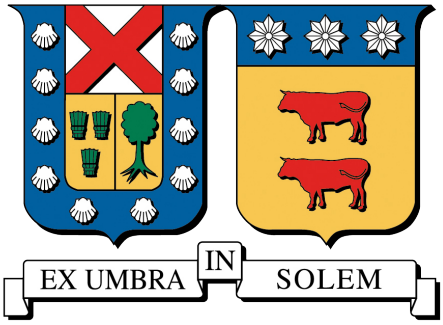
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VAPARAÍSO - CHILE



DISEÑO, IMPLEMENTACIÓN Y EVALUACIÓN DE UNA PLATAFORMA EXPERIMENTAL PARA EL ESTUDIO DEL CONTROL DE UNA FLOTA DE ROBOTS MÓVILES.

CARLOS MATÍAS ESCOBAR VALDIVIA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELECTRÓNICO

PROFESOR GUÍA	:	SR. FRANCISCO VARGAS PARRA.
PROFESOR CORREFERENTE	:	SR. GONZALO CARVAJAL BARRERA.
PROFESOR CORREFERENTE	:	SR. JUAN YUZ EISSMANN.

NOVIEMBRE 2021

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mi profesor guía Dr. Francisco Vargas Parra, quien con sus conocimientos y apoyo me orientó en de cada una de las etapas de este trabajo para alcanzar los resultados y conclusiones que buscaba. Agradezco también al Dr. Gonzalo Carvajal Barrera y al Dr. Juan Yuz Eissmann, por el apoyo e interés en mi trabajo.

Quiero agradecer a la Universidad Técnica Federico Santa María por apoyar el desarrollo de esta memoria a través principalmente del proyecto interno PI-LII-2020-38 “Diseño de técnicas de control para un sistema piloto de vehículos a escala autónomos considerando restricciones de comunicación entre vehículos”, y también a través del Centro Avanzado de Ingeniería Eléctrica y Electrónica, AC3E, Centro Basal ANID FB0008 y su Programa de Apoyo a Estudiantes Colaboradores.

Agradecer al profesor Andrés Peters, por su interés y apoyo en la gestión del proceso de fabricación de los componentes estructurales de los robots. También agradecer a Vicente Hernández Palominos, jefe de carrera del área de electrónica del Liceo Politécnico Andes, quien tuvo la gentileza y disposición de facilitar instrumentos de medición. Agradezco también a Sebastián Bórquez Gonzáles, Ing. en informática y compañero de la Universidad, por su asistencia en el procesamiento de los datos obtenidos en este trabajo. Su ayuda fue esencial para cumplir con los objetivos de mi memoria.

Por último, quiero agradecer a todos mis amigos y a mi familia, por apoyarme en todo momento y ofrecer su ayuda incondicional. En especial, quiero mencionar a mis padres, por inspirarme con su esfuerzo que han puesto desde siempre y nunca darse por vencidos.

Muchas gracias a todos.

RESUMEN

El objetivo de este trabajo es desarrollar una plataforma experimental para el estudio del control de flotas de robots móviles. Para este propósito, se reúnen los requisitos generales de diseño, los cuales, son definidos previamente por integrantes del proyecto interno PI-LII-2020-38, UTFSM, con participación del autor de esta memoria.

Como primera parte del trabajo, se estudian los materiales para implementar la superficie y trayectoria de navegación, y se diseña el hardware para un robot móvil de tracción diferencial (DDMR), utilizando herramientas computacionales para el diseño mecánico y electrónico.

En segundo lugar, se estudia el modelo matemático que describe el movimiento del DDMR sobre una trayectoria de navegación y la distancia que mantiene este con respecto a otro robot predecesor (modelo principal). El desarrollo del modelo se analiza desde un enfoque cinemático y dinámico. Adicionalmente, se construye un modelo complementario que incluye, además de las salidas de orientación y distancia, la velocidad lineal del DDMR. Por último, se analizan las principales propiedades del modelo como: estabilidad, observabilidad y controlabilidad.

A partir del modelo principal, se desarrolla un esquema de control basado en el regulador LQR, con el propósito de llevar las salidas a un punto de operación deseado. Además, se desarrolla un esquema de control utilizando el modelo complementario, basado en controladores PID para cada salida del modelo. Los parámetros de cada controlador se obtienen mediante un análisis de la respuesta a escalón de las salidas y las señales de control, para el sistema en lazo cerrado.

A continuación, se describen los modelos y algoritmos que permiten la implementación de los esquemas de control, la medición de señales de interés, el accionamiento de motores y la comunicación inalámbrica, a partir de un análisis del hardware utilizado y las condiciones del entorno de navegación. Para esto, se utilizan aproximaciones de los modelos de tiempo continuo a tiempo discreto, y algunas librerías ya desarrolladas por terceros.

Finalmente, se realizan diferentes pruebas experimentales que permiten validar el funcionamiento de la plataforma experimental. Las pruebas se enfocan en la información obtenida por el monitoreo de variables en línea para una flota compuesta por tres robots móviles diferenciales, para cada uno de los esquemas de control propuesto, y para diferentes escenarios de operación, donde se concluye que el desempeño de la navegación depende de la elección de los parámetros de cada esquema de control y de la velocidad en que se desplaza el agente líder (punto de operación). Se comprueba la capacidad de comunicación entre robots y como esta ayuda a mejorar el desempeño de la navegación de la flota. Finalmente, se comprueba la capacidad de la plataforma experimental para realizar estudios sobre el control de flotas de robots móviles.

Palabras Claves: plataforma experimental multi-agentes, robot móvil diferencial, navegación cooperativa, comunicación robot a robot, seguimiento de trayectorias, seguimiento de predecesor, flotas de robots.

ABSTRACT

The objective of this work is to develop an experimental platform for the study of mobile robot fleet control. For this purpose, the general design requirements are gathered, which are previously defined by members of the UTFSM internal project PI-LII-2020-38, and the author of this report.

Firstly, the materials to implement the navigation surface and trajectory are studied, and it is designed a hardware for a differential-drive mobile robots (DDMR), using computational tools for mechanical and electronic design.

Secondly, the mathematical model that describes the movement of the DDMR on a navigation trajectory and the distance it maintains concerning another predecessor robot (main model) is studied. The development of the model is analyzed from a kinematic and dynamic approach. Additionally, a complementary model is built that includes, in addition to the orientation and distance outputs, the linear velocity of the DDMR. Finally, the main properties of the model are analyzed: stability, observability, and controllability.

From the main model, it is developed an LQR based control scheme to bring the outputs to a desired operating point. In addition, a control scheme is developed by using the complementary model, based on PID controllers for each model output. The parameters of each controller are obtained by analyzing, the outputs' and the signals of control step response for the closed-loop system.

Next, the models and algorithms that allow the implementation of the control schemes, measuring signals of interest, the drive of motors, and wireless communication are described, from an analysis of the hardware used and the conditions of the navigation environment. For this, approximations of the continuous-time models to discrete-time and some libraries already developed by third parties are used.

To conclude, several experimental tests are performed to validate the operation of the experimental platform. The tests focus on the information obtained by monitoring online variables for a fleet composed of three differential mobile robots, for each of the proposed control schemes, and for different operating scenarios, where it is concluded that the navigation performance depends on the choice of parameters of each control scheme, and the speed at which the leading agent moves (operating point). The communication capability between robots and how it helps improve the navigation performance of the fleet is tested. Finally, the platform's experimental capability to perform studies on the control of mobile robot fleets is proved.

Keywords. multiagent experimental platform, differential mobile robot, autonomous navigation, robot-to-robot communication, trajectory tracking, predecessor tracking, robot fleets.

ABREVIACIONES

Mayúsculas

DDMR	: Robot Móvil de Tracción Diferencial (Diferential-Drive Movile Robot)
LQR	: Regulador Cuadrático Lineal (Linear-Quadratic Regulator)
PID	: Proporcional-Integral-Derivativo (Proportional-Integral-Derivarive)
V2N	: Vehículo a la Red (Vehicle to Network)
V2V	: Vehículo a Vehículo (Vehicle to Vehicle)
WLAN	: Red de Área Local Inalámbrica (Wireless Local Area Network)
LIDAR	: Tecnología de Detección por Luz (Light Detection and Ranging)
WiFi	: Tecnología de Comunicación Inalambrica (Wireless Fidelity)
ROS	: Sistema Operativo Robótico (Robot Operating System)
DC	: Corriente Directa (Direct Current)
MCU	: Microcontrolador (Microcontroller Unit)
MQTT	: Transporte de Telemetría de Cola de Mensajes (Message Queing Telemetry Transport)
CAD	: Diseño Asistido por Computadora (Computer-Aided Design)
MIMO	: Multiples Entradas, Multiples Salidas (Multiple Input, Multiple Output)
SISO	: Una Entrada, Una salida (Single Input, Single Output)
UDP	: Protocolo de Datagramas de Usuario (User Datagram Protocol)
IP	: Protocolo de Internet (Internet Protocol)
EVA	: Etileno-Vinil Acetato (Ethylene-Vinyl Acetate)
LED	: Diodo Emisor de Luz (Light Emitting Diode)
ToF	: Tiempo de Vuelo (Time of Flight)
IR	: Infrarrojo (Infra Red)
I2C	: Circuito Inter-Integrado (Inter-Integrated Circuit)
GPIO	: Entradas/Salidas de Propósito General (General-Purpose Input/Output)
PCB	: Tarjeta de Circuito Impreso (Printed Board Circuit)
E-L	: Euler-Lagrange
PWM	: Modulación por Ancho de Pulsos (Pulse-Width Modulation)
LTI	: Lineal e Invariante en el Tiempo (Linear Time-Invariant)
R1	: Robot Número Uno
R2	: Robot Número Dos
R3	: Robot Número Tres
HMI	: Interfaz Humano-Máquina (Human-Machine-Interface)
ADC	: Conversor Análogo-Digital (Analog-to-Digital Converter)
N-S	: Norte-Sur
BLE	: Bluetooth de Baja Energía (Bluetooth Low Energy)

SIMBOLOGÍA

Matrices

Desarrollo E-L:

$\mathbf{M}$: Matriz de inercias, $\mathbf{M} \in \mathbb{R}^{4\times4}$
$\mathbf{V_m}$: Matriz de interconexión y disipación de energía, $\mathbf{V_m} \in \mathbb{R}^{4\times4}$
$\mathbf{S}$: Matriz de entrada, $\mathbf{S} \in \mathbb{R}^{4\times2}$

Modelo E-L en Espacio de Estados:

$\mathbf{A}$: Matriz de estados, $\mathbf{A} \in \mathbb{R}^{4\times4}$
$\mathbf{B}$: Matriz de entrada, $\mathbf{B} \in \mathbb{R}^{4\times2}$
$\mathbf{C}$: Matriz de salida, $\mathbf{C} \in \mathbb{R}^{2\times4}$
$\mathbf{D}$: Matriz de paso directo, $\mathbf{D} \in \mathbb{R}^{2\times2}$

Modelo del Entorno de Navegación:

$\tilde{\mathbf{A}}$: Matriz de estados, $\tilde{\mathbf{A}} \in \mathbb{R}^{2\times2}$
$\tilde{\mathbf{B}}$: Matriz de entrada, $\tilde{\mathbf{B}} \in \mathbb{R}^{2\times2}$
$\tilde{\mathbf{C}}$: Matriz de salida, $\tilde{\mathbf{C}} \in \mathbb{R}^{2\times2}$
$\tilde{\mathbf{D}}$: Matriz de paso directo, $\tilde{\mathbf{D}} \in \mathbb{R}^{2\times2}$

Modelo Principal:

$\bar{\mathbf{A}}$: Matriz de estados, $\bar{\mathbf{A}} \in \mathbb{R}^{6\times6}$
$\bar{\mathbf{B}}$: Matriz de entrada, $\bar{\mathbf{B}} \in \mathbb{R}^{6\times2}$
$\bar{\mathbf{C}}$: Matriz de salida, $\bar{\mathbf{C}} \in \mathbb{R}^{2\times6}$
$\bar{\mathbf{D}}$: Matriz de paso directo, $\bar{\mathbf{D}} \in \mathbb{R}^{2\times2}$

Modelo Complementario:

$\mathbf{A_c}$: Matriz de estados, $\mathbf{A_c} \in \mathbb{R}^{6\times6}$
$\mathbf{B_c}$: Matriz de entrada, $\mathbf{B_c} \in \mathbb{R}^{6\times2}$
$\mathbf{C_c}$: Matriz de salida, $\mathbf{C_c} \in \mathbb{R}^{3\times6}$
$\mathbf{D_c}$: Matriz de paso directo, $\mathbf{D_c} \in \mathbb{R}^{3\times2}$

LQR:

$\mathbf{P}$: Matriz solución de la Ecuación Algebraica de Riccati
$\mathbf{K}$: Matriz de realimentación óptima

$\mathbf{Q}$: Matriz de ponderación de estados $\mathbf{x}$
$\mathbf{R}$: Matriz de ponderación de entradas $\mathbf{u}$

Vectores

Desarrollo E-L:

$\mathbf{q(t)}$: Coordenadas generalizadas, $\mathbf{q(t)} = [\phi_r(t), \phi_l(t), Q_{mr}(t), Q_{ml}(t)]^T$
$\mathbf{u(t)}$: Señales de entrada, $\mathbf{u(t)} = [u_{mr}(t), u_{ml}(t)]^T$

Modelo E-L en Espacio de Estados:

$\mathbf{x}$: Vector de estados, $\mathbf{x(t)} = [\dot{\phi}_r(t), \dot{\phi}_l(t), \dot{Q}_{mr}(t), \dot{Q}_{ml}(t)]^T$
$\mathbf{u(t)}$: Señales de entrada, $\mathbf{u(t)} = [u_{mr}(t), u_{ml}(t)]^T$
$\mathbf{y(t)}$: Señales de salida, $\mathbf{y(t)} = [v(t), \dot{\theta}(t)]^T$

Modelo del Entorno de Navegación:

$\tilde{\mathbf{x}}\mathbf{(t)}$: Vector de estados, $\tilde{\mathbf{x}}\mathbf{(t)} = [\phi_r(t), \phi_l(t)]^T$
$\tilde{\mathbf{u}}\mathbf{(t)}$: Señales de entrada, $\tilde{\mathbf{u}}\mathbf{(t)} = [v(t), \dot{\theta}(t))]^T$
$\tilde{\mathbf{y}}\mathbf{(t)}$: Señales de salida, $\tilde{\mathbf{y}}\mathbf{(t)} = [d(t), \theta(t)]^T$

Modelo Principal:

$\bar{\mathbf{x}}\mathbf{(t)}$: Vector de estados, $\bar{\mathbf{x}}\mathbf{(t)} = [\dot{\phi}_r(t), \dot{\phi}_l(t), \dot{Q}_{mr}(t), \dot{Q}_{ml}(t), \phi_r(t), \phi_l(t)]^T$
$\mathbf{u(t)}$: Señales de entrada, $\mathbf{u(t)} = [u_{mr}(t), u_{ml}(t)]^T$
$\tilde{\mathbf{y}}\mathbf{(t)}$: Señales de salida, $\tilde{\mathbf{y}}\mathbf{(t)} = [d(t), \theta(t)]^T$

Modelo Complementario:

$\mathbf{x_c(t)}$: Vector de estados, $\mathbf{x_c(t)} = [\dot{\phi}_r(t), \dot{\phi}_l(t), \dot{Q}_{mr}(t), \dot{Q}_{ml}(t), \phi_r(t), \phi_l(t)]^T$
$\mathbf{u(t)}$: Señales de entrada, $\mathbf{u(t)} = [u_{mr}(t), u_{ml}(t)]^T$
$\mathbf{y_c(t)}$: Señales de salida, $\mathbf{y_c(t)} = [v(t), d(t), \theta(t)]^T$

Escalares

$v(t)$: Velocidad lineal
$d(t)$: Distancia lineal
$\theta(t)$: Ángulo de orientación
$F(t)$: Fuerza externa
$\tau(t)$: Torque externo
$\phi(t)$: Posición angular
$Q(t)$: Carga eléctrica
$u(t)$: Señal de voltaje
$e(t)$: Fuerza contraelectromotriz
$\mathcal{L}(q, \dot{q})$: Funcional Lagrangiano (E-L)
$T(q, \dot{q})$: Energía cinética
$V(q)$: Energía potencial

$W'_m(q,\dot{q})$: Co-energía magnética
$W_e(q)$: Energía de campo electrico
$\mathcal{F}$: Función disipativa de Rayleight
$J(u)$: Funcional de costo cuadrático (LQR)

Subíndices

r, l : Rueda del lado derecho e izquierdo
ml, mr : Motor del lado derecho e izquierdo
$mín$: Mínimo
$máx$: Máximo
ref : Referencia
ed : Error de seguimiento de distancia
$e\theta$: Error de seguimiento de orientación

ÍNDICE

1. INTRODUCCIÓN **1**
1.1. Descripción del problema . . . 1
1.2. Estado del Arte . . . 1
1.3. Contribuciones . . . 3
1.4. Objetivo General . . . 3
1.5. Objetivos Específicos . . . 3
1.6. Metodología . . . 4
1.7. Contenido de la Memoria . . . 5

2. DISEÑO DE LA PLATAFORMA EXPERIMENTAL **6**
2.1. Superficie . . . 7
2.1.1. Trayectoria de navegación . . . 7
2.2. Robot móvil diferencial DDMR . . . 8
2.2.1. Hardware . . . 8
2.2.1.1. Unidad de procesamiento . . . 8
2.2.1.2. Sensores . . . 8
2.2.1.3. Actuadores . . . 10
2.2.1.4. Fuente de energía . . . 11
2.2.1.5. Interconexión del hardware . . . 11
2.2.2. Estructura . . . 14

3. MODELO MATEMÁTICO DDMR **17**
3.1. Cinemática . . . 17
3.2. Dinámica . . . 21
3.3. Modelo principal . . . 26
3.4. Modelo complementario . . . 27
3.5. Parámetros . . . 28
3.6. Simulación . . . 34

4. DISEÑO DE CONTROLADORES **41**
4.1. Esquema de Control basado en el regulador LQR . . . 41
4.1.1. Elección de las matrices Q y R . . . 42
4.1.2. Simulación . . . 43
4.2. Esquema de Control basado en controladores PID . . . 51
4.2.1. Simulación . . . 56

5. IMPLEMENTACIÓN DE LA PLATAFORMA EXPERIMENTAL **60**
5.1. Superficie y trayectoria . . . 60
5.2. Robot móvil diferencial . . . 61
5.3. Sensores . . . 62
5.3.1. Sensor de orientación . . . 62

5.3.2. Sensor de velocidad . . . 70
5.3.3. Sensor de distancia . . . 71
5.4. Actuadores . . . 73
5.5. Comunicación . . . 74
5.5.1. Comunicación entre robots . . . 75
5.5.2. Monitoreo . . . 77
5.5.3. Configuración de parámetros remota . . . 77
5.6. Controladores . . . 79
5.6.1. Esquema LQI . . . 79
5.6.2. Esquema PID . . . 80
5.6.3. Esquema PID y comunicación R2R . . . 80
5.6.4. Automatización basada en estados . . . 81
5.7. Resultados . . . 82
5.7.1. Resultados de navegación con controlador LQI . . . 82
5.7.1.1. Resultados en trayectoria recta - Controlador LQI . . . 82
5.7.1.2. Resultados en trayectoria cerrada con curvas - Controlador LQI . . . 86
5.7.2. Resultados de navegación con controlador PID . . . 87
5.7.2.1. Resultados en trayectoria recta - Controlador PID . . . 87
5.7.2.2. Resultados en trayectoria cerrada con curvas - Controlador PID . . . 91
5.7.3. Resultados de navegación con controlador PID y comunicación R2R . . . 95
5.7.3.1. Resultados de seguimiento de velocidad, con distancia constante entre robots - PID y comunicación R2R . . . 95
5.7.3.2. Resultados de seguimiento de distancia, con velocidad de navegación constante- PID y comunicación R2R . . . 98

6. CONCLUSIONES **102**
6.1. Resumen . . . 102
6.2. Conclusiones . . . 103
6.3. Trabajo Futuro . . . 105

BIBLIOGRAFÍA **106**

Índice de Tablas

2.1. Pines utilizados por sensores y actuadores en la tarjeta **D1 R32** 14

3.1. Relación de voltaje versus velocidad del motor DC 31
3.2. Relación de corriente versus velocidad de la rueda 33
3.3. Parámetros del DDMR 34

4.1. Parámetros de diseño del controlador de velocidad 54
4.2. Parámetros de diseño del controlador de distancia 55
4.3. Parámetros de diseño del controlador de orientación 56

5.1. Relación entre la secuencia de control y el sensor leído por el MCU 63
5.2. Relación entre los encoders y los objetos de la librería 70
5.3. Relación entre los pines de control del módulo puente H y los pines del MCU 73
5.4. Relación entre el signo de las variables **pwm_r** y **pwm_l**, con los pines de control del módulo puente H 74
5.5. Listado de comandos definidos para el operador de la plataforma experimental 78

Índice de Figuras

2.1. Diseño 3D de la plataforma experimental 6
2.2. Material de EVA modular 7
2.3. Diseño de la superficie y trayectoria de navegación modular 7
2.4. D1 R32: Tarjeta de desarrollo ESP32 basada en Arduino UNO 8
2.5. VL53L0X *Breakout Board* 9
2.6. Arreglo de sensores QRE113 *Breakout Board* 9
2.7. CD74HC4067 *Breakout Board* 9
2.8. Encoder de cuadratura, efecto Hall - acoplado a motor N20 10
2.9. Motor DC N20 - con caja reductora y *encoder* 10
2.10. TB6612FNG *Breakout Board* 10
2.11. Baterías Li-Ion 18650 11
2.12. Conexión entre el sensor de distancia y la tarjeta de desarrollo 11
2.13. Conexión entre los arreglos de sensores IR y la tarjeta de desarrollo 12
2.14. Conexión entre los motores N20 y la tarjeta de desarrollo 13
2.15. Diagrama de bloques de interconexión del hardware en el DDMR 14
2.16. Partes estructurales del DDMR 15
2.17. Diseño de la PCB para la interconexión de hardware del DDMR 15
2.18. Modelo 3D del DDMR 16
2.19. Modelo 3D del DDMR desarmado 16

3.1. Representación del DDMR con ambas ruedas aportando la misma velocidad. 17
3.2. Desplazamiento del DDMR cuando emplea un motor a la vez, operando en ambos casos a la misma velocidad 18
3.3. Desplazamiento del DDMR operando con ambos motores a la misma velocidad, pero diferente sentido 19
3.4. Interconexión mecánica motor DC - Rueda 20
3.5. Representación esquemática del motor DC 21
3.6. Diagrama del modelo de velocidades del DDMR en espacio de estados 24
3.7. Ángulo entre el DDMR y la trayectoria de navegación 25
3.8. Distancia del DDMR respecto a un robot líder 25
3.9. Diagrama del modelo de navegación del DDMR en espacio de estados 26
3.10. Diagrama de conexión serie de los modelos 3.6 y 3.9 26
3.11. Diagrama del modelo completo del DDMR en espacio de estados 26
3.12. Representación del movimiento rotacional del DDMR 29
3.13. Diagrama esquemático del circuito de medición empleado en motor DC con rotor detenido . 30
3.14. Respuesta temporal voltaje de entrada y corriente eléctrica medida en osciloscopio. 30
3.15. Voltaje inducido versus velocidad del motor DC 31
3.16. Circuito esquemático empleado para la estimación la inercia del motor DC 32
3.17. Respuesta transitoria del voltaje en el motor equivalente a la dinámica de la velocidad 32
3.18. Esquema de medición en el robot móvil diferencial con una rueda detenida 33
3.19. Velocidad rueda versus torque disipado 33

3.20. Esquema implementado en Simulink para simular el modelo principal del DDMR 36
3.21. Evolución temporal de las entradas y salidas del DDMR 36
3.22. Evolución temporal de los estados del DDMR - Respuesta a escalón 37
3.23. Evolución temporal de las entradas y salidas del DDMR 38
3.24. Evolución temporal de los estados del DDMR - Respuesta a escalón 38
3.25. Modelo en Simulink del accionamiento de los motores del DDMR 39
3.26. Señal PWM de entrada 39
3.27. Evolución temporal de las entradas y salidas del DDMR 39
3.28. Evolución temporal de los estados del DDMR - Respuesta a escalón 40
3.29. Rizado en los estados del modelo producto del control PWM 40

4.1. Esquema de control LQI 42
4.2. Esquema de control LQI implementado en Simulink 45
4.3. Respuesta temporal de las salidas del DDMR frente a cambios de referencia 45
4.4. Respuesta temporal de las señales de entradas del DDMR 46
4.5. Respuesta temporal de los estados del sistema 46
4.6. Respuesta temporal del error de seguimiento y su integral en el tiempo 47
4.7. Respuesta temporal de las salidas del DDMR frente a cambios de referencia 49
4.8. Respuesta temporal de las señales de entradas del DDMR 49
4.9. Respuesta temporal de los estados del sistema 50
4.10. Respuesta temporal del error de seguimiento y su integral en el tiempo 50
4.11. Esquema de control propuesto 51
4.12. Ilustración de sistemas conectados en cascada 55
4.13. Esquema de control PID implementado en Simulink 57
4.14. Respuesta tempo 57
4.15. Esquema de control PID implementado en Simulink 58
4.16. Esquema de control PID implementado en Simulink 58

5.1. Resultados de la implementación de la superficie y trayectoria modular 60
5.2. Ejemplo de ensamble de una trayectoria de navegación cerrada 61
5.3. Versiones prototipo del DDMR 61
5.4. Resultados de la implementación del diseño propuesto para el DDMR 61
5.5. Perspectivas del DDMR, utilizando 3 robots 62
5.6. Respuesta ideal del sensor infrarrojo 64
5.7. Función de normalización en función de los valores Mín. y Máx. de cada sensor 64
5.8. Ubicación de los sensores infrarrojos para la medición de la orientación 66
5.9. Arco de una circunferencia 66
5.10. Escenario de medición de la trayectoria utilizando un arreglo de sensores 67
5.11. Mediciones normalizadas del arreglo de sensores ubicado sobre la trayectoria de navegación 67
5.12. Representación del modelo de ponderación para el arreglo de sensores IR 68
5.13. Ejemplo de lectura y área generada por el sensor D1 69
5.14. Esquema de conexión de la plataforma experimental a la red Local 75
5.15. Topología de comunicación en cascada - comunicación entre robots 75
5.16. Topología de comunicación en estrella - monitoreo de señales 77
5.17. Topología de comunicación en estrella - configuración de parámetros 78
5.18. Descripción de estados de operación del DDMR 82
5.19. Seguimiento de distancia con controlador LQI - cambios de referencia - trayectoria recta . . 83
5.20. Acercamiento de la respuesta temporal del seguimiento de distancia de la Figura 5.19 83
5.21. Seguimiento de orientación con controlador LQI - trayectoria recta - prueba 1 84
5.22. Seguimiento de distancia con controlador LQI - perturbaciones con formación fija - trayectoria recta 85
5.23. Acercamiento de la respuesta temporal del seguimiento de distancia de la Figura 5.22 85
5.24. Seguimiento de orientación con controlador LQI - trayectoria recta - prueba 2 85

5.25. Seguimiento de distancia con controlador LQI - perturbaciones con formación fija - trayectoria con curvas 86
5.26. Seguimiento de orientación con controlador LQI - trayectoria con curvas - prueba 3 87
5.27. Seguimiento de distancia con controlador PID - cambios de referencia - trayectoria recta . . 88
5.28. Acercamiento de la respuesta temporal del seguimiento de distancia de la Figura 5.27 88
5.29. Seguimiento de velocidad con controlador PID - trayectoria recta - prueba 4 89
5.30. Seguimiento de orientación con controlador PID - trayectoria recta - prueba 4 90
5.31. Seguimiento de velocidad con controlador PID - cambios de referencia para el robot R1 - trayectoria recta 90
5.32. Seguimiento de distancia con controlador PID - R1 líder - trayectoria recta 91
5.33. Seguimiento de orientación con controlador PID - trayectoria recta - prueba 5 91
5.34. Resultados del seguimiento de velocidad con controlador PID 93
5.35. Resultados del seguimiento de distancia con controlador PID 94
5.36. Resultados del seguimiento de orientación con controlador PID 95
5.37. Resultados del seguimiento de velocidad con controlador PID 96
5.38. Resultados del seguimiento de distancia con controlador PID 97
5.39. Resultados del seguimiento de orientación con controlador PID 98
5.40. Resultados del seguimiento de velocidad con controlador PID 99
5.41. Resultados del seguimiento de distancia con controlador PID 100
5.42. Resultados del seguimiento de orientación con controlador PID 101

6.1. PMA: Plataforma multi-agentes 102

1 | INTRODUCCIÓN

1.1. Descripción del problema

El análisis de la navegación autónoma de robots móviles sobre una superficie es un problema que ha sido ampliamente estudiado, generalmente el enfoque se basa en controlar la posición del robot sobre un espacio o trayectoria de navegación conocida por el robot, desde un marco de referencia global, utilizando diferentes técnicas de control [29, 30, 12, 19, 9]. Analizar el posicionamiento y navegación del robot móvil resulta en la mayoría de los trabajos en modelos no lineales, debido a la necesidad de transformar las variables de interés desde el marco de referencia no inercial, a otro del tipo inercial, donde, algunos trabajos no consideran el modelo eléctrico de los actuadores en la dinámica del movimiento del robot.

Si bien, se ha estudiado bastante el problema de seguimiento de trayectoria, junto con la evasión de obstáculos por parte de un robot móvil, el desarrollo de este trabajo se enfoca en aquellos problemas que involucran la navegación autónoma de grupos de robots móviles [18, 8, 28, 11, 16]. Realizar este tipo de estudios implica contar con la tecnología e infraestructura correspondiente, por lo que desarrollar una plataforma experimental multi-agentes de bajo costo, permite facilitar la adopción a esta tecnología y avanzar en el estudio e investigación del comportamiento grupal y cooperativo de robots móviles.

1.2. Estado del Arte

En el artículo "*Duckietown: an Open, Inexpensive and Flexible Platform for Autonomy Education and Research*" [21], se presenta una plataforma abierta, flexible y de bajo costo, compuesta por vehículos autónomos (*Duckiebots*), ciudades (*Duckietowns*) con carreteras, señalética, semáforos, obstáculos y ciudadanos (*duck-ies*). Los *Duckiebots* navegan por la ciudad gracias a una cámara monocular y sensores que son procesados en una tarjeta de desarrollo Raspberry Pi 2, esto los hace capaces de navegar evitando obstáculos, peatones y otros *Duckiebots*. La plataforma presenta desarrollos que permiten a los robots localizarse en un mapa global de la ciudad y coordinarse con otros *Duckiebots* para evitar colisiones. También logra controlar de forma descentralizada la navegación autónoma de los vehículos, a través de una serie de estímulos visuales, y una coordinación entre robots, gracias a la configuración *fleet* que incorpora LED como medio para la comunicación entre los *Duckiebots*, basada en la detección por cámara de la frecuencia de encendido de estos. El artículo describe a grandes rasgos los componentes que posee cada *Duckiebot*, también presenta una tabla comparativa entre diversas plataformas de bajo costo en función de sus capacidades y características.

El artículo "*Networked and Autonomous Model-scale Vehículos for Experiments in Research and Education*" [24], presenta un vehículo a escala, el cual, es utilizado para el estudio del control de sistemas multi-vehículos no tripulados, donde existe una comunicación entre los agentes a través de hardware externo (*Vehicle-to-Network* – V2N). El vehículo posee dos modos de operación, el primero consiste en calcular las señales de control en hardware externo, transmitirlos hacia el vehículo a través de WLAN, y aplicar las señales recibidas a los actuadores; el segundo modo de operación consiste en transmitir una trayectoria de referencia a través de WLAN, que luego es seguida por el vehículo gracias a un controlador que se ejecuta en la placa Raspberry Pi Zero W. El diseño permite utilizar vehículos idénticos, para realizar experimentos

con una gran cantidad de agentes en red. El artículo presenta enlaces a material audiovisual con propósitos demostrativos, además abarca temas como la configuración del vehículo, descripción y comparación de los componentes, descripción del entorno e interfaces, modelo y control de los vehículos, junto con la presentación de plataformas recientes.

En la hoja de datos "*Self-Driving Car Research Studio Brochure*" [7], se presenta una plataforma desarrollada para propósitos experimentales y académicos. Donde el vehículo presentado posee una dirección tipo Ackermann. Dentro de los componentes principales se encuentra, el microcomputador Nvidia Jetson TX2, con capacidad de comunicación WiFi, y un sensor LIDAR, los cuales, facilitan a los vehículos navegar en espacios abiertos gracias al procesamiento de la información capturada del entorno. Es posible utilizar la plataforma para el estudio sobre el control colaborativo multi-vehículos en una red de comunicación V2V (*Vehicle-to-Vehicle*). La plataforma desarrollada por la compañía canadiense Quanser, proporciona funciones clave necesarias para la investigación de multi-vehículos a través de una variedad de módulos personalizables. Basándose en un conjunto de herramientas de software, que incluyen Simulink, Python, TensorFlow y ROS, el estudio permite a los investigadores crear aplicaciones de alto nivel y reconfigurar procesos de bajo nivel que son compatibles con bibliotecas y módulos prediseñados.

