante exactas. En el apéndice A1. Desarrollo y validación del sistema de visión se detalla más sobre el sistema de visión en caso de que al lector le interese.

Además, se realizó un diseño de experimentos con la ayuda de la M. Sc. Cindy Calderón, coordinadora del proyecto PROE E1F2. Estos experimentos se hicieron bajo las siguientes condiciones: la constante proporcional k_P del control de posición tuvo un valor de 9; la frecuencia de muestreo del sistema de control de velocidad y posición fue de 100 Hz; y la velocidad teórica de los movimientos del robot fue de 3 cm/s para el avance lineal y 7.85 rad/s para los giros.

Con estos experimentos lo que se buscó propiamente fue: encontrar si para un avance lineal había una relación entre el error obtenido y la distancia avanzada; identificar si existía una diferencia en el error al girar 90° o -90°; y cuantificar en general la precisión y exactitud del sistema de control de la navegación del robot móvil desarrollado. Para determinar lo anterior, se tomaron datos avanzando una distancia primero de 30 cm y luego de 15 cm, además de datos girando 90° y girando -90°.

Para llevar a cabo las pruebas, se imprimió un patrón de dos círculos de diferentes tamaños que se pegó al robot móvil. En la Figura 4.7 se aprecia el antes y el después de un movimiento de avanzar 15 cm en línea recta, aquí se puede observar el patrón de dos círculos, esto es lo que el sistema de visión detecta y no al robot en sí. La cámara captura una imagen al inicio y otra al final del movimiento, y luego genera un vector por cada par de círculos de las dos imágenes. Seguidamente, estos vectores se superponen en una sola imagen y se mide las diferencias entre sus posiciones y orientaciones, de esta forma se obtiene el desplazamiento lineal y angular que sufrió el robot debido al movimiento.

Además, es notorio que hay otro patrón cuadrado que asemeja un tablero de ajedrez, este patrón lo utiliza el sistema de visión para eliminar las distorsiones en cada captura realizada y alcanzar la exactitud ya comentada. Para otorgarle una dirección a los movimientos angulares se

utilizó la regla de la mano derecha alrededor del eje que “sale” del suelo observado en la Figura 4.7. Es decir, positivo será un movimiento antihorario y negativo uno horario.

a) b)

Figura 4.7. Movimiento del robot durante el experimento de avanzar 15 cm: a) antes del movimiento; b) después de la acción de control.

Fuente: elaboración propia.

En la Tabla 4.3 y la Tabla 4.4 se muestran los resultados del primer experimento, indicarle al robot que avance 30 cm. La Tabla 4.3 muestra: la referencia que se le indicó al control del robot en todas las repeticiones, las mediciones de la acción de control tomadas mediante el sistema de visión y el error que existió entre la referencia dada al sistema de control y lo que midió el sistema de visión. Este error se calculó como relativo, tomando como dato teórico la referencia indicada al robot y como dato experimental lo medido por el sistema de visión.

La Tabla 4.4 muestra: las mediciones de la acción de control tomadas mediante el sistema de visión, que se interpretan como el verdadero movimiento que realizó el robot; las estimaciones que realizó el control del robot mediante la cinemática directa y el error que existió en la estimación hecha por el robot. En este caso el error se calculó también como relativo, pero aquí se estimó tomando como dato teórico lo medido por el sistema de visión, y como dato experimental, lo estimado por la cinemática directa.

En la Tabla 4.5 y la Tabla 4.6 se muestran los mismo resultados, pero ahora para el experimento de avanzar 15 cm. Los errores se calcularon igual que como en la Tabla 4.3 y en la Tabla 4.4. Es importante resaltar que ambos experimentos muestran la acción de control como un desplazamiento del punto controlado *P* del robot móvil, y no como un dato coordenada a coordenada del punto *P*, pues esto facilita el análisis de los datos.

Tabla 4.3. Error existente en el control del desplazamiento en el experimento de avanzar 30 cm.

Fuente: elaboración propia.

Valor teórico (cm)	Sistema de visión (cm ± 0.37%)	Error del control del desplazamiento (%)
30.00	30.14	-0.47
30.00	30.76	-2.54
30.00	32.91	-9.71
30.00	30.41	-1.37
30.00	30.94	-3.14
30.00	31.18	-3.92
30.00	30.64	-2.12
30.00	29.88	0.40
30.00	31.26	-4.19
30.00	30.73	-2.44
30.00	30.49	-1.63
30.00	29.39	2.03
30.00	30.03	-0.11
30.00	30.42	-1.41
30.00	30.52	-1.73
Promedio del error:		-2.16
Desviación estándar asociada:		2.65

Tabla 4.4. Error existente en la estimación del desplazamiento en el experimento de avanzar 30 cm.

Fuente: elaboración propia.

Sistema de visión (cm ± 0.37%)	Estimación del robot (± 0.01 cm)	Error de la estimación del desplazamiento (%)
30.14	30.08	0.21
30.76	29.86	2.93
32.91	30.52	7.27
30.41	30.33	0.27
30.94	29.98	3.11
31.18	30.77	1.30
30.64	29.57	3.48
29.88	29.49	1.30
31.26	31.17	0.28
30.73	30.28	1.47
30.49	30.31	0.59
29.39	30.51	-3.81
30.03	29.50	1.77
30.42	30.23	0.63
30.52	30.53	-0.04
Promedio del error:		1.38
Desviación estándar asociada:		2.37

Tabla 4.5. Error existente en el control del desplazamiento en el experimento de avanzar 15 cm.

Fuente: elaboración propia.

Valor teórico (cm)	Sistema de visión (cm ± 0.37%)	Error del control del desplazamiento (%)
15.00	16.02	-6.82
15.00	16.18	-7.89
15.00	15.37	-2.45
15.00	15.07	-0.44
15.00	15.38	-2.51
15.00	15.36	-2.37
15.00	15.38	-2.54
15.00	15.21	-1.43
15.00	15.98	-6.52
15.00	14.94	0.37
15.00	14.60	2.70
15.00	15.03	-0.20
15.00	15.24	-1.59
15.00	15.01	-0.06
15.00	14.52	3.22
Promedio del error:		-1.90
Desviación estándar asociada:		3.21

Tabla 4.6. Error existente en la estimación del desplazamiento en el experimento de avanzar 15 cm.

Fuente: elaboración propia.

Sistema de visión (cm ± 0.37%)	Estimación del robot (± 0.01 cm)	Error de la estimación del desplazamiento (%)
16.02	15.43	3.69
16.18	15.70	2.98
15.37	15.22	0.94
15.07	15.19	-0.83
15.38	15.48	-0.69
15.36	15.49	-0.88
15.38	15.56	-1.16
15.21	15.25	-0.24
15.98	16.13	-0.95
14.94	15.08	-0.93
14.60	14.54	0.37
15.03	15.22	-1.26
15.24	15.43	-1.26
15.01	15.14	-0.87
14.52	14.58	-0.43
Promedio del error:		-0.10
Desviación estándar asociada:		1.53

En cuanto a los experimentos al hacer girar al robot ± 90°, los resultados se muestran en las tablas: Tabla 4.7, Tabla 4.8, Tabla 4.9 y Tabla 4.10 y los errores se calcularon de la misma manera ya explicada. En estos dos experimentos, lo que se muestra en las tablas es la orientación final del robot móvil, la cual se controló implícitamente con el punto de control *P* del robot móvil. Se buscaba que el centro de giro del robot se quedara completamente estático y no sufriera ningún desplazamiento.

Tabla 4.7. Error existente en el control de la orientación en el experimento de girar 90°.

Fuente: elaboración propia.

Valor teórico (°)	Sistema de visión (± 0.01°)	Error del control de la orientación (%)
90	91.40	-1.56
90	89.70	0.33
90	89.63	0.41
90	88.59	1.57
90	88.72	1.42
90	89.02	1.09
90	87.47	2.81
90	88.04	2.18
90	92.31	-2.56
90	91.16	-1.29
90	91.16	-1.28
90	91.02	-1.13
90	85.96	4.49
90	91.29	-1.43
90	90.00	0.01
Promedio del error:		0.34
Desviación estándar asociada:		1.94

Tabla 4.8. Error existente en la estimación de la orientación en el experimento de girar 90°.

Fuente: elaboración propia.

Sistema de visión (± 0.01°)	Estimación del robot (± 0.01 °)	Error de la estimación de la orientación (%)
91.40	91.32	0.09
89.70	88.71	1.10
89.63	92.32	-3.00
88.59	90.11	-1.72
88.72	92.79	-4.58
89.02	90.28	-1.41
87.47	90.50	-3.46
88.04	91.29	-3.69
92.31	92.63	-0.35
91.16	91.76	-0.67
91.16	92.09	-1.02
91.02	91.93	-1.00
85.96	88.62	-3.10
91.29	91.53	-0.26
90.00	91.29	-1.44
Promedio del error:		-1.63
Desviación estándar asociada:		1.61

Tabla 4.9. Error existente en el control de la orientación en el experimento de girar -90°.

Fuente: elaboración propia.

Valor teórico (°)	Sistema de visión (± 0.01°)	Error del control de la orientación (%)
-90	-89.30	0.78
-90	-91.84	-2.04
-90	-92.45	-2.72
-90	-90.85	-0.94
-90	-90.68	-0.76
-90	-93.60	-4.00
-90	-91.74	-1.94
-90	-86.69	3.68
-90	-88.57	1.59
-90	-87.92	2.31
-90	-89.61	0.43
-90	-89.69	0.34
-90	-89.11	0.99
-90	-89.65	0.39
-90	-90.70	-0.78
Promedio del error:		-0.18
Desviación estándar asociada:		2.01

Tabla 4.10. Error existente en la estimación de la orientación en el experimento de girar -90°.

Fuente: elaboración propia.

Sistema de visión (± 0.01°)	Estimación del robot (± 0.01 °)	Error de la estimación de la orientación (%)
-89.30	-89.26	0.05
-91.84	-92.71	-0.95
-92.45	-91.77	0.73
-90.85	-92.64	-1.97
-90.68	-92.16	-1.63
-93.60	-91.77	1.96
-91.74	-90.99	0.83
-86.69	-88.78	-2.41
-88.57	-90.68	-2.38
-87.92	-90.28	-2.68
-89.61	-91.78	-2.42
-89.69	-91.56	-2.07
-89.11	-90.19	-1.21
-89.65	-92.96	-3.70
-90.70	-92.32	-1.79
Promedio del error:		-1.31
Desviación estándar asociada:		1.56

Como ya se dijo, aquí se presentaron errores de control y estimación para el desplazamiento en los movimientos de avance, y para la orientación en los movimientos de giro; sin embargo, es inevitable que exista un error asociado en la orientación del robot al avanzar, y un error asociado en el desplazamiento del robot al girar. Estos errores colaterales de movimiento no son esenciales, pues aquí se buscaba representar el error asociado al movimiento respectivo; no obstante, es importante tenerlos presentes porque a la larga afectan la navegación del robot. Por lo que en el apéndice: A2. Errores colaterales asociados a los movimientos del robot móvil, se muestran las tablas que agrupan los errores de orientación al avanzar y los errores de desplazamiento al girar. Estos errores se calcularon como ya se ha explicado en este apartado, con la diferencia de que son absolutos y no relativos porque se debían comparar con una referencia de magnitud igual a cero.