El artículo "*PL-TOON: A Low-Cost Experimental Platform for Teaching and Research on Decentralized Cooperative Control*" [22], presenta una plataforma experimental de bajo costo de implementación, cuyo propósito es servir de apoyo en cursos de teoría de control, al mismo tiempo que permite el estudio y emulación de resultados teóricos referente al control de un pelotón de vehículos no tripulados en una dimensión. En este se describe que los agentes que interactúan en la plataforma corresponden a trenes autónomos accionados por un motor DC, se presentan sus principales capacidades, donde destaca la unidad de procesamiento (MCU) de bajo costo, con capacidad de comunicación inalámbrica basada en tecnología WiFi. Cada MCU tiene acceso a la velocidad instantánea de su tren, la distancia a su predecesor inmediato, y algunas medidas de otros trenes mediante el uso del protocolo MQTT. Usando controladores PID y un esquema de red altamente descentralizado, las MCU alimentan los motores DC para llegar a un consenso de velocidad y una formación deseada, donde los vehículos mantienen espaciamientos fijos, siempre que sea posible. Adicionalmente, se presenta un enlace a un repositorio web, donde se pueden encontrar diseños 3D, códigos de ejemplo y material audiovisual del funcionamiento de la plataforma.

Las plataformas antes mencionadas, corresponden a algunas de las plataformas existentes. En particular, estas tienen en común que presentan un robot móvil y un escenario de operación en donde pueden interactuar múltiples agentes. Si bien todas presentan un vehículo capaz de establecer comunicación inalámbrica con otro agente y/o con su entorno, ninguna presenta una aplicación específica donde se utilice la cooperación entre los agentes. En algunas plataformas como Duckietown y PL-TOON, se puede destacar que existe una cierta coordinación entre los agentes gracias a los sensores que posee cada robot, pero esto limita el tipo de información que se comparten entre ellos. Además, esta coordinación ocurre en casos particulares, como cuando los vehículos se encuentran de frente, o en la misma dirección. Debido a lo anterior, también se limita el número de agentes que pueden interactuar de forma simultánea. Estas plataformas en general presentan vehículos de bajo costo y de baja complejidad en su construcción, debido al hardware y la configuración cinemática que emplean. Desde el punto de vista económico, la plataforma PL-TOON resulta ser la más económica si se piensa en estudiar aplicaciones con un gran número de agentes en red, además se puede aprovechar su hardware para establecer un canal de comunicación entre los agentes de forma inalámbrica a través de WIFI, pero su gran desventaja se presenta en las restricciones de movimiento que poseen sus agentes. Por otra parte, las plataformas citadas en [24] y [7], muestran vehículos de altas prestaciones, donde el hardware permite realizar aplicaciones relacionadas con la navegación autónoma y/o cooperativa de flotas, de un alto nivel y sin mayores restricciones en lo que respecta al movimiento de los vehículos, lo anterior, eleva el costo de estas plataformas, situación que dificulta las posibilidades de adquirir e implementar una aplicación con múltiples agentes en red de forma económica.

1.3. Contribuciones

La contribución de este trabajo es el desarrollo de una plataforma experimental para el estudio del control de flotas de robots móviles. La plataforma es completamente modular, modificable, reproducible y de bajo costo. Los robots móviles poseen una tracción diferencial, lo que les permite desplazarse en dos dimensiones, realizando movimientos de avance y retroceso en una sola dirección, con la capacidad irrestricta de modificar su ángulo de orientación (como una silla de ruedas). Cada robot posee la capacidad de identificar una trayectoria de navegación predefinida en la superficie de navegación y calcular su orientación respecto a esta, medir la distancia desde su parte frontal hasta un robot predecesor u objeto cercano, medir la posición y velocidad angular de cada rueda activa y comunicarse con otros robots, permitiendo la implementación de rutinas y esquemas de control cooperativas. Adicionalmente, es posible comunicarse con cada robot a través de un dispositivo externo a fin de realizar una configuración remota de los parámetros y adquisición en línea de los datos locales de cada robot.

El desarrollo de un modelo matemático del movimiento del robot diseñado, un estudio experimental de sus parámetros, junto con el desarrollo de un esquema de control LQI y otro esquema basado en controladores PID, son implementados para evaluar las capacidades de la plataforma experimental multi-agentes. La propuesta de modelos y algoritmos diseñados e implementados para el uso de sensores, actuadores y comunicación, junto con la propuesta e implementación de una estrategia de navegación cooperativa que emplea la información de velocidad del robot predecesor, se describen en este trabajo. Se puede acceder a todos los archivos de diseño, lista de materiales, simulaciones, datos experimentales, códigos de prueba, ejemplos demostrativos y material audiovisual, en el repositorio https://github.com/ProyectoMultiagentes/openPMA.

1.4. Objetivo General

El objetivo general de este trabajo es el diseño, implementación, y caracterización de una plataforma experimental de bajo costo, compuesta por una superficie y trayectoria de navegación, y una flota de robots móviles diferenciales con capacidad de comunicación inalámbrica usando tecnología WiFi.

1.5. Objetivos Específicos

Los objetivos específicos de este trabajo son:

1. Implementar una superficie, trayectoria de navegación y una flota compuesta por robots móviles diferenciales.
2. Posibilitar el objetivo 1, mediante el diseño mecánico y electrónico, basado en los requisitos derivados del proyecto interno PI-LII-2020-38 “Diseño de técnicas de control para un sistema piloto de vehículos a escala autónomos considerando restricciones de comunicación entre vehículos”.
3. Evaluar las capacidades de la plataforma experimental, mediante la automatización de la flota de robots móviles.
4. Posibilitar el objetivo 3, mediante el diseño de un esquema de control LQI para el seguimiento de trayectoria y distancia respecto a un robot predecesor, y un esquema basado en controladores PID para el seguimiento de trayectoria, seguimiento de predecesor, y control de velocidad lineal.
5. Posibilitar el objetivo 3, mediante el diseño de algoritmos para la lectura de sensores, accionamiento de actuadores, aproximación de controladores, comunicación entre robots, monitoreo de señales y configuración remota de parámetros.
6. Posibilitar el objetivo 3 y 4, mediante la obtención de un modelo matemático principal, a partir de un análisis cinemático y dinámico del DDMR, cuya entrada sea el voltaje aplicado a los motores, y su

salida, la orientación del robot respecto de una trayectoria de navegación y la distancia que mantiene este respecto de otro robot predecesor. Además, un modelo complementario que corresponde a una extensión del modelo principal, que incorpora la dinámica de la velocidad lineal del DDMR.

7. Validar los diseños y desarrollos propuestos, mediante pruebas de funcionamiento, a fin de apreciar de forma explícita las capacidades de la plataforma experimental presentada.

1.6. Metodología

Inicialmente, se realiza un trabajo de diseño mecánico y electrónico utilizando el software Autodesk Fusion 360 y el software Autodesk Eagle, respectivamente. Utilizando estas herramientas, se obtiene un CAD (diseño asistido por computadora) de los componentes de la plataforma experimental. A partir de los diseños, se genera una lista de materiales y los archivos de fabricación para la estructura del DDMR. Luego, se gestiona la compra y fabricación de todos los componentes de la plataforma experimental, para lo cual, se considera un total de tres robots móviles diferenciales para efecto del desarrollo y evaluación de este trabajo.

A partir del diseño CAD y un prototipo del DDMR, se realiza un trabajo teórico, en el cual, se obtiene el modelo matemático del robot móvil diferencial. Esto se realiza utilizando la ecuación de Euler-Lagrange (E-L), basada en el Cálculo de Variaciones. Luego, se obtiene una representación en el espacio de estados, y se estiman todos los parámetros del modelo matemático mediante una serie de pruebas experimentales, las cuales, se basan principalmente en el estudio de la respuesta a escalón del sistema real (DDMR). Una vez desarrollado el modelo en forma teórica, se realiza un análisis de sus propiedades y simulaciones del mismo mediante el software MATLAB y Simulink.

El modelo matemático desarrollado es el punto de partida para el desarrollo conceptual de dos esquemas de control. El primero corresponde a un esquema basado en el regulador LQR y acción integral, el que se diseña para controlar la navegación del robot a través de una trayectoria, al mismo tiempo, que se preocupa de controlar la distancia de seguimiento que mantiene respecto de otro robot predecesor. El segundo esquema de control, aprovecha la simetría que posee el modelo del DDMR para "transformar" el modelo MIMO obtenido, en el cual, las salidas corresponden a una combinación lineal de las entradas, a un modelo multivariable donde cada salida está relacionada con una sola entrada, esto permite diseñar controladores PID SISO para cada relación entrada salida, además es posible incorporar la información de la velocidad lineal del DDMR para diseñar un esquema de control en cascada, donde el controlador de distancia (lazo externo) regula una referencia para el controlador de velocidad (lazo interno). Este último esquema aporta a cada robot la capacidad de seguir referencias de velocidad "apagando" el controlador de distancia.

Los esquemas de control se implementan en el microcontrolador ESP32 a través del software Arduino IDE, esto se realiza mediante la aproximación de los modelos de tiempo continuo a tiempo discreto, luego son transcritos al lenguaje de programación C++. Las funciones necesarias para medir las señales de interés, comandar los actuadores, establecer y coordinar la comunicación entre robots, el monitoreo y configuración remota de variables, se desarrollan a partir del estudio del funcionamiento de cada uno de los componentes, y son implementadas en el microcontrolador, en algunos casos, utilizando librerías desarrolladas por terceros.

Los resultados experimentales son obtenidos realizando pruebas de funcionamiento a la plataforma. Gracias a la capacidad de los robots para enviar información utilizando el protocolo de comunicación UDP, a través de una red WiFi local, cada robot hace envío de sus señales procesadas localmente a una dirección IP y puerto de un computador conectado a la misma red local. A través de un script en Python es posible abrir el puerto local, acceder a los datos de forma remota y almacenarlos en un archivo de datos *.csv*. Además, cada una de las pruebas realizadas tiene asociado un material audiovisual que permite realizar un análisis cualitativo del desempeño y capacidades de la plataforma experimental.

1.7. Contenido de la Memoria

- En el Capítulo 2 se describen los componentes principales de la plataforma experimental propuesta, y se presenta el diseño mecánico y electrónico del DDMR.
- En el Capítulo 3 se desarrolla y se obtiene un modelo matemático del robot móvil diferencial, necesario para el diseño y validación de esquemas de control que permiten ilustrar y evaluar, las capacidades del DDMR y las aplicaciones de la plataforma experimental.
- En el Capítulo 4 se presenta un esquema de control LQI para el seguimiento de orientación y distancia del DDMR respecto de una trayectoria de navegación y un agente predecesor, respectivamente. También se describe un esquema de control basado en controladores PID SISO que incorpora la información de la velocidad lineal del DDMR.
- En el Capítulo 5 se presenta la implementación de la plataforma experimental desarrollada, los esquemas de control diseñados, algoritmos para la adquisición de datos de sensores y operación del DDMR, y topologías de comunicación. Además, se muestran los resultados experimentales obtenidos, los cuales, ilustran sus capacidades, y validan los esquemas de control propuestos.
- En el Capítulo 6 se entregan las conclusiones del trabajo.

2 | DISEÑO DE LA PLATAFORMA EXPERIMENTAL

En este capítulo se presenta el diseño mecánico y electrónico de los componentes de la plataforma experimental, los cuales, se desarrollan haciendo uso de los programas Autodesk Fusion 360 y Autodesk Eagle, respectivamente. Los componentes de la plataforma experimental se pueden clasificar, de forma general, como:

1. **Superficie**: Corresponde a la zona sobre la cual se debe desplazar uno o más robots móviles.
    - **Trayectoria de navegación**: Corresponde a un patrón continuo, definido sobre la superficie, el cual, tiene el propósito de definir un camino para los robots.
2. **Robot móvil diferencial**: Corresponde a un sistema mecatrónico, que posee la capacidad de desplazarse por el accionamiento de dos ruedas activas, paralelas entre sí y no orientables.
    - **Hardware**: Corresponde a los componentes electrónicos que permiten el funcionamiento autónomo del robot.
    - **Estructura**: Corresponde a un soporte mecánico para el hardware, este define el tamaño y forma del robot.

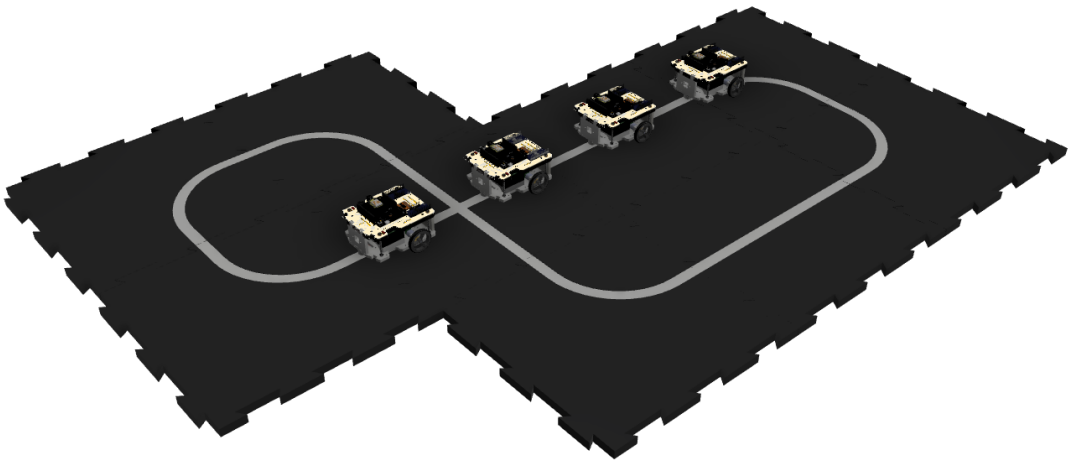


Figura 2.1: Diseño 3D de la plataforma experimental

2.1. Superficie

Para el diseño de la superficie, se estudia un material de EVA (etileno-vinil-acetato), comercializados en rectángulos modulares, tal como muestra la Figura 2.2. Esto le brinda la característica modular, y modificable, además de facilitar su transporte y almacenamiento. Se escoge en color negro, para evitar la aparición de manchas que se produzcan del uso de la plataforma.

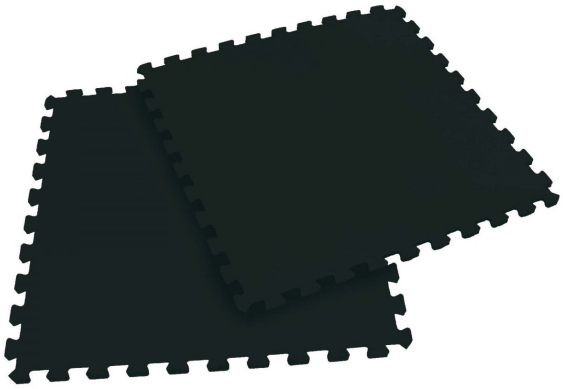


Figura 2.2: Material de EVA modular

2.1.1. Trayectoria de navegación

Sobre la superficie modular de color negro descrita anteriormente, se diseña una trayectoria de 2 cm de ancho y de color blanco, tomando en consideración que el contraste de color entre la superficie y trayectoria es importante para facilitar la navegación de los robots. Se diseñan secciones de trayectoria para cada pieza de EVA, que en conjunto puedan formar un circuito de navegación, abierto o cerrado. Inicialmente, se proponen las secciones tipo: curvas, rectas y cruces, tal como se muestra en la Figura 2.3. Así, conectando módulos individuales, se pueden formar superficies y trayectorias simples o complejas.

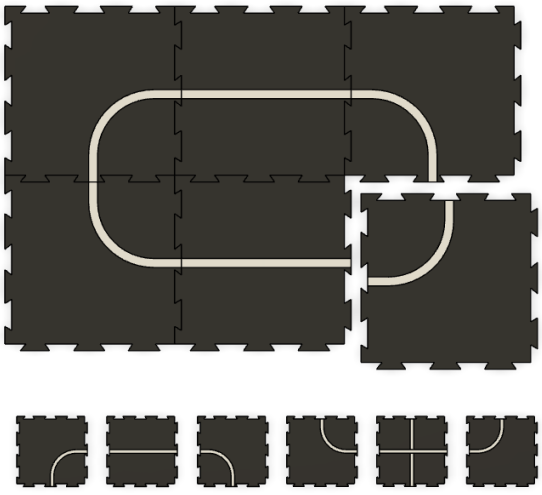


Figura 2.3: Diseño de la superficie y trayectoria de navegación modular

2.2. Robot móvil diferencial DDMR

2.2.1. Hardware

Los componentes del DDMR corresponden principalmente a *Breakout Boards*, las cuales, son tarjetas de circuito impreso diseñadas para realizar una función específica (según el circuito que implementen), y se pueden utilizar directamente a través de sus pines. Estas tarjetas están listas para ser usadas, y no requieren de algún tipo de diseño de circuito extra, de esta manera, la parte electrónica del DDMR corresponde a una integración de tarjetas de desarrollo que cumplen una función específica. Las principales características que motivaron la elección de los componentes son sus capacidades, costo, documentación asociada, y su demanda. También se tomó en consideración algunos de los componentes utilizados en plataformas similares.

2.2.1.1. Unidad de procesamiento

El DDMR se diseña con el microcontrolador ESP32 [5], el cual, posee un procesador doble núcleo Xtensa LX6, que opera a 240 MHz. Uno de los núcleos, está principalmente dedicado a la comunicación inalámbrica WiFi y Bluetooth de bajo consumo energético (BLE). Posee múltiples periféricos que permiten la interacción con dispositivos externos. La ESP32 se utiliza mediante la tarjeta de desarrollo D1 R32[1], la cual, está basada en la tarjeta de desarrollo Arduino UNO (ver Figura 2.4), e incluye un puerto de programación USB, un botón de reinicio, indicadores LED, un conjunto de pines conectados a los periféricos de entrada y salida de la ESP32, y un circuito acondicionador de voltaje para energizar el módulo y algunos dispositivos externos.

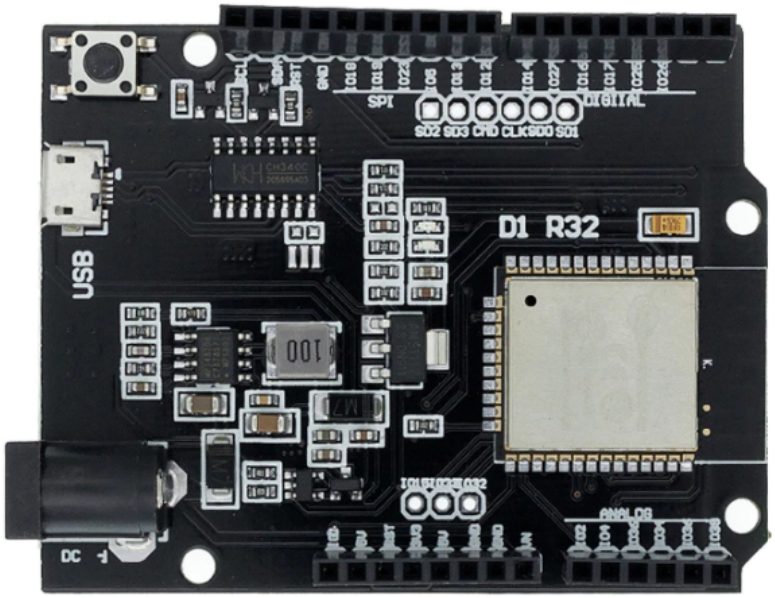


Figura 2.4: D1 R32: Tarjeta de desarrollo ESP32 basada en Arduino UNO

2.2.1.2. Sensores

El DDMR utiliza el sensor VL53L0X [4] en su parte frontal, para medir la distancia que mantiene respecto de su robot predecesor. Este es un dispositivo de tamaño reducido, basado en la medición del tiempo que toma un haz de luz láser en ser reflejado por el objeto más cercano (ToF), puede medir distancias absolutas

[1] Documentación: https://github.com/SmartArduino/SZDOITWiKi/wiki/ESP8266---ESPduino-32

de hasta 2 m en menos de 30 ms. Posee una interfaz I2C para controlar el dispositivo y transferir los datos. El sensor mencionado se puede apreciar en la Figura 2.5.

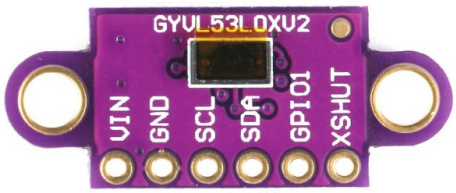


Figura 2.5: VL53L0X *Breakout Board*

Por otro lado, el robot utiliza un arreglo con 8 sensores infrarrojos QRE1113 [6], para medir su orientación respecto a la trayectoria de navegación (cuando está sobre la trayectoria). El sensor QRE1113 posee un diodo que emite luz infrarroja (IR), y un fototransistor que recibe la luz que se refleja en un objeto cercano. El arreglo de sensores que se muestra en la Figura 2.6 permite energizar y poner en un punto de operación a cada uno de los 8 sensores, esto permite que cada sensor genere una señal de voltaje analógica, que varía en el rango de 0 V y 3.3 V, en función de la intensidad de la luz reflejada hacia cada uno.

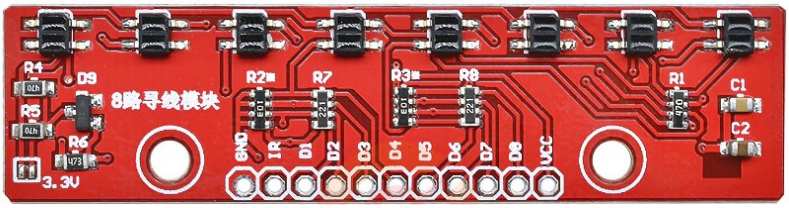


Figura 2.6: Arreglo de sensores QRE113 *Breakout Board*

Se utiliza un arreglo de sensores en la parte delantera del robot y otro en la parte de atrás, cada uno, orientado de forma horizontal, apuntando hacia la superficie. Los arreglos de sensores pueden medir la orientación del DDMR por separado, y cada uno es usado según el sentido del movimiento del robot (hacia adelante o hacia atrás).

Para conectar las salidas de ambos sensores a la tarjeta de desarrollo D1 R32, se utiliza el multiplexor analógico CD74HC4067 [2], el cual, posee 16 canales, esto permite conectar directamente los dos arreglos de sensores. A través de la secuencia lógica configurada en los pines de control del multiplexor, se puede seleccionar cada uno de los canales y acceder a su valor analógico a través de la salida del multiplexor.

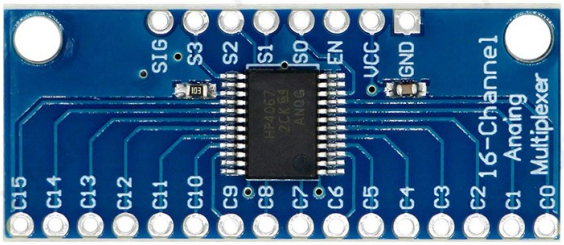
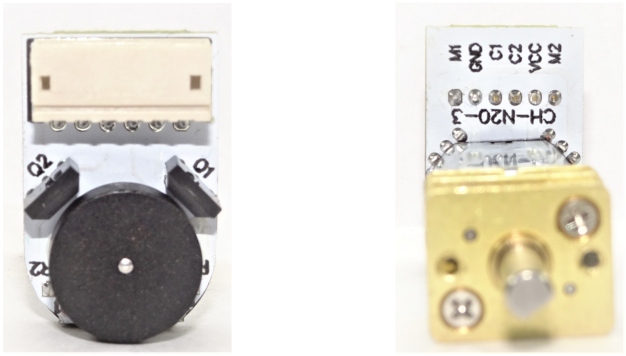


Figura 2.7: CD74HC4067 *Breakout Board*

El DDMR emplea un *encoder* de cuadratura (ver Figura 2.8) para medir la posición y velocidad angular de cada rueda activa (uno por cada motor). El módulo está basado en sensores de efecto Hall, los cuales operan en corte y saturación, en función del sentido del campo magnético que miden desde el disco magnético de 14 polos N-S (7 pares de polos) acoplado en la parte de atrás del motor DC, posee dos salidas

digitales que están desfasadas en 90° (cuadratura), por lo que usando ambas señales, se puede determinar el sentido de giro del motor.



(a) Vista frontal del *encoder*

(b) Vista posterior del *encoder*

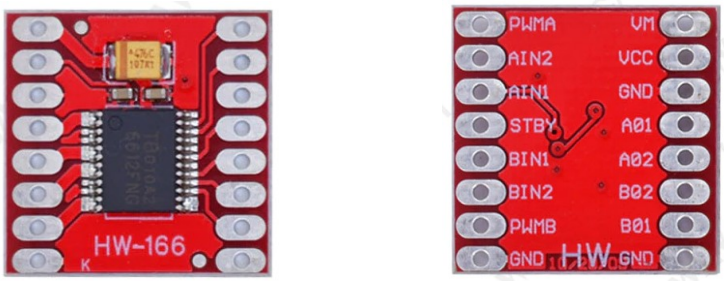
Figura 2.8: Encoder de cuadratura, efecto Hall - acoplado a motor N20

2.2.1.3. Actuadores

El DDMR cuenta con dos motores DC N20 (ver Figura 2.9), el cual, incorpora el *encoder* de cuadratura mencionado anteriormente y un mecanismo que reduce la velocidad en el eje de salida según la razón entrada/salida de 100/1. Su velocidad nominal sin carga es de 15000 RPM a 6 V, es decir, 150 RPM de salida. Su consumo máximo es de 0.55 A en el arranque. Para controlar ambos motores de forma independiente, se utiliza el circuito integrado, driver Puente H TB6612FNG [3], el cual, posee dos canales independientes, donde cada uno soporta una corriente promedio de 1.2 A y una corriente *peak* de 3.2 A. Este circuito integrado se utiliza sobre el módulo que aparece en la Figura 2.10.



Figura 2.9: Motor DC N20 - con caja reductora y *encoder*



(a) Vista frontal del módulo Puente H

(b) Vista frontal del módulo Puente H

Figura 2.10: TB6612FNG *Breakout Board*

.

2.2.1.4. Fuente de energía

Para energizar el DDMR, se utilizan dos baterías de Iones de Litio, conectadas en serie, entregando un voltaje nominal de 7.2 V, y de 8.4 V como máximo. Su capacidad en promedio es de 2200 mAh, lo cual, es suficiente para entregar una gran autonomía a los robots. Su formato cilíndrico 18650, es globalmente comercializado. Las baterías se conectan directamente al pin VM del módulo puente H, y al pin VIN de la tarjeta D1 R32, los sensores y circuitos externos son alimentados a través de los pines 5V y 3V3 provenientes de los reguladores de voltaje internos de la tarjeta de desarrollo.



Figura 2.11: Baterías Li-Ion 18650

2.2.1.5. Interconexión del hardware

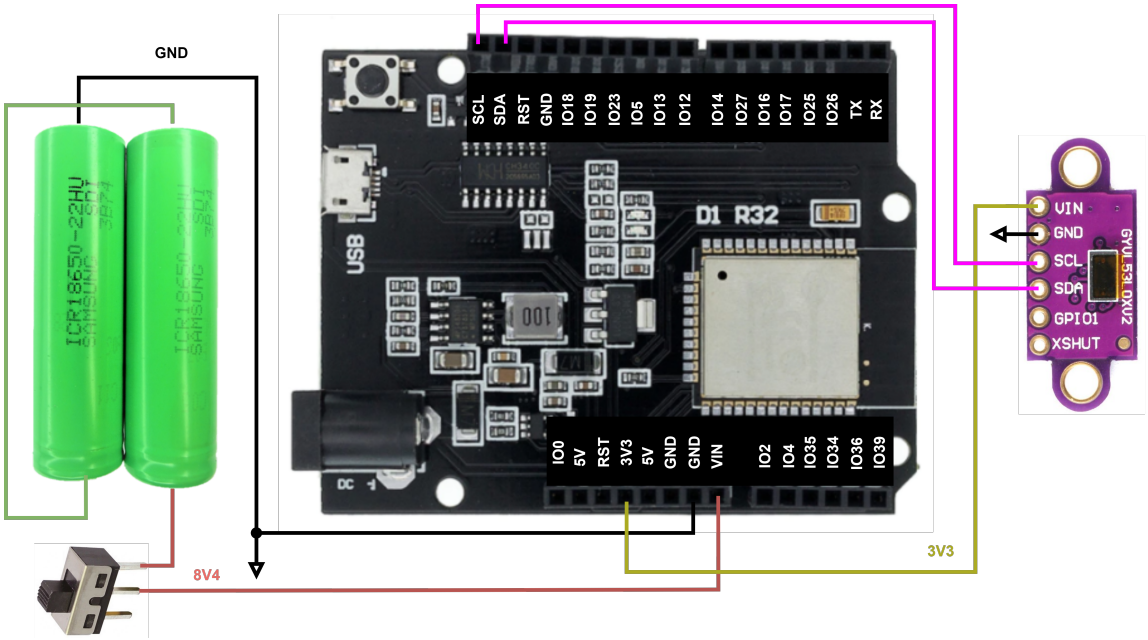


Figura 2.12: Conexión entre el sensor de distancia y la tarjeta de desarrollo

La Figura 2.12 muestra la conexión utilizada para que la ESP32 pueda interactuar con el sensor de distancia VL53L0X. El sensor se energiza desde el regulador interno de 3.3 V de la tarjeta D1 R32, y utiliza los pines SDA y SCL de la interfaz I2C, para comunicarse con el microcontrolador.

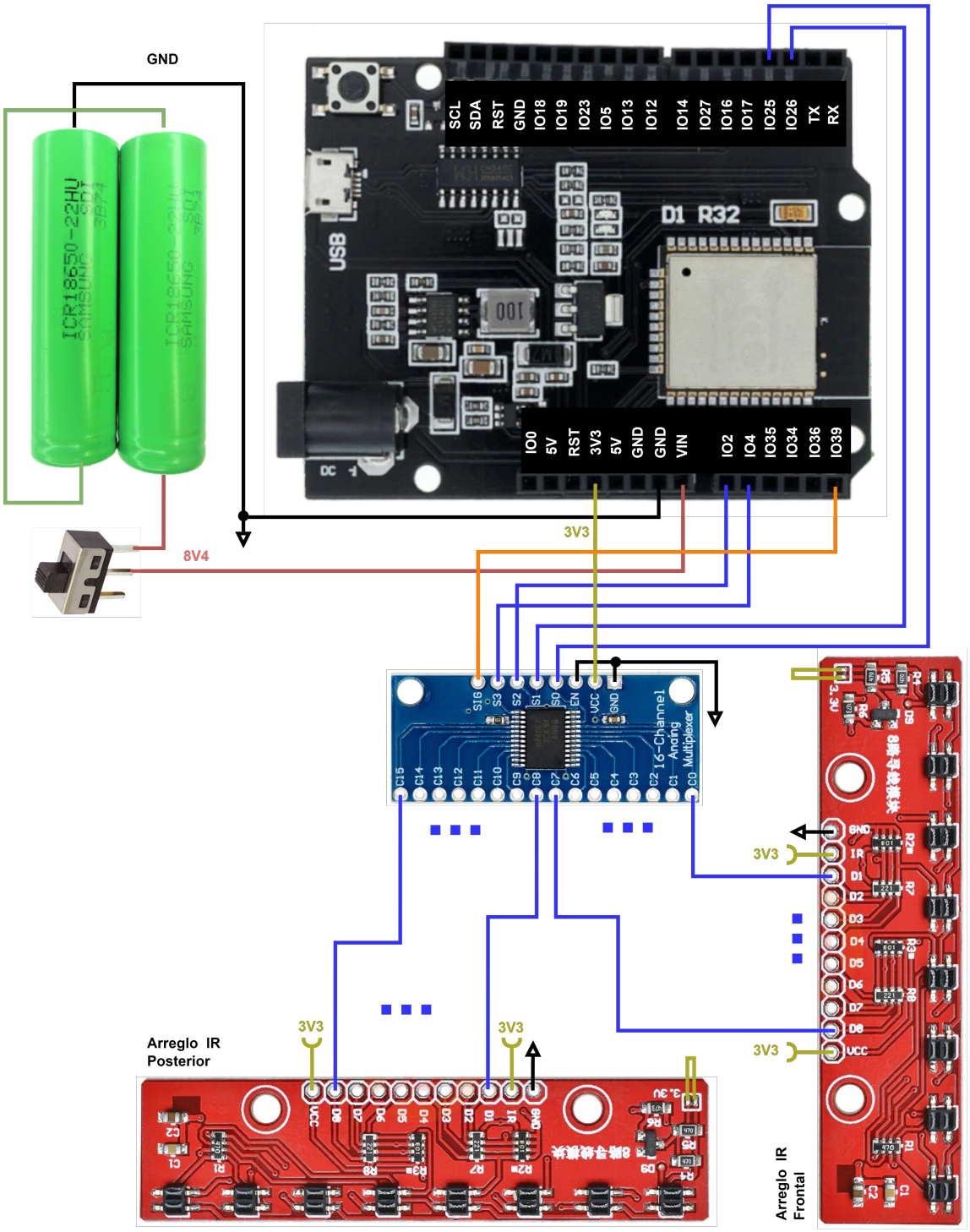


Figura 2.13: Conexión entre los arreglos de sensores IR y la tarjeta de desarrollo

La Figura 2.13 muestra la conexión utilizada para que la ESP32 pueda tomar muestras de cada uno de los 16 sensores IR dispuestos en los arreglos. La lectura se realiza a través del pin análogo IO39 de la ESP32, el cual, está conectado con la salida del multiplexor a través del pin SIG. Según la secuencia de control que reciba el multiplexor en sus entradas S0, S1, S2 y S3, selecciona una de las 16 entradas y la conecta internamente con la salida.

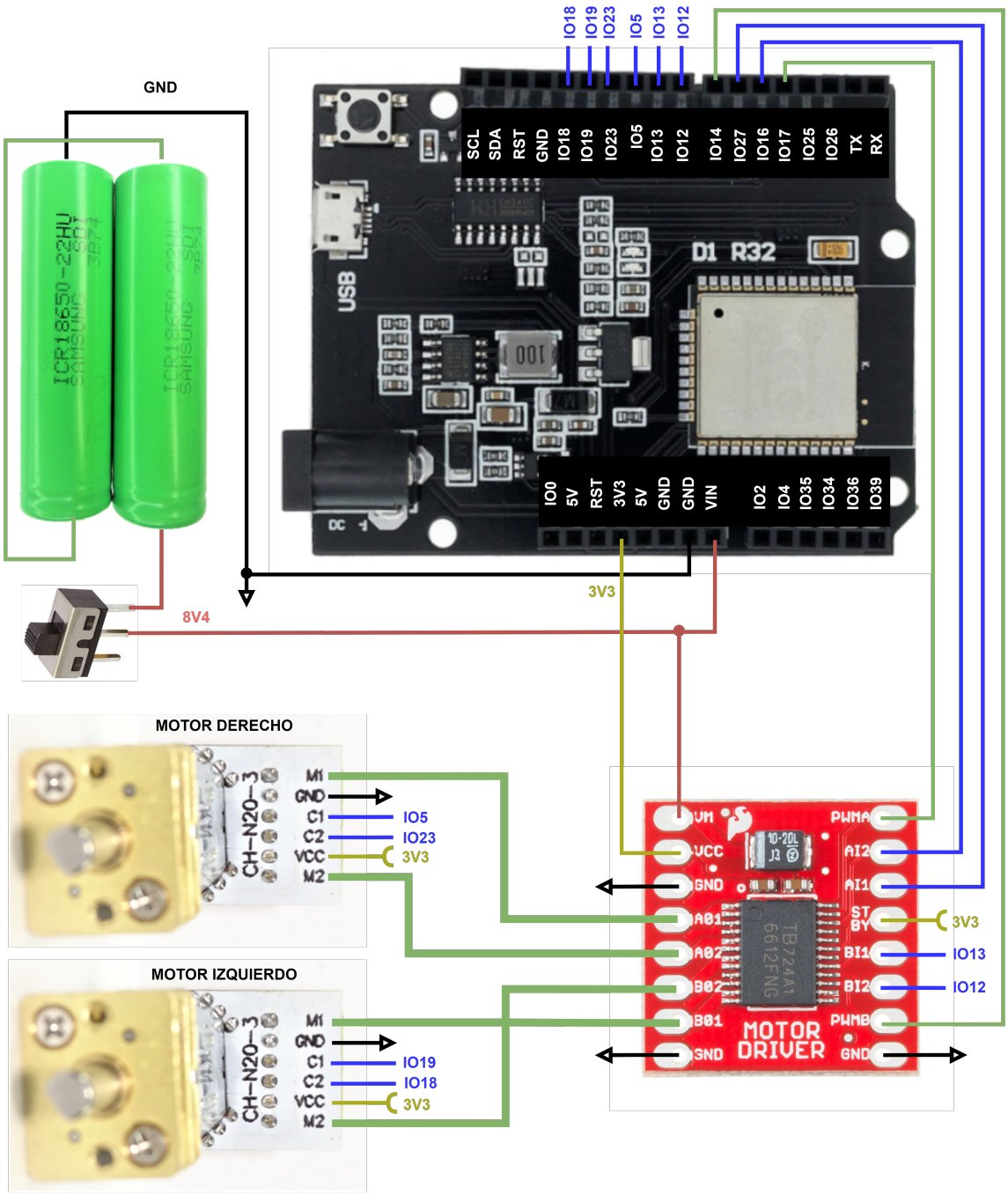


Figura 2.14: Conexión entre los motores N20 y la tarjeta de desarrollo

La Figura 2.14 muestra la conexión utilizada para que la ESP32 pueda controlar el accionamiento de los motores DC, a través del driver puente H TB6612FNG, y además se muestra la conexión de los *encoders* de cuadratura acoplados en la parte posterior de cada motor.

En la Tabla 2.1, se presenta un resumen de las conexiones de entrada y salida del hardware del DDMR, seguido, se muestra un diagrama de bloques de las conexiones del robot (ver Figura 2.15).

Tabla 2.1: Pines utilizados por sensores y actuadores en la tarjeta **D1 R32**

	Pines tarjeta de desarrollo: **D1 R32**								
Componente	**ANÁLOGO**	**DIGITAL**					**PWM**	**I2C**	
Encoder DER	-	IO5		IO23			-	-	
Encoder IZQ	-	IO19		IO18			-	-	
Canal A Puente H	-	IO16		IO27			IO17	-	
Canal B Puente H	-	IO13		IO12			IO14	-	
Sensor de distancia	-	-					-	SDA	SCL
Multiplexor	IO39	IO25	IO26	IO2	IO4		-	-	
Arreglo de sensores IR	-	-					-	-	

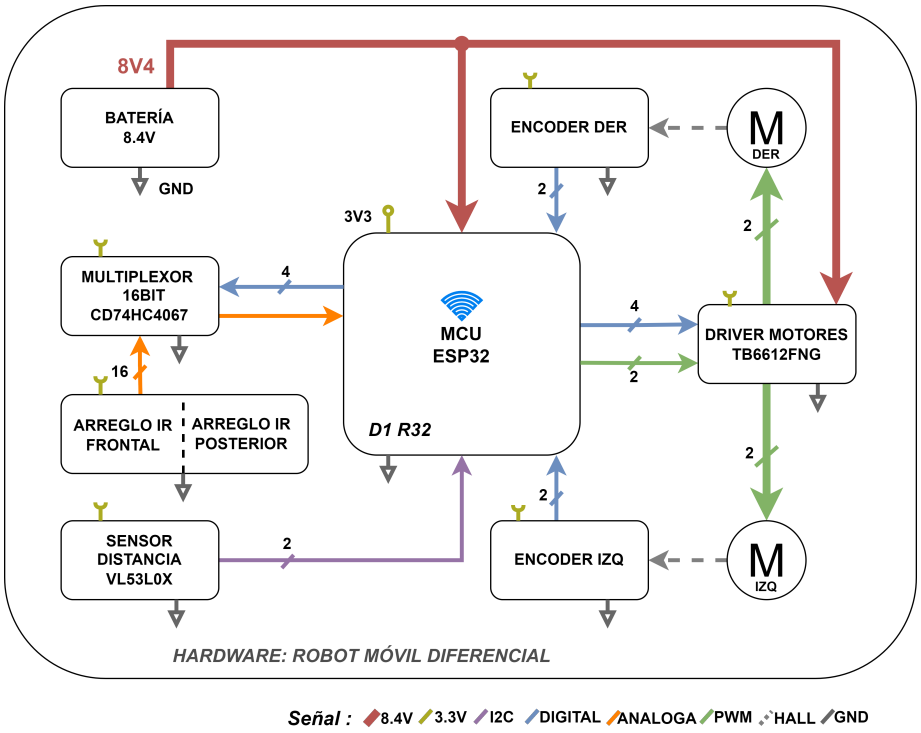
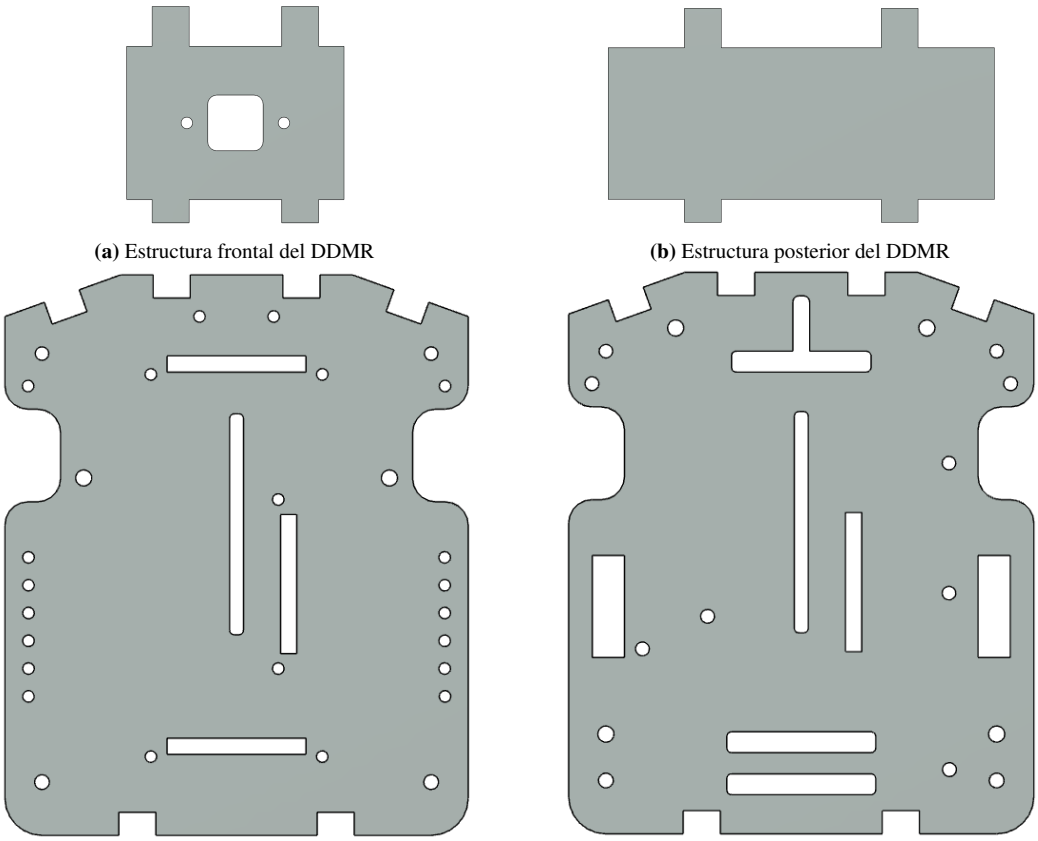


Figura 2.15: Diagrama de bloques de interconexión del hardware en el DDMR

2.2.2. Estructura

La estructura del DDMR está compuesta principalmente por cuatro piezas planas (ver Figura 2.16), las cuales, se ensamblan a través de cuatro pernos M3x50 mm. Cada una de estas piezas, está diseñada para fabricarse principalmente en acrílico mediante corte láser, o en plástico a través de impresión 3D. El hardware se monta sobre la estructura diseñada, para lo cual, se utilizan pernos M2x10 mm. A fin de facilitar las conexiones del hardware, y homogeneizar el aspecto visual del DDMR, se diseña una tarjeta de circuito impreso (PCB), con pines para montar cada uno de los componentes del robot, el cual, se presenta en la Figura 2.17. Además de las partes mencionadas anteriormente, se incorporan dos ruedas de 43 mm de diámetro, que van acopladas a cada motor, una rueda libre que funciona como pivote y mantiene la estructura del DDMR nivelada, y un soporte para las baterías. El diseño completo del robot se puede observar en las figuras 2.18 y 2.19.



(a) Estructura frontal del DDMR

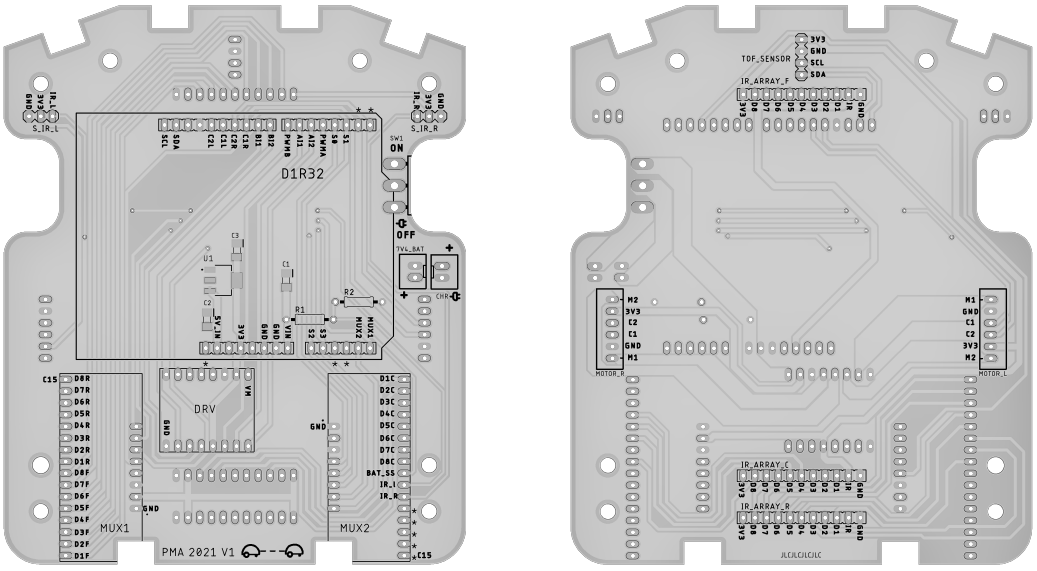
(b) Estructura posterior del DDMR

(c) Estructura inferior DDMR

(d) Estructura superior DDMR

Figura 2.16: Partes estructurales del DDMR

.

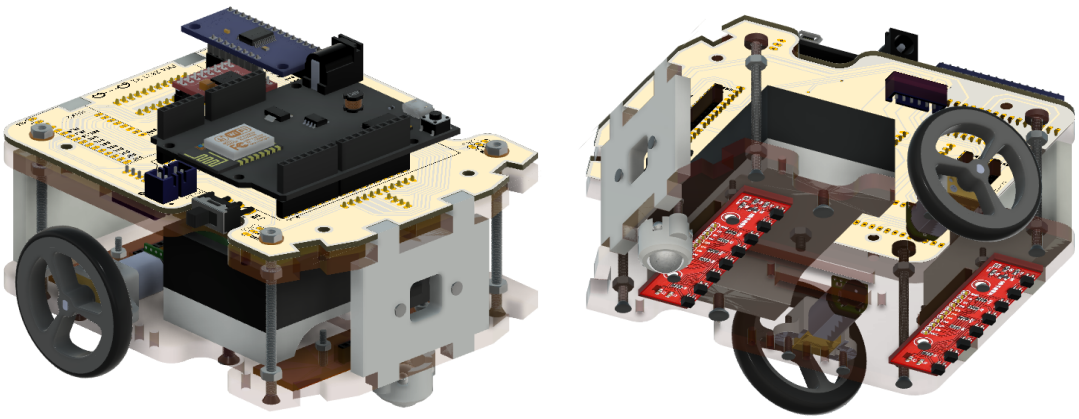


(a) Vista superior PCB

(b) Vista Inferior PCB

Figura 2.17: Diseño de la PCB para la interconexión de hardware del DDMR

.



(a) Modelo visto desde en frente **(b)** Modelo visto desde abajo

Figura 2.18: Modelo 3D del DDMR

.

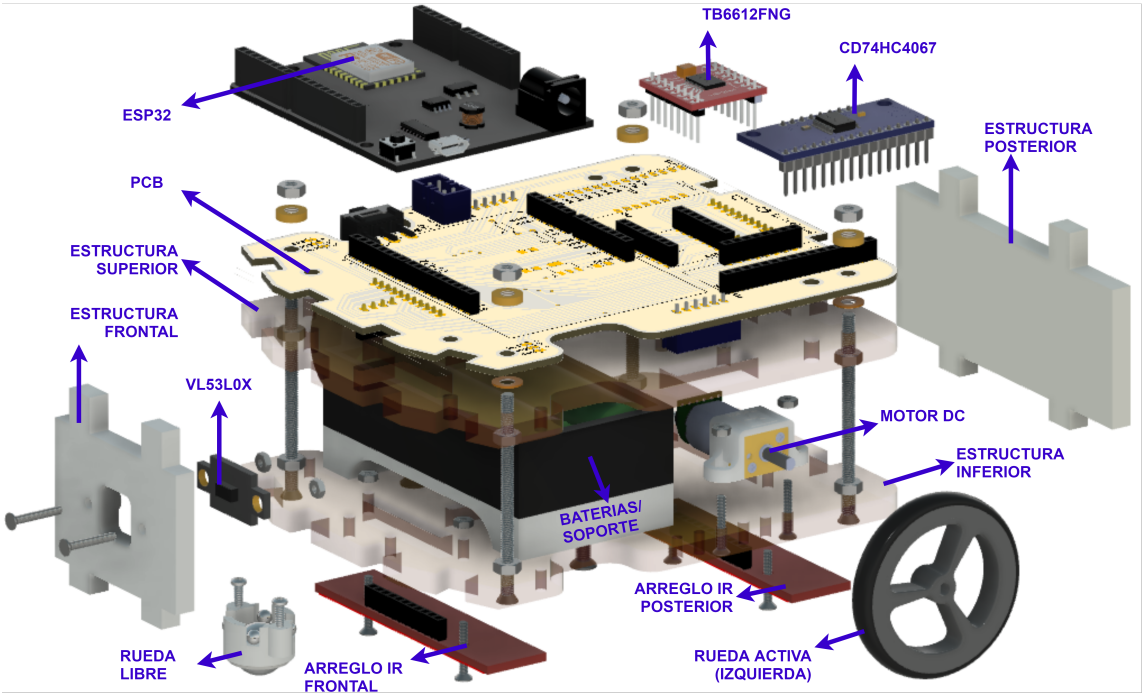


Figura 2.19: Modelo 3D del DDMR desarmado

NOTA: Si el lector desea obtener más información al respecto, puede visitar el repositorio https://github.com/ProyectoMultiagentes/openPMA.

3 | MODELO MATEMÁTICO DDMR

A partir del diseño CAD propuesto en el Capítulo anterior se desarrolla un modelo matemático del robot diferencial. En particular, se modela la relación entre la velocidad del DDMR y su orientación respecto de la trayectoria de navegación, ambas variables en función del voltaje aplicado a cada uno de los motores. Todos los parámetros y magnitudes físicas definidos a continuación poseen unidades en el sistema internacional.

3.1. Cinemática

A fin de comprender el comportamiento del DDMR frente a los cambios de velocidad en sus ruedas, se inicia el desarrollo del modelo cinemático, el cual, permite describir el desplazamiento de un cuerpo sin tomar en cuenta la naturaleza de las fuerzas o torques que lo provocan. Dado que el objetivo no es encontrar el modelo de la posición del robot en un marco global sino que su velocidad lineal $v(t)$ y orientación respecto de la trayectoria de navegación $\theta(t)$, se define un marco de referencia local definido en el centro de giro del robot denotado por O, luego se definen las velocidades lineales v_l y v_r que entrega un par de ruedas de radio r que permiten el movimiento del robot en función de sus velocidades angulares. Finalmente, se define como l la distancia entre el centro de giro del robot O y el centro de su rueda derecha.

Según lo anterior, se definen dos escenarios de operación

- Escenario 1: Ambas ruedas aportando velocidad lineal de igual magnitud y sentido.

  Debido a que la estructura del robot está compuesta por materiales rígidos, se puede anticipar que todo el cuerpo se desplazará con la misma velocidad tangencial de ambas ruedas. Este caso está representado por la Figura 3.1, donde para encontrar la relación entre la velocidad del DDMR en función de la velocidad de cada rueda se puede utilizar el principio de superposición determinando la velocidad v(t) en función de la velocidad tangencial de la rueda derecha e izquierda por separado, para finalmente sumar ambos resultados y obtener la expresión buscada. La Figura 3.2 se muestran los casos de operación resultantes.

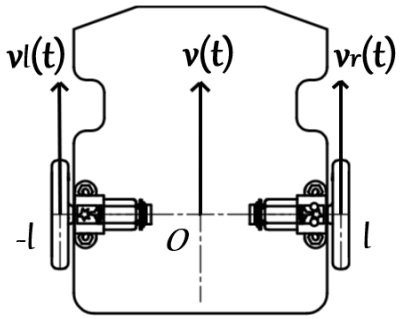
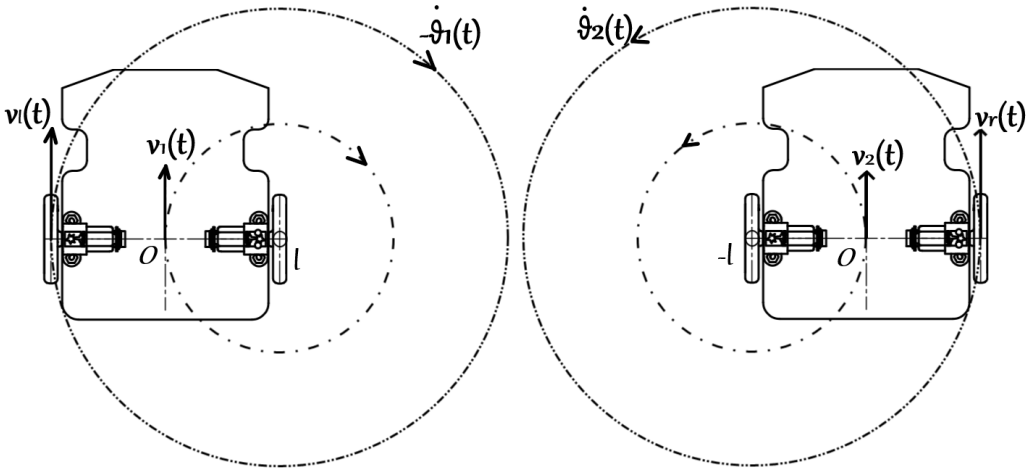


Figura 3.1: Representación del DDMR con ambas ruedas aportando la misma velocidad.



(a) Operación con el motor derecho apagado $v_r = 0$ **(b)** Operación con el motor izquierdo apagado $v_l = 0$

Figura 3.2: Desplazamiento del DDMR cuando emplea un motor a la vez, operando en ambos casos a la misma velocidad

A partir de la Figura 3.2a se observa que el desplazamiento del DDMR describe una circunferencia de radio $2l$ en sentido horario, cuando la rueda izquierda aporta una velocidad tangencial $v_l(t)$ positiva y la rueda derecha se mantiene detenida, lo anterior considerando un modelo ideal de la rueda, donde esta se apoya en un único punto sobre la superficie y no permite desplazamientos en forma ortogonal a esta. A partir de este caso se puede obtener las siguientes relaciones:

$$-\dot{\theta}_1(t) \cdot l = v_1(t) \tag{3.1}$$
$$-\dot{\theta}_1(t) \cdot 2l = v_l(t) \tag{3.2}$$

con lo cual, se define la velocidad lineal del DDMR como:

$$v_1(t) = \frac{v_l(t)}{2} \tag{3.3}$$

De la Figura 3.2b se obtienen conclusiones similares, donde se observa que el desplazamiento del DDMR describe una circunferencia de radio $2l$ en sentido antihorario, ya que en este caso la rueda izquierda se mantiene detenida y la rueda derecha entrega una velocidad tangencial positiva, de este caso se obtienen las siguientes relaciones:

$$\dot{\theta}_2(t) \cdot l = v_2(t) \tag{3.4}$$
$$\dot{\theta}_2(t) \cdot 2l = v_r(t) \tag{3.5}$$

donde la velocidad lineal del DDMR en función de la velocidad de la rueda derecha corresponde a:

$$v_2(t) = \frac{v_r(t)}{2} \tag{3.6}$$

De esta forma se obtiene la velocidad lineal total del carro producto de la velocidad de giro de cada rueda del DDMR según:

$$\begin{aligned} v(t) &= v_1(t) + v_2(t) \\ &= \frac{v_l(t) + v_r(t)}{2} \end{aligned} \tag{3.7}$$

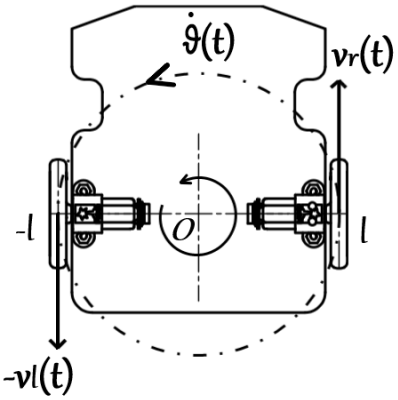
De esta expresión se concluye que la velocidad del DDMR corresponde al promedio entre las velocidades tangenciales aportadas por ambas ruedas del robot. De igual forma se puede expresar la velocidad lineal del DDMR en función de las velocidades angulares de sus ruedas como:

$$v(t) = \frac{r}{2}(\dot{\phi}_r(t) + \dot{\phi}_l(t)) \tag{3.8}$$

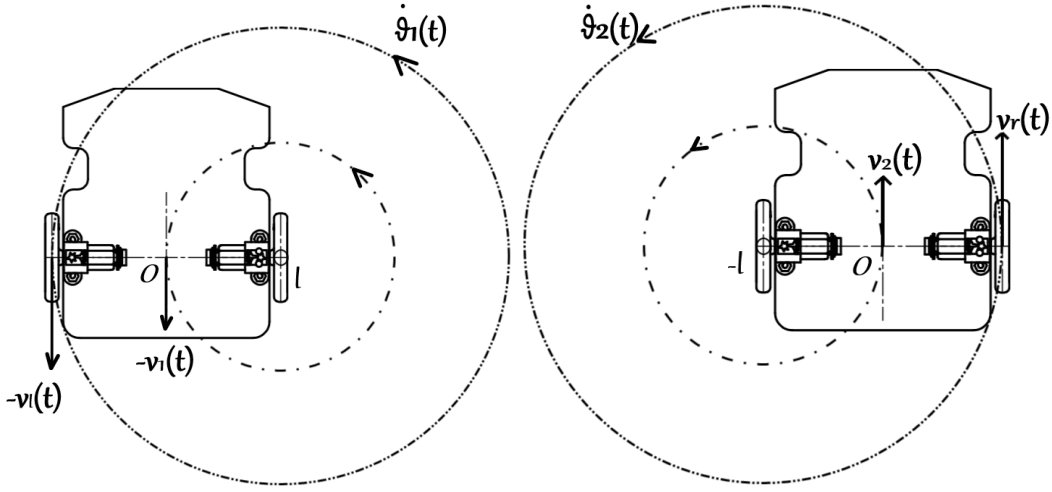
Donde $\dot{\phi}_r(t)$, $\dot{\phi}_l(t)$ corresponde a las velocidades angulares de la rueda derecha e izquierda respectivamente y r el radio de ambas ruedas.

- Escenario 2: Ambas ruedas girando con velocidad lineal de igual magnitud y diferente sentido.

  En este escenario se puede anticipar que el robot comenzará a girar con velocidad angular constante en torno al marco de referencia definido en su centro de giro O, pero es posible aplicar el mismo método de superposición para encontrar la relación entre la velocidad angular del DDMR en función de la velocidad de cada rueda. La Figura 3.3 muestra el escenario 2 y la Figura 3.3b y 3.3c presenta los casos de operación resultantes.



(a) Operación ambos motores escenario 2



(b) Operación con el motor derecho apagado $v_r = 0$ **(c)** Operación con el motor izquierdo apagado $v_l = 0$

Figura 3.3: Desplazamiento del DDMR operando con ambos motores a la misma velocidad, pero diferente sentido

De las figuras obtenidas del escenario 2 de operación del DDMR se obtienen las siguientes relaciones:

$$\dot{\theta}_1(t) = -\frac{v_l(t)}{2l} \tag{3.9}$$

$$\dot{\theta}_2(t) = \frac{v_r(t)}{2l} \tag{3.10}$$

Aplicando el principio de superposición se obtiene la velocidad angular total del robot según:

$$\begin{aligned}\dot{\theta}(t) &= \dot{\theta}_1(t) + \dot{\theta}_2(t) \\ &= \frac{-v_l(t) + v_r(t)}{2l}\end{aligned} \tag{3.11}$$

De esta expresión se concluye que la velocidad angular del DDMR vista desde el origen del sistema de referencia local es proporcional a la diferencia entre las velocidades tangenciales aportadas por ambas ruedas del robot. De igual forma se puede expresar la velocidad angular del DDMR en función de las velocidades angulares de sus ruedas como:

$$\dot{\theta}(t) = \frac{r}{2l}(\dot{\phi}_r(t) - \dot{\phi}_l(t)) \tag{3.12}$$

Las expresiones obtenidas en (3.8) y (3.12) nos permiten encontrar una relación para el torque que produce el movimiento rotacional y la fuerza que produce el movimiento lineal en el DDMR, en función de los torques de entrada de los motores.

$$F(t) = \frac{2}{r}(\tau_r(t) + \tau_l(t)) \tag{3.13}$$

$$\tau(t) = \frac{2l}{r}(\tau_r(t) - \tau_l(t)) \tag{3.14}$$

La relación que existe entre la velocidad de giro de las ruedas, el torque que pueden entregar y las velocidades y torques del motor está determinado por un mecanismo reductor de factor de reducción $N \in [0, 1]$ y rendimiento $\eta \in [0, 1]$ (que representa el porcentaje de potencia que puede transferir este mecanismo). Entonces las velocidades angulares $\dot{\phi}_r(t)$, $\dot{\phi}_l(t)$ y los torques de carga $\tau_r(t)$, $\tau_l(t)$ a la salida del reductor, están relacionados con $\dot{\phi}_{mr}(t)$, $\dot{\phi}_{ml}(t)$ y $\tau_{mr}(t)$,$\tau_{ml}(t)$ en el eje de cada motor (o entrada de cada reductor) según las siguientes expresiones:

$$\dot{\phi}_r(t) = N_r\dot{\phi}_{mr}(t) \tag{3.15}$$

$$\tau_r(t) = \frac{\eta_r}{N_r}\tau_{mr}(t) \tag{3.16}$$

$$\dot{\phi}_l(t) = N_l\dot{\phi}_{ml}(t) \tag{3.17}$$

$$\tau_l(t) = \frac{\eta_l}{N_l}\tau_{ml}(t) \tag{3.18}$$

Donde N_r, N_l es el módulo del factor de transformación del mecanismo reductor del motor derecho e izquierdo respectivamente y η_r, η_l sus respectivas eficiencias.

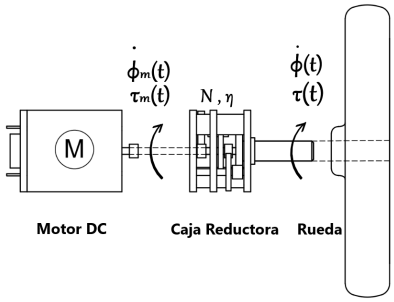


Figura 3.4: Interconexión mecánica motor DC - Rueda

3.2. Dinámica

Para comprender y obtener un modelo más completo del DDMR es que se analiza la dinámica del movimiento del robot, es decir, las relaciones entre el movimiento y las fuerzas que lo producen. El modelo comienza a partir de la mecánica Lagrangiana que es una reformulación de la mecánica clásica, puede ser aplicada a cualquier sistema de referencia, sean estos inerciales o no inerciales. Estos últimos sistemas resultan en un conjunto de ecuaciones más complejas desde el punto de vista de la mecánica clásica, puesto que se hace necesario introducir la suma vectorial de todas las fuerzas que actúan sobre un cuerpo, apareciendo fuerzas inerciales (tales como la fuerza de Coriolis o la fuerza centrífuga) para poder explicar los movimientos con respecto a dichos sistemas de referencia. El enfoque de Lagrange permite modelar el movimiento (variación de cantidad de movimiento) de sistemas mecánicos de cuerpos rígidos a partir de la energía total del sistema (magnitud escalar) definida por la variable auxiliar $\mathcal{L}$ denominada "Lagrangiano", tal como se muestra a continuación:

$$\mathcal{L} = T(q, \dot{q}) - V(q) \tag{3.19}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}_i}\right) - \frac{\partial \mathcal{L}}{\partial q_i} = \tau_i - \frac{\partial \mathcal{F}}{\partial \dot{q}_i} \tag{3.20}$$

Donde $T(q, \dot{q})$ y $V(q)$ corresponde a la energía cinética y la energía potencial total del cuerpo, la ecuación (3.20) corresponde a la ecuación de Euler-Lagrange (E-L) la que indica que la variación de la cantidad de movimiento generalizado es igual a los torques y fuerzas generalizadas τ_i que producen movimiento en la i-ésima componente del vector q de coordenadas generalizadas, más los torques y fuerzas disipativos $-\frac{\partial \mathcal{F}}{\partial \dot{q}_i}$, que pueden ser deducidas a partir de la función disipativa de Rayleigh $\mathcal{F}$ (utilizada habitualmente en el modelado de sistemas en general). Además, es posible extender esta ecuación a sistemas eléctricos incorporando la co-energía magnética y la energía del campo eléctrico total del sistema en el Lagrangiano.