4.2 Resultados de la integración de las funcionalidades de la *IMU* para mejorar el error de orientación

Los primeros datos obtenidos con la *IMU* fueron las constantes de calibración para los tres sensores. En la Figura 4.8 se muestran los resultados obtenidos con el programa de *PC* que ayuda a calibrar el magnetómetro.

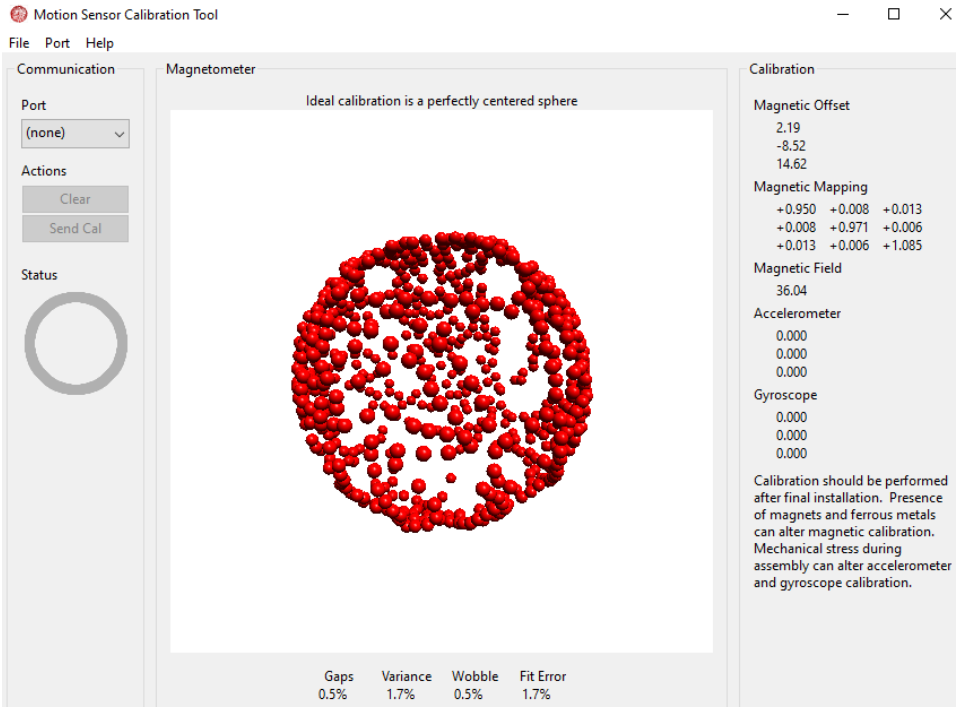

Figura 4.8. Resultado obtenido de la calibración del magnetómetro en el programa Motion Cal.

Fuente: elaboración propia.

De esto se obtuvo entonces que:

$$M_m \cdot S \cdot SI = \begin{bmatrix} 0.95 & 0.008 & 0.013 \\ 0.008 & 0.971 & 0.006 \\ 0.013 & 0.006 & 1.085 \end{bmatrix} \tag{4.3}$$

$$O = \begin{bmatrix} 2.19 \\ -8.52 \\ 14.62 \end{bmatrix} \tag{4.4}$$

Por otro lado, para calibrar el acelerómetro y giroscopio los sensores recopilaron aproximadamente 1600 datos crudos por cada eje de medición sin ningún estímulo aplicado, y

luego este conjunto de datos se ponderaron para obtener el vector de desplazamiento de la medición de cada sensor. Se obtuvo que:

$$O_a = \begin{bmatrix} 0.300 \\ -0.132 \\ -0.310 \end{bmatrix} \tag{4.5}$$

$$O_g = \begin{bmatrix} 1.802 \\ -4.523 \\ 0.418 \end{bmatrix} \tag{4.6}$$

Con estas constantes de calibración obtenidas, se procedió a medir la exactitud y precisión de la *IMU* utilizando un montaje especial que construyó el proyecto PROE E1F2. El mismo se observa en la Figura 4.9, donde se nota que el montaje posee unas divisiones, estas permitieron medir cada 2.5°.

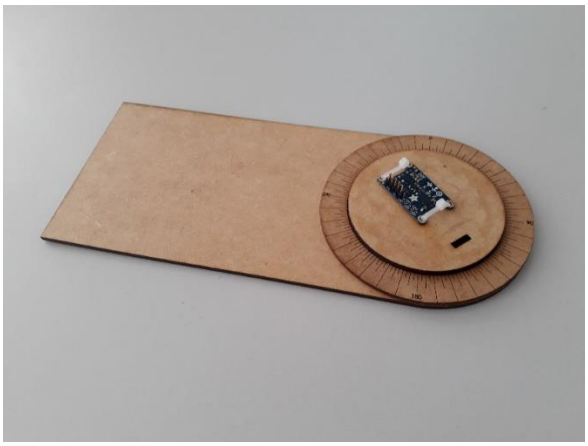

Figura 4.9. Montaje para realizar mediciones con la *IMU*.

Fuente: elaboración propia.

Con lo anterior se obtuvo la Tabla 4.11, aquí se pueden observar los datos arrojados por la *IMU* trabajando con el filtro Madgwick para diferentes valores de ángulos. Se observa además el error absoluto de cada valor medido al igual que su desviación estándar.

Tabla 4.11. Error en la *IMU* con el filtro Madgwick.

Fuente: elaboración propia.

	Valores Medidos (± 0.1°)				
	47.5	**135**	**180**	**235**	**345**
	46.9	134.2	179.7	234.2	345.2
	46.8	134.2	179.9	234.3	345.1
	46.7	134.5	179.8	234.3	344.8
	46.7	134.4	179.7	234.4	345.1
	46.9	134.6	179.6	234.1	344.8
	46.7	134.3	179.3	234.5	344.9
	46.7	134.8	179.5	234.5	345.0
	46.9	134.8	179.4	234.4	344.8
	46.7	134.5	179.4	234.5	345.0
	46.9	134.6	179.2	234.5	345.2
	46.9	134.3	179.2	234.5	344.8
	47.4	134.5	179.4	234.7	345.3
	46.9	134.7	179.1	234.4	344.3
	46.9	135.0	179.4	234.3	345.2
Promedio	46.9	134.5	179.5	234.4	345.0
Error	0.6	0.5	0.5	0.6	0.0
Desviación	0.2	0.2	0.2	0.2	0.3

Una vez calibrada e implementada la orientación medida con la *IMU*, se realizaron los debidos experimentos para cuantificar su aporte a la disminución del error de orientación ante errores del dipo deslizamiento para diferentes movimientos. Estos experimentos se diseñaron en conjunto con los hechos al control automático en el subapartado anterior, por lo que la información se recopiló de la misma manera. Solo hubo una diferencia: dado que el objetivo fue medir la disminución de error de orientación en errores de tipo deslizamiento, se colocó una placa de material MDF de 3 mm de espesor frente a la rueda derecha del robot para todos los experimentos.

En total se realizaron seis experimentos, tres al avanzar 15 cm y girar ± 90° sin retroalimentación de la *IMU*, y otros tres haciendo lo mismo pero con la medición de los sensores de la *IMU*. La Tabla 4.12 y la Tabla 4.13 muestran los resultados del error existente en el control de la orientación y del error en la estimación de la orientación que realizó el robot, respectivamente, ambos al avanzar 15 cm sin retroalimentación de la *IMU*. La Tabla 4.14 y la Tabla 4.15 muestran los mismos errores y en el mismo orden, pero en el caso donde sí hubo retroalimentación de la *IMU* al avanzar los 15 cm. Las tablas: Tabla 4.16, Tabla 4.17, Tabla 4.18 y Tabla 4.19, agrupan los resultados obtenidos al girar 90°. Las primeras dos muestran respectivamente el error existente en el control de la orientación y el error en la estimación de la orientación realizada por el robot cuando no hubo retroalimentación de la *IMU*, mientras que la Tabla 4.18 y la Tabla 4.19 lo muestran para el caso en el que sí hubo intervención por parte de la *IMU*. De igual forma, las tablas: Tabla 4.20, Tabla 4.21, Tabla 4.22 y Tabla 4.23, agrupan los resultados cuando se giró -90° en el mismo orden ya dicho para los casos de avanzar 15 cm y girar 90°.

Al igual que los experimentos del subapartado anterior, los errores de control se calcularon tomando como valor teórico la referencia dada al sistema de control de navegación, y como valor experimental lo medido por el sistema de visión; y en los errores de la estimación se utilizó como valor teórico lo medido con el sistema de visión y como valor experimental lo estimado por la cinemática directa del control del robot móvil. Sin embargo, para este análisis los errores se calcularon como absolutos y no como relativos, ya que se debieron comparar con una referencia de magnitud igual cero en algunas ocasiones.

También es importante recordar que la medición de la *IMU* solo afecta al control del robot móvil por medio del subsistema de vigilancia de orientación ya explicado en el capítulo 3, y que este subsistema en realidad no interviene con la estimación que hace la cinemática directa del control del robot, que se basa solamente en los datos medidos con los codificadores.

Tabla 4.12. Error existente en el control de la orientación en el experimento de avanzar 15 cm con obstáculo y sin ayuda de la *IMU*.

Fuente: elaboración propia.

Valor teórico (°)	Sistema de visión (± 0.01°)	Error del control de la orientación (°)
0	-15.24	15.24
0	-6.56	6.56
0	-3.44	3.44
0	-12.56	12.56
0	-5.02	5.02
0	-12.15	12.15
0	-4.41	4.41
0	-26.09	26.09
0	-15.58	15.58
0	-9.40	9.40
0	-9.63	9.63
0	-7.91	7.91
0	-8.77	8.77
0	-5.16	5.16
0	-3.26	3.26
Promedio del error:		9.68
Desviación estándar asociada:		6.05

Tabla 4.13. Error existente en la estimación de la orientación en el experimento de avanzar 15 cm con obstáculo y sin ayuda de la *IMU*.