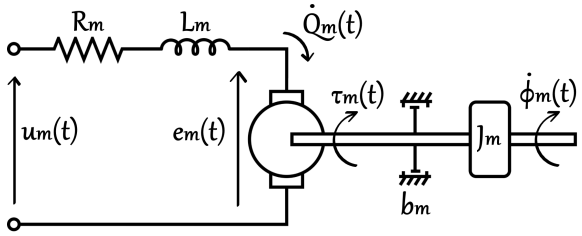


Figura 3.5: Representación esquemática del motor DC

En primer lugar, se define el vector de coordenadas generalizadas:

$$q = \left[\phi_r(t), \phi_l(t), Q_{mr}(t), Q_{ml}(t)\right]^T \tag{3.21}$$

donde $\phi_r(t)$, $\phi_l(t)$ corresponden a las posiciones angulares de las ruedas derecha e izquierda del DDMR respectivamente, y $Q_{mr}(t)$, $Q_{ml}(t)$ representan la carga eléctrica que circula a través del circuito eléctrico del motor derecho e izquierdo del robot. Se continúa el modelo dinámico con la obtención de la energía del DDMR en función del vector de coordenadas generalizadas q y velocidades generalizadas $\dot{q}$.

$$T(t) = \frac{1}{2}mv(t)^2 + \frac{1}{2}j\dot{\theta}(t)^2 + \frac{1}{2}j_{mr}\dot{\phi}_{mr}(t)^2 + \frac{1}{2}j_{ml}\dot{\phi}_{ml}(t)^2 \tag{3.22}$$

La ecuación (3.22) representa la energía cinética total del DDMR, esto es, la energía cinética lineal y rotacional del robot, junto con la energía rotacional de los motores derecho e izquierdo respectivamente. Utilizando las expresiones obtenidas en (3.8), (3.12), (3.15) y (3.17) se puede expresar $T(t)$ en función del vector $\dot{q}$.

$$T(\dot{q}) = \frac{r^2 m}{8}(\dot{\phi}_r(t) + \dot{\phi}_l(t))^2 + \frac{r^2 j}{8l^2}(\dot{\phi}_r(t) - \dot{\phi}_l(t))^2 + \frac{j_{mr}}{2N_r}\dot{\phi}_r(t)^2 + \frac{j_{ml}}{2N_l}\dot{\phi}_l(t)^2 \tag{3.23}$$

La ecuación (3.23) representa la energía cinética total en función de las coordenadas generalizadas $\dot{q}$. Ahora se define la co-energía magnética total del robot como:

$$W'_m(\dot{q}) = \frac{1}{2}L_{mr}\dot{Q}_{mr}(t)^2 + \frac{1}{2}L_{ml}\dot{Q}_{ml}(t)^2 \tag{3.24}$$

que corresponde a la suma de las energías almacenadas en el campo magnético del los inductores L_{mr} y L_{ml} presente en los motores derecho e izquierdo respectivamente.

Luego la energía potencial de nuestro robot es cero, puesto que nuestro robot navega en todo momento sobre una superficie plana.

$$V(q) = 0 \tag{3.25}$$

Finalmente, nos queda expresar la energía de campo eléctrico, la cual es cero, puesto que en el circuito eléctrico del motor no existen elementos capacitivos que almacenen energía en el potencial eléctrico.

$$W_e(q) = 0 \tag{3.26}$$

Luego el Lagrangiano extendido a nuestro sistema electromecánico queda determinado por:

$$\begin{aligned}\mathcal{L} &= (T(\dot{q}) + W'_m(\dot{q})) - (V(q) + W_e(q)) \\ &= T(\dot{q}) + W'_m(\dot{q})\end{aligned} \tag{3.27}$$

A continuación se puede obtener el modelo dinámico a través de la ecuación de E-L, donde se calcula la derivada parcial del Lagrangiano con respecto del vector de velocidades generalizadas $\dot{q}$, como se expresa en la ecuación (3.20).

$$\frac{\partial \mathcal{L}}{\partial \dot{q}} = \left[\frac{\partial \mathcal{L}}{\partial \dot{\phi}_r(t)}, \frac{\partial \mathcal{L}}{\partial \dot{\phi}_l(t)}, \frac{\partial \mathcal{L}}{\partial \dot{Q}_{mr}(t)}, \frac{\partial \mathcal{L}}{\partial \dot{Q}_{ml}(t)}\right]^T \tag{3.28}$$

Se calculan los valores para cada derivada parcial, obteniendo:

$$\frac{\partial \mathcal{L}}{\partial \dot{\phi}_r(t)} = \frac{r^2 m}{4}(\dot{\phi}_r(t) + \dot{\phi}_l(t)) + \frac{r^2 j}{4l^2}(\dot{\phi}_r(t) - \dot{\phi}_l(t)) + \frac{j_{mr}}{N_r}\dot{\phi}_r(t) \tag{3.29}$$

$$\frac{\partial \mathcal{L}}{\partial \dot{\phi}_l(t)} = \frac{r^2 m}{4}(\dot{\phi}_r(t) + \dot{\phi}_l(t)) - \frac{r^2 j}{4l^2}(\dot{\phi}_r(t) - \dot{\phi}_l(t)) + \frac{j_{ml}}{N_l}\dot{\phi}_l(t) \tag{3.30}$$

$$\frac{\partial \mathcal{L}}{\partial \dot{Q}_{mr}(t)} = L_{mr}\dot{Q}_{mr}(t) \tag{3.31}$$

$$\frac{\partial \mathcal{L}}{\partial \dot{Q}_{ml}(t)} = L_{ml}\dot{Q}_{ml}(t) \tag{3.32}$$

Con las derivadas parciales calculadas, se obtienen las derivadas temporales de cada una.

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) = \left[\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\phi}_r(t)}\right), \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\phi}_l(t)}\right), \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{Q}_{mr}(t)}\right), \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{Q}_{ml}(t)}\right)\right]^T \tag{3.33}$$

Cuyas componentes corresponden a:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\phi}_r(t)}\right) = \frac{r^2 m}{4}(\ddot{\phi}_r(t) + \ddot{\phi}_l(t)) + \frac{r^2 j}{4l^2}(\ddot{\phi}_r(t) - \ddot{\phi}_l(t)) + \frac{j_{mr}}{N_r}\ddot{\phi}_r(t) \tag{3.34}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\phi}_l(t)}\right) = \frac{r^2 m}{4}(\ddot{\phi}_r(t) + \ddot{\phi}_l(t)) - \frac{r^2 j}{4l^2}(\ddot{\phi}_r(t) - \ddot{\phi}_l(t)) + \frac{j_{ml}}{N_l}\ddot{\phi}_l(t) \tag{3.35}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{Q}_{mr}(t)}\right) = L_{mr}\ddot{Q}_{mr}(t) \tag{3.36}$$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{Q}_{ml}(t)}\right) = L_{ml}\ddot{Q}_{ml}(t) \tag{3.37}$$

con lo cual, es posible expresar de forma matricial en función del vector de aceleraciones generalizadas $\ddot{q}$

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) = \begin{bmatrix} \frac{r^2 m}{4} + \frac{r^2 j}{4l^2} + \frac{j_{mr}}{N_r} & \frac{r^2 m}{4} - \frac{r^2 j}{4l^2} & 0 & 0 \\ \frac{r^2 m}{4} - \frac{r^2 j}{4l^2} & \frac{r^2 m}{4} + \frac{r^2 j}{4l^2} + \frac{j_{ml}}{N_l} & 0 & 0 \\ 0 & 0 & L_{mr} & 0 \\ 0 & 0 & 0 & L_{ml} \end{bmatrix} \begin{bmatrix} \ddot{\phi}_r(t) \\ \ddot{\phi}_l(t) \\ \ddot{Q}_{mr}(t) \\ \ddot{Q}_{ml}(t) \end{bmatrix} \tag{3.38}$$

Por otro lado, la derivada del Lagrangiano respecto a las coordenadas generalizadas q es cero, ya que, el Lagrangiano en nuestro modelo solo depende de las velocidades generalizadas, es decir $\dot{q}$.

$$\frac{\partial \mathcal{L}}{\partial q} = 0 \tag{3.39}$$

Luego, en el lado derecho de la ecuación de E-L se tiene en primer lugar los torques generalizados τ_i que producen movimiento en cada componente del vector de coordenadas generalizadas q.

$$\tau_i = \left[\tau_r(t),\ \tau_l(t),\ u_{mr}(t) - e_{mr}(t),\ u_{ml}(t) - e_{ml}(t)\right]^T \tag{3.40}$$

$\tau_r(t)$, $\tau_l(t)$ corresponden a los torques en las ruedas derecha e izquierda respectivamente, los cuales producen un cambio en la posición angular de estas. Por otro lado u_{mr}, u_{ml} los voltajes aplicados a los motores derecho e izquierdo respectivamente, y e_{mr}, e_{ml} la fuerza contraelectromotriz del motor derecho e izquierdo, que aparece producto del acoplamiento electromecánico y que induce en el circuito eléctrico del motor una caída de tensión. Por otra parte, es posible usar las expresiones obtenidas en (3.16) y (3.18) para expresar el torque de las ruedas en función del torque producido por los motores. Además, se puede relacionar estos torques con las corrientes de los motores y relacionar la fuerza contraelectromotriz con la velocidad de los motores y utilizando las expresiones (3.8) y (3.12) expresarlo en función de la velocidad de las ruedas, esto es:

$$\tau_{mr}(t) = kt_r \dot{Q}_{mr}(t) \tag{3.41}$$
$$\tau_{ml}(t) = kt_l \dot{Q}_{ml}(t) \tag{3.42}$$
$$e_{mr}(t) = kv_r \dot{\phi}_{mr}(t) \tag{3.43}$$
$$e_{ml}(t) = kv_l \dot{\phi}_{ml}(t) \tag{3.44}$$
$$\tag{3.45}$$

Luego

$$\tau_r(t) = \frac{\eta_r kt_r}{N_r} \dot{Q}_{mr}(t) \tag{3.46}$$
$$\tau_l(t) = \frac{\eta_l kt_l}{N_l} \dot{Q}_{ml}(t) \tag{3.47}$$
$$e_{mr}(t) = \frac{kv_r}{N_r} \dot{\phi}_r(t) \tag{3.48}$$
$$e_{ml}(t) = \frac{kv_l}{N_l} \dot{\phi}_l(t) \tag{3.49}$$
$$\tag{3.50}$$

De esta manera se reescribe el vector de torques generalizados en función del voltaje de entrada y las velocidades generalizadas $\dot{q}$

$$\begin{aligned} \tau_i &= \left[\tfrac{\eta_r kt_r}{N_r} \dot{Q}_{mr}(t), \tfrac{\eta_l kt_l}{N_l} \dot{Q}_{ml}(t), u_{mr}(t) - \tfrac{kv_r}{N_r} \dot{\phi}_r(t), u_{ml}(t) - \tfrac{kv_l}{N_l} \dot{\phi}_l(t)\right]^T \\ &= \begin{bmatrix} 0 & 0 & \frac{\eta_r kt_r}{N_r} & 0 \\ 0 & 0 & 0 & \frac{\eta_l kt_l}{N_l} \\ -\frac{kv_r}{N_r} & 0 & 0 & 0 \\ 0 & -\frac{kv_l}{N_l} & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\phi}_r(t) \\ \dot{\phi}_l(t) \\ \dot{Q}_{mr}(t) \\ \dot{Q}_{ml}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_{mr}(t) \\ u_{ml}(t) \end{bmatrix} \end{aligned} \tag{3.51}$$

Finalmente nos queda determinar el vector de fuerzas disipativas $-\frac{\partial \mathcal{F}}{\partial \dot{q}_i}$, donde solo se consideran elementos disipativos lineales presentes en nuestro robot, tales como, el torque de fricción viscosa y la resistencia eléctrica, en tal caso la función disipativa de Rayleigh $\mathcal{F}$ queda definida como:

$$\mathcal{F} = \frac{1}{2} b_r \dot{\phi}_r(t)^2 + \frac{1}{2} b_l \dot{\phi}_l(t)^2 + \frac{1}{2} R_{mr} \dot{Q}_{mr}(t)^2 + \frac{1}{2} R_{ml} \dot{Q}_{ml}(t)^2 \tag{3.52}$$

Para este caso $2\mathcal{F}$ equivale a la tasa de disipación de energía o potencia total disipada, con b_r, b_l los coeficientes de fricción viscosa de las ruedas derecha e izquierda respectivamente (estos coeficientes incluyen el roce viscoso del los ejes de cada motor, de las ruedas activas con el suelo y de la rueda pasiva con el suelo), y R_{mr}, R_{ml} las resistencias eléctricas de los motores derecho e izquierdo respectivamente. De esta forma se calculan las fuerzas disipativas según:

$$\begin{aligned} -\frac{\partial \mathcal{F}}{\partial \dot{q}} &= -\left[\frac{\partial \mathcal{F}}{\partial \dot{\phi}_r(t)}, \frac{\partial \mathcal{F}}{\partial \dot{\phi}_l(t)}, \frac{\partial \mathcal{F}}{\partial \dot{Q}_{mr}(t)}, \frac{\partial \mathcal{F}}{\partial \dot{Q}_{ml}(t)} \right]^T \\ &= -\left[b_r \dot{\phi}_r(t), b_l \dot{\phi}_l(t), R_{mr} \dot{Q}_{mr}, R_{ml} \dot{Q}_{ml} \right]^T \\ &= -\begin{bmatrix} b_r & 0 & 0 & 0 \\ 0 & b_l & 0 & 0 \\ 0 & 0 & R_{mr} & 0 \\ 0 & 0 & 0 & R_{ml} \end{bmatrix} \begin{bmatrix} \dot{\phi}_r(t) \\ \dot{\phi}_l(t) \\ \dot{Q}_{mr}(t) \\ \dot{Q}_{ml}(t) \end{bmatrix} \end{aligned} \tag{3.53}$$

Ahora se puede escribir el modelo E-L a partir de las expresiones obtenidas en (3.38), (3.39), (3.51) y (3.53)

$$M\ddot{q} = V_m \dot{q} + S u$$

$$\begin{bmatrix} \frac{r^2 m}{4} + \frac{r^2 j}{4l^2} + \frac{j_{mr}}{N_r} & \frac{r^2 m}{4} - \frac{r^2 j}{4l^2} & 0 & 0 \\ \frac{r^2 m}{4} - \frac{r^2 j}{4l^2} & \frac{r^2 m}{4} + \frac{r^2 j}{4l^2} + \frac{j_{ml}}{N_l} & 0 & 0 \\ 0 & 0 & L_{mr} & 0 \\ 0 & 0 & 0 & L_{ml} \end{bmatrix} \begin{bmatrix} \ddot{\phi}_r(t) \\ \ddot{\phi}_l(t) \\ \ddot{Q}_{mr}(t) \\ \ddot{Q}_{ml}(t) \end{bmatrix} = \begin{bmatrix} -b_r & 0 & \frac{\eta_r k t_r}{N_r} & 0 \\ 0 & -b_l & 0 & \frac{\eta_l k t_l}{N_l} \\ -\frac{k v_r}{N_r} & 0 & -R_{mr} & 0 \\ 0 & -\frac{k v_l}{N_l} & 0 & -R_{ml} \end{bmatrix} \begin{bmatrix} \dot{\phi}_r(t) \\ \dot{\phi}_l(t) \\ \dot{Q}_{mr}(t) \\ \dot{Q}_{ml}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_{mr}(t) \\ u_{ml}(t) \end{bmatrix} \tag{3.54}$$

De esta manera, la dinámica del DDMR queda expresada en función de los voltajes de entrada de los motores del robot, la cual, se puede ordenar convenientemente de la forma:

$$\begin{aligned} \ddot{q} &= M^{-1} V_m \dot{q} + M^{-1} S u \\ \dot{x}(t) &= A x(t) + B u(t) \end{aligned} \tag{3.55}$$

Donde $x(t) = \dot{q}$ nuestro vector de estados, $A = M^{-1} V_m$ la matriz de estados, $B = M^{-1} S$ la matriz de entrada y $u(t)$ el vector de entrada del sistema correspondiente al voltaje de los motores del DDMR. Por otra parte, la salida de interés de este sistema corresponde a la velocidad lineal $v(t)$ y rotacional del robot $\dot{\theta}(t)$ en unidades de cm/s y rad/s respectivamente, por lo cual, es posible escribir estas salidas como una combinación lineal de los estados $x(t)$ utilizando las expresiones (3.8) y (3.12).

$$\begin{aligned} y(t) &= C x(t) + D u(t) \\ \begin{bmatrix} v(t) \\ \dot{\theta}(t) \end{bmatrix} &= \begin{bmatrix} 100\frac{r}{2} & 100\frac{r}{2} & 0 & 0 \\ \frac{r}{2l} & -\frac{r}{2l} & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\phi}_r(t) \\ \dot{\phi}_l(t) \\ \dot{Q}_{mr}(t) \\ \dot{Q}_{ml}(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{mr}(t) \\ u_{ml}(t) \end{bmatrix} \end{aligned} \tag{3.56}$$

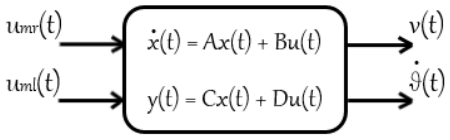


Figura 3.6: Diagrama del modelo de velocidades del DDMR en espacio de estados

Así, nuestro modelo del DDMR en espacio de estado queda representado por las matrices A, B, C y D, con entrada el voltaje aplicado a cada uno de los motores y salida las velocidades lineal y rotacional del robot, tal como se indica en las expresiones (3.55) y (3.56)

Ahora bien, es necesario incorporar al modelo los componentes que interactúan con el robot en la plataforma experimental, esto es, la trayectoria de navegación y un robot guía o líder, lo anterior se representa en las figuras 3.7 y 3.8.

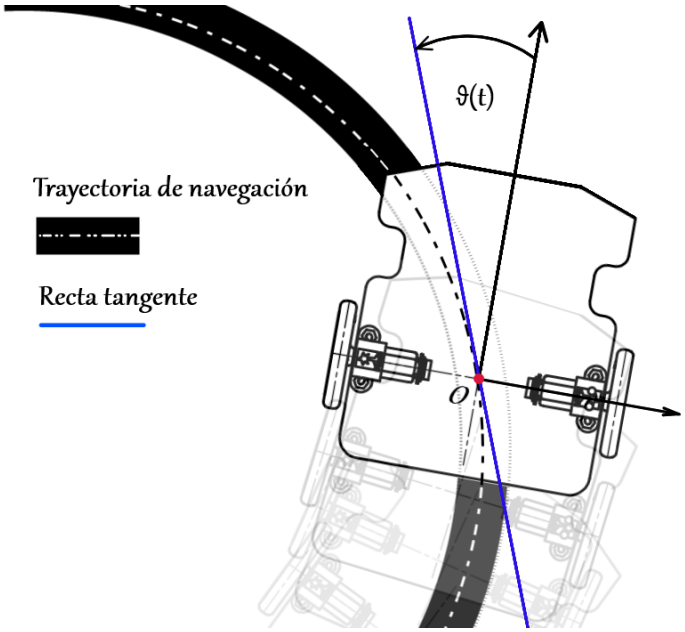


Figura 3.7: Ángulo entre el DDMR y la trayectoria de navegación

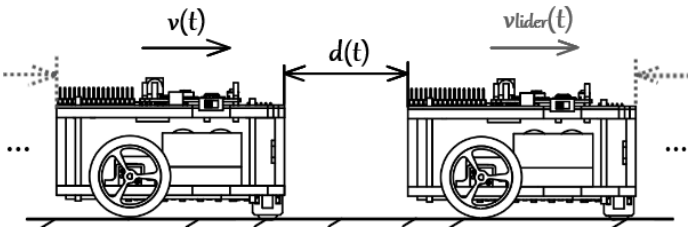


Figura 3.8: Distancia del DDMR respecto a un robot líder

El escenario anterior se puede modelar según:

$$\begin{aligned} d(t) &= -\int^{t} v(t)dt \\ &= -\frac{r}{2}(\phi_r(t) + \phi_l(t)) \end{aligned} \tag{3.57}$$

$$\begin{aligned} \theta(t) &= \int^{t} \dot{\theta}(t)dt \\ &= \frac{r}{2l}(\phi_r(t) - \phi_l(t)) \end{aligned} \tag{3.58}$$

Definiendo $\bar{x}(t)$ como las posiciones angulares de cada rueda, se puede representar la navegación del robot sobre la plataforma experimental en función de sus velocidades $v(t)$ y $\dot{\theta}(t)$ en el espacio de estados,

según:

$$\begin{aligned} \dot{\tilde{x}}(t) &= \tilde{A}\tilde{x}(t) + \tilde{B}\tilde{u}(t) \\ \begin{bmatrix} \dot{\phi}_r(t) \\ \dot{\phi}_l(t) \end{bmatrix} &= \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \phi_r(t) \\ \phi_l(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{100r} & \frac{l}{r} \\ \frac{1}{100r} & \frac{-l}{r} \end{bmatrix} \begin{bmatrix} v(t) \\ \dot{\theta}(t) \end{bmatrix} \end{aligned} \tag{3.59}$$

$$\begin{aligned} \tilde{y}(t) &= \tilde{C}\tilde{x}(t) + \tilde{D}\tilde{u}(t) \\ \begin{bmatrix} d(t) \\ \theta(t) \end{bmatrix} &= \begin{bmatrix} -100\frac{r}{2} & -100\frac{r}{2} \\ \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix} \begin{bmatrix} \phi_r(t) \\ \phi_l(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v(t) \\ \theta(t) \end{bmatrix} \end{aligned} \tag{3.60}$$

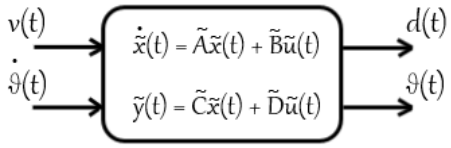


Figura 3.9: Diagrama del modelo de navegación del DDMR en espacio de estados

Por último, se puede obtener un modelo que integre la dinámica del movimiento del robot junto con la interacción con los componentes de la plataforma experimental, interconectando la salida del modelo 1 con la entrada del modelo 2 (conexión serie), tal como muestra la Figura 3.10.

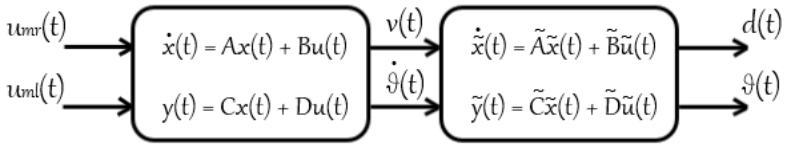


Figura 3.10: Diagrama de conexión serie de los modelos 3.6 y 3.9

En tal condición, el modelo completo del sistema queda determinado por:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{\tilde{x}}(t) \end{bmatrix} = \begin{bmatrix} A & 0 \\ \tilde{B}C & \tilde{A} \end{bmatrix} \begin{bmatrix} x(t) \\ \tilde{x}(t) \end{bmatrix} + \begin{bmatrix} B \\ \tilde{B}D \end{bmatrix} u(t) \tag{3.61}$$

$$\tilde{y}(t) = \begin{bmatrix} \tilde{D}C & \tilde{C} \end{bmatrix} \begin{bmatrix} x(t) \\ \tilde{x}(t) \end{bmatrix} + \tilde{D}Du(t) \tag{3.62}$$

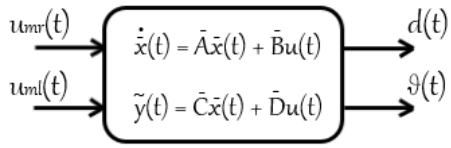


Figura 3.11: Diagrama del modelo completo del DDMR en espacio de estados

3.3. Modelo principal

En resumidas cuentas se ha obtenido un modelo matemático que describe el comportamiento del DDMR. Esta Sección tiene el propósito de identificar el modelo resultante (modelo principal), a fin de

aclarar sus componentes y diferenciarlo de los otros modelos que fueron surgiendo en el desarrollo de este Capítulo.

El modelo principal del DDMR está representado por las expresiones:

$$\dot{\bar{x}}(t) = \bar{A}\bar{x}(t) + \bar{B}u(t) \tag{3.63}$$

$$\tilde{y}(t) = \bar{C}\bar{x}(t) + \bar{D}u(t) \tag{3.64}$$

Donde el vector de estados $\bar{x}(t)$ está dado por:

$$\begin{aligned}\bar{x}(t) &= \begin{bmatrix} x(t) & \tilde{x}(t) \end{bmatrix}^T \\ &= \begin{bmatrix} \dot{\phi}_r(t) & \dot{\phi}_l(t) & \dot{Q}_{mr}(t) & \dot{Q}_{ml}(t) & \phi_r(t) & \phi_l(t) \end{bmatrix}^T \end{aligned} \tag{3.65}$$

Los estados $\dot{\phi}_r(t)$, $\dot{\phi}_r(t)$, corresponden a las velocidades angulares de las ruedas del robot, derecha e izquierda respectivamente, las cuales, están en unidades de radianes por segundo. Los estados $\dot{Q}_{mr}(t)$, $\dot{Q}_{ml}(t)$, corresponden a las corrientes de armadura en amperios, que circula por los motores derecho e izquierdo respectivamente. Por último, los estados $\phi_r(t)$, $\phi_l(t)$ corresponden a las posiciones angulares de las ruedas derecha e izquierda del DDMR, las cuales, están en unidades de radianes.

Las matrices $\bar{A}$,$\bar{B}$,$\bar{C}$,$\bar{D}$ están descritas como:

$$\bar{A} = \begin{bmatrix} A & 0 \\ \tilde{B}C & \tilde{A} \end{bmatrix} \tag{3.66}$$

$$\bar{B} = \begin{bmatrix} B \\ \tilde{B}D \end{bmatrix} \tag{3.67}$$

$$\bar{C} = \begin{bmatrix} \tilde{D}C & \tilde{C} \end{bmatrix} \tag{3.68}$$

$$\bar{D} = \begin{bmatrix} \tilde{D}D \end{bmatrix} \tag{3.69}$$

luego el vector de entrada $u(t)$ corresponde a:

$$u(t) = \begin{bmatrix} u_{mr}(t) & u_{ml}(t) \end{bmatrix}^T \tag{3.70}$$

con $u_{mr}(t)$ y $u_{ml}(t)$ las señales de entrada en unidades de voltios, correspondiente al voltaje aplicado a los motores derecho e izquierdo del robot. Por último, el vector de salida $\tilde{y}(t)$:

$$\tilde{y}(t) = \begin{bmatrix} d(t) & \theta(t) \end{bmatrix}^T \tag{3.71}$$

donde $d(t)$ corresponde a la distancia en centímetros entre el DDMR y otro agente en la trayectoria de navegación, y $\theta(t)$ su orientación medida como el ángulo entre el eje central del robot y la recta tangente a la trayectoria de navegación, en radianes.

Adicionalmente, como es posible medir la velocidad del DDMR, puede ser conveniente contar con un modelo que incluya como salida la velocidad lineal del robot. Lo anterior resulta de gran utilidad teniendo en consideración que la flota de robots móviles es dirigida por un robot líder, el cual, no necesariamente navega siguiendo una distancia determinada, más bien, este se desplaza con velocidad constante a través de la trayectoria de navegación. Como la velocidad del DDMR depende de la velocidad de las ruedas, las cuales, son variables de estado en el modelo definido anteriormente, solo se debe ampliar la matriz de salida $\bar{C}$ para incluir como salida la velocidad del robot.

3.4. Modelo complementario

A partir del modelo principal se deduce un modelo complementario. Este tiene como propósito el estudio y diseño de otras arquitecturas de control donde se aproveche la posibilidad de medir las salidas de velocidad, distancia y orientación del DDMR. El modelo complementario queda definido por las

expresiones:

$$\dot{x}_c(t) = A_c x_c(t) + B_c u(t) \tag{3.72}$$
$$y_c(t) = C_c x_c(t) + D_c u(t) \tag{3.73}$$

donde la ecuación de estados (3.72) es igual a la ecuación de estados presentada en el modelo principal (3.63), es decir:

$$x_c(t) = \bar{x}(t) \tag{3.74}$$
$$A_c = \bar{A} \tag{3.75}$$
$$B_c = \bar{B} \tag{3.76}$$

Para la ecuación de salida, las magnitudes físicas de interés corresponden a la velocidad del DDMR en centímetros por segundo, la distancia entre el robot y un agente externo en centímetros y su orientación respecto de la trayectoria de navegación en radianes.

$$y_c(t) = \begin{bmatrix} v(t) & d(t) & \theta(t) \end{bmatrix}^T \tag{3.77}$$

Con esto las matrices C_c y D_c quedan definidas por:

$$C_c = \begin{bmatrix} 100\frac{r}{2} & 100\frac{r}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -100\frac{r}{2} & -100\frac{r}{2} \\ 0 & 0 & 0 & 0 & \frac{r}{2l} & -\frac{r}{2l} \end{bmatrix} \tag{3.78}$$

$$D_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \tag{3.79}$$

Para terminar es necesario aclarar que las señales de entrada corresponden al voltaje aplicado a cada uno de los motores del DDMR, del mismo modo que en el modelo principal.

$$u(t) = \begin{bmatrix} u_{mr}(t) & u_{ml}(t) \end{bmatrix}^T \tag{3.80}$$

3.5. Parámetros

Para validar el modelo principal obtenido anteriormente es necesario realizar una simulación y analizar su respuesta frente a diferentes estímulos a través de su entrada. En este análisis se espera obtener una respuesta que represente de la mejor forma al sistema real y, de ser necesario, realizar algunos ajustes al modelo. En primer lugar, se requiere conocer todos los parámetros del modelo, algunos de estos se pueden obtener directamente de la geometría del robot (modelo CAD) como son el radio de las ruedas r y la distancia l entre cada una de las ruedas y el centro de giro del DDMR. La masa m se obtiene a través de la medición directa en una báscula. A partir de estos datos es posible estimar el momento de inercia j del robot cuando este se encuentra girando en torno a su centro de giro, a partir del teorema de Steiner.

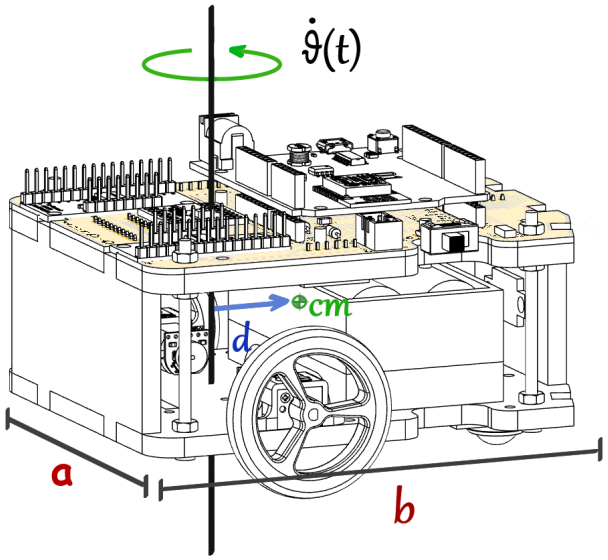


Figura 3.12: Representación del movimiento rotacional del DDMR

$$j = j_{cm} + md^2 \tag{3.81}$$

donde j_{cm} corresponde al momento de inercia del cuerpo respecto de su centro de masa, el cual, se puede aproximar como:

$$j_{cm} \approx \frac{1}{12} m(a^2 + b^2) \tag{3.82}$$

puesto que la geometría del DDMR se puede aproximar por un prisma rectangular, y el centro de masa obtenido a partir del modelo CAD se encuentra muy próximo al eje central de este prisma.

Las resistencias eléctricas R_{mr} y R_{ml} se miden directamente utilizando un multitester a la entrada de cada motor, a diferencia de las inductancias que fueron obtenidas experimentalmente analizando la respuesta temporal de la corriente eléctrica en el motor con el rotor detenido, en tal condición el modelo eléctrico del motor DC se reduce a un circuito LR. Para esto se utilizó un osciloscopio donde, el canal 1 (CH1) mide el voltaje que cae sobre una resistencia auxiliar $R_{aux} = 2{,}3\ \Omega$, el cual, representa la dinámica de la corriente eléctrica $\dot{Q}_m(t)$, y el canal 2 (CH2) mide el voltaje total aplicado, tal como se muestra en la Figura 3.13. El circuito equivalente puede ser modelado como un sistema de primer orden, donde la inductancia está directamente relacionada con la constante de tiempo eléctrica t_e de la respuesta temporal del sistema.

$$L_m = t_e \cdot (R_m + R_{aux}) \tag{3.83}$$

Los valores de las resistencias son conocidos, y la constante de tiempo t_e puede ser medida como el intervalo de tiempo en que la corriente eléctrica llega a un 63,2 % de su valor final desde su condición inicial, en este caso 0A. Dicho intervalo se observa en la curva obtenida desde el CH1 del osciloscopio presentado en la Figura 3.14

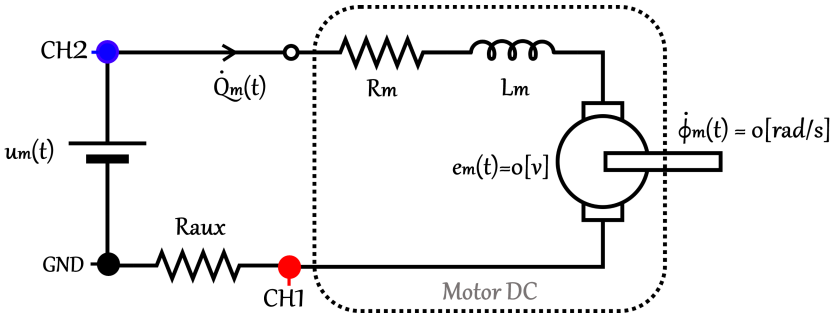


Figura 3.13: Diagrama esquemático del circuito de medición empleado en motor DC con rotor detenido

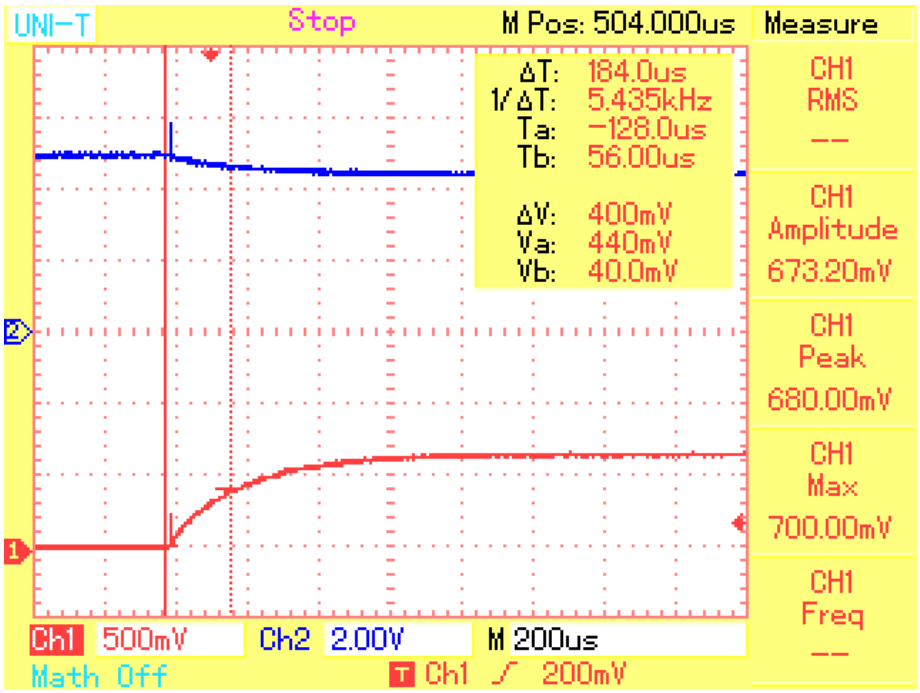


Figura 3.14: Respuesta temporal voltaje de entrada y corriente eléctrica medida en osciloscopio.