Fuente: elaboración propia.

Sistema de visión (± 0.01°)	Estimación del robot (± 0.01°)	Error de la estimación de la orientación (°)
-15.24	-4.14	-11.10
-6.56	-1.39	-5.17
-3.44	3.12	-6.56
-12.56	-4.06	-8.50
-5.02	0.49	-5.52
-12.15	-0.60	-11.55
-4.41	0.01	-4.42
-26.09	-1.08	-25.01
-15.58	-2.81	-12.77
-9.40	-4.76	-4.64
-9.63	-0.13	-9.51
-7.91	-3.60	-4.31
-8.77	-2.96	-5.81
-5.16	-2.47	-2.69
-3.26	0.75	-4.01
Promedio del error:		-8.10
Desviación estándar asociada:		5.60

Tabla 4.14. Error existente en el control de la orientación en el experimento de avanzar 15 cm con obstáculo y con ayuda de la *IMU*.

Fuente: elaboración propia.

Valor teórico (°)	Sistema de visión (± 0.01°)	Error del control de la orientación (°)
0	-0.17	0.17
0	-2.42	2.42
0	-0.86	0.86
0	1.25	-1.25
0	-1.47	1.47
0	0.38	-0.38
0	0.57	-0.57
0	0.23	-0.23
0	-0.94	0.94
0	-1.90	1.90
0	-1.43	1.43
0	-0.99	0.99
0	-1.01	1.01
0	-0.34	0.34
0	-0.16	0.16
Promedio del error:		0.62
Desviación estándar asociada:		0.99

Tabla 4.15. Error existente en la estimación de la orientación en el experimento de avanzar 15 cm con obstáculo y con ayuda de la *IMU*.

Fuente: elaboración propia.

Sistema de visión (± 0.01°)	Estimación del robot (± 0.01°)	Error de la estimación de la orientación (°)
-0.17	12.41	-12.58
-2.42	12.32	-14.74
-0.86	4.85	-5.70
1.25	14.55	-13.30
-1.47	2.99	-4.46
0.38	15.27	-14.88
0.57	12.88	-12.31
0.23	14.03	-13.80
-0.94	8.60	-9.53
-1.90	16.96	-18.86
-1.43	9.83	-11.26
-0.99	10.73	-11.72
-1.01	13.60	-14.61
-0.34	10.24	-10.58
-0.16	16.02	-16.18
Promedio del error:		-12.30
Desviación estándar asociada:		3.74

Tabla 4.16. Error existente en el control de la orientación en el experimento de girar 90° con obstáculo y sin ayuda de la *IMU*.

Fuente: elaboración propia.

Valor teórico (°)	Sistema de visión (± 0.01°)	Error del control de la orientación (°)
90	105.56	-15.56
90	99.30	-9.30
90	113.88	-23.88
90	91.37	-1.37
90	118.25	-28.25
90	91.68	-1.68
90	102.37	-12.37
90	94.78	-4.78
90	98.20	-8.20
90	99.06	-9.06
90	110.78	-20.78
90	100.32	-10.32
90	128.33	-38.33
90	104.21	-14.21
90	100.26	-10.26
Promedio del error:		-13.89
Desviación estándar asociada:		10.17

Tabla 4.17. Error existente en la estimación de la orientación en el experimento de girar 90° con obstáculo y sin ayuda de la *IMU*.

Fuente: elaboración propia.

Sistema de visión (± 0.01°)	Estimación del robot (± 0.01°)	Error de la estimación de la orientación (°)
105.56	90.16	15.40
99.30	85.75	13.55
113.88	88.20	25.68
91.37	92.64	-1.27
118.25	94.87	23.39
91.68	86.13	5.55
102.37	88.75	13.62
94.78	87.39	7.39
98.20	91.16	7.05
99.06	89.89	9.18
110.78	85.87	24.91
100.32	85.75	14.57
128.33	86.47	41.86
104.21	91.37	12.83
100.26	87.03	13.22
Promedio del error:		15.13
Desviación estándar asociada:		10.44

Tabla 4.18. Error existente en el control de la orientación en el experimento de girar 90° con obstáculo y con ayuda de la *IMU*.

Fuente: elaboración propia.

Valor teórico (°)	Sistema de visión (± 0.01°)	Error del control de la orientación (°)
90	88.04	1.96
90	87.60	2.40
90	87.85	2.15
90	85.88	4.12
90	87.02	2.98
90	85.93	4.07
90	86.42	3.58
90	88.00	2.00
90	86.70	3.30
90	84.84	5.16
90	85.72	4.28
90	86.57	3.43
90	87.80	2.20
90	87.19	2.81
90	86.62	3.38
Promedio del error:		3.19
Desviación estándar asociada:		0.96

Tabla 4.19. Error existente en la estimación de la orientación en el experimento de girar 90° con obstáculo y con ayuda de la *IMU*.

Fuente: elaboración propia.

Sistema de visión (± 0.01°)	Estimación del robot (± 0.01°)	Error de la estimación de la orientación (°)
88.04	67.66	20.38
87.60	86.49	1.11
87.85	80.13	7.72
85.88	80.24	5.63
87.02	79.27	7.75
85.93	77.94	7.99
86.42	75.97	10.46
88.00	86.91	1.10
86.70	73.20	13.51
84.84	80.12	4.72
85.72	85.36	0.36
86.57	86.09	0.48
87.80	75.62	12.18
87.19	73.04	14.14
86.62	79.17	7.46
Promedio del error:		7.67
Desviación estándar asociada:		5.79

Tabla 4.20. Error existente en el control de la orientación en el experimento de girar -90° con obstáculo y sin ayuda de la *IMU*.

Fuente: elaboración propia.

Valor teórico (°)	Sistema de visión (± 0.01°)	Error del control de la orientación (°)
-90	-98.69	8.69
-90	-101.22	11.22
-90	-96.97	6.97
-90	-92.11	2.11
-90	-90.76	0.76
-90	-92.11	2.11
-90	-94.45	4.45
-90	-84.73	-5.27
-90	-88.82	-1.18
-90	-86.53	-3.47
-90	-86.92	-3.08
-90	-92.78	2.78
-90	-90.93	0.93
-90	-83.58	-6.42
-90	-89.89	-0.11
Promedio del error:		1.37
Desviación estándar asociada:		5.01

Tabla 4.21. Error existente en la estimación de la orientación en el experimento de girar -90° con obstáculo y sin ayuda de la *IMU*.

Fuente: elaboración propia.

Sistema de visión (± 0.01°)	Estimación del robot (± 0.01°)	Error de la estimación de la orientación (°)
-98.69	-91.98	-6.71
-101.22	-93.30	-7.92
-96.97	-93.08	-3.89
-92.11	-93.63	1.52
-90.76	-92.30	1.54
-92.11	-92.01	-0.10
-94.45	-93.55	-0.90
-84.73	-85.70	0.97
-88.82	-90.69	1.88
-86.53	-89.35	2.82
-86.92	-88.08	1.16
-92.78	-92.12	-0.66
-90.93	-92.56	1.63
-83.58	-85.46	1.88
-89.89	-91.70	1.82
Promedio del error:		-0.33
Desviación estándar asociada:		3.28

Tabla 4.22. Error existente en el control de la orientación en el experimento de girar -90° con obstáculo y con ayuda de la *IMU*.

Fuente: elaboración propia.

Valor teórico (°)	Sistema de visión (± 0.01°)	Error del control de la orientación (°)
-90	-88.78	-1.22
-90	-90.80	0.80
-90	-91.12	1.12
-90	-90.58	0.58
-90	-87.93	-2.07
-90	-87.04	-2.96
-90	-90.62	0.62
-90	-91.10	1.10
-90	-90.37	0.37
-90	-91.04	1.04
-90	-88.81	-1.19
-90	-91.11	1.11
-90	-89.48	-0.52
-90	-89.12	-0.88
-90	-91.33	1.33
Promedio del error:		-0.05
Desviación estándar asociada:		1.34

Tabla 4.23. Error existente en la estimación de la orientación en el experimento de girar -90° con obstáculo y con ayuda de la *IMU*.

Fuente: elaboración propia.

Sistema de visión (± 0.01°)	Estimación del robot (± 0.01°)	Error de la estimación de la orientación (°)
-88.78	-71.254	-17.53
-90.80	-65.07	-25.74
-91.12	-57.49	-33.64
-90.58	-48.33	-42.25
-87.93	-63.45	-24.48
-87.04	-40.68	-46.37
-90.62	-84.82	-5.81
-91.10	-75.25	-15.85
-90.37	-79.81	-10.56
-91.04	-73.74	-17.31
-88.81	-87.75	-1.06
-91.11	-81.58	-9.53
-89.48	-78.34	-11.14
-89.12	-75.68	-13.44
-91.33	-77.47	-13.86
Promedio del error:		-19.24
Desviación estándar asociada:		13.01

4.3 Resultados del sistema de comunicación

Los primeros resultados obtenidos respecto al sistema de comunicación fue el haber medido el desfase existente por el término $n \cdot \tau$ de la ecuación (2.4) para poder determinar empíricamente este valor y mejorar la sincronización. Se realizaron cinco pruebas, donde un único nodo estaba conectado a la base y la unidad central le enviaba su reloj. Se midió el desfase existente con ayuda del programa Extra PuTTY, que permite desplegar más de un puerto serial a la vez y adjuntar estampas de tiempo a los mensajes recibido en el puerto serial. Se obtuvo en las cinco pruebas que el desfase era constante, y tenía un valor de 2 ms.

Una vez corregido esto, se decidió medir la estabilidad de la sincronización de los relojes de un nodo y la base central a través del tiempo. Para esto, se realizó tres veces el siguiente experimento: se dio la sincronización inicial, y de ahí en adelante se midió el desfase entre los dos relojes a los minutos 0, 7, 15 y 90 de iniciada la comunicación. Los resultados se pueden observar en la Tabla 4.24, y con ellos fue que se decidió que el espacio de sobra en la ventana de tiempo fuera inicialmente de 10 ms, pues se previó un desfase máximo de ± 5 ms.

Tabla 4.24. Desfases entre el reloj del máster y el de un nodo según el tiempo transcurrido en cada prueba.

Fuente: elaboración propia.

	Desfases (± 1 ms)			
Iteración	**Minuto 0**	**Minuto 7**	**Minuto 15**	**Minuto 90**
1	0	0	-3	4
2	0	0	2	5
3	0	0	-2	3

Al realizarse el experimento anterior y determinarse entonces que la ventana de tiempo óptima para cada nodo sería de 25 ms, se decidió corroborar que el sistema de comunicación serviría con este valor de tiempo utilizando la mayor cantidad de nodos disponibles. El proyecto PROE E2F1 poseía cinco microcontroladores Feather M0, por lo que se pudo probar el sistema con un máximo de cuatro nodos y la base central. Entonces, se realizó un experimento donde los cuatro nodos le enviaban a la base central un mensaje de la forma “Prueba #ID”. En la Figura 4.10

se observa el puerto serial de la base central durante este experimento y se ve que cada mensaje recibido posee su estampa de tiempo.

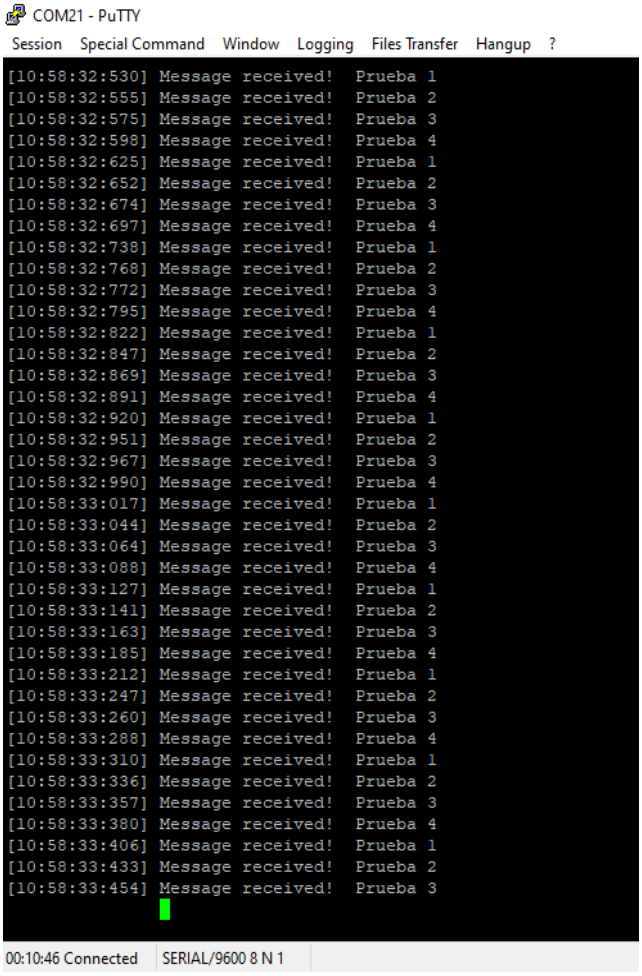

Figura 4.10. Puerto serial de la base central durante una prueba de sincronización del protocolo *TDMA* con cuatro nodos.

Fuente: elaboración propia.

Una vez que se corroboró que el sistema de comunicación funcionó con la ventana de tiempo de 25 ms, se procedió a probarlo con la trama completa de 32 bytes que realmente se enviaría a la base central. Se repitió la prueba mostrada en la Figura 4.10 y se descubrió que una ventana de tiempo de 25 ms no sería viable para un mensaje tan largo, pues se detectaron pérdidas considerables en los mensajes enviados, y desacomodo en el tiempo de los nodos. Por ende, se fue aumentando la longitud de la ventana de tiempo hasta que ya no hubiera pérdidas notables. Se

determinó que el valor de 50 ms daba un buen desempeño, por lo que se pasó a realizar la prueba final al sistema de comunicación desarrollado.

Este experimento final constó de poner en funcionamiento el sistema cuatro veces seguidas durante 15 minutos con 34 segundos cada una, y luego analizar cuántos mensajes se perdieron para determinar el porcentaje de recibimiento del sistema. Además, de los siete datos $(ID, x_P, y_P, \emptyset, tipo, \rho, \alpha)$ que se enviaron: el ID correspondió al de cada nodo en específico; x_P, y_P, $tipo$ y ρ correspondieron a números aleatorios generados en cada nodo; pero $\emptyset$ y α correspondieron a números estáticos, para así también observar la integridad de una parte de los bytes enviados. Concretamente, $\emptyset = 1.57$ y $\alpha = 281.65$ para los cuatro nodos.

En la Tabla 4.25 se observan los resultados de las cuatro iteraciones del experimento. Se muestra la cantidad de mensajes recibidos por cada nodo según la iteración correspondiente y, además, se muestra la cantidad total de mensajes recibidos en cada iteración y la cantidad teórica que se debería entonces de tener por cada nodo. Esta dato de mensajes por ID corresponde nada más a dividir entre cuatro la cantidad total de mensajes.

Tabla 4.25. Cantidad de mensajes recibidos en las pruebas al comunicar cuatro nodos con la base central.

Fuente: elaboración propia.

	Cantidad de mensajes recibidos			
ID	**Iteración 1**	**Iteración 2**	**Iteración 3**	**Iteración 4**
1	4767	4765	4767	4765
2	4755	4764	4761	4764
3	4765	4765	4766	4765
4	4767	4771	4774	4771
Total de mensajes	19054	19065	19068	19065
Mensajes por ID	4763.5	4766.25	4767	4766.25

5 Análisis de los resultados

En este apartado se analizarán los resultados expuestos en el capítulo 4, y su relación con el cumplimiento de los objetivos del presente proyecto de graduación.

5.1 Análisis de resultados del control automático de la navegación

Los resultados que se aprecian en la Figura 4.1 corroboraron que con la estrategia de control escogida era posible controlar la orientación del robot de manera implícita. Aquí se le indicó al robot que se desplazara a la coordenada $(15, 15)$ cm, en la imagen d) de la Figura 4.1 se ve la diagonal realizada por el robot al desplazarse y en la imagen c) de la Figura 4.1 se observa que se logró que tuviera una orientación final de aproximadamente 0.732 radianes (41.9°), que es cercano a la orientación que se esperaba (45°) al indicarle que se moviera de esa forma. Sin embargo, sí es importante resaltar que esta simulación no incluyó los controladores *PID* para la velocidad de los motores, y asumió que los motores seguían a la perfección la señal generada por la cinemática inversa, de ahí que las coordenadas (x_P, y_P) presentaron una salida ideal sin ruido ni error.

Por otro lado, es interesante observar de la ecuación (4.1) y la ecuación (4.2) que los polos del motor derecho son conjugados, mientras que los del motor izquierdo no. Estas diferencias en las plantas se deben a que, al ser motores de baja gama, los procesos de manufactura no tienen altos controles de calidad, generando así diferencias físicas de un motor a otro, como por ejemplo: masa, resistencia de armadura, fricciones mecánicas en las escobillas y los cojinetes y demás. Es importante que, a medida que el enjambre crezca en cantidad de robots, se dé importancia a este aspecto, pues para un enjambre de 20 robots se tendrían 40 modelos diferentes de motores CD, y por ende, 40 tríos de constantes diferentes para los controladores *PID* de la velocidad de los motores.

También es importante resaltar de la Figura 4.2, que los datos recopilados para obtener mediante MATLAB las ecuaciones (4.1) y (4.2), poseían oscilaciones importantes. Esto se observa en las crestas de la velocidad angular del motor izquierdo, donde se ve el mayor ruido en las mediciones realizadas durante los 8 y 10 segundos, y los 12 y 14 segundos. Esto se debió a que los datos fueron tomados con el motor bajo carga mecánica, que fue el peso del robot; por lo tanto, existió una fricción constante entre la rueda acoplada al motor y la superficie sobre la que se encontraba moviéndose el robot.

A pesar de haber estado presente estas perturbaciones en el modelado de los motores CD, en la Figura 4.4 se observa cómo los controles *PID* de velocidad permitieron que los motores giraran de manera tal que siguieran la referencia deseada sin problema alguno. Se observa además que los motores presentan el comportamiento de un sistema que se encuentra entre críticamente amortiguado y sobre amortiguado, que fue lo deseado al seleccionar las constantes de los controles *PID*; esto indica que no existe sobre impulso en la respuesta de los motores. Con los datos de la Tabla 4.1 se comprueba que el mayor error obtenido para ese experimento fue de 0.26%, lo que comprueba que los controladores *PID* pueden seguir referencias sin presentar sobre impulso y con errores en estado estacionario menores a 1.5%.

Cabe resaltar de los datos de la Tabla 4.1 que existe una desviación estándar considerable debida al ruido que se observa en los gráficos de tono claro en la Figura 4.4. Esto se debió al cableado de los codificadores, pues se corroboró que los cables al aire le incorporaron ruido a las mediciones de la velocidad. Se observa que este ruido no varió según la referencia y se mantuvo constante durante todo el experimento, lo que indica que fácilmente se podría filtrar. De la Figura 4.5, la Figura 4.6 y la Tabla 4.2 se puede observar que el ruido se presentó para todas las mediciones y experimentos hechos sobre los motores del robot, lo cual era esperado, pues los cables siempre estuvieron al aire y afectaron las mediciones.

Ahora bien, respecto a los resultados vistos en la Figura 4.5, es importante recordar que el objetivo de control no englobaba un controlador inmune a perturbaciones, sin embargo, se demostró en que los controladores *PID* permiten a los motores recuperarse ante perturbaciones bruscas como lo fue frenar por completo el motor; eso sí, con un efecto colateral, un sobre impulso considerable al recuperarse. Este sobre impulso se debe a la parte integrativa del controlador *PID*, que acumula el error conforme pasa el tiempo, y puede solucionarse fácilmente limitando el valor que esta componente del control puede alcanzar.

En la Figura 4.5 también se observar una peculiaridad: existe una leve relación entre la respuesta de un motor y el otro, pues cuando una rueda se frenó la otra presentó una pequeña disminución en su salida, y cuando a la rueda frenada se le permitió girar de nuevo, se ve que la otra rueda presenta un leve sobre impulso en su salida que es corregido rápidamente por el respectivo controlador *PID*. Se cree que esto se debe a las variaciones en el consumo de corriente del motor frenado, ya que un rotor estático crear un mayor consumo de corriente, haciendo que el

otro motor que sí gira reciba menos potencia. Al soltar la rueda y permitir que al motor contrario le llegue más corriente, se produce el pequeño sobre impulso.

Finalmente, en la Figura 4.6 se observa que cuando los motores poseen la carga del peso del robot igual logran seguir la referencia sin problema, pero con un poco más de oscilaciones. La Tabla 4.2 muestra que el mayor error de los controladores *PID* se obtuvo en este experimento, y tuvo un valor de 1.31%. Con estos datos se corrobora que los controladores *PID* diseñados cumplieron los requisitos del primer objetivo específico del proyecto.