Los módulos del sistema reductor de los motores izquierdo y derecho N_r, N_l, junto con sus respectivas eficiencias η_r, η_l se extraen a partir de la hoja de datos del motor DC. Las constantes de la fuerza contraelectromotriz de cada motor kv_r y kv_l se obtienen de forma experimental midiendo la velocidad angular del motor sin carga $\dot{\phi}_m(t)$ y la corriente eléctrica $\dot{Q}_m(t)$ para diferentes valores de voltaje de entrada, puesto que en un motor DC cuando este se encuentra en rotación aparece en el inducido una tensión $e_m(t)$ proporcional al producto del flujo por la velocidad angular, por lo que en estado estacionario la tensión inducida es directamente proporcional a la velocidad angular.

$$kv = \frac{u_m(t)}{\dot{\phi}_m(t)} \tag{3.84}$$

Tabla 3.1: Relación de voltaje versus velocidad del motor DC

$u_m(t)$ $[V]$	$\dot{Q}_m(t)$ $[A]$	$e_m(t)$ $[V]$	$\dot{\phi}_m(t)$ $[rad/s]$
8.2	0.0132	8.0323	5000
4.88	0.0262	4.5466	2985
4.1	0.0103	3.9686	2450
3.3	0.0228	3.0104	1955
-3.3	-0.0229	-3.0091	-1955
-4.1	-0.0107	-3.9637	-2450
-4.88	-0.0263	-4.5459	-2990
-8.2	-0.0141	-8.0209	-5000

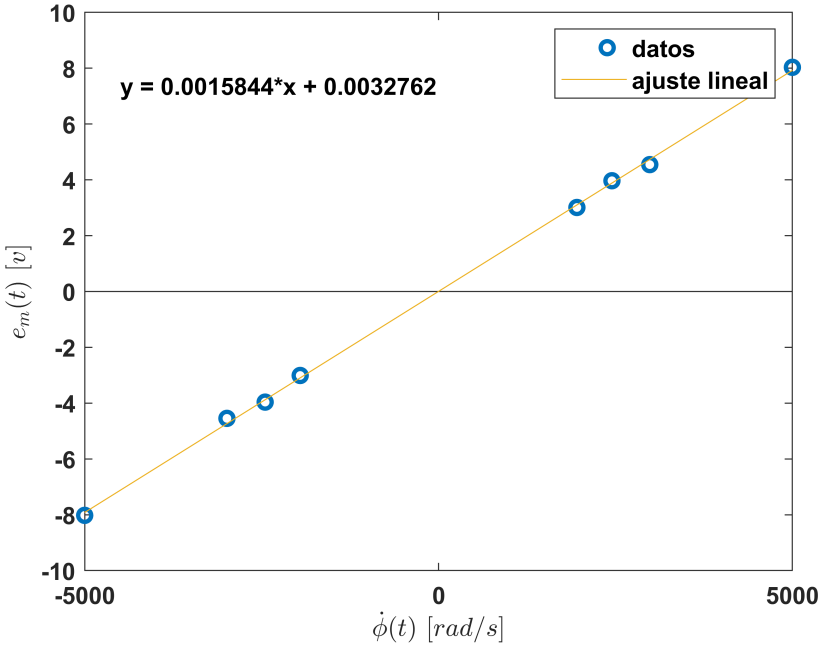


Figura 3.15: Voltaje inducido versus velocidad del motor DC

Los resultados presentados en la Tabla 3.2 se representan de forma gráfica en la Figura 3.15 donde se observa la relación lineal obtenida entre el voltaje inducido y la velocidad del motor DC. El experimento mencionado anteriormente fue realizado a ambos motores del DDMR obteniendo resultados similares. Además, en un motor DC esta constante kv se relaciona directamente con la constante de torque kt, de hecho, estos parámetros son iguales cuando se expresan en el mismo sistema de unidades.

También es necesario estimar las inercias de los motores derecho e izquierdo j_{mr} y j_{ml} respectivamente, para esto se obtiene la constante de tiempo mecánica t_m del motor DC, la cual, está involucrada directamente en la respuesta transitoria de la velocidad del motor frente a un escalón de voltaje entre sus terminales y se relaciona con su inercia y el resto de parámetros de un motor DC según:

$$j_m = \frac{t_m \, kt \, kv}{R_m} \tag{3.85}$$

Como el resto de parámetros fueron obtenidos anteriormente, solo basta determinar t_m, para lo cual se observa la dinámica del voltaje con un osciloscopio conectado en los terminales del motor. Al aplicar un voltaje tipo escalón a la entrada del motor, el osciloscopio construirá la característica equivalente de velocidad. El circuito esquemático utilizado se presenta en la Figura 3.16

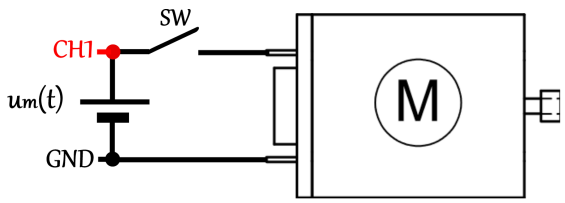


Figura 3.16: Circuito esquemático empleado para la estimación la inercia del motor DC

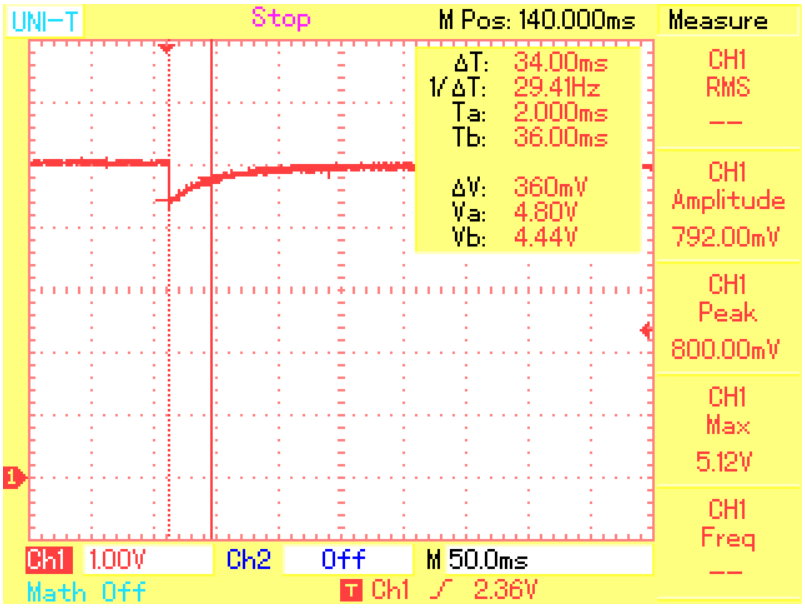


Figura 3.17: Respuesta transitoria del voltaje en el motor equivalente a la dinámica de la velocidad

En la Figura 3.19 se muestra la respuesta transitoria que tiene el voltaje de entrada del motor producto de la dinámica de la velocidad del rotor, a partir de esta es posible determinar la constante de tiempo mecánica t_m como el intervalo de tiempo que toma el voltaje en alcanzar el 63 % de su valor final luego de haber cerrado el circuito a través del interruptor **SW**, para luego estimar la inercia del motor DC utilizando los parámetros ya obtenidos experimentalmente.

Por último se estiman los coeficientes de fricción viscosa b_r y b_l que relacionan la fuerza de fricción con la velocidad de giro de las ruedas del DDMR, para esto se utiliza el robot y se aplica un voltaje de entrada solo a uno de los motores con el objetivo de que el robot comience a girar y alcance un estado estacionario, en dicho punto de operación se mide la velocidad de la rueda acoplada al motor activo, por medio de los encoders del robot, y la corriente que circula por este a través de un amperímetro (los cables que se utilizan son lo suficientemente largos y flexibles a modo de no interferir en el movimiento del robot), luego el coeficiente de fricción viscosa, para una operación en estado estacionario, se puede estimar según:

$$\begin{aligned} b &= \frac{\eta\, kt \dot{Q}_m(t)}{N\, \dot{\phi}(t)} \\ &= \frac{\tau(t)}{\dot{\phi}(t)} \end{aligned} \tag{3.86}$$

Este procedimiento es válido para obtener ambos coeficientes de fricción viscosa, el cual se aplica siempre con una de las ruedas detenida.

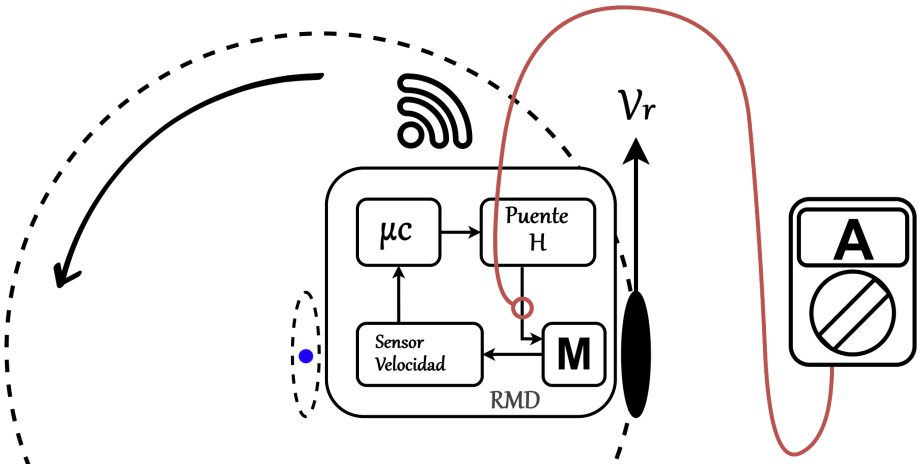


Figura 3.18: Esquema de medición en el robot móvil diferencial con una rueda detenida

Tabla 3.2: Relación de corriente versus velocidad de la rueda

$\dot{Q}_m(t)$ $[A]$	$\dot{\phi}(t)$ $[rad/s]$
0.065	4.5111
0.07	7.1228
0.085	9.7345
0.099	12.1088
0.107	14.7205
0.109	16.6199

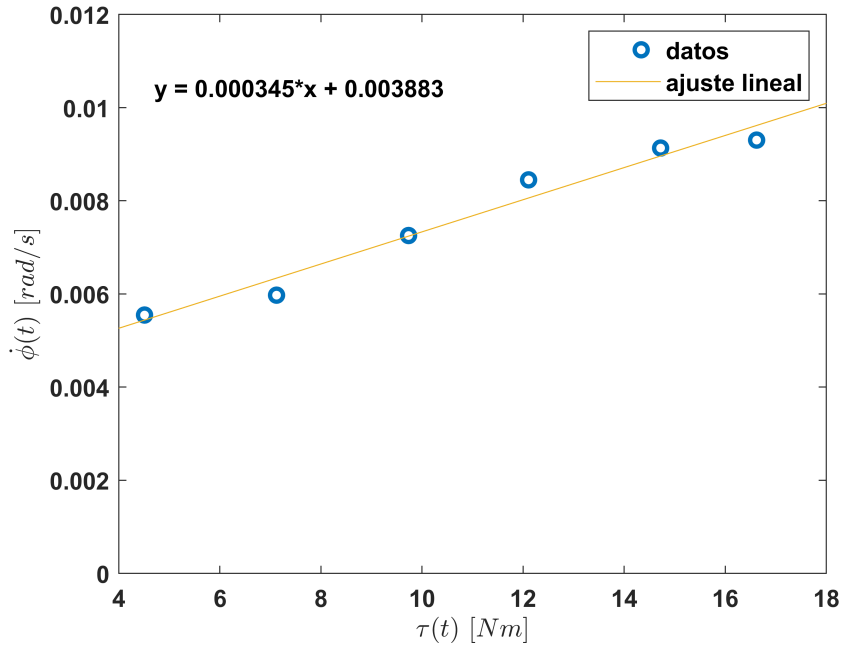


Figura 3.19: Velocidad rueda versus torque disipado

Tabla 3.3: Parámetros del DDMR

Símbolo	Valor	Unidad (SI)	Parámetro
r	0.0215	m	Radio de las ruedas activas
l	0.056	m	Distancia entre las ruedas activas y el centro de giro del DDMR
m	0.445	kg	Masa del DDMR
j	1.1968e-3	$kg\ m^2$	Momento de inercia del DDMR
j_{mr}, j_{ml}	6.7738e-9	$kg\ m^2$	Momento de inercia de los motores
N_r, N_l	0.01	–	Módulo de transformación mecanismo reductor
η_r, η_l	0.2	–	Eficiencias de los motores
kv_r, kv_l	1.5844e-3	$V\ s/rad$	Constante de la fuerza contraelectromotriz de los motores
kt_r, kt_l	1.5844e-3	$A/N\ m$	Constante de torque de los motores
b_r, b_l	3.45e-4	$N\ s/rad$	Constante de roce viscoso del DDMR
L_{mr}, L_{ml}	2.7416e-3	H	Inductancia de armadura de los motores
R_{mr}, R_{ml}	12.6	Ω	Resistencia de armadura motores

3.6. Simulación

La simulación del modelo principal se realiza en el software MATLAB, donde en primer lugar se declaran todos los parámetros del modelo, luego se define el modelo en espacio de estados (3.61), (3.62), del cual, se obtiene como resultado:

$$\dot{\bar{x}}(t) = \bar{A}\bar{x}(t) + \bar{B}u(t)$$
$$\bar{y}(t) = \bar{C}\bar{x}(t) + \bar{D}u(t)$$

$$\begin{bmatrix} \ddot{\phi}_r(t) \\ \ddot{\phi}_l(t) \\ \dot{Q}_{mr}(t) \\ \dot{Q}_{ml}(t) \\ \dot{\phi}_r(t) \\ \dot{\phi}_l(t) \end{bmatrix} = \begin{bmatrix} -3{,}606 & 0{,}274 & 331{,}290 & -25{,}209 & 0 & 0 \\ 0{,}274 & -3{,}606 & -25{,}209 & 331{,}290 & 0 & 0 \\ -57{,}791 & 0 & -4595{,}856 & 0 & 0 & 0 \\ 0 & -57{,}791 & 0 & -4595{,}856 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\phi}_r(t) \\ \dot{\phi}_l(t) \\ \dot{Q}_{mr}(t) \\ \dot{Q}_{ml}(t) \\ \phi_r(t) \\ \phi_l(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 364{,}750 & 0 \\ 0 & 364{,}750 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{mr}(t) \\ u_{ml}(t) \end{bmatrix} \tag{3.87}$$

$$\begin{bmatrix} d(t) \\ \theta(t) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & -1{,}0750 & -1{,}0750 \\ 0 & 0 & 0 & 0 & 0{,}191 & -0{,}191 \end{bmatrix} \begin{bmatrix} \dot{\phi}_r(t) \\ \dot{\phi}_l(t) \\ \dot{Q}_{mr}(t) \\ \dot{Q}_{ml}(t) \\ \phi_r(t) \\ \phi_l(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_{mr}(t) \\ u_{ml}(t) \end{bmatrix} \tag{3.88}$$

De esta forma queda determinado el modelo que representa la distancia $d(t)$ en centímetros entre el DDMR y otro agente, y su orientación $\theta(t)$ en radianes respecto de una trayectoria de navegación, donde la entrada al sistema corresponde al voltaje aplicado a los motores $u_{mr}(t)$, $u_{ml}(t)$ derecho e izquierdo respectivamente. El vector de estados de este sistema está compuesto por las velocidades angulares $\dot{\phi}_r(t)$, $\dot{\phi}_l(t)$ de las ruedas en radianes por segundo, las corrientes de armadura de los motores $\dot{Q}_{mr}(t)$, $\dot{Q}_{ml}(t)$ en amperios, y las posiciones angulares de las ruedas $\phi_r(t)$, $\phi_l(t)$, relativas a la orientación y distancia del DDMR con la trayectoria de navegación y otro agente, en radianes.

A partir del modelo se puede analizar la estabilidad de este, observando los autovalores de la matriz de estados $\bar{A}$, los cuales corresponden a los polos del sistema. Además, es necesario estudiar la observabilidad y controlabilidad del sistema para posteriormente diseñar los controladores deseados. Existen diversas formas de realizar los estudios señalados anteriormente, además el software MATLAB entrega un amplio conjunto de comandos que permiten realizar esta tarea. En particular, se utiliza el comando:

```
H=canon(sys,'modal');
```

el cual, transforma nuestro modelo lineal definido por la variable **sys**, en una representación canónica en espacio de estados y lo almacena en la variable **H**.

```
H =

A =
        x1      x2      x3      x4      x5      x6
 x1      0       0       0       0       0       0
 x2      0       0       0       0       0       0
 x3      0       0   -4592       0       0       0
 x4      0       0       0   -4591       0       0
 x5      0       0       0       0  -8.372       0
 x6      0       0       0       0       0  -7.187

B =
            u1          u2
 x1      0.4228           0
 x2           0      0.8457
 x3      -8.071      -8.071
 x4       8.074      -8.074
 x5       0.568      -0.568
 x6      0.5446      0.5446

C =
            x1          x2          x3          x4          x5          x6
 y1        -8.6        -4.3   0.0007063           0           0       6.687
 y2       1.536     -0.7679           0   0.0001469      -1.145           0

D =
     u1  u2
 y1   0   0
 y2   0   0

Continuous-time state-space model.
```

Esta representación canónica tiene la particularidad que la matriz de estados A es diagonal, en ese caso los autovalores corresponden a los valores de la diagonal. Los autovalores del sistema están asociados a modos naturales que componen la solución del sistema en el dominio temporal, en nuestro caso existen dos autovalores de la matriz A (polos del sistema) iguales a cero, esto quiere decir que los estados asociados a esos autovalores se mantendrán en un punto de operación estable cuando la entrada al sistema sea nula. La interpretación física del comportamiento de estos estados se explica teniendo en cuenta que en nuestro modelo dos de los estados corresponden a las posiciones angulares de las ruedas del DDMR, dichas posiciones se mantendrán en un punto de operación si y solo si el voltaje aplicado a los motores es cero. Por otra parte, se tiene que el resto de los autovalores son negativos, por consiguiente los modos naturales asociados a dichos estados se mantiene acotados y convergen a cero a medida que el tiempo tiende a infinito. En nuestro modelo, los polos más rápidos están asociados a la dinámica de la corriente de los motores y los polos dominantes (más lentos) están asociados a la dinámica de las velocidades de las ruedas del DDMR. Los estados obtenidos de la representación canónica corresponden a una transformación lineal de los estados presentados en (3.87), (3.88) por lo que no necesariamente representan la misma magnitud física.

De igual manera se analiza la observabilidad de los estados del sistema, observando la matriz de entrada **B** obtenida de la transformación canónica de nuestro modelo. Dado que dicha matriz no posee filas nulas, se concluye que todos los estados del sistema son observables. Además, se puede ver que existen estados que solo dependen de una de las entradas y otros de una combinación lineal de las entradas.

Por último, se analiza la controlabilidad de los estados del sistema observando la matriz **C** obtenida de la transformación canónica. Puesto que dicha matriz no posee columnas nulas, se asegura que todos los estados son controlables.

A continuación se presenta la implementación del modelo del DDMR en la herramienta Simulink de MATLAB, lo cual, se puede apreciar en la Figura 3.20. En la Figura 3.21 se aprecia la evolución temporal de las salidas de interés $d(t)$, $\theta(t)$, distancia y orientación, respectivamente, frente a un escalón de entrada a partir del instante $t = 1\ s$, este escalón corresponde a un voltaje de amplitud 6 volts aplicado en ambos motores en el mismo instante de tiempo. Inicialmente, el DDMR se encuentra correctamente posicionado en sobre una trayectoria de navegación ($\theta(0) = 0\ rad$) y se encuentra a una distancia de veinte centímetros ($d(0) = 20\ cm$), respecto a otro agente que se encuentra en reposo. Dado que el voltaje aplicado a los motores es positivo y de la misma magnitud, el DDMR comienza a avanzar hacia adelante, sin cambiar su orientación, de este modo la distancia entre el DDMR y el agente en reposo comienza a disminuir hasta que se encuentran en la misma posición aproximadamente en $t = 1{,}5\ s$, luego el DDMR 'adelanta' al agente en reposo y sigue avanzando de forma indefinida.

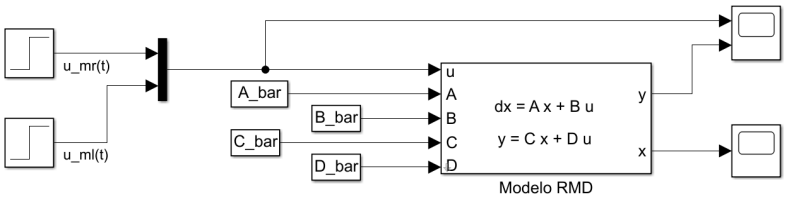


Figura 3.20: Esquema implementado en Simulink para simular el modelo principal del DDMR

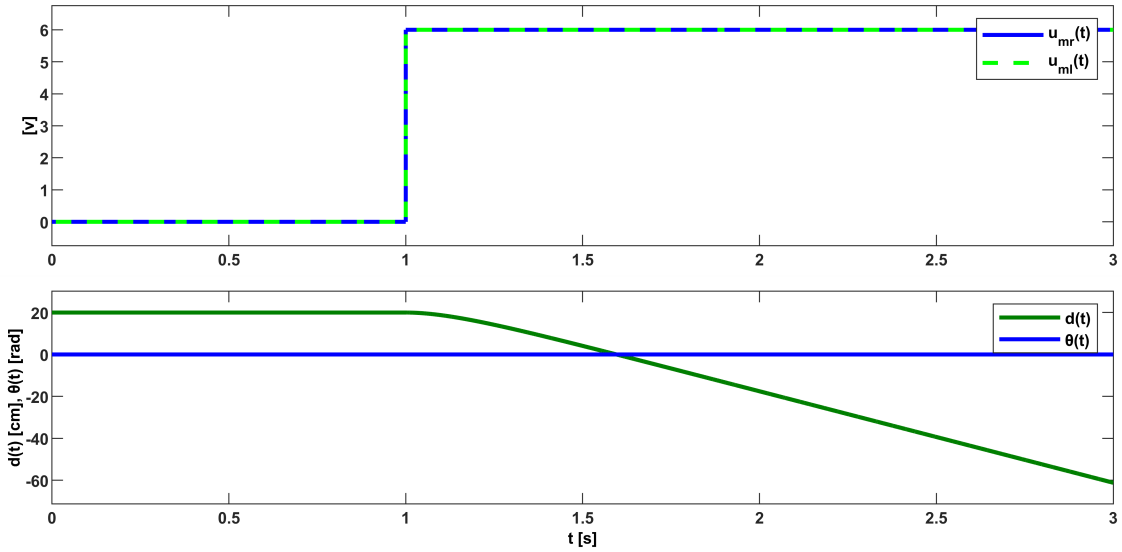


Figura 3.21: Evolución temporal de las entradas y salidas del DDMR

Lo anterior también se puede apreciar en la evolución temporal de los estados del sistema (Figura 3.22), donde se aprecia que las velocidades angulares de las ruedas del DDMR tienen el mismo valor, partiendo desde el reposo hasta alcanzar una velocidad aproximada de 20 rad/s. Se observa la dinámica de la corriente de armadura de los motores, la cual, tiene un *peak* de 0.5 A asociado al torque inicial que permite sacar al DDMR del reposo, luego dicha corriente disminuye hasta alcanzar un valor aproximado de 0.2 A. Finalmente, se aprecia que las posiciones angulares de las ruedas aumenta de forma indefinida desde el instante en que se energizan los motores, esto nos indica que el desplazamiento del robot es hacia adelante,

partiendo de un valor inicial de -9.3 rad ($\phi_r(0) = \phi_l(0) = -9{,}3\ rad$), los cuales permiten posicionar el DDMR a una distancia inicial de 20 cm respecto del agente en reposo.

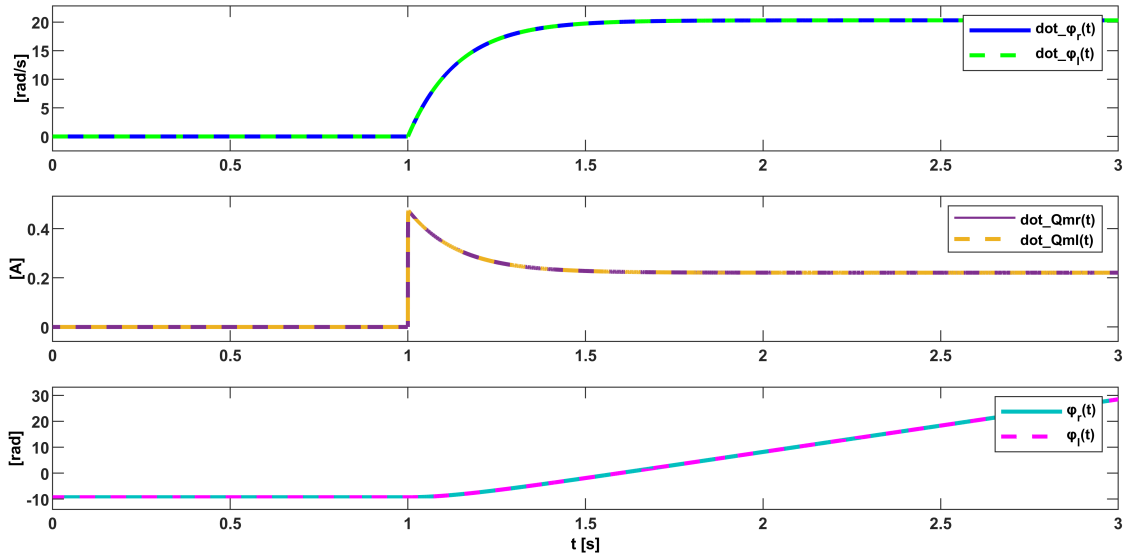


Figura 3.22: Evolución temporal de los estados del DDMR - Respuesta a escalón

De igual manera se analiza el comportamiento del DDMR cuando el voltaje aplicado a los motores posee la misma magnitud, pero distinto sentido para un mismo instante de tiempo. Este escenario se aprecia en las figuras 3.23, 3.24, donde el voltaje aplicado al motor derecho es de 6 V y el voltaje aplicado al motor izquierdo es de -6 V en el instante $t = 1\ s$. A partir de ese momento la velocidad angular de la rueda derecha comienza a aumentar hasta alcanzar los 20 rad/s. De igual modo, pero en sentido contrario, la velocidad angular de la rueda izquierda aumenta hasta alcanzar una velocidad de -20 rad/s. El sentido de giro de las ruedas también se puede observar a través de las corrientes que circulan por cada uno de los motores, la cual, está relacionada con el torque que produce el movimiento del DDMR, y se puede ver directamente en los estados asociados a las posiciones angulares de cada rueda. Las salidas del sistema nos indican que en este escenario de operación, el DDMR comienza a girar de forma indefinida en sentido antihorario, puesto que el ángulo que describe la orientación del DDMR respecto de la trayectoria de navegación aumenta de forma positiva. La distancia entre el robot y el agente en reposo se mantiene constante en 20 cm (condición inicial), puesto que la velocidad lineal del DDMR es nula.

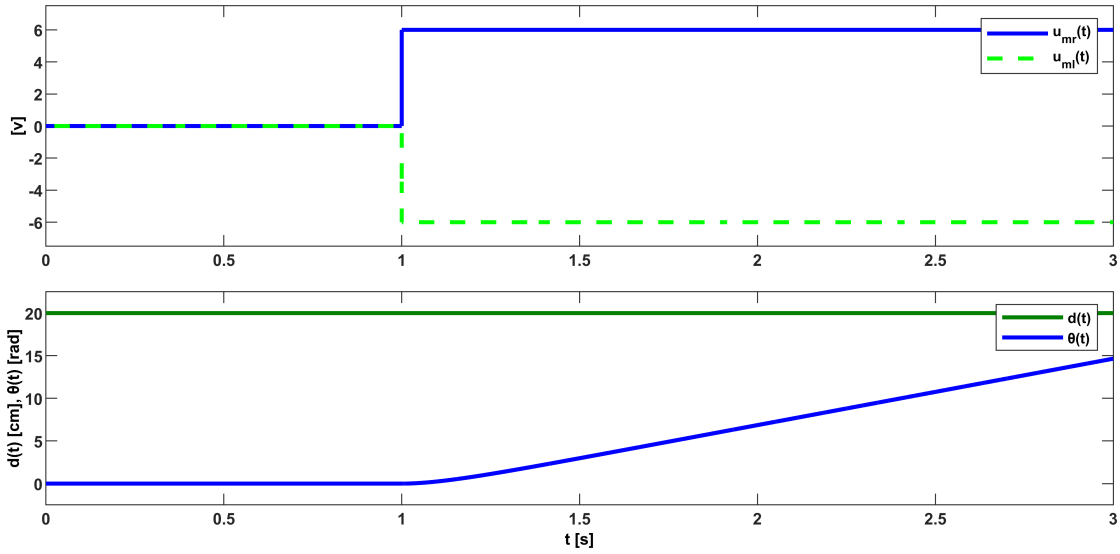


Figura 3.23: Evolución temporal de las entradas y salidas del DDMR

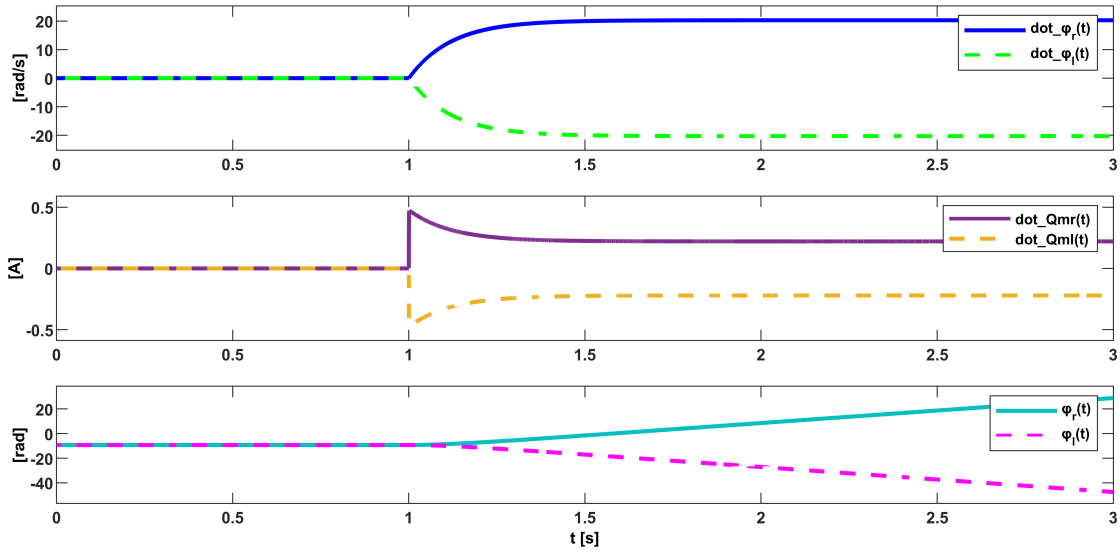


Figura 3.24: Evolución temporal de los estados del DDMR - Respuesta a escalón

En síntesis se han presentado los resultados de simulación para la operación del DDMR cuando este es excitado a través del voltaje aplicado a sus motores, empleando señales de voltajes constantes. En la implementación propuesta para el accionamiento de los motores se emplea un puente H que permite controlar el sentido de giro de los motores, y una señal de disparo del tipo PWM para controlar el nivel de voltaje aplicado a los motores. este nivel lo controla un microcontrolador que varía el ciclo de trabajo de la señal PWM. La amplitud de esta señal corresponde al voltaje aportado por la batería del DDMR compuesta por dos celdas de Li-Ion 18650, las que poseen un voltaje nominal de 7.4 V, y un voltaje de 8.4 V cuando la carga está completa. Para analizar el comportamiento del DDMR frente a una excitación del tipo PWM se implementa este tipo de señal considerando el voltaje máximo que aporta la batería del robot y que el ciclo de trabajo que varía entre 0 y 1 es controlado por el microcontrolador a través de una variable de 10 bits, es decir que un ciclo de trabajo del 0 % equivale a 0 y un ciclo de trabajo del 100 % equivale a 1023 en el valor que alcanza

la variable en el microcontrolador. La implementación de lo anterior se realiza en Simulink y se muestra en la Figura 3.25. En la Figura 3.26 se muestra la señal PWM que es aplicada a cada uno de los motores del robot, la cual, posee una frecuencia de 500 Hz, amplitud 8.4 V y valor medio 6 V. Las figuras 3.27 y 3.28 muestran el comportamiento del DDMR cuando los motores son accionados a través de señales PWM. A modo de comparar los resultados se escogen valores para la variable de control (microcontrolador) que generen una señal PWM de valor medio equivalente a 6 V para excitar el motor derecho y de -6 V para excitar el motor izquierdo.

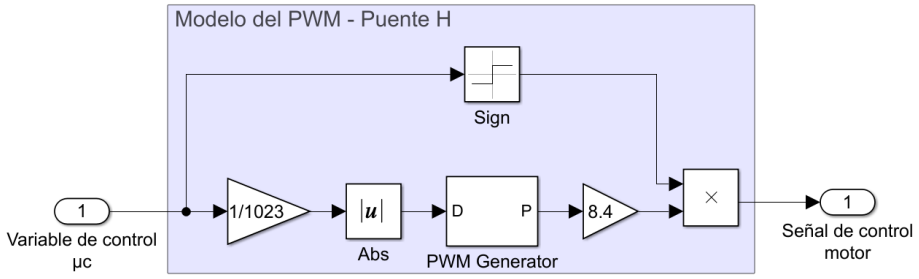


Figura 3.25: Modelo en Simulink del accionamiento de los motores del DDMR

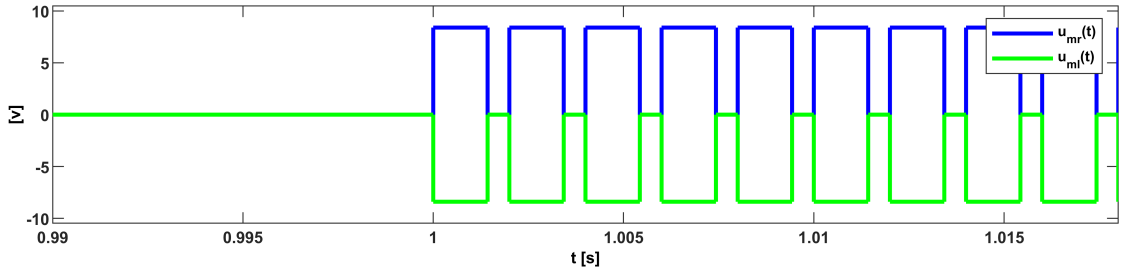


Figura 3.26: Señal PWM de entrada

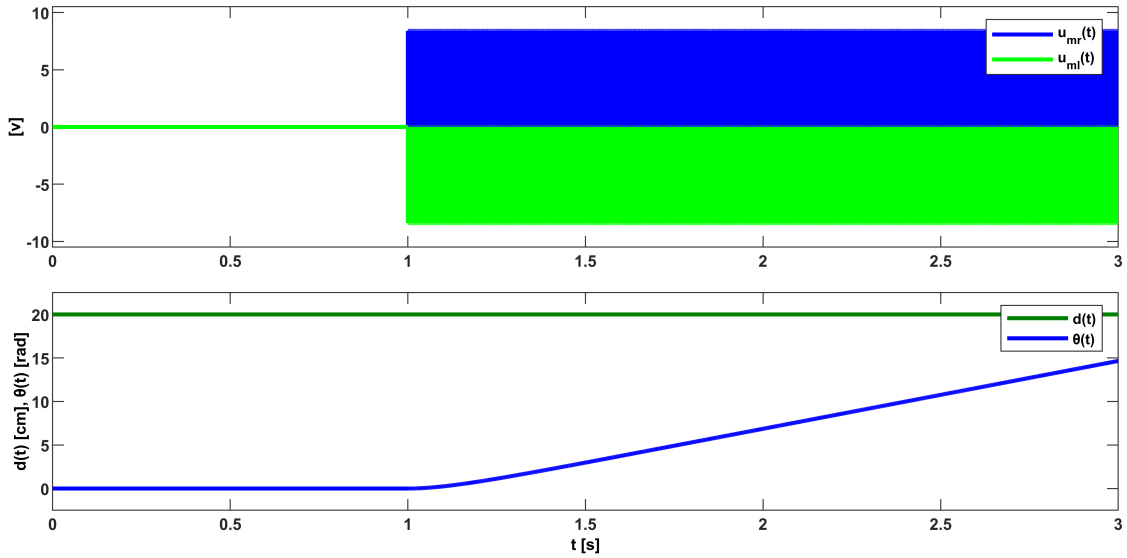


Figura 3.27: Evolución temporal de las entradas y salidas del DDMR

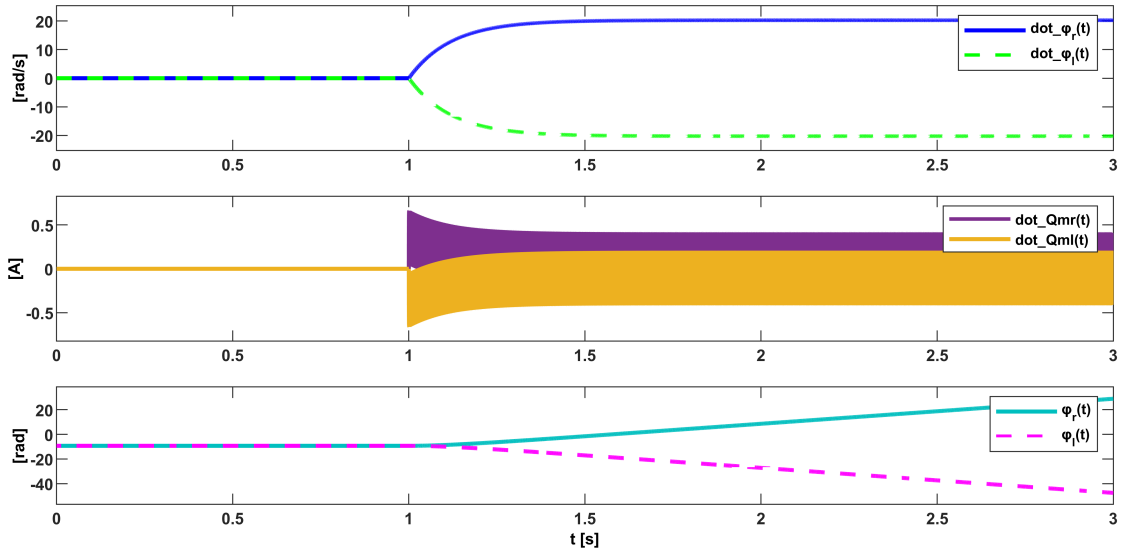


Figura 3.28: Evolución temporal de los estados del DDMR - Respuesta a escalón

Al comparar las salidas obtenidas en 3.23 y 3.27 se observa que tienen el mismo comportamiento, de hecho al comparar los estados del sistema en 3.24 y 3.28 se obtiene la misma dinámica para los estados de velocidad y posición angular de las ruedas. La corriente eléctrica que circula a través de los motores se ve más afectada por la señal PWM de entrada, pero esta dinámica interna no afecta de forma significativa las salidas de interés de nuestro modelo, y solo provoca un pequeño rizado en las velocidades de las ruedas del DDMR (Figura 3.29). La amplitud del rizado para las velocidades angulares de las ruedas no es significativa con respecto al valor promedio de estas señales, además en los estados correspondientes a las posiciones angulares de las ruedas el rizado no es apreciable. A partir de los resultados obtenidos y para efectos del diseño de controladores se considera la transferencia entre el microcontrolador y la entrada de los motores (modelo de la Figura 3.25) como una ganancia $K_{PWM} = \frac{8,4}{1023}$.

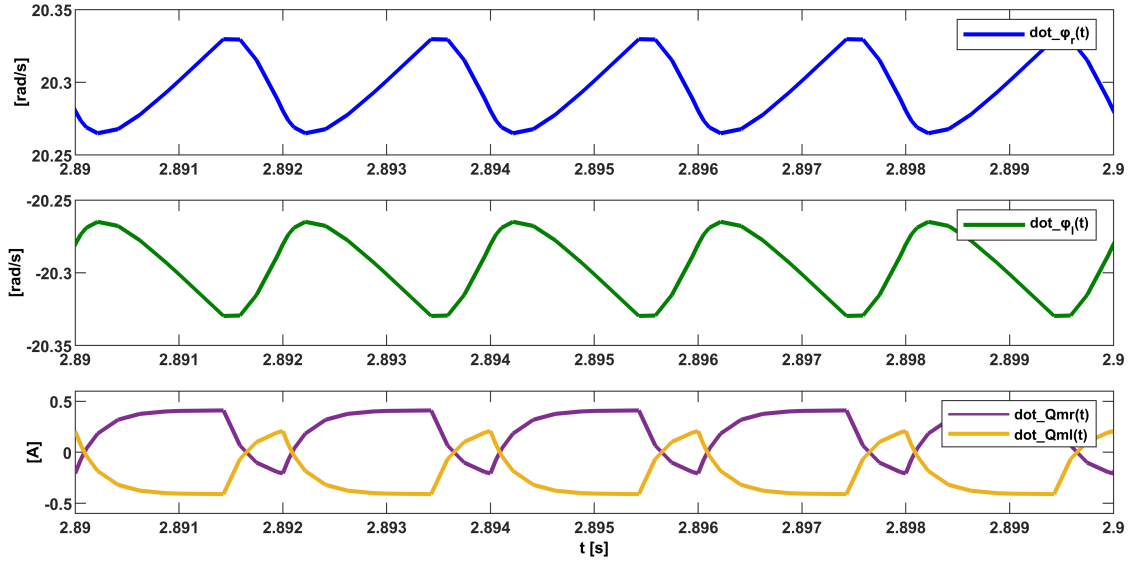


Figura 3.29: Rizado en los estados del modelo producto del control PWM

4 | DISEÑO DE CONTROLADORES

El problema que se desarrolla en este Capítulo consiste en el diseño de controladores que permitan llevar las salidas de interés del DDMR a un punto de operación deseado a partir de los modelos matemáticos obtenidos en el Capítulo anterior. El objetivo es diseñar la arquitectura y parámetros del controlador utilizando herramientas de la *control system toolbox* de MATLAB, a fin de validar dichos controladores y ajustar sus parámetros considerando la estabilidad y velocidad de convergencia de las señales de interés, y la amplitud máxima de la señal de control. Es de aclararse que los controladores diseñados en este Capítulo no buscan ser los más óptimos, más bien, pretenden presentar alternativas de controladores que pueden ser implementados en el DDMR.

4.1. Esquema de Control basado en el regulador LQR

El regulador LQR es un tipo de control lineal óptimo [20], donde la ley de control $u(t)$ (señal de control) corresponde a la solución factible óptima de un problema de minimización del funcional $J(u)$.

$$J(u) = \int_0^\infty \{x^T(t)Qx(t) + u^T(t)Ru(t)\}dt \tag{4.1}$$

Donde el funcional presentado en (4.1) está definido para un horizonte infinito, es decir, en régimen permanente, con el objetivo de encontrar una solución óptima que sea invariante en el tiempo. El problema de minimización está sujeto al sistema dinámico $\dot{x}(t) = Ax(t) + Bu(t)$. Donde el objetivo del regulador LQR estándar consiste en llevar los estados del sistema hacia el origen de forma óptima dentro de un intervalo de tiempo finito, por lo que es necesario que el modelo sea controlable y observable. Para asegurar que el problema tenga solución óptima, tanto el dominio del funcional como el de las restricciones deben ser regiones convexas. Si se analiza $J(u)$ se puede ver que corresponde a una función cuadrática en u(t), de este modo para que esta función tenga un mínimo las matrices Q y R deben ser semidefinida positiva y definida positiva respectivamente y sus autovalores reales. Además, el sistema dinámico debe ser LTI para asegurar que la solución óptima no dependa del tiempo.

De cumplirse las condiciones anteriormente mencionadas, la acción de control que minimiza el funcional $J(u)$ está dada por:

$$u(t) = -R^{-1}B^T Px(t) \tag{4.2}$$

Donde P es una matriz real simétrica definida positiva, la cual se obtiene a partir de la *ecuación de Riccati algebraica*

$$A^T P + PA + Q - PBR^{-1}B^T P = 0 \tag{4.3}$$

Por lo cual, escogiendo adecuadamente las matrices Q, R y conociendo el modelo dinámico del sistema definido por las matrices A y B, matriz de estados y matriz de entrada respectivamente se puede conocer la ganancia de realimentación $K = R^{-1}B^T P$ que minimiza el funcional $J(u)$.

Ahora bien, como se plantea al inicio de este Capítulo, nuestro controlador debe ser capaz de llevar las salidas de interés a un punto de operación deseado. Por lo que se define un estado adicional x_i como la integral del error de seguimiento, esto es:

$$x_i(t) = \int^t (y_d - y(t))dt \tag{4.4}$$

De esta forma se incorpora la dinámica del error de seguimiento $\dot{x}_i(t)$ en el modelo dinámico del sistema y se resuelve el problema de minimización para este nuevo sistema ampliado, así pues, se garantiza que la ley de control encontrada permita llevar la integral del error de seguimiento a cero en un intervalo de tiempo finito. La Figura 4.1 muestra el esquema de control basado en el regulador LQR que incluye acción integral, también se observa la implementación de la ley de control $u(t)$ obtenida a partir de la solución del problema de minimización.

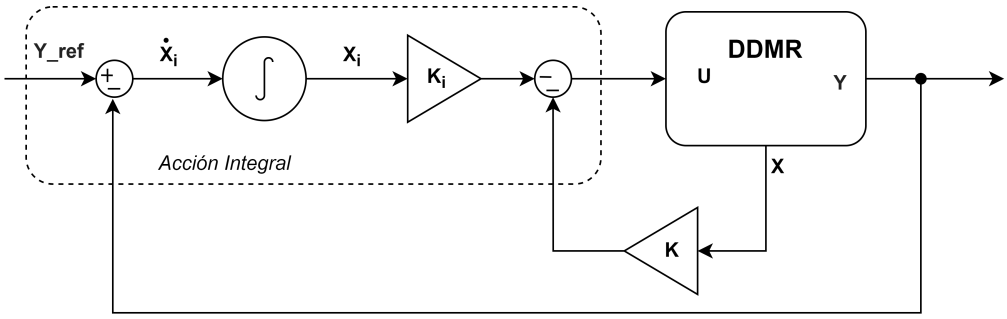


Figura 4.1: Esquema de control LQI

La obtención de la ganancia de realimentación óptima se realiza utilizando las expresiones (4.2) y (4.3), a partir del modelo dinámico ampliado y escogiendo adecuadamente las matrices Q y R. De igual manera se puede hacer uso de las herramientas que nos entrega MATLAB, en particular el comando:

```
K=lqi(sys,Q,R);
```

nos entrega directamente la ganancia de realimentación óptima dado un sistema dinámico y las matrices Q y R adecuadas. Además, este comando considera la acción integral, por lo que resuelve el problema de optimización considerando los estados definidos por la integral del error de seguimiento para las salidas del sistema, tal como fue explicado anteriormente.

4.1.1. Elección de las matrices Q y R

Las matrices de ponderación Q y R permiten definir un compromiso entre la velocidad en que los estados tienden al origen y el esfuerzo de las señales de control empleadas para dicho propósito. Con esto se logra la respuesta deseada moderando el gasto energético o nivel de amplitud de las señales de control. También permiten escalar los valores de los estados y señales de control cuando existen variaciones paramétricas entre el modelo dinámico y el sistema real (en la mayoría de los casos, el modelo dinámico es una aproximación del sistema real). Si bien no existe una regla general definida para determinar las matrices de ponderación, existen algunas propiedades que nos sugieren la dirección de sintonización.

- Para una matriz R constante, si Q tiende a cero, los polos estables del sistema tienden a los polos de lazo abierto, mientras que los polos inestables se reflejan respecto del eje jw. El caso de variar R se puede comprender de la misma forma, es decir, al hacer que Q tienda a cero se puede entender como que R se aumenta, y viceversa.
- Otro criterio que se puede seguir es, si se desea un mejor desempeño o respuesta de la $x_i(t)$ variable de estado, entonces aumentar q_{ii}. Por el contrario, si no interesa el movimiento de $x_i(t)$, se puede dar un valor pequeño a q_{ii}.

- Si se desea un ahorro de esfuerzo de una señal de control $u_j(t)$, entonces se debe aumentar el valor de r_{jj}, de forma similar, si no interesa el esfuerzo de control, se pueden dar valores pequeños a r_{jj}.

Una elección simple de las matrices de ponderación se puede realizar de acuerdo a la *regla de Bryson*, la cual sugiere escoger las matrices Q y R diagonales con:

$$q_{ii} = \frac{1}{x_{i_{max}}^2}, \qquad i = 1, 2, ..., n \tag{4.5}$$

$$r_{jj} = \frac{1}{u_{j_{max}}^2}, \qquad j = 1, 2, ..., m \tag{4.6}$$

Si bien, la *regla de Bryson* da buenos resultados en la mayoría de los casos, este método puede ser considerado como un punto de partida a la hora de definir las matrices de ponderación. Además, es de gran utilidad conocer el comportamiento del sistema dinámico, sobre todo cuando los sistemas poseen más de una señal de control y las salidas de interés corresponden a una combinación lineal de los estados, en dichos casos no basta con enfocarse solamente en los valores de la diagonal de las matrices de ponderación. En general, una selección adecuada de Q y R se dará mediante un procedimiento iterativo en el cual se observe el desempeño del sistema de control.

4.1.2. Simulación

Para simular el comportamiento del esquema de control propuesto se definen en primer lugar las matrices de ponderación Q y R considerando el sistema dinámico del DDMR presentado en (3.63) y (3.64), el cual, se presenta con mayor detalle en (3.87) y (3.88). El modelo dinámico resultante es observable, controlable e invariante en el tiempo según el análisis realizado en el Capítulo anterior.

En primer lugar, se define el vector de estados ampliado según:

$$\begin{aligned} z(t) &= \begin{bmatrix} \bar{x}(t) & \vert & x_i(t) \end{bmatrix}^T \\ &= \begin{bmatrix} \dot{\phi}_r(t) & \dot{\phi}_l(t) & \dot{Q}_{mr}(t) & \dot{Q}_{ml}(t) & \phi_r(t) & \phi_l(t) & \vert & I_{ed}(t) & I_{e\theta}(t) \end{bmatrix}^T \end{aligned} \tag{4.7}$$

Donde $I_{ed}(t)$ y $I_{e\theta}(t)$ corresponden a la integral del error de seguimiento para las salidas de distancia $d(t)$ y orientación $\theta(t)$ respectivamente. Luego se definen las matrices de ponderación Q y R siguiendo la *regla de Bryson*. Los estados asociados a las velocidades angulares de las ruedas $\dot{\phi}_r(t)$ y $\dot{\phi}_l(t)$ tienen unidades de radianes por segundo, por lo cual se define 20 rad/s como el valor máximo aceptable para dichos estados, esto implica que:

$$q_{11} = q_{22} = \frac{1}{20^2} = 0{,}0025 \tag{4.8}$$

Los estados correspondientes a las corrientes de armadura de los motores derecho e izquierdo, $\dot{Q}_{mr}(t)$ y $\dot{Q}_{ml}(t)$ respectivamente, se definen en unidades de amperios, de esta forma se escoge 0,5 A como valor máximo aceptable en estos estados.

$$q_{33} = q_{44} = \frac{1}{0{,}5^2} = 4 \tag{4.9}$$

Luego los estados $\phi_r(t)$ y $\phi_l(t)$, asociados a las posiciones angulares de las ruedas del DDMR relativas a la distancia respecto de un agente, y a la orientación respecto de la trayectoria de navegación, están definidas en unidades de radianes. Dado que el objetivo de control es mantenerse sobre la trayectoria y mantener la distancia entre el DDMR y otro agente, el valor máximo aceptable para las posiciones angulares relativas se define en 0,0465 rad equivalente a una distancia de 1 mm para una rueda de radio 21,5 mm, por lo cual:

$$q_{55} = q_{66} = \frac{1}{0{,}0465^2} = 462{,}25 \tag{4.10}$$

Por otro lado, los estados adicionales $I_{ed}(t)$ y $I_{e\theta}(t)$ asociados a la integral del error de seguimiento para las salidas del sistema, se analizan por separado debido a que representan magnitudes físicas diferentes. Primero el estado $I_{ed}(t)$ corresponde a la integral de la diferencia entre una distancia de seguimiento deseada d_{ref} y la distancia $d(t)$ correspondiente a la salida del sistema, estas distancias están en unidades de centímetros, por lo cual, se define como valor máximo aceptable para la integral del error de seguimiento 0.025 cm s.

$$q_{77} = \frac{1}{0{,}025^2} = 1600 \tag{4.11}$$

Segundo el estado $I_{e\theta}(t)$ corresponde a la integral de la diferencia entre una orientación deseada θ_{ref} y la orientación $\theta(t)$ correspondiente a la salida del sistema, estas magnitudes están en unidades de radianes, por lo cual, se define como valor máximo aceptable para la integral del error de seguimiento 0.01 rad s.

$$q_{88} = \frac{1}{0{,}01^2} = 10000 \tag{4.12}$$

Con esto se define la matriz de ponderación Q utilizada inicialmente en el lazo de control. Finalmente, nos queda definir la matriz de ponderación R. Las señales de control del sistema corresponden al voltaje aplicado a los motores del DDMR por lo que se define 8,4 V como la amplitud de entrada máxima aceptable por los motores DC.

$$r_{11} = r_{22} = \frac{1}{8{,}4^2} = 0{,}0142 \tag{4.13}$$

Las matrices de ponderación quedan definidas según la regla de Bryson como:

$$Q = \begin{bmatrix} 0{,}0025 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0{,}0025 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 462{,}25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 462{,}25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1600 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10000 \end{bmatrix} \tag{4.14}$$

$$R = \begin{bmatrix} 0{,}0142 & 0 \\ 0 & 0{,}0142 \end{bmatrix} \tag{4.15}$$

Luego se obtiene la ganancia de realimentación utilizando el comando en MATLAB:

```
K = lqi(ss(A_bar,B_bar,C_bar,D_bar),Q,R)/K_PWM;
K1 = K(1:2,1:6);
K2 = K(1:2,7:8);
```

Donde K es la matriz de realimentación óptima para el sistema dinámico ampliado y el conjunto de parámetros escogidos para Q y R. Además, K considera la ganancia que existe entre la variable de control en el microcontrolador y el voltaje de entrada de los motores K_{PWM} proveniente del accionamiento de los motores DC mencionado en el Capítulo anterior. Para implementar el esquema de control se utiliza Simulink, para lo cual, es útil separar la matriz K en una matriz K1 que contiene las ganancias asociadas a los estados del sistema dinámico original, es decir, los elementos de la primera a la sexta columna, y en una matriz K2 contiene los elementos de las columnas séptima y octava de la matriz K equivalente a las ganancias asociadas a los estados adicionales que integran el error de seguimiento de las salidas respecto a una salida deseada.

```
K1 =

    1.0e+04 *

    0.0559    0.0029    0.1045   -0.0001    2.3196    0.0506
```

```
    0.0029    0.0559   -0.0001    0.1045    0.0506    2.3196


K2 =

    1.0e+04 *

    2.8907   -7.2267
    2.8907    7.2267
```

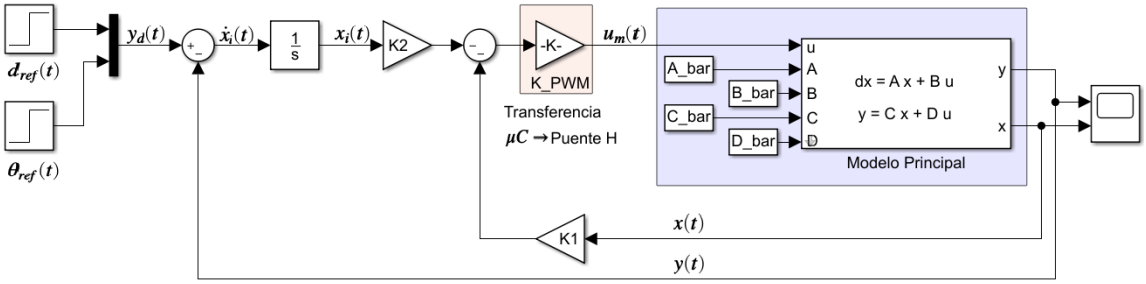


Figura 4.2: Esquema de control LQI implementado en Simulink

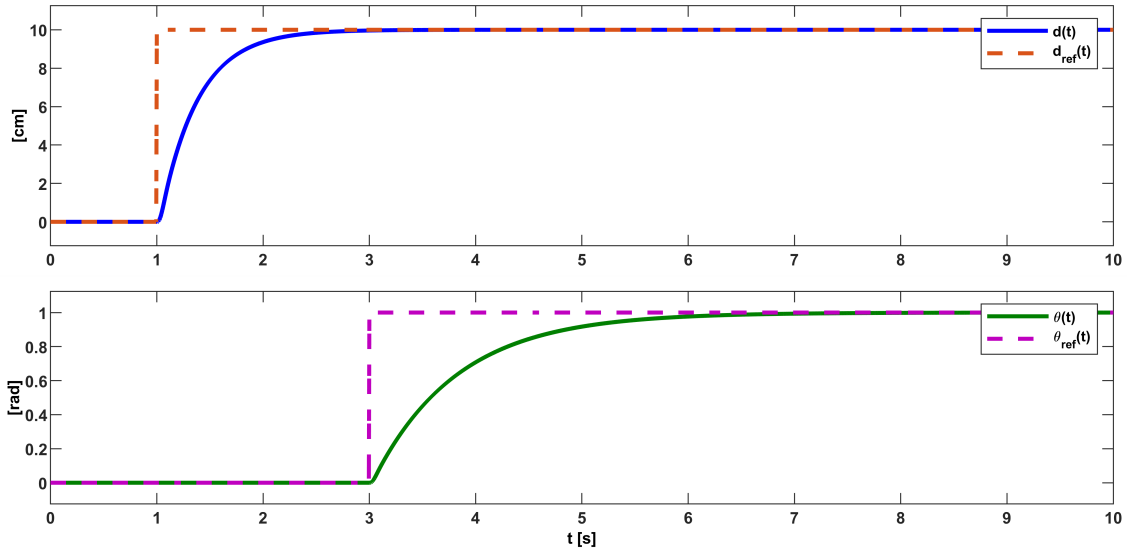


Figura 4.3: Respuesta temporal de las salidas del DDMR frente a cambios de referencia

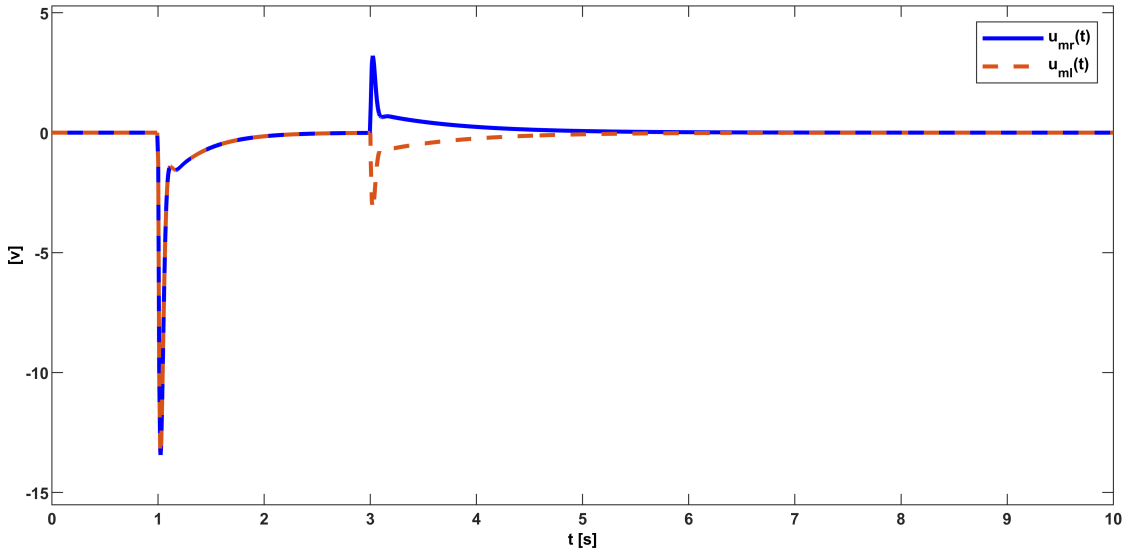


Figura 4.4: Respuesta temporal de las señales de entradas del DDMR

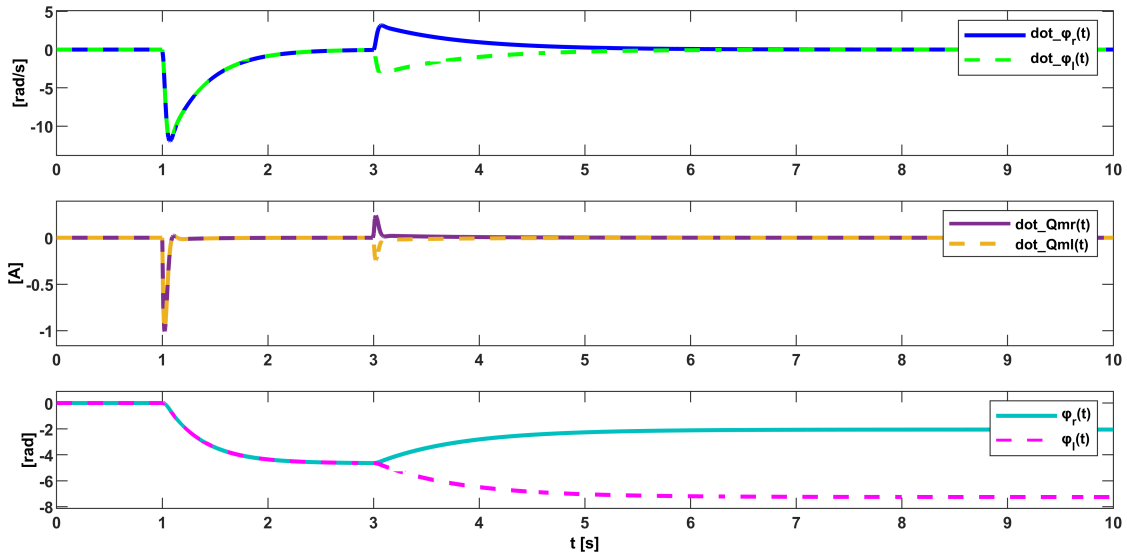


Figura 4.5: Respuesta temporal de los estados del sistema

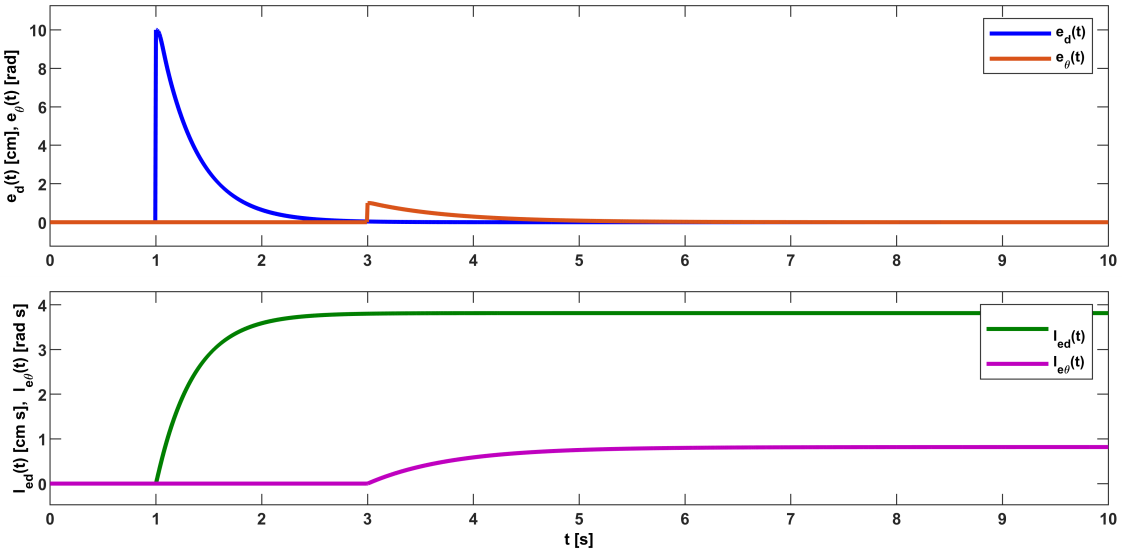


Figura 4.6: Respuesta temporal del error de seguimiento y su integral en el tiempo

La Figura 4.2 muestra la implementación en Simulink del esquema de control LQI basado en el regulador LQR, y la incorporación de acción integral para conseguir seguimiento de referencia y error en estado estacionario cero. El esquema realiza una realimentación de los estados del sistema para implementar la ley de control obtenida del problema de optimización. Las ganancias de realimentación $K1$ y $K2$ corresponden a submatrices de la matriz de ganancia de realimentación óptima K, esto permite implementar el esquema utilizando el modelo dinámico original del DDMR y construir los estados adicionales correspondientes a las integrales del error de seguimiento para las salidas, sin la necesidad de construir un nuevo modelo en espacio de estados que incorpore la dinámica del error de seguimiento.