Respecto a los resultados del controlador de posición, para el avance de 30 cm se observa en la Tabla 4.3 y la Tabla 4.4 que el error de la estimación es 0.78 puntos porcentuales (p.p.) menor (en valor absoluto) que el error del control. Además, se observa una leve disminución de la desviación de los datos en la estimación comparado con el control. En cuanto al valor del error en sí, el control obtuvo un error promedio de -2.16%, no logrando cumplir con el objetivo de obtener un error menor a ± 1.5% por 0.56 p.p. Sin embargo, la estimación del desplazamiento sí logró obtener un error promedio de 1.38% en el desplazamiento, teniendo varias mediciones del error por debajo del 0.5%, pero obteniendo un error máximo de 7.27% en una de las mediciones.

Es interesante observar el comportamiento del signo del error, pues en los errores de la acción de control de la Tabla 4.3, 13 de los 15 errores medidos son negativos. Esto quiere decir que el robot siempre se está movimiento ligeramente más de lo que se le indicó. En los errores de la estimación se observa que solo hay 2 de 15 mediciones con error negativo, que significa que el robot prácticamente siempre subestimó su ubicación.

En la Tabla 4.5 y la Tabla 4.6 se observan los resultados del experimento de avanzar 15 cm. Se nota una importante disminución en los promedios y las desviaciones estándar de los errores, a excepción de la desviación estándar del error de desplazamiento del control, ya que más bien aumentó 0.56 p.p. La estimación de la posición que realizó el robot presentó una excelente respuesta, pues el error en la estimación del desplazamiento presentó solamente en 2 de las 15 mediciones realizadas un error mayor a ±1.5%. Estos resultados demuestran que sí existe una relación entre el desplazamiento que realiza el robot y la cantidad de error que se acumula, por así decirlo, en la medición. Esto cumple con la teoría que se explica en los sistemas de odometría, donde a mayor distancia recorrida, mayor probabilidad de una medición con más error.

Es posible observar en la Tabla 4.7 y la Tabla 4.8 que a pesar de que la orientación se controla de manera implícita, al girar 90°, el control logra manejar el error en la orientación bastante bien, obteniendo un error promedio de 0.34%, pero en cuanto a la estimación, sí se obtuvo un error que se sale de lo deseado, teniendo un valor de -1.63%. En cuanto al giro de -90°, en los datos de la Tabla 4.9 y la Tabla 4.10 se puede observar que tanto para el control y la estimación del giro se obtienen porcentajes de error aceptables, estando ambos por debajo del 1.5% que se deseaba. Con lo anterior además se observa que no hay una aparente diferencia entre girar 90° o girar -90°, lo cual se deseaba corroborar con el diseño de experimento.

En resumen, en la Tabla 5.1 se pueden observar los errores de control y de estimación obtenidos para los diferentes movimientos. Se observa que de los ocho resultados obtenidos, solamente en tres no se logró cumplir con el objetivo de obtener un error menor a ± 1.5%, y la máxima diferencia entre lo obtenido y lo deseado fue de 0.56 p.p. Por lo que aun si no se logró cumplir con lo deseado, se obtuvo un muy buen resultado. Sin embargo, sí es necesario resaltar que las desviaciones estándar fueron altas, y podrían generar duda de si siempre el control y la estimación logran cumplir con el objetivo deseado.

Tabla 5.1. Errores obtenido para el sistema de control de la navegación del robot móvil.

Fuente: elaboración propia.

Experimento	Error de la navegación (%)	Error de la estimación (%)
Avanzar 30 cm	-2.16	1.38
Avanzar 15 cm	-1.90	-0.10
Girar 90°	0.34	-1.63
Girar -90°	-0.18	1.31

Lo anterior se puede deber a cuatro diferentes motivos:

1. **Calibración de odometría:** como se explicó en el marco teórico, existen dos tipos de errores sistemáticos que afectan a la odometría de un robot del tipo diferencial: una estimación errónea del valor de la distancia entre las ruedas del robot (el valor $2d$ en la Figura 2.4); y diámetros desiguales en las ruedas. Mediante experimentos,

se puede calibrar el robot por así decirlo, y disminuir esos errores. Lo anterior no se hizo en este proyecto, lo que significa que, aunque el control sea perfecto, siempre habrá un error asociado a la construcción mecánica del prototipo.

2. **Condiciones para alcanzar el error en estado estacionario:** se debe recordar al lector lo dicho en el párrafo que se encuentra debajo de la Figura 3.10, y es que para indicarle al robot que ya había alcanzado el estado estacionario, se manejó un margen de error en la coordenada x y y de 0.5 cm, mientras que para los movimientos en arco fue de 5°. O sea que se le permitió al control tener un margen de error, porque si no el robot nunca iba a salir de la acción de control y se quedaría oscilando alrededor de la referencia.
3. **Inexistencia de un método de frenado:** el robot no cuenta con ningún método de frenado, lo que podría introducir error a la hora de que el control de velocidad se acerque a velocidades de referencia igual a 0. El control de posición es capaz de controlar sobre impulsos, pero de todas formas debido a lo explicado en el punto 2, no tener un método de frenado puede generar errores por la inercia que no sean compensados.
4. **Ruido en la medición de los codificadores:** como ya se explicó en el análisis de la respuesta de los controladores *PID* de la velocidad de los motores, se observó una gran cantidad de ruido en la medición de los codificadores. Este ruido nunca se filtró durante los experimentos aquí presentados, y es muy posible que afectara la estimación de la cinemática directa del control, y por ende, también la acción de control.

Debido a lo anterior, es entendible que para el controlador de posición, no siempre se haya alcanzado el error en estado estacionario de 1.5% y que se obtuvieran desviaciones estándar altas.

5.2 Análisis de resultados de la integración de las funcionalidades de la *IMU* para mejorar el error de la orientación

Con los datos mostrados en la Tabla 4.11 se puede ver que tanto los errores como las desviaciones estándar de las mediciones tomadas con la *IMU* y el filtro de Madgwick son bastante bajas, y si se diera como dato general promediando los errores y las desviaciones, se podría decir

entonces que el instrumento presentó un error absoluto igual a 0.44 ± 0.22°, demostrando que la calibración de los sensores y la implementación del filtro fueron exitosos.

Es posible observar en las tablas correspondientes a los experimentos sin retroalimentación de la *IMU*, o sea las tablas: Tabla 4.12, Tabla 4.13, Tabla 4.16, Tabla 4.17, Tabla 4.20 y Tabla 4.21, que tanto los errores que presentó el control como los presentados por la estimación, fueron bastante altos. Los peores errores se obtuvieron del experimento de girar 90° (Tabla 4.16, Tabla 4.17), donde el error del control de la orientación fue de -13.89 ± 10.17° y el error de la estimación de la orientación fue de 15.13 ± 10.44°, ambos de la forma *promedio* ± *desviación estándar*; sin embargo, este resultado es de esperar, pues se le introdujo un obstáculo al robot que este no esperaba.

Mientras que, por otro lado, al introducir la medición de la *IMU* al sistema de control del robot móvil se observa en las tablas: Tabla 4.14, Tabla 4.18 y Tabla 4.22, que los errores de control de la orientación obtuvieron una considerable mejora respecto a los experimentos sin ayuda de la *IMU*. Eso sí, los errores de estimación de la orientación presentes en las tablas: Tabla 4.15, Tabla 4.19, y Tabla 4.23, no presentaron mejora notable. Lo anterior se debe a que, como ya se mencionó, el valor de la *IMU* se utiliza en la lógica de control mediante el subsistema de vigilancia de la orientación, pero nunca afecta a la estimación que realiza la cinemática directa del control de posición, la cual se basa enteramente en lo medido por los codificadores. Por lo tanto, aunque la *IMU* mejore la acción de control, la estimación que hace el robot no se ve mejorada por esta intervención.

En la Tabla 5.2 se resumen los errores promedio del control según el movimiento y si existía o no intervención de la *IMU*, de la forma *error promedio* ± *desviación estándar*. Se aprecia fácilmente que para cada uno de los casos se obtuvo una gran disminución del error en el control de la orientación del robot móvil, siendo la mayor reducción la presente en el experimento de girar 90°, donde se obtuvo una reducción de 10.70° (tomándolos como valor absoluto) en el promedio del error y una reducción de 9.21° en la desviación estándar. Con los resultados aquí mostrados se corrobora el cumplimiento del objetivo específico 2 del presente proyecto de graduación.

Tabla 5.2. Comparación de los errores de control de la orientación según el uso de la *IMU* ante errores del tipo deslizamiento.

Fuente: elaboración propia.

Experimento	Error del control de orientación sin *IMU* (°)	Error del control de orientación con *IMU* (°)
Avanzar 15 cm	9.68 ± 6.05	0.62 ± 0.99
Girar 90°	-13.89 ± 10.17	3.19 ± 0.96
Girar -90°	1.37 ± 5.01	-0.05 ± 1.34

5.3 Análisis de resultados del sistema de comunicación

En el experimento presente en la Figura 4.10, el sistema de comunicación llevaba casi 11 minutos de sincronizado, y es posible ver cómo la base recibió a la perfección el mensaje estático que estaba enviando cada nodo, que era "Prueba #ID", donde el ID era 1, 2, 3 o 4 para cada nodo. Además, es posible apreciar en la estampa de tiempo de los mensajes, la cual tiene la forma de [HH:MM:SS:MSMSMS], que los mensajes se recibían cada ~25 ms, como se determinó en un inicio para la ventana de envió de información. Por ejemplo, de arriba abajo, el primer mensaje se recibe a los 530 ms y el segundo a los 555 ms, luego de 555 ms a 575 ms; aquí se observa una diferencia primero de 25 ms y luego de 20 ms. El lector podrá ir viendo las diferencias de tiempo a lo largo de los mensajes recibidos que se muestran en la Figura 4.10, pero se corrobora que en general estas estuvieron alrededor de 25 ms.

Esta fluctuación del tiempo que separa el recibimiento de un mensaje y otro se debe a la manera en la que se inicializa el sistema de comunicación, pues se recuerda que la base central envía en un inicio 10 mensajes de sincronización, y no necesariamente todos los nodos reciban el reloj del máster en el mismo mensaje de sincronización. Entre un mensaje de sincronización y otro, existió aproximadamente 6 ms de separación, lo que evidentemente generará estos pequeños desfases con solo que un nodo reciba el reloj un mensaje después de los demás.

Ahora bien, respecto al experimento final que se le realizó al sistema de comunicación, teóricamente si cada nodo hablaba cada 50 ms durante un periodo de 15 minutos con 34 segundos, se debían recibir 18680 mensajes. En la Tabla 4.25 se observan los resultados de las cuatro

iteraciones del experimento y en promedio se recibieron 19063 mensajes por iteración, 383 mensajes de más. Esto indica que existieron leves desfases en la sincronización de los nodos. Concretamente, para que durante los 15 minutos y 34 segundos se recibieran en promedio 19063 mensajes fue porque los nodos estuvieron enviando información con una ventana de tiempo promedio igual a 48.995 ms. Esto demuestra que al igual que cuando la ventana de tiempo era de 25 ms, los mensajes no se recibían exactamente cada 50 ms.

También es posible ver de la Tabla 4.25 que en realidad el sistema tuvo una pérdida de datos bastante baja, por ejemplo, para la tercera iteración del experimento, se recibieron exactamente la cantidad de mensajes del nodo con ID 1 que se esperaban. Con las diferencias entre los mensajes esperados por nodo para cada iteración y los realmente recibidos, se crea la Tabla 5.3, que representa la cantidad total de mensajes fallados por nodo y el porcentaje de mensajes recibido correctamente por iteración.

Tabla 5.3. Cantidad total de errores presentes en el sistema de comunicación por nodo.

Fuente: elaboración propia.

	Cantidad de mensajes erróneos			
ID	**Iteración 1**	**Iteración 2**	**Iteración 3**	**Iteración 4**
1	+3.5	-1.25	0	-1.25
2	-8.5	-2.25	-6	-2.25
3	+1.5	-1.25	-1	-1.25
4	+3.5	+4.75	+7	+4.75
Total de fallos	17	9.5	14	9.5
Mensajes recibidos correctamente (%)	99.91	99.95	99.93	99.95

En la Tabla 5.3, un signo positivo frente al número significa que se recibió más de lo esperado, y un signo negativo que se recibió menos. Primero, se debe aclarar que no se puede recibir una fracción de mensaje, pero este resultado se obtiene entre comparar lo que teóricamente se debía recibir por nodo y lo que verdaderamente se recibió. Es interesante notar que siempre, lo

que se recibió demás por alguno o algunos nodos, es lo que se recibió de menos por otro u otros nodos. También resalta que el nodo 4 siempre recibió demás, y fue el que más mensajes recibió extra; mientras que por otro lado, el nodo 2 siempre recibió de menos, y fue el que menos recibió de los cuatro nodos del sistema.

Todos los nodos fueron programados con el mismo código, por lo que al menos en cuanto a software no existe una razón específica que explique este comportamiento. Sin embargo, en cuanto a hardware, se observó que escasas ocasiones, los nodos eran capaces de enviar un mensaje seguido de otro, seguramente por algún error a la hora de limpiar la bandera de la interrupción que indicaba que ya se debía enviar un mensaje. Además, es importante mencionar que ningún reloj *RTC* será igual a otro, lo que genera que los relojes se desvíen del valor teórico de conteo a tasas diferentes, esto puede explicar que a la larga, el nodo 4 envió más mensajes y el nodo 2 menos.

Con los resultados anteriores, se obtiene un porcentaje promedio de mensajes recibidos de 99.935%. Este porcentaje de error general se obtuvo promediando los cuatro porcentajes de error de cada iteración, obtenidos al dividir la cantidad de mensajes fallados entre la cantidad total de mensajes enviados. Además, el recibimiento de la información en sí también fue totalmente exitosa, pues el promedio de los $\emptyset$ y α recibidos en cada iteración dio justamente el valor enviado, junto a una desviación estándar de cero. Esto significa que toda la información que se envió fue aquella que se recibió, teniendo así un sistema de comunicación exitoso.

También hubo un comportamiento interesante en los resultados de este experimento. Y es que, teóricamente, los nodos debían enviar sus mensajes de la forma: nodo 1, nodo 2, nodo 3, nodo 4, nodo 1, nodo2, … Justo como se muestra en la Figura 4.10. Sin embargo, para este experimento, en las iteraciones 1, 2 y 3, el patrón con el que se recibieron los mensajes en la base central fue de 1, 3, 2, 4 y así consecutivamente. Solamente para la cuarta iteración sí se recibió el patrón 1, 2, 3, 4.

Se cree que esto se dio por la manera en la que la base central inicializa el sistema, pues como ya se dijo, ella en realidad envía su reloj diez veces al inicio y no solamente una. Si cada mensaje de sincronización tuvo una separación de aproximadamente 6 ms en el tiempo, esto es suficiente para que si un nodo recibe el reloj en el primer envío, pero otro nodo lo recibe en el octavo, exista un desfase de 48 ms entre el recibimiento del reloj, pudiendo crear ese cambió entre las ventanas de tiempo del nodo 3 y el nodo 2 en el patrón de recibimiento. Dados los resultados

de la Tabla 5.3 al menos se corrobora que no existe diferencia entre los errores de la cuarta iteración y las otras tres, lo que indica que este cambio en el patrón de recibimiento no afecta el porcentaje de mensajes recibidos con éxito.

Se obtuvo así entonces un sistema de comunicación que logró recibir correctamente el 99.935% de los mensajes enviados a la base central, cumpliendo exitosamente con el objetivo específico 3 de este proyecto de graduación.

6 Análisis económico

Como se mencionó anteriormente en este informe, el proyecto PROE E1F2 ya poseía el prototipo de robot que se utilizó para implementar el control de la navegación y el sistema de comunicación, además de todo el equipo secundario que se requirió para desarrollar este proyecto de graduación. No obstante, es importante cuantificar el costo total de la solución planteada para llevar un control de costos en caso de ser necesario en futuras fases de PROE. En la Tabla 6.1 se detalla el coste de todos los equipos, licencias, horas de trabajo y demás material utilizados en este proyecto.

Tabla 6.1. Inversión necesaria para llevar a cabo el proyecto de graduación.

Fuente: elaboración propia.

Descripción	Costo Unitario (₡)	Unidades	Costo Total (₡)
Chasís del robot	1.875	1	1.875
Paquete de tuercas, tornillos y separadores	3.510	1	3.510
Puente H	3.475	1	3.475
Convertidor de voltaje CD-CD	6.395	1	6.395
Feather M0	20.995	6	125.970
Paquete de cables	1.170	1	1.170
Motor con codificador óptico y rueda	9.315	2	18.630
Rueda de apoyo	5.780	1	5.780
Batería de Litio	9.900	1	9.900
IMU	11.650	1	11.650
Cámara	65.000	1	65.000
Trípode	12.000	1	12.000
Cables USB	2.000	5	10.000
Licencias[8] de MATLAB	2.198.176[9]	1	2.198.176
Software Arduino IDE	Gratis	1	Gratis
Software Motion Cal	Gratis	1	Gratis
Software Extra PutTTY	Gratis	1	Gratis
Pago mensual al desarrollador	83.600	4	334.400
Total			**2.807.931**

[8] Incluye todas las licencias necesarias para llevar a cabo el proyecto: MATLAB, *Simulink*, *System Identification*, Adquisición de imágenes y Procesamiento de imágenes.
[9] Precio por licencia anual, no permanente.

Es importante notar que el 78.3% del costo total del proyecto lo abarca las licencias del programa y las herramientas de MATLAB, por lo que sería prudente que el proyecto PROE E1F2 valore utilizar software matemático de código abierto como Octave para disminuir de manera considerable los costos asociados al desarrollo tecnológico. Además, el otro rubo grande del proyecto es el pago mensual de la remuneración económica, que representa el 11.9% del costo total de desarrollo. Si bien es cierto que no es mucho, sigue siendo una parte significativa, ya que es el segundo rubro más grande. Lo bueno es que este gasto se debe hacer una sola vez, y de ahí en adelante la solución se replica.

Como se acaba de mencionar, esos dos rubros son los que más consumieron dinero, haciendo un total del 90.2% del costo asociado al proyecto, lo que deja al restante un costo de 275 mil colones, que son aproximadamente $471. Si se compara esto con el costo del robot Khepera IV, que como se dijo anteriormente supera los $3800, se nota entonces que el proyecto PROE E1F2 va en buen camino para lograr su objetivo de tener un sistema multirobot de tipo enjambre de bajo costo para mapear escenarios estáticos.

7 Conclusiones y recomendaciones

7.1 Conclusiones

En el presente proyecto de graduación se desarrolló e implementó las rutinas de navegación necesarias para un primer prototipo de robot móvil que se utilizará en un sistema multirobot de tipo enjambre. Además, se diseñó e implementó con éxito las rutinas de un sistema de comunicación basado en el protocolo *TDMA*, para sincronizar efectivamente el flujo de información en el sistema multirobot de tipo enjambre.

Se desarrollaron controles *PID* para el control de la velocidad de los motores, que tuvieron errores en estado estacionario menores al 1.5% deseado y no presentaron sobre impulso alguno en la respuesta del sistema. Además, se implementó una estrategia de control de posición que, si bien es cierto no cumplió en 3 de los 8 experimentos con obtener un porcentaje de error en estado estacionario menor a 1.5%, sí demostró eficazmente la viabilidad y el potencial de esta solución, y sentó las bases para un futuro sistema más robusto. También se probó que es posible controlar implícitamente la orientación del robot móvil mediante un controlador lineal e invariante en el tiempo que regula la posición de un punto *P*.

Además, se demostró la contribución a la disminución del error del control de orientación en estado estacionario frente a errores del tipo deslizamiento que generó la integración de las funcionalidades de una *IMU* al sistema, logrando inclusive una reducción de 10.70° en el error promedio para el movimiento de girar 90°.

Por último, se demostró la eficacia de las técnicas *TDMA* y *DMTS* para crear un protocolo de acceso al medio que fiscalice el flujo de información en un sistema de comunicación para robótica de enjambre, recibiendo un 99.935% de los mensajes enviados a la base central. Además, se corroboró la robustez del sistema para enviar y recibir información del tipo flotante, conservando la integridad de los datos recibidos en todos los mensajes.