La Figura 4.3 muestra como las salidas siguen a la referencia deseada. Inicialmente, el robot se encuentra detenido junto a un agente en reposo, con $d(0) = 0$ *cm* la distancia entre el agente y el DDMR, y posicionado sobre una trayectoria de navegación en la misma dirección que la recta tangente a la trayectoria en dicha posición, esto es, $\theta(0) = 0$ *rad*. Luego, en el instante $t = 1$ *s*, la referencia para la distancia de seguimiento se define en 10 cm, por lo que el controlador calcula la señal de control que permita llevar la distancia de seguimiento a dicho valor, llegando al valor deseado en aproximadamente dos segundos. En el instante $t = 3$ *s* se define la referencia para el ángulo formado entre el eje del DDMR y la recta tangente a la curva, en 1 radian. Debido a que para efectos de la simulación la trayectoria de navegación es uniforme, el robot debe realizar un giro sobre su eje de rotación equivalente a un radián, lo cual se alcanza en aproximadamente 3 s.

Los esfuerzos de control empleados para mover el DDMR se muestran en la Figura 4.4. Puesto que la referencia para la distancia de seguimiento se define en 10 *cm*, el robot debe retroceder para lograr posicionarse en dicha referencia, por lo cual, el controlador aplica voltajes negativos a la entrada de los motores DC del DDMR, alcanzando rápidamente un valor cercano a los -13 *V* el cual tiende a cero cuando la salida es más cercana a la referencia. Para lograr hacer girar el robot en 1 radian (sentido antihorario) el controlador aplica señales de control de la misma magnitud, pero distinto signo, siendo la señal de control asociada al motor derecho una señal de voltaje positiva, y una señal negativa para la entrada asociada al motor izquierdo. También se observa que la magnitud de las señales de control poseen magnitudes mayores cuando se trata de controlar la distancia de seguimiento, respecto del control de la orientación del DDMR, lo cual, es posible respaldar teniendo en cuenta que la orientación del robot tarda más tiempo en alcanzar la referencia deseada. Además, la actuación puede llegar a superar el valor máximo de voltaje que pueden entregar las baterías utilizadas por el robot. Lo anterior nos sugiere cambiar las matrices de ponderación escogidas para lograr una actuación que esté dentro de las capacidades del robot, y hacer más rápida la dinámica de la orientación respecto de la dinámica de la distancia de seguimiento, lo que permite desacoplar las salidas y

que estas no se afecten mutuamente de forma significativa en la implementación real.

A partir de la Figura 4.5 se observa la dinámica de los estados del sistema. El comportamiento de las velocidades angulares de las ruedas del robot es similar al comportamiento de las corrientes de armadura en los motores, siendo esta última la dinámica más rápida del sistema. Dichos estados tienden a cero al alcanzar las referencias deseadas, puesto que para efectos de simulación el agente a seguir se encuentra en reposo. También es posible ver el comportamiento de las posiciones angulares de las ruedas del DDMR, donde para conseguir la referencia de distancia es evidente que ambas ruedas se mueven en el mismo sentido provocando que el robot se desplace hacia atrás. Luego el movimiento de las ruedas es en sentido opuesto para provocar un giro del robot en sentido antihorario, cuando se alcanza la referencia deseada las posiciones angulares de las ruedas quedan en una posición final estable distinta de la posición inicial.

Por último, en la Figura 4.6 se aprecia el error de seguimiento de las salidas y los estados adicionales correspondientes a la integral de dicho error. El error de seguimiento aumenta cuando hay cambios de referencia, lo cual, es de esperarse, puesto que el error depende directamente de la referencia. A medida que el robot se acerca a la referencia se puede ver como el error de seguimiento tiende a cero.

Ahora bien, si se escogen las matrices de ponderación:

$$Q = \begin{bmatrix} 10 & 0{,}5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0{,}5 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 462{,}25 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 462{,}25 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1600 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 5e5 \end{bmatrix} \tag{4.16}$$

$$R = \begin{bmatrix} 5 & 0{,}5 \\ 0{,}5 & 5 \end{bmatrix} \tag{4.17}$$

se obtiene la ganancia de realimentación:

```
K1 =

    1.0e+03 *

    0.1884   -0.0102    0.0174   -0.0021    2.1006   -0.4880
   -0.0102    0.1884   -0.0021    0.0174   -0.4880    2.1006


K2 =

    1.0e+04 *

    0.1469   -2.8705
    0.1469    2.8705
```

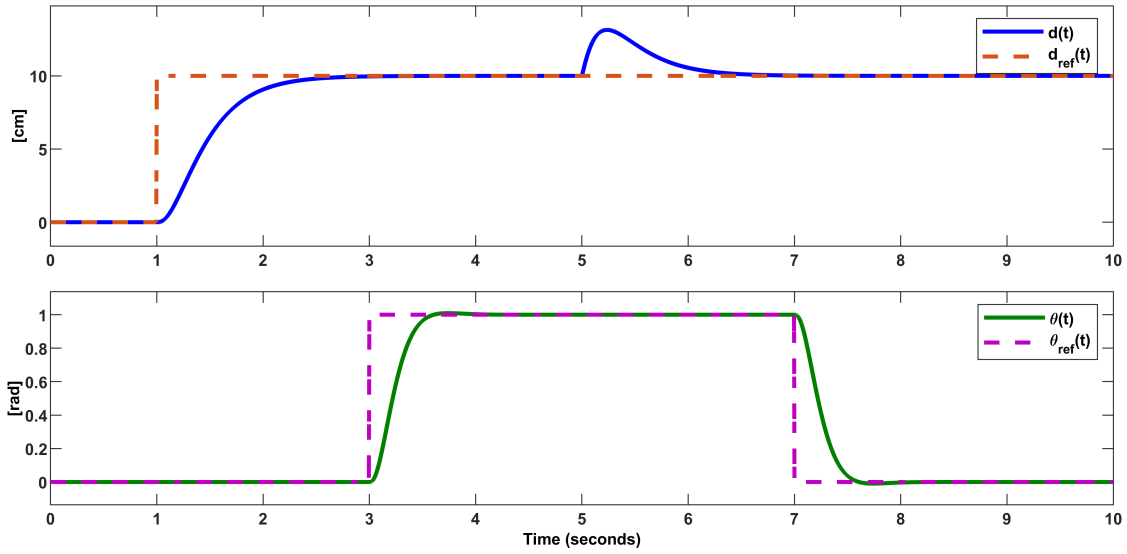


Figura 4.7: Respuesta temporal de las salidas del DDMR frente a cambios de referencia

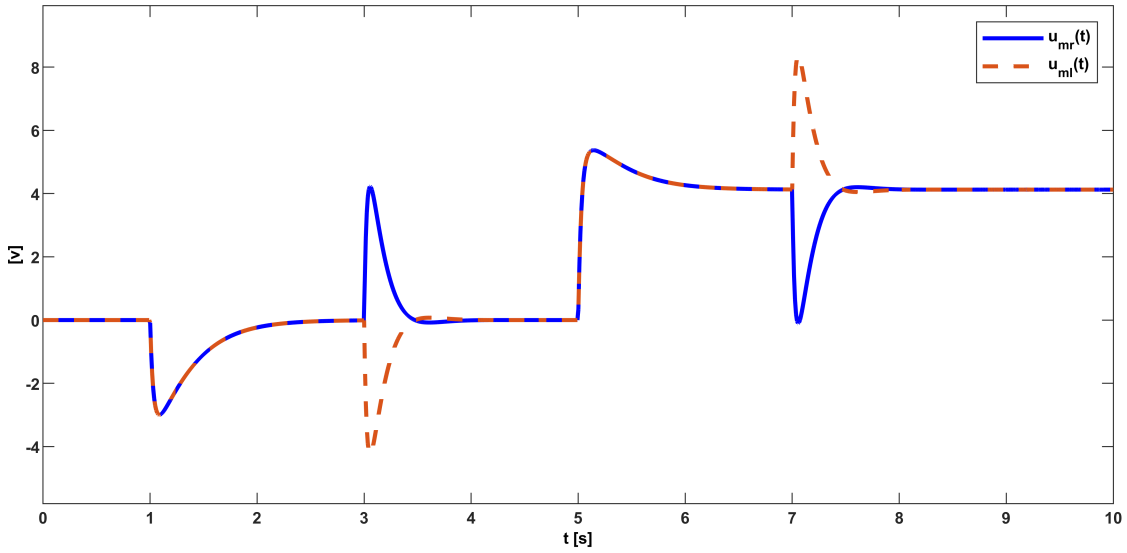


Figura 4.8: Respuesta temporal de las señales de entradas del DDMR

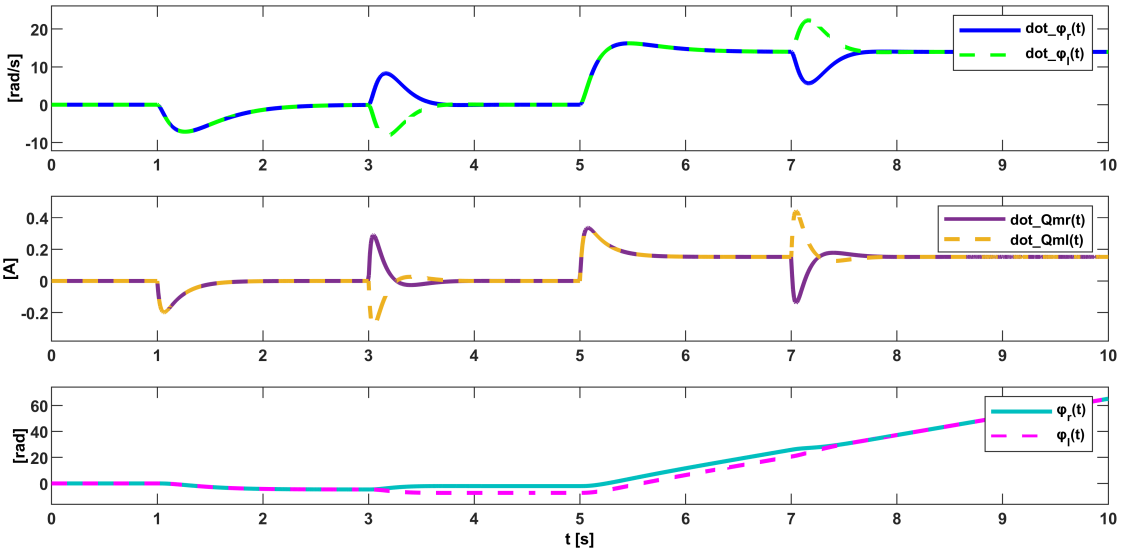


Figura 4.9: Respuesta temporal de los estados del sistema

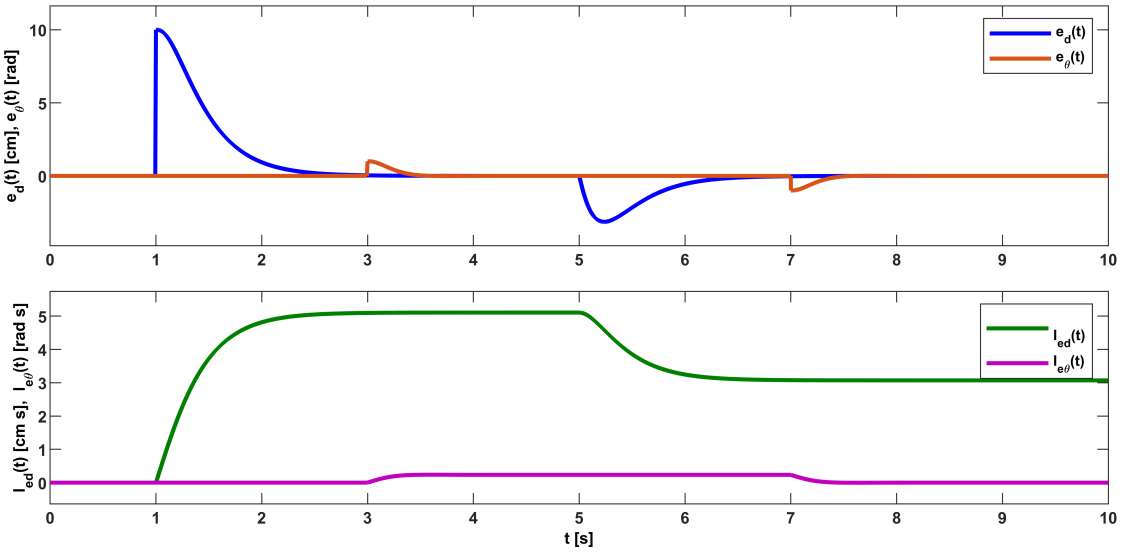


Figura 4.10: Respuesta temporal del error de seguimiento y su integral en el tiempo

A partir de la Figura 4.7 se observa que de la orientación ahora es más rápida que la dinámica del seguimiento de distancia. Además, en el instante $t = 5\ s$ se introduce una perturbación en la salida correspondiente a la distancia de seguimiento a través de una rampa de pendiente 30, la cual, permite emular el movimiento del agente inicialmente en reposo, a desplazarse a 30 cm/s, esto provoca que la distancia de seguimiento aumente respecto de la referencia, pero rápidamente es compensado por el controlador para nuevamente cumplir con la referencia de seguimiento definida pero ahora desplazándose a velocidad constante distinta de cero. Al instante $t = 7\ s$ se fija la referencia para la orientación en 0 *rad* para orientar nuevamente el DDMR respecto de la trayectoria de navegación.

En la Figura 4.8 se puede ver que las amplitudes para las señales de control se encuentran en el rango de operación del robot y también se puede observar como la actuación queda en un valor constante distinto de cero para lograr mantener la referencia de distancia cuando el otro agente está en movimiento. Algo similar

se puede observar en los estados del sistema a través de la Figura 4.9, en primer lugar se observa como la dinámica de la velocidad se ha vuelto más rápida, pero aún sigue siendo la dinámica dominante. A diferencia del ejemplo anterior, ahora la velocidad final del robot tiende a la velocidad del otro agente, producto del esfuerzo de control que permite mantener la referencia. También se puede observar que la posición angular de las ruedas del DDMR aumenta constantemente, lo que nos siguiere que el robot se desplaza de forma uniforme.

En la Figura 4.10 se aprecia el aumento de los errores de seguimiento, y además el error que provoca la perturbación de salida. También se comprueba a través de las señales de error, que el controlador permite llevar las salidas a una referencia deseada, aun cuando el agente externo se encuentra en movimiento.

4.2. Esquema de Control basado en controladores PID

A fin de complementar el diseño de controladores y entregar un esquema alternativo que otorgue mayor flexibilidad a la arquitectura de control, aprovechando el modelo complementario descrito en el Capítulo anterior, se propone diseñar un esquema de control basado en controladores PID, para las variables de velocidad y orientación del DDMR, mientras que la distancia se controla a través de un controlador PID conectado en cascada al lazo de velocidad, tal como se muestra en la Figura 4.11.

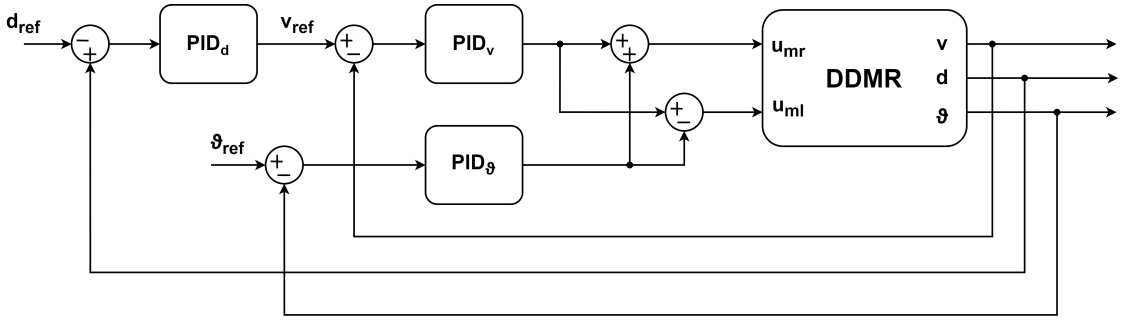


Figura 4.11: Esquema de control propuesto

El esquema propone controladores del tipo PID, puesto que es un tipo de controlador clásico y ampliamente estudiado y utilizado en el mundo, a raíz de lo anterior, es posible encontrar una serie de herramientas que nos ayudan a diseñar sus parámetros, analizar su desempeño, como también librerías que nos permiten implementarlos fácilmente. El diseño de los controladores PID se realiza considerando una representación en función de transferencia del modelo complementario descrito por las ecuaciones (3.72) y (3.73). Lo anterior se puede obtener aplicando la transformada de Laplace al modelo en variable de estados para obtener la representación deseada. De manera equivalente, es posible usar el software MATLAB para convertir el modelo dinámico descrito en espacio de estados a una representación en función de transferencia, usando el comando *zpk(sys)*. Este comando se puede aplicar a sistemas del tipo MIMO, entregando una función de transferencia SISO por cada relación entrada-salida que describe al sistema, además entrega de forma explícita los ceros, polos y la ganancia del sistema dinámico. También se puede emplear el comando *minreal(sys)*, el cual, realiza una cancelación polo-cero en caso de ser necesario. A continuación se obtiene la representación del modelo complementario en forma de función de transferencia.

```
sys_c = ss(A_c, B_c, C_c, D_c);
G = minreal(zpk(sys_c));
```

Donde la variable *sys_c* corresponde a la representación en espacio de estado del modelo complementario descrito por las matrices A_c, B_c, C_c y D_c. En la variable G se asigna la representación en función de transferencia del modelo complementario.

```
G =

   From input 1 to output...
           1.2002e+05
    1:  ------------------
       (s+4592) (s+7.187)

           -1.2002e+05
    2:  --------------------
       s (s+4592) (s+7.187)

              24962
    3:  --------------------
       s (s+4591) (s+8.372)

   From input 2 to output...
           1.2002e+05
    1:  ------------------
       (s+4592) (s+7.187)

           -1.2002e+05
    2:  --------------------
       s (s+4592) (s+7.187)

              -24962
    3:  --------------------
       s (s+4591) (s+8.372)

Continuous-time zero/pole/gain model.
```

Un primer aspecto que vale la pena señalar es que los polos, correspondientes a las raíces del denominador de cada función de transferencia, son iguales a los derivados del modelo en espacio de estados en el Capítulo anterior, esto ya que ambas representaciones son equivalentes. Un segundo aspecto a señalar es la igualdad entre las funciones de transferencias obtenidas para ambas entradas del sistema, lo cual, nos indica que este sistema posee una simetría en sus entradas debido a la geometría del robot diferencial y los parámetros de los componentes utilizados.

Según el modelo complementario, la entrada 1 y 2 corresponden a las señales de control $u_{mr}(t)$ y $u_{ml}(t)$, mientras que las salidas 1, 2 y 3 corresponden a $v(t)$, $d(t)$ y $\theta(t)$ respectivamente. De esta forma se puede escribir el modelo en función de transferencia de forma matricial según:

$$\begin{aligned} Y(s) &= G(s)U(s) \\ \begin{bmatrix} V(s) \\ D(s) \\ \Theta(s) \end{bmatrix} &= \begin{bmatrix} g_{11}(s) & g_{12}(s) \\ g_{21}(s) & g_{22}(s) \\ g_{31}(s) & g_{32}(s) \end{bmatrix} \begin{bmatrix} U_{mr}(s) \\ U_{ml}(s) \end{bmatrix} \end{aligned} \tag{4.18}$$

con

$$g_{11}(s) = g_{12}(s) = \frac{1{,}2002e5}{(s+4592)(s+7{,}187)} \tag{4.19}$$

$$g_{21}(s) = g_{22}(s) = \frac{-1{,}2002e5}{s(s+4592)(s+7{,}187)} \tag{4.20}$$

$$g_{31}(s) = -g_{32}(s) = \frac{24962}{s(s+4591)(s+8{,}372)} \tag{4.21}$$

A partir de las expresiones (4.19) y (4.20) se obtiene la relación

$$g_{21}(s) = -\frac{g_{11}(s)}{s} \tag{4.22}$$

Así, se pueden expresar la funciones de transferencia de la forma:

$$\frac{V(s)}{U_{mr}(s) + U_{ml}(s)} = g_{11}(s) \tag{4.23}$$

$$\frac{D(s)}{U_{mr}(s) + U_{ml}(s)} = g_{21}(s) \tag{4.24}$$

$$\frac{\Theta(s)}{U_{mr}(s) - U_{ml}(s)} = g_{31}(s) \tag{4.25}$$

Además, se considera que la velocidad de ambos motores DC, $u_{mr}(t)$, $u_{ml}(t)$, se controla a través de una señal PWM proveniente desde el microcontrolador del DDMR. Esto se puede representar matemáticamente como:

$$\begin{bmatrix} u_{mr}(t) \\ u_{ml}(t) \end{bmatrix} = K_{PWM} \begin{bmatrix} pwm\_r(t) \\ pwm\_l(t) \end{bmatrix} \tag{4.26}$$

donde $pwm\_r(t)$ y $pwm\_l(t)$ son variables dentro del microcontrolador que definen el ciclo de trabajo de la señal PWM que disparan cada uno de los canales del módulo Puente H para accionar los motores DC del robot y K_{PWM} la ganancia que relaciona dichas señales con la entrada de los motores $u_{mr}(t)$ y $u_{ml}(t)$. Con esto se puede representar las funciones de trasferencia (4.23), (4.25) y (4.24) en función de las variables provenientes del microcontrolador.

$$\frac{V(s)}{pwm\_r(s) + pwm\_l(s)} = K_{PWM} \cdot g_{11}(s) \tag{4.27}$$

$$\frac{D(s)}{pwm\_r(s) + pwm\_l(s)} = K_{PWM} \cdot g_{21}(s) \tag{4.28}$$

$$\frac{\Theta(s)}{pwm\_r(s) - pwm\_l(s)} = K_{PWM} \cdot g_{31}(s) \tag{4.29}$$

Luego se definen las entradas auxiliares:

$$U_v(s) = \frac{pwm\_r(s) + pwm\_l(s)}{2} \tag{4.30}$$

$$U_\theta(s) = \frac{pwm\_r(s) - pwm\_l(s)}{2} \tag{4.31}$$

donde $U_v(s)$ y $U_\theta(s)$ corresponden a las salidas de los controladores de velocidad y orientación respectivamente. A partir de estas relaciones se obtiene una expresión para las variables que controlan el accionamiento de los motores del DDMR:

$$pwm_r(s) = U_v(s) + U_\theta(s) \tag{4.32}$$

$$pwm_l(s) = U_v(s) - U_\theta(s) \tag{4.33}$$

Finalmente, se obtienen las funciones de transferencia que relacionan las salidas de interés con las señales provenientes de cada controlador SISO.

$$G1(s) = \frac{V(s)}{U_v(s)} = 2K_{PWM} \cdot g_{11}(s) \tag{4.34}$$

$$G2(s) = \frac{D(s)}{V(s)} = \frac{-1}{s} \tag{4.35}$$

$$G3(s) = \frac{\Theta(s)}{U_\theta(s)} = 2K_{PWM} \cdot g_{31}(s) \tag{4.36}$$

Un controlador PID utiliza como entrada el error entre una salida deseada y la salida real de un sistema, y establece una ley de control que es proporcional al error, a su integral y su derivada, en el dominio temporal. Este puede ser descrito en el dominio de Laplace según:

$$\begin{aligned} C(s) &= \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s \\ &= \frac{K_d s^2 + K_p s + K_i}{s} \end{aligned} \tag{4.37}$$

donde los parámetros K_p, K_i y K_d permiten definir una dinámica deseada para el sistema en lazo cerrado. Esta dinámica está determinada por las raíces del polinomio de lazo cerrado $Acl(s)$, las cuales corresponden a los polos del sistema en lazo cerrado.

El $Acl(s)$ para un esquema de control SISO con realimentación negativa se puede escribir como:

$$Acl(s) = A(s)L(s) + B(s)P(s) \tag{4.38}$$

Siendo $A(s)$, $B(s)$ el denominador y numerador de la función de transferencia del modelo del sistema, y $L(s)$, $P(s)$ el denominador y numerador de la función de transferencia del controlador. De esta forma se obtiene la dinámica de lazo cerrado del sistema en función de los parámetros del controlador, y así escoger las ganancias que garanticen un desempeño deseado en lazo cerrado.

Es importante aclarar que el sistema en lazo cerrado define un lugar geométrico para las raíces del $Acl(s)$, por lo que la dinámica deseada debe pertenecer a dicho lugar geométrico. Además de la dinámica de los polos del sistema, resulta necesario escoger los parámetros del controlador que garanticen que el esfuerzo de control se encuentre dentro de los rangos de operación del robot, como también aspectos propios de la respuesta temporal de salida como son el sobre-impulso y sub-impulso máximos aceptables. Por lo que el diseño de los parámetros del controlador va más allá de definir una dinámica de los polos en el lazo cerrado.

Para diseñar los parámetros de los controladores propuestos, se utiliza el comando $pidTuner(sys, type)$ en MATLAB, el cual, inicia una herramienta de diseño de controladores PID donde es posible seleccionar el tipo de controlador, y ajustar su desempeño observando de forma gráfica las respuestas tanto en el dominio temporal como en frecuencia, de las principales señales del lazo de control, tales como, el seguimiento de referencia, el esfuerzo de control, rechazo a perturbaciones de entrada y de salida, respuesta del sistema en lazo abierto, etc.

En primer lugar, se diseña el controlador de velocidad utilizando el modelo definido en (4.34).

```
G1 = G(1,1)*2*K_PWM;
pidTuner(G1,'pi')
```

Definiendo el desempeño deseado en la herramienta de diseño se obtienen los siguientes parámetros:

Tabla 4.1: Parámetros de diseño del controlador de velocidad

Parámetros del controlador	
Kp	20.2912
Ki	145.2779
Kd	0
Desempeño y Robustez	
Tiempo de Levantamiento	0.253 s
Tiempo de Asentamiento	0.452 s
Sobre-impulso	0 %
Valor Pico	0.999
Margen de Ganancia	Inf dB @ Inf rad/s
Margen de Fase	90 deg @ 8.7 rad/s
Estabilidad de lazo cerrado	Estable

luego el controlador diseñado se exporta al workspace de MATLAB en la variable C1.

```
C1 =

           1
Kp + Ki * ---
           s

with Kp = 20.3, Ki = 145

Continuous-time PI controller in parallel form.
```

En segundo lugar, se diseña el controlador de distancia C2(s), el cual, se conecta en cascada al lazo de control de velocidad, por lo que este controlador regula la distancia del DDMR respecto a otro agente modificando la referencia de velocidad. En la medida que el lazo interno (lazo de velocidad) sea más rápido que el lazo externo (lazo de distancia) la dinámica que ve el controlador externo se aproxima a un integrador G2 (4.35). De todas formas, el controlador de distancia está conectado a un sistema en lazo cerrado definido por $G4(s)$ tal como se muestra en la Figura 4.12.

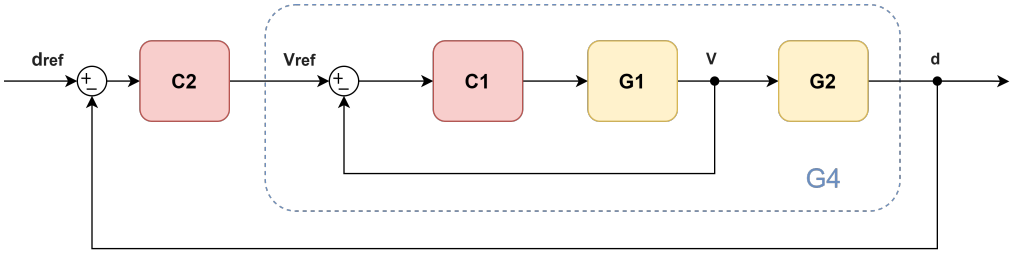


Figura 4.12: Ilustración de sistemas conectados en cascada

Luego, el controlador de distancia C2(s) se puede diseñar a partir de la función de transferencia del sistema G4(s):

$$G4(s) = \left(\frac{G1(s)C1(s)}{1 + G1(s)C1(s)}\right)G2(s) \tag{4.39}$$

```
G2 = tf([0 -1],[1 0]);
G4 = minreal((G1*C1/(1+G1*C1))*G2);
pidTuner(G4,'pid')
```

Definiendo el desempeño deseado en la herramienta de diseño se obtienen los siguientes parámetros:

Tabla 4.2: Parámetros de diseño del controlador de distancia

Parámetros del controlador	
Kp	-8.0013
Ki	-5.6025
Kd	-1.0077
Desempeño y Robustez	
Tiempo de Levantamiento	0.245 s
Tiempo de Asentamiento	2.4 s
Sobre-impulso	6.56 %
Valor Pico	1.07
Margen de Ganancia	-Inf dB @ 0 rad/s
Margen de Fase	90 deg @ 8 rad/s
Estabilidad de lazo cerrado	Estable

Por último, el controlador diseñado se exporta al workspace de MATLAB en la variable C2.

```
C2 =

             1
  Kp + Ki * --- + Kd * s
             s

  with Kp = -8, Ki = -5.6, Kd = -1.01

Continuous-time PID controller in parallel form.
```

En tercer y último lugar, se diseña el controlador de orientación utilizando el modelo definido en (4.36).

```
G3 = G(3,1)*2*K_PWM;
pidTuner(G3,'pid')
```

Definiendo el desempeño deseado en la herramienta de diseño se obtienen los siguientes parámetros:

Tabla 4.3: Parámetros de diseño del controlador de orientación

Parámetros del controlador	
Kp	2755.3558
Ki	6250.5618
Kd	303.6521
Desempeño y Robustez	
Tiempo de Levantamiento	0.0723 s
Tiempo de Asentamiento	0.765 s
Sobre-impulso	6.34 %
Valor Pico	1.06
Margen de Ganancia	-Inf dB @ 0 rad/s
Margen de Fase	87.8 deg @ 26.6 rad/s
Estabilidad de lazo cerrado	Estable

Por último, el controlador diseñado se exporta al workspace de MATLAB en la variable C3.

```
C3 =

             1
  Kp + Ki * --- + Kd * s
             s

  with Kp = 2.76e+03, Ki = 6.25e+03, Kd = 304

Continuous-time PID controller in parallel form.
```

4.2.1. Simulación

Para la simulación se implementa el esquema de control propuesto en la Figura 4.11 en Simulink, el esquema se muestra en la Figura 4.13.