7.2 Recomendaciones

Una vez finalizada la solución del presente proyecto de graduación, se tiene una lista de recomendaciones que facilitarán y guiarán el futuro trabajo del equipo PROE para que sea de máximo provecho el trabajo que se realizó aquí. Estas son:

- Es de importancia que se haga una revisión de la teoría expuesta por Borenstein y Feng en [16] para así mejorar el desempeño del control automático y disminuir los errores en el control de navegación del robot móvil.
- El control de posición desarrollado tiene una forma sumamente básica: la retroalimentación proporcional, por lo que una vez comprendido lo que se desarrolló aquí, se recomienda explorar controladores más complejos e incluso evaluar la posibilidad de integrar un controlador *PID* o similar para mejorar la respuesta de la posición.
- Si bien es cierto que en los experimentos se observó una relación entre el desplazamiento, y el error final del control, no se determinó un avance óptimo para minimizar el error. Por lo tanto, sería interesante determinar concretamente el desplazamiento que genera un menor error en estado estacionario para la navegación.
- En el desarrollo de este proyecto nunca se consideró la opción de filtrar la velocidad medida por los codificadores, y es posible que esto generara error en la estimación que realiza la cinemática directa del control de posición, y por ende, también en la respuesta del control de posición en sí. Por lo que se recomienda evaluar el desempeño de los controladores aplicando algún tipo de filtro a estos sensor.
- Se recomienda fuertemente crear una placa electrónica propia pronto, para así evitar de raíz el ruido que se tiene en la medición de la velocidad angular de los motores por medio de los codificadores.
- Se recomienda investigar a fondo lo expuesto por Gong et al. en [31], para tener previsto desde un inicio cómo se estructurará la red de comunicación más adelante, para así optar por la manera más eficiente y asegurar la escalabilidad del protocolo *TDMA*.
- En este proyecto se utilizó el *RTC* a una frecuencia de 1 kHz para facilitar los cálculos de tiempo, y no se probaron diferentes frecuencias del reloj. Por lo que se recomienda estudiar el sistema de comunicación con mayores frecuencias en los *RTC* de los nodos, para mejorar la sincronización.
- La biblioteca Radio Head posee múltiples capas de protocolos de comunicación, siendo la más compleja una topología del tipo malla. Se recomienda evaluar la posibilidad de utilizar estas funcionalidades junto al protocolo *TDMA* aquí implementado, para así apuntar a topologías aún más robustas que permitan cumplir los objetivos finales de comunicación en un enjambre de robots.

Bibliografía

[1] C. Calderón, R. Solis, and T. Bustillos, "Path Planning on Static Environments based on Exploration with a Swarm Robotics and RRG Algorithms.," *2018 IEEE 38th Cent. Am. Panama Conv. (CONCAPAN XXXVIII)*, 2018.

[2] M. Dorigo, M. Birattari, and M. Brambilla, "Swarm robotics," *Scholarpedia*, vol. 9, no. 1, p. 1463, 2014.

[3] F. Arvin, J. Espinosa, B. Bird, A. West, S. Watson, and B. Lennox, "Mona: an Affordable Open-Source Mobile Robot for Education and Research," *J. Intell. Robot. Syst. Theory Appl.*, vol. 94, no. 3–4, pp. 761–775, 2019.

[4] H. Hamann, *Swarm Robotics: A Formal Approach*. 2018.

[5] I. Navarro and F. Matía, "An Introduction to Swarm Robotics," *ISRN Robot.*, vol. 2013, pp. 1–10, 2013.

[6] L. Parker, *Chap. 40 Multiple Mobile Robot System*. 2008.

[7] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," *Lect. Notes Comput. Sci.*, vol. 3342, pp. 10–20, 2005.

[8] B. Khaldi and F. Cherif, "An Overview of Swarm Robotics: Swarm Intelligence Applied to Multi-robotics," *Int. J. Comput. Appl.*, vol. 126, no. 2, pp. 31–37, 2015.

[9] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 3293–3298, 2012.

[10] J. Hilder, A. Horsfield, A. G. Millard, and J. Timmis, "The PSI swarm: A low-cost robotics platform and its use in an education setting," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9716, pp. 158–164, 2016.

[11] S. Wilson *et al.*, "Pheeno, A Versatile Swarm Robotic Research and Education Platform," *IEEE Robot. Autom. Lett.*, vol. 1, no. 2, pp. 884–891, 2016.

[12] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. 2004.

[13] K. Lynch and F. C. Park, *Modern robotics : mechanics, planning, and control*, no. May. 2017.

[14] A. Mendoza, "Vehículo Acuático Autónomo de Superficie para el Laboratorio de Visión, Gráfica y Robótica (VGR-Lab)," Instituto Tecnológico de Costa Rica, 2019.

[15] J. de Jesús Rubio, V. Aquino, and M. Figueroa, "Inverse kinematics of a mobile robot," *Neural Comput. Appl.*, vol. 23, no. 1, pp. 187–194, 2013.

[16] J. Borenstein and L. Feng, "Measurement and Correction of Systematic Odometry Errors in Mobile Robots," *IEEE Trans. Robot. Autom.*, vol. 12, no. 6, pp. 845–857, 1996.

[17] B. Kuo and F. Golnaraghi, *Automatic Control Systems*, 9th ed. John Wiley & Sons.

[18] K. Ogata, *Ingeniería de control moderna*, vol. 5, no. 10. 2018.

[19] T. Wildi, *Máquinas eléctricas y sistemas de potencia*. Pearson Educación, 2007.

[20] R. H. Bishop, *The mechatronics handbook*. 2002.

[21] Williams, "Programming Embedded Systems Second Edition Programming Embedded Systems, Second Edition with C and GNU Development Tools," pp. 1–288, 2017.

[22] A. Kuncar, M. Sysel, and T. Urbanek, "Calibration of low-cost triaxial magnetometer," *MATEC Web Conf.*, vol. 76, pp. 1–5, 2016.

[23] D. Tedaldi, "IMU calibration without mechanical equipment (Calibrazione di IMU svincolata da apparati meccanici)," Università degli Studi di Padova, 2013.

[24] S. Salehi, N. Mostofi, and G. Bleser, "A practical in - field magnetometer calibration method for IMUs," no. May, 2014.

[25] V. Y. Skvortzov, H. K. Lee, S. W. Bang, and Y. B. Lee, "Application of electronic compass for mobile robot in an indoor environment," *Proc. - IEEE Int. Conf. Robot. Autom.*, no. April, pp. 2963–2970, 2007.

[26] M. Looney, "A simple calibration for MEMS gyroscopes," *EDN Eur.*, no. July, pp. 28–31, 2010.

[27] S. Madgwick, "An efficient orientation filter for inertial and inertial/magnetic sensor arrays," 2010.

[28] J. P. Blázquez, "Introducción a los sistemas de comunicación inalámbricos," 2015.

[29] S. Rackley, *Wireless Networking Technology: From Principles to Successful Implementation*. Newnes, 2007.

[30] E. Leão, C. Montez, R. Moraes, P. Portugal, and F. Vasques, "Alternative path communication in wide-scale cluster-tree wireless sensor networks using inactive periods," *Sensors (Switzerland)*, vol. 17, no. 5, pp. 1–36, 2017.

[31] H. Gong, M. Liu, L. Yu, and X. Wang, "An event driven TDMA protocol for wireless sensor networks," *Proc. - 2009 WRI Int. Conf. Commun. Mob. Comput. C. 2009*, vol. 2, pp. 132–136, 2009.

[32] S. Bhandari and P. Gautam, "Implementation of RF communication with TDMA algorithm in swarm robots," *2008 IEEE Int. Conf. Technol. Pract. Robot Appl. TePRA*, no. 1, pp. 68–73, 2008.

[33] X. Che, I. Wells, G. Dickers, and P. Kear, "TDMA frame design for a prototype underwater RF communication network," *Ad Hoc Networks*, vol. 10, no. 3, pp. 317–327, 2012.

[34] I. K. Rhee, J. Lee, J. Kim, E. Serpedin, and Y. C. Wu, "Clock synchronization in wireless sensor networks: An overview," *Sensors*, vol. 9, no. 1, pp. 56–85, 2009.

[35] S. Ping, "Delay measurement time synchronization for wireless sensor networks," *IRB-TR-03-013, Intel Res. Berkeley Lab*, p. 12, 2003.

[36] D. Aneesh, "Tracking controller of mobile robot," *2012 Int. Conf. Comput. Electron. Electr. Technol. ICCEET 2012*, no. 4, pp. 343–349, 2012.

[37] R. Anushree and B. K. S. Prasad, “Design and development of novel control strategy for trajectory tracking of mobile robot: Featured with tracking error minimization,” *2016 IEEE Annu. India Conf. INDICON 2016*, pp. 1–6, 2017.

[38] Y. Kanayama, Y. Kimura, F. Miyasaki, and T. Noguchi, “A stable tracking control method for an autonomous mobile robot,” in *Proc. IEEE Int. Conf. on Robotics and Automation*, 1990, pp. 384–389.

[39] B. Sandeep Kumar Malu, J. Majumdar, S. Kumar Malu α, and J. Majumdar σ, “Kinematics, Localization and Control of Differential Drive Mobile Robot Global Journal of Researches in Engineering: H Kinematics, Localization and Control of Differential Drive Mobile Robot,” *Type Double Blind Peer Rev. Int. Res. J. Publ. Glob. Journals Inc*, vol. 14, no. 1, 2014.

[40] S. He, “Feedback control design of differential-drive wheeled mobile robots,” *2005 Int. Conf. Adv. Robot. ICAR ’05, Proc.*, vol. 2005, pp. 135–140, 2005.

[41] D. Diaz and R. Kelly, “On modeling and position tracking control of the generalized differential driven wheeled mobile robot,” *2016 IEEE Int. Conf. Autom. ICA-ACCA 2016*, no. 1, pp. 1–6, 2016.

[42] R. Halir and J. Flusser, “Numerically Stable Direct Least Squares Fitting Of Ellipses,” 1998.

[43] A. W. Fitzgibbon, M. Pilu, and R. B. Fisher, “Direct least squares fitting of ellipses,” *Proc. - Int. Conf. Pattern Recognit.*, vol. 1, pp. 253–257, 1996.

[44] W. C. Lee and C. W. Cai, “An orientation sensor for mobile robots using differentials,” *Int. J. Adv. Robot. Syst.*, vol. 10, 2013.

[45] M. McCauley, “RadioHead Packet Radio library for embedded microprocessors,” *https://www.airspayce.com/mikem/arduino/RadioHead/*, 2014. .

[46] Microchip Technology Inc., *SAM D21 Family*. 2018.

[47] MathWorks, “Measuring Planar Objects with a Calibrated Camera,” *https://www.mathworks.com/help/vision/examples/measuring-planar-objects-with-a-calibrated-camera.html*. .

Apéndices

A1. Desarrollo y validación del sistema de visión de PROE E1F2

El sistema de visión utiliza la webcam modelo *HD webcam Pro C920*, la cual permitió una resolución máxima de 2304x1536 pixeles para la toma de imágenes. La calibración utilizada en el sistema para eliminar las diferentes distorsiones presentes en la cámara fue la expuesta en uno de los foros guía de MATLAB [47]. La misma utiliza el *Camera Calibrator App* de la caja de herramientas de MATLAB para adquirir las imágenes correspondientes a la calibración y posteriormente exportarlas a un código de MATLAB para su procesamiento.

La buena toma de imágenes durante la calibración es muy importante para obtener datos correctos. Se utiliza un patrón de calibración tipo tablero para la misma. La calidad de la calibración se puede observar en el dato de error de proyección realizado por la aplicación, cuyo resultado lo arroja en pixeles; en la Figura A1.1 se muestra una captura de pantalla de la aplicación *Camera Calibrator App* de MATLAB, en la esquina superior derecha se observa el error promedio que se obtuvo en esa calibración. En general se trabajó con errores de menos de 0.35 pixeles.

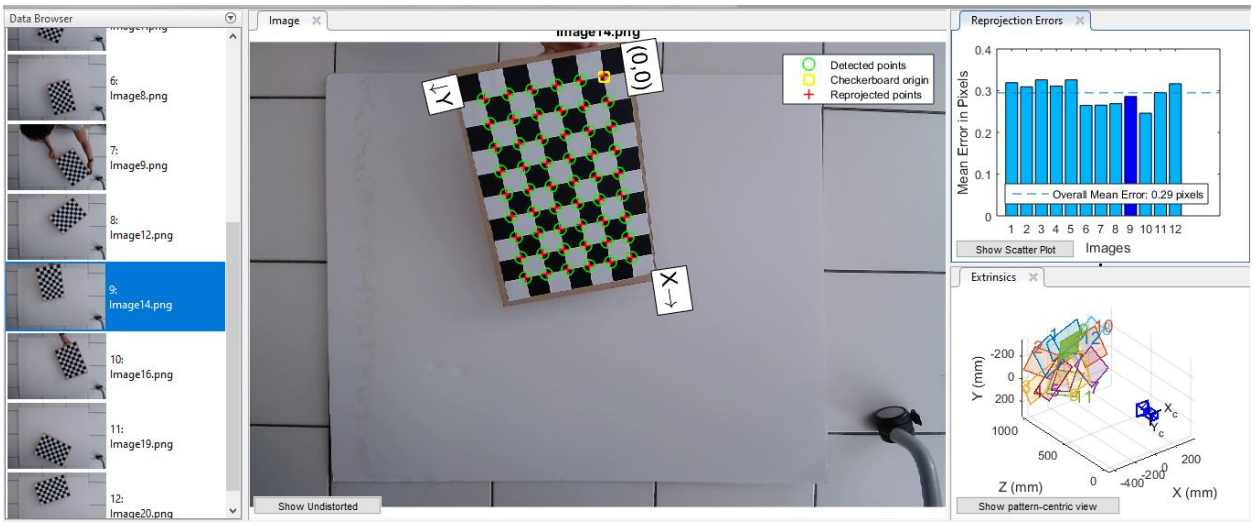

Figura A1.1. Captura de pantalla del proceso de calibración de la cámara.

Fuente: elaboración propia.

Se validó el sistema de visión realizando un diseño de experimento, donde se midió el diámetro de dos patrones circulares distintos, uno de 50 mm de diámetro y otro de 100 mm de diámetro, utilizando dos calibraciones distintas para la cámara. En la Tabla A1.1 se observan los

resultados obtenidos para la validación del sistema, donde para cada uno de los cuatro experimentos realizados se tomó un total de 10 mediciones y en la tabla se muestra el promedio de los errores obtenidos. Se determinó entonces que el error que presenta el sistema de visión es de ±0.37%, se escoge representarlo como error relativo para que este escale conforme escala la medición que se vaya a realizar

Tabla A1.1. Errores presentes en la validación del sistema de visión.

Fuente: elaboración propia.

Experimento		**Error absoluto (mm)**	**Error relativo (%)**
Patrón de 50 mm	**Primera calibración**	-0.053 ± 0.586	-0.106 ± 1.173
	Segunda calibración	-0.184 ± 0.247	-0.369 ± 0.494
Patrón de 100 mm	**Primera calibración**	0.045 ± 0.793	0.045 ± 0.793
	Segunda calibración	-0.260 ± 0.995	-0.260 ± 0.995

A2. Errores colaterales asociados a los movimientos del robot móvil

Tabla A2.1. Error existente en el control de la orientación en el experimento de avanzar 30 cm.

Fuente: elaboración propia.

Valor teórico (°)	Sistema de visión (± 0.01°)	Error del control de la orientación (°)
0	5.00	-5.00
0	5.76	-5.76
0	6.73	-6.73
0	4.15	-4.15
0	8.45	-8.45
0	7.24	-7.24
0	8.29	-8.29
0	9.44	-9.44
0	2.04	-2.04
0	5.86	-5.86
0	6.72	-6.72
0	8.17	-8.17
0	1.14	-1.14
0	7.04	-7.04
0	5.20	-5.20
Promedio del error:		-6.08
Desviación estándar asociada:		2.33

Tabla A2.2. Error existente en la estimación de la orientación en el experimento de avanzar 30 cm.

Fuente: elaboración propia.

Sistema de visión (± 0.01°)	Estimación del robot (± 0.01°)	Error de la estimación de la orientación (°)
5.00	1.58	4.93
5.76	2.66	8.59
6.73	-2.12	8.15
4.15	0.25	3.84
8.45	-0.86	9.23
7.24	-0.24	6.13
8.29	1.08	9.93
9.44	-1.02	7.24
2.04	-0.55	5.02
5.86	-2.51	4.37
6.72	1.89	6.66
8.17	0.38	5.48
1.14	-0.47	2.81
7.04	0.63	6.00
5.20	-1.26	5.36
Promedio del error:		6.25
Desviación estándar asociada:		2.05

Tabla A2.3. Error existente en el control de la orientación en el experimento de avanzar 15 cm.

Fuente: elaboración propia.

Valor teórico (°)	Sistema de visión (± 0.01°)	Error del control de la orientación (°)
0	2.57	-2.57
0	7.04	-7.04
0	-0.94	0.94
0	0.35	-0.35
0	1.15	-1.15
0	-0.28	0.28
0	1.49	-1.49
0	-0.12	0.12
0	-0.24	0.24
0	-1.77	1.77
0	0.02	-0.02
0	-0.38	0.38
0	-0.58	0.58
0	0.58	-0.58
0	-0.94	0.94
Promedio del error:		-0.53
Desviación estándar asociada:		2.09

Tabla A2.4. Error existente en la estimación de la orientación en el experimento de avanzar 15 cm.

Fuente: elaboración propia.

Sistema de visión (± 0.01°)	Estimación del robot (± 0.01°)	Error de la estimación de la orientación (°)
2.57	1.58	0.99
7.04	2.66	4.38
-0.94	-2.12	1.18
0.35	0.25	0.10
1.15	-0.86	2.01
-0.28	-0.24	-0.04
1.49	1.08	0.41
-0.12	-1.02	0.91
-0.24	-0.55	0.31
-1.77	-2.51	0.74
0.02	1.89	-1.87
-0.38	0.38	-0.75
-0.58	-0.47	-0.11
0.58	0.63	-0.05
-0.94	-1.26	0.32
Promedio del error:		0.57
Desviación estándar asociada:		1.38

Tabla A2.5. Error existente en el control del desplazamiento en el experimento de girar 90°.

Fuente: elaboración propia.

Valor teórico (cm)	Sistema de visión (cm ± 0.37%)	Error del control del desplazamiento (cm)
0.00	0.84	-0.84
0.00	0.52	-0.52
0.00	0.40	-0.40
0.00	0.40	-0.40
0.00	0.34	-0.34
0.00	0.35	-0.35
0.00	0.16	-0.16
0.00	0.19	-0.19
0.00	0.57	-0.57
0.00	0.56	-0.56
0.00	0.41	-0.41
0.00	0.37	-0.37
0.00	0.03	-0.03
0.00	0.63	-0.63
0.00	0.33	-0.33
Promedio del error:		-0.41
Desviación estándar asociada:		0.20

Tabla A2.6. Error existente en la estimación del desplazamiento en el experimento de girar 90°.

Fuente: elaboración propia.

Sistema de visión (cm ± 0.37%)	Estimación del robot (± 0.01 cm)	Error de la estimación del desplazamiento (cm)
0.84	1.14	-0.29
0.52	1.17	-0.65
0.40	1.45	-1.05
0.40	1.37	-0.97
0.34	1.57	-1.24
0.35	1.23	-0.88
0.16	1.33	-1.16
0.19	1.35	-1.15
0.57	1.50	-0.94
0.56	1.47	-0.91
0.41	1.52	-1.11
0.37	1.45	-1.09
0.03	1.12	-1.09
0.63	1.59	-0.97
0.33	1.36	-1.03
Promedio del error:		-0.97
Desviación estándar asociada:		0.23

Tabla A2.7. Error existente en el control del desplazamiento en el experimento de girar -90°.

Fuente: elaboración propia.

Valor teórico (cm)	Sistema de visión (cm ± 0.37%)	Error del control del desplazamiento (cm)
0.00	0.62	-0.62
0.00	0.86	-0.86
0.00	1.20	-1.20
0.00	0.55	-0.55
0.00	0.63	-0.63
0.00	1.19	-1.19
0.00	0.90	-0.90
0.00	0.34	-0.34
0.00	0.35	-0.35
0.00	0.20	-0.20
0.00	0.40	-0.40
0.00	0.30	-0.30
0.00	0.36	-0.36
0.00	0.42	-0.42
0.00	0.45	-0.45
Promedio del error:		-0.59
Desviación estándar asociada:		0.32

Tabla A2.8. Error existente en la estimación del desplazamiento en el experimento de girar -90°.

Fuente: elaboración propia.

Sistema de visión (cm ± 0.37%)	Estimación del robot (± 0.01 cm)	Error de la estimación del desplazamiento (cm)
0.62	0.95	-0.33
0.86	1.24	-0.38
1.20	1.70	-0.50
0.55	1.52	-0.97
0.63	1.43	-0.80
1.19	1.39	-0.20
0.90	1.21	-0.31
0.34	1.05	-0.72
0.35	1.23	-0.89
0.20	1.12	-0.92
0.40	1.34	-0.94
0.30	1.28	-0.97
0.36	1.10	-0.74
0.42	1.54	-1.12
0.45	1.39	-0.94
Promedio del error:		-0.71
Desviación estándar asociada:		0.29