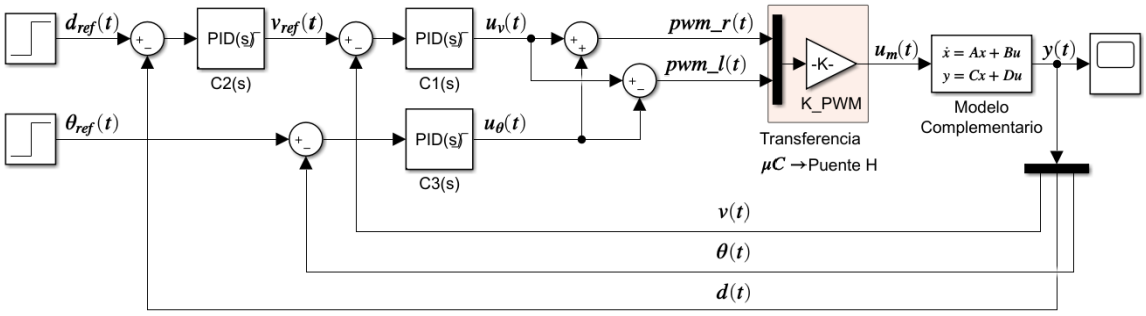


Figura 4.13: Esquema de control PID implementado en Simulink

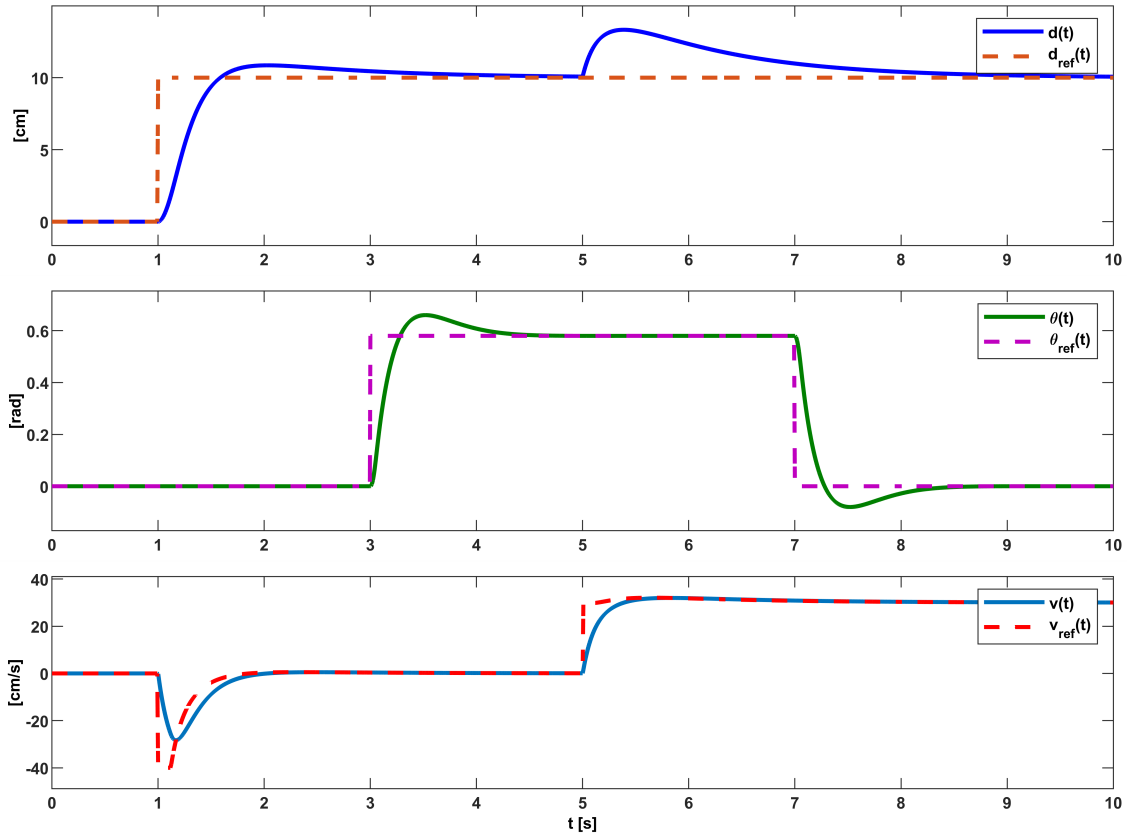


Figura 4.14: Respuesta tempo

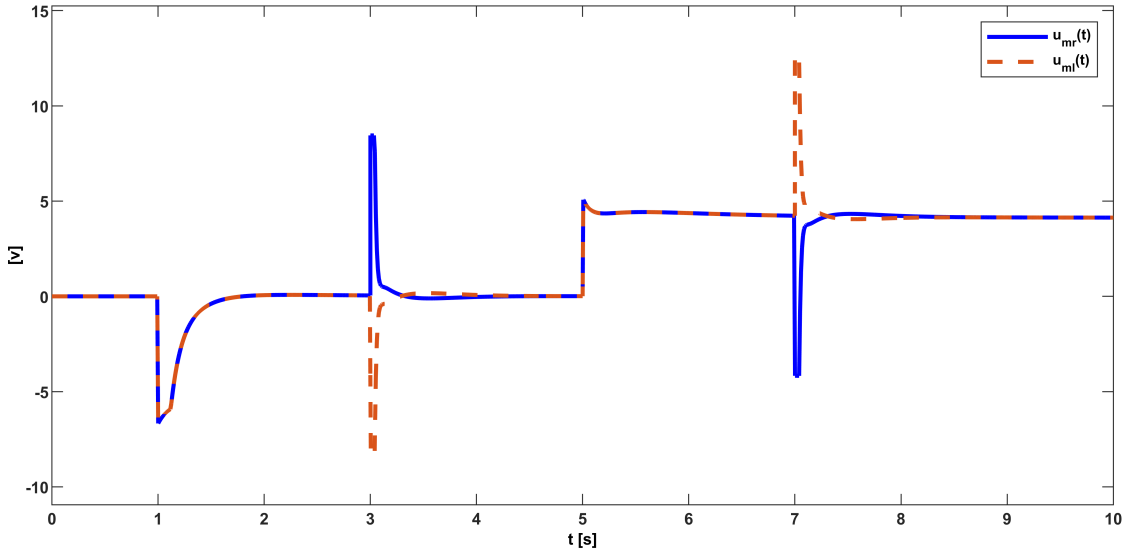


Figura 4.15: Esquema de control PID implementado en Simulink

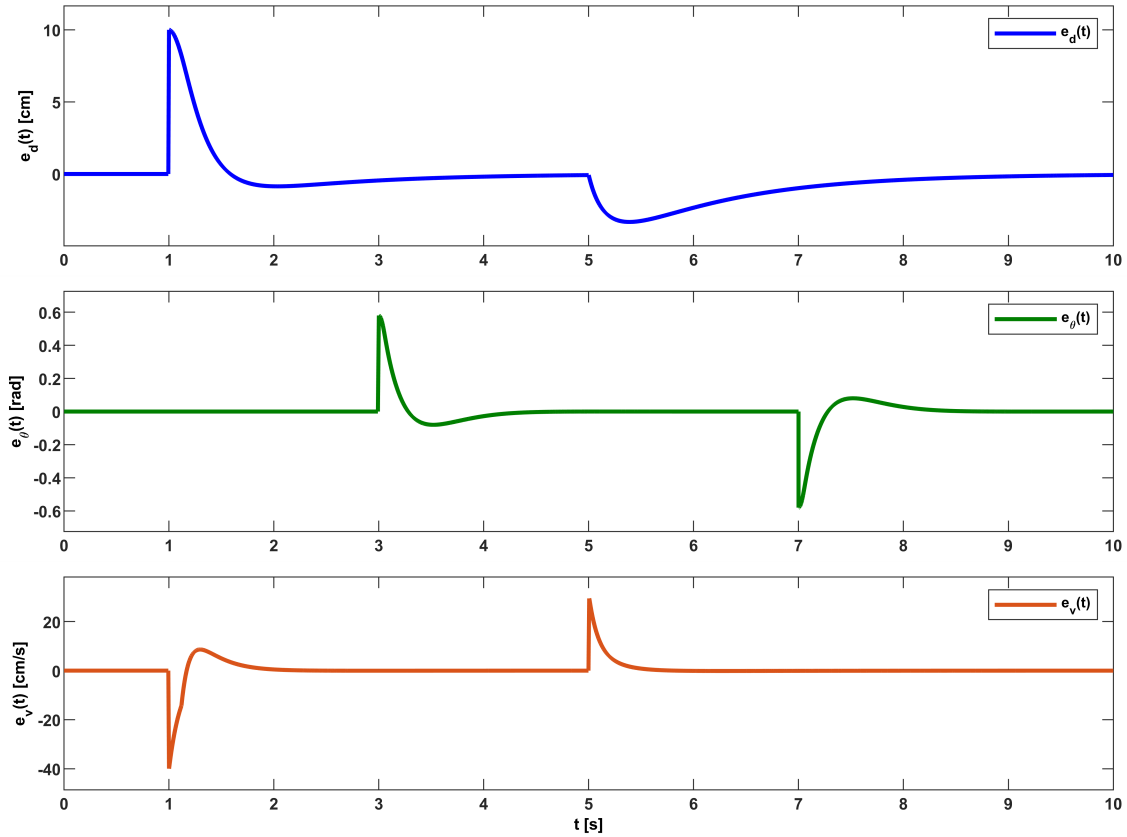


Figura 4.16: Esquema de control PID implementado en Simulink

En la Figura 4.14 se aprecia como las salidas del DDMR son capaces de seguir las referencias impuestas, en el instante $t = 1\ s$ la referencia de distancia entre el robot y otro agente en reposo se define en 10 *cm*, en ese instante el controlador de distancia modifica la referencia de velocidad para que el robot

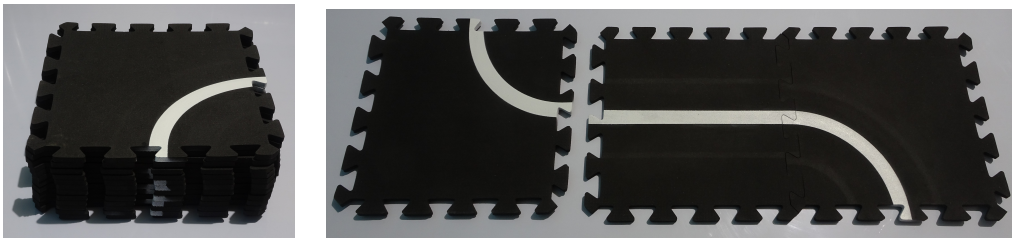
retroceda y logre posicionarse a la distancia deseada. Luego en el instante $t = 3\ s$ se modifica la referencia de orientación del robot a 1 *rad*, siendo el controlador de orientación capaz de generar las acciones de control para llevar la salida al valor de la referencia. En el instante $t = 5\ s$ se introduce una perturbación en la distancia de salida a través de una señal tipo rampa de pendiente 30. Esto representa que el agente externo inicialmente en reposo comienza a desplazarse con velocidad constante de 30 cm/s, siendo el controlador de distancia capaz de realizar las acciones de control para seguir la referencia de distancia dé 10 *cm*. Por último, en el instante $t = 7\ s$ se define la referencia de orientación en 0 *rad* siendo el controlador de orientación capaz de posicionar el robot en la orientación deseada respecto de la trayectoria de navegación, aun cuando este se desplaza a una velocidad constante dé 30 cm/s. En la Figura 4.15 se puede ver las señales de control que son aplicadas a cada uno de los motores del DDMR con el objetivo de seguir la referencia impuesta y llevar el error de seguimiento a cero, tal como se muestra en la Figura 4.16, donde se aprecia que independientemente de los cambios de referencia y perturbaciones ingresadas al sistema el error de seguimiento finalmente tiende a cero en un periodo de tiempo finito.

5 | IMPLEMENTACIÓN DE LA PLATAFORMA EXPERIMENTAL

La implementación de la plataforma experimental es motivada principalmente por la necesidad de realizar investigación aplicada referente al control de flotas de robots móviles. Cada uno de los componentes que integran la plataforma fue diseñado con un propósito específico, los tipos de componentes en esta plataforma abarcan, estructuras mecánicas, componentes electrónicos, hasta funciones matemáticas transcritas en algún lenguaje de programación. El conjunto de componentes va conformando estructuras más complejas, las que a su vez atienden a otros propósitos, como son, otorgar la capacidad a cada robot de detectar una trayectoria definida en una superficie de navegación, disponer de las mediciones de posición y velocidad angular de las ruedas del los robots, que los robots puedan navegar sobre una trayectoria de forma coordinada, disponer de la capacidad de monitoreo de las variables de interés de cada robot, de forma remota, posibilitar la capacidad de comunicación entre robots, etc.

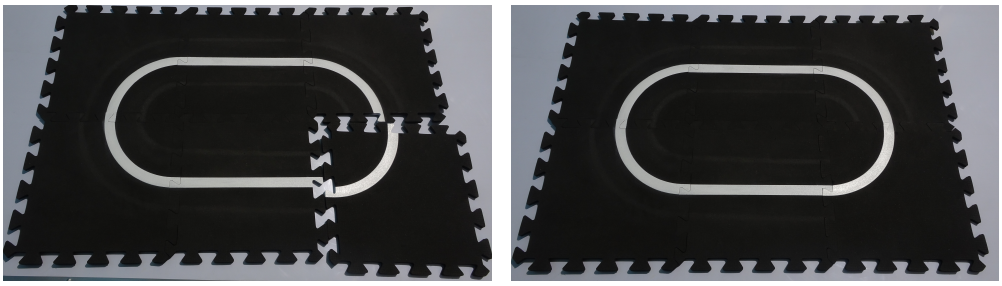
5.1. Superficie y trayectoria

De acuerdo a lo propuesto en el Capítulo 2, dedicado al diseño de la plataforma experimental, la superficie de navegación se implementa en un material de goma EVA modular. Cada módulo corresponde a una sección de una trayectoria, la cual, es definida utilizando materiales como: pintura en spray; papel; cinta adhesiva. Los resultados se pueden apreciar en las figuras 5.1 y 5.2. La característica modular de la superficie, otorga la capacidad de construir escenarios de navegación variados, y facilita su traslado, almacenado y su montaje en espacios reducidos.



(a) Módulos apilados **(b)** Ejemplo de ensamble de la superficie

Figura 5.1: Resultados de la implementación de la superficie y trayectoria modular

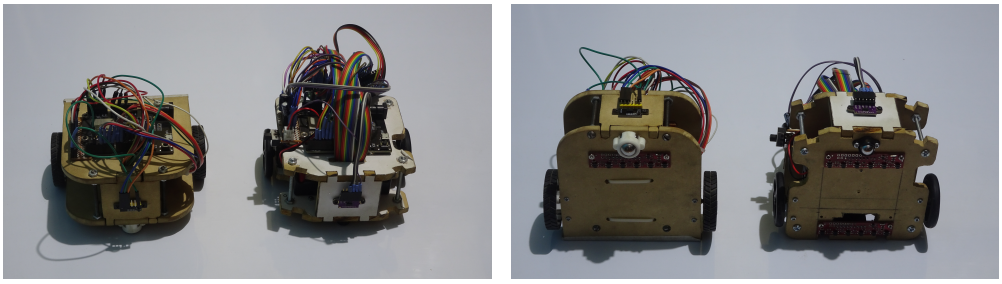


(a) Ensamble de trayectoria cerrada **(b)** Trayectoria de navegación cerrada

Figura 5.2: Ejemplo de ensamble de una trayectoria de navegación cerrada

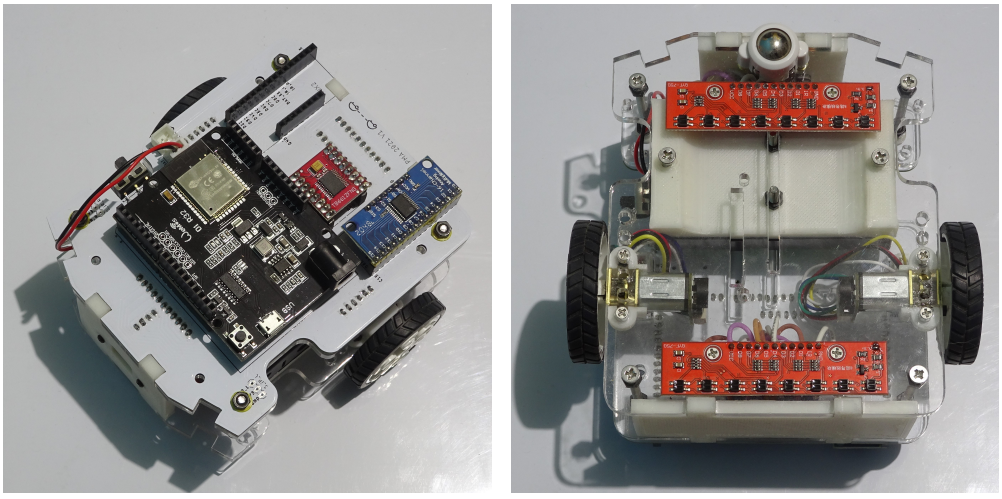
5.2. Robot móvil diferencial

La implementación final de los robots móviles de tracción diferencial diseñados en este trabajo, queda definida a partir de un proceso de diseño iterativo, del cual, surgen dos versiones prototipo, previas al desarrollo de la versión final presentada en este trabajo. En la Figura 5.3, se puede apreciar los prototipos de robots implementados. Los resultados obtenidos de los prototipos permiten modificar el diseño, y mejorar aspectos funcionales y estéticos del robot, los cuales, se pueden observar en la figura 5.4



(a) Prototipos del DDMR - Vista superior **(b)** Prototipos del DDMR - Vista inferior

Figura 5.3: Versiones prototipo del DDMR



(a) Vista superior del DDMR **(b)** Vista inferior del DDMR

Figura 5.4: Resultados de la implementación del diseño propuesto para el DDMR

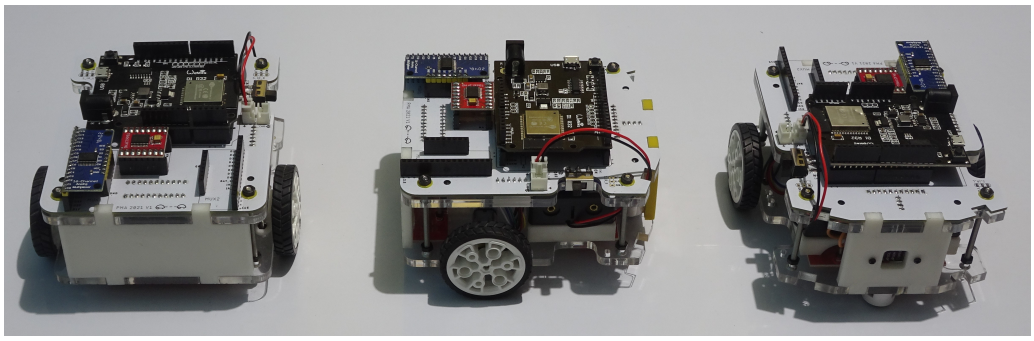


Figura 5.5: Perspectivas del DDMR, utilizando 3 robots

La Figura 5.5 muestra 3 robots implementados según el diseño propuesto. Están ubicados en distintos ángulos para mostrar el resultado obtenido. Visualmente, los robots lucen idénticos, lo cual, permite garantizar la capacidad reproducible del DDMR.

5.3. Sensores

Tal como se indicó al inicio de este Capítulo, es necesario otorgar capacidades a los robots de la plataforma experimental. En esta Sección se describen las funciones matemáticas que permiten procesar las señales provenientes de los sensores del DDMR, y convertirlas en variables que representen las magnitudes físicas de interés, junto con el respectivo pseudocódigo que ilustra la implementación de estas funciones en el MCU del robot.

5.3.1. Sensor de orientación

Como fue descrito en el Capítulo de diseño, cada arreglo emplea 8 sensores de reflectancia QRE1113 [6], los cuales, entregan un valor analógico entre 0 V y 3.3 V, en función de la luz que se refleja hacia el sensor. A estos valores se accede a través del multiplexor analógico CD74HC4067 [2], conectado al MCU ESP32 [5], este último, digitaliza dichos valores y los cuantiza utilizando 12 bits, es decir, las lecturas en el MCU toman valores entre 0 y 4095.

En primer lugar, se requiere poder leer cada uno de los sensores infrarrojos, por lo cual, se necesita acceder a cada una de las entradas del multiplexor variando sus señales de control *S0*, *S1*, *S2* y *S3*, las cuales, están conectadas al MCU a través de los pines GPIO25, GPIO26, GPIO2 y GPIO4 respectivamente. La medición está disponible a través del pin *SIG* del multiplexor, conectado al pin de lectura análoga GPIO39. La Tabla 5.1 muestra la relación entre la secuencia de control y el sensor medido.

Tabla 5.1: Relación entre la secuencia de control y el sensor leído por el MCU

Lectura GPIO39		Secuencia de control MCU→ MUX1			
Arreglo	Sensor	GPIO4	GPIO2	GPIO26	GPIO25
Frontal	D1	L	L	L	L
	D2	L	L	L	H
	D3	L	L	H	L
	D4	L	L	H	H
	D5	L	H	L	L
	D6	L	H	L	H
	D7	L	H	H	L
	D8	L	H	H	H
Posterior	D1	H	L	L	L
	D2	H	L	L	H
	D3	H	L	H	L
	D4	H	L	H	H
	D5	H	H	L	L
	D6	H	H	L	H
	D7	H	H	H	L
	D8	H	H	H	H

Algoritmo 1 Lectura de sensores infrarrojos

1: Seleccionar una secuencia de control (secuencia binaria de 4 bits)
2: Asignar la secuencia a los pines de control GPIO25; GPIO26; GPIO2; GPIO4
3: Leer el pin analógico GPIO39

El algoritmo 1 muestra los pasos a seguir para acceder a la medición de cualquiera de los sensores infrarrojos del DDMR, esta rutina se puede implementar dentro de un ciclo recursivo para así optimizar la lectura de los arreglos cuando sea necesario.

El porcentaje de luz que se refleja hacia el sensor depende en gran medida del color del material sobre el cual se hace incidir la luz infrarroja. Al ser la superficie de color negro y la trayectoria de navegación de color blanco, se otorgan dos zonas que son fáciles de identificar utilizando el arreglo de sensores infrarrojo, puesto que el color negro absorbe la luz en mayor porcentaje que el color blanco. Para discriminar la trayectoria de navegación del resto de la superficie en función de las mediciones de los sensores, se debe tener en cuenta las condiciones lumínicas del entorno donde se instale la plataforma experimental, para lo cual, se diseña una rutina de calibración que permite a cada sensor identificar las zonas en la superficie de mayor y menor reflexión de la luz, de esta manera, mitigar el efecto que pueda producir el entorno. El objetivo de esta rutina es llevar las mediciones de los sensores a una escala común, independiente de las condiciones lumínicas del entorno, donde los máximos y los mínimos de esta escala dependen de los colores de la superficie y trayectoria de navegación.

La Figura 5.6 muestra la respuesta ideal del sensor infrarrojo frente a variaciones en el porcentaje de luz que recibe el fototransistor, además se muestra el valor cuantizado a través del conversor ADC del MCU. También se ilustra un ejemplo de respuesta real de los sensores infrarrojos, donde a pesar de tener un comportamiento lineal, los valores mínimos y máximos medidos para los colores blanco y negro difieren de la respuesta ideal.

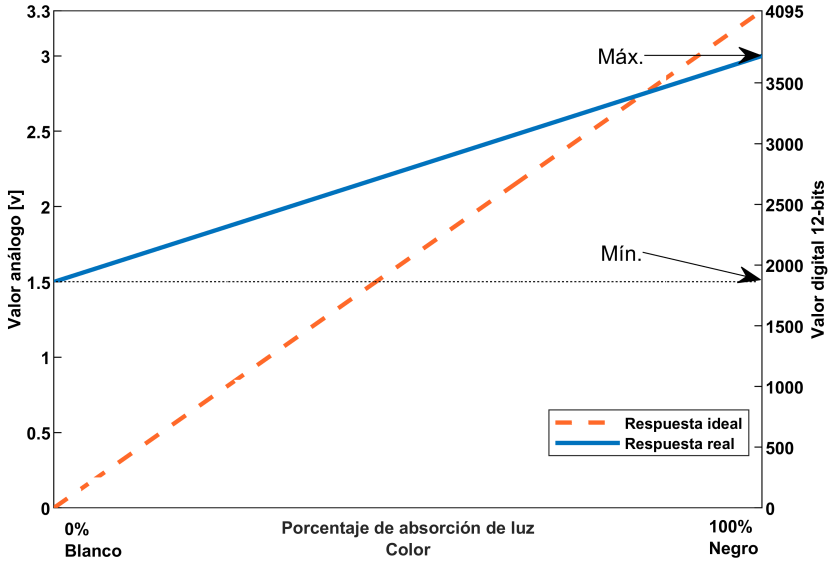


Figura 5.6: Respuesta ideal del sensor infrarrojo

Como se dijo anteriormente, la rutina de calibración busca los valores mínimo y máximo de cada sensor y lo lleva a una escala común. Para esto, se define una función de normalización 1, definida entre 0 y 100, siendo el valor cero, el que representa el color de la superficie (negro), y el valor 100, el que representa el color de la trayectoria (blanco). También se define una función de normalización 2, la cual, está pensada en el caso en que los colores de la superficie y la trayectoria de navegación estén invertidos, es decir, superficie color blanco y trayectoria color negro. Lo anterior se puede ver gráficamente en la Figura 5.7.

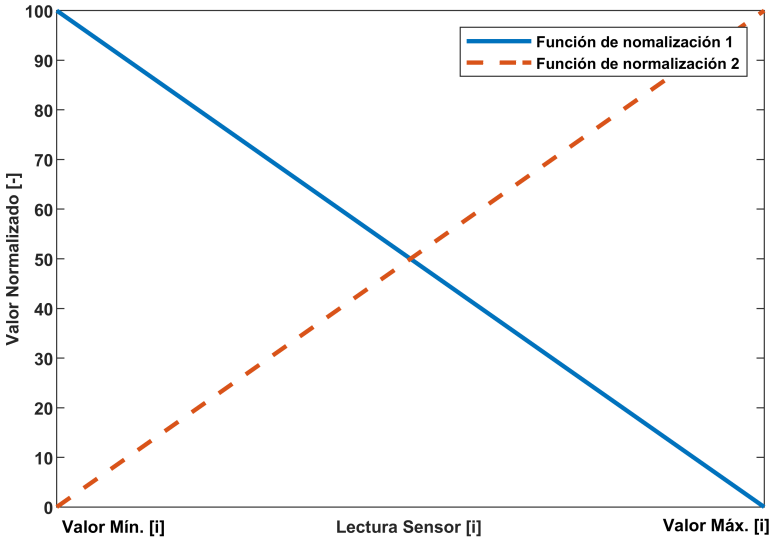


Figura 5.7: Función de normalización en función de los valores Mín. y Máx. de cada sensor

La función de normalización 1 posee pendiente negativa, de esta forma, el valor normalizado para la lectura de un sensor dado, es mayor cuando el sensor consultado está posicionado sobre la trayectoria de

navegación (color blanco). En ocasiones el color de la superficie y la trayectoria de navegación pueden estar invertidos, en ese caso, se puede utilizar la función de normalización 2, la cual, posee pendiente positiva, de esta forma, la función de normalización siempre tomará valores cercanos a 100, cuando las lecturas provengan de sensores que posicionados sobre la trayectoria, y valores cercanos a 0 a aquellos que provengan de mediciones que estén fuera de la trayectoria, siempre y cuando se indique en que tipo de escenario va a operar la plataforma (color de la línea).

Estas funciones de normalización pueden ser escritas de forma matemática como:

$$valor\_normalizado_i = \begin{cases} \dfrac{-100}{máx_i - mín_i}(Lectura_i - mín_i) + 100 & si \quad trayectoria = blanco, \\ & \quad superficie = negro \\ \dfrac{100}{máx_i - mín_i}(Lectura_i - mín_i) & si \quad trayectoria = negro, \\ & \quad superficie = blanco \end{cases} \tag{5.1}$$

El algoritmo 2 muestra de forma resumida la rutina implementada para calibrar los sensores infrarrojos. Para que la calibración cumpla con su propósito de forma correcta, se debe posicionar el robot sobre la superficie, y asegurar que mientras se ejecute la rutina de calibración, el robot se desplace alternando su posición entre la trayectoria de navegación y la superficie (se sugiere que el robot gire sobre la trayectoria de navegación).

Algoritmo 2 Calibración de sensores infrarrojos

1: **mientras** El DDMR se mueve sobre la superficie (un número finito de veces) **hacer**
2: Leer cada sensor utilizando el algoritmo 1
3: Buscar valores máximos y mínimos para cada sensor
4: **si** se encuentran valores máximos ó mínimos para el sensor i-ésimo **entonces**
5: Asignar valores a la variable $máx_i$ o $mín_i$ respectivamente
6: **fin si**
7: **fin mientras**

El algoritmo 3 muestra la secuencia que permite realizar lecturas de cada sensor infrarrojo en una escala normalizada. Para esto es necesario que la rutina de calibración 2 se haya ejecutado.

Algoritmo 3 Lectura normalizada de sensores infrarrojos

Entrada: Definir un escenario de operación (línea blanca o negra); Algoritmo 2
1: Leer cada sensor utilizando el algoritmo 1
2: Evaluar la lectura del sensor i-ésimo en la función de normalización (5.1)

Para poder estimar el ángulo entre el DDMR y la trayectoria de navegación, se dispone el arreglo de sensores de forma horizontal y al centro del robot. La Figura 5.8 muestra la ubicación del arreglo de sensores, además es posible observar que el robot cuenta con un arreglo de sensores frontal y uno posterior, ambos se utilizan para medir el ángulo entre el DDMR y la trayectoria de navegación, pero la medición se realiza de forma independiente para cada uno, en función de la dirección de desplazamiento del robot, es decir, cuando el robot va hacia adelante, se utiliza el arreglo de sensores frontal, y cuando el robot va hacia atrás, se utiliza el arreglo de sensores posterior.

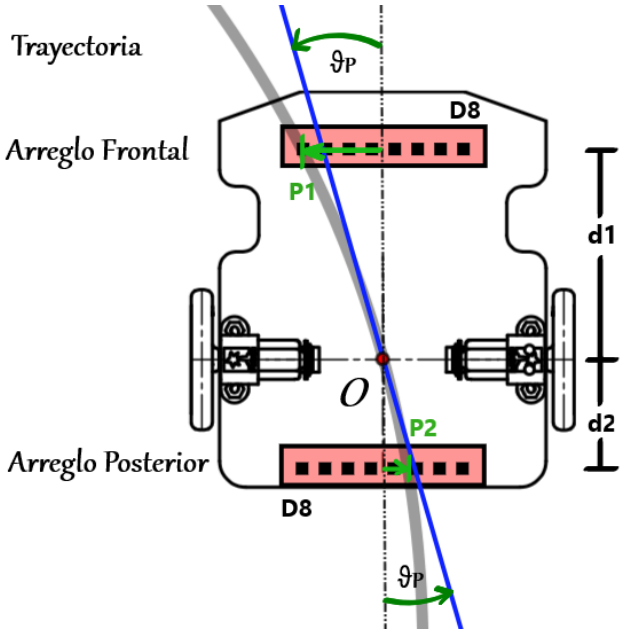
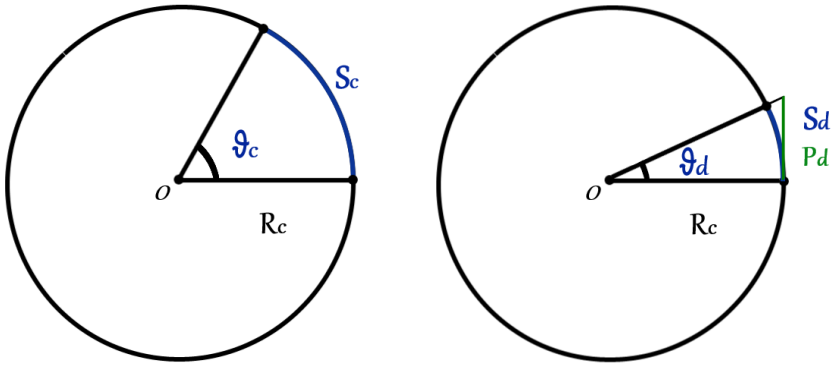


Figura 5.8: Ubicación de los sensores infrarrojos para la medición de la orientación



(a) Ejemplo de arco de circunferencia **(b)** Arco de circunferencia para un ángulo pequeño

Figura 5.9: Arco de una circunferencia

Tomando en consideración la Figura 5.9a, se obtiene el ángulo θ_c a partir de la expresión:

$$\theta_c = \frac{S_c}{R_c} \tag{5.2}$$

donde S_c corresponde al arco de circunferencia. Si se considera el caso mostrado en la Figura 5.9b, se observa que el arco de circunferencia S_d se aproxima al segmento P_d, lo cual, es válido solo para ángulos pequeños. Según lo anterior, se puede utilizar la expresión (5.2) para medir el ángulo del DDMR, aproximando el arco de circunferencia por la distancia entre la trayectoria y el centro del arreglo de sensores, indicada en la Figura 5.8 como P1 para el arreglo frontal y P2 para el arreglo posterior. Lo anterior queda definido en la siguiente

expresión:

$$\theta_P \approx \begin{cases} \dfrac{P1}{d1} & con \quad Arreglo\ frontal \\ \dfrac{P2}{d2} & con \quad Arreglo\ posterior \end{cases} \tag{5.3}$$

con d1, d2, la distancia entre el centro de giro del DDMR y el punto central del arreglo de sensores frontal y posterior respectivamente. Con esta aproximación, el problema de medición del ángulo de orientación del robot, se reduce a encontrar la distancia entre el centro del arreglo de sensores y la trayectoria de navegación.

La Figura 5.10 muestra el escenario de medición de la trayectoria de navegación. La superficie se muestra de un color negro, y la trayectoria de navegación de color blanco, mientras que el arreglo de sensores se sitúa sobre la superficie de navegación. Dado que el arreglo de sensores se encuentra alineado con el eje de simetría del DDMR (ver anexo **??**), se debe medir la posición de la trayectoria respecto del centro del arreglo. A medida que la distancia entre la trayectoria y el centro de arreglo tiende a cero, el ángulo de orientación lo hace de la misma manera. La Figura 5.11 muestra los valores de lectura normalizados, a partir de las lecturas de cada sensor, tomando en consideración el escenario de operación presentado en la Figura 5.10.

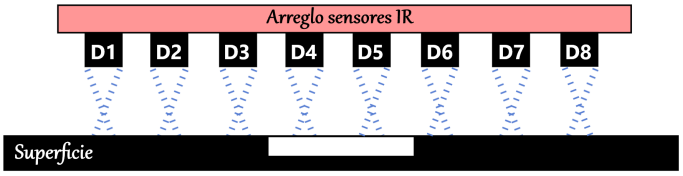


Figura 5.10: Escenario de medición de la trayectoria utilizando un arreglo de sensores

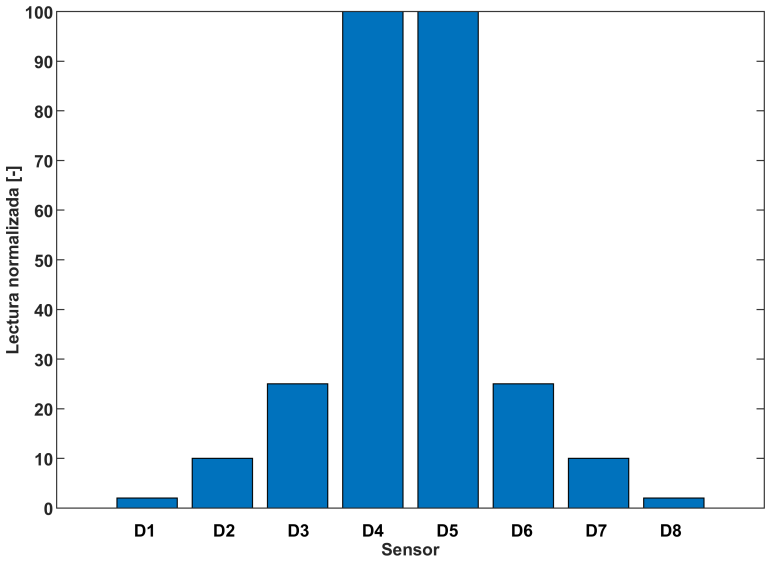


Figura 5.11: Mediciones normalizadas del arreglo de sensores ubicado sobre la trayectoria de navegación

Al comprender el funcionamiento del arreglo de sensores, se construye un modelo que permite determinar la posición de la trayectoria de navegación, tomando como referencia el centro del arreglo. La distribución espacial de cada sensor es discreta, ya que, existen posiciones a lo largo del arreglo donde no hay

sensores, a causa de lo anterior se plantea utilizar una función continua para modelar el comportamiento del arreglo, y de esta forma medir la posición de la trayectoria en un rango continuo. El modelo planteado se puede observar en la Figura 5.12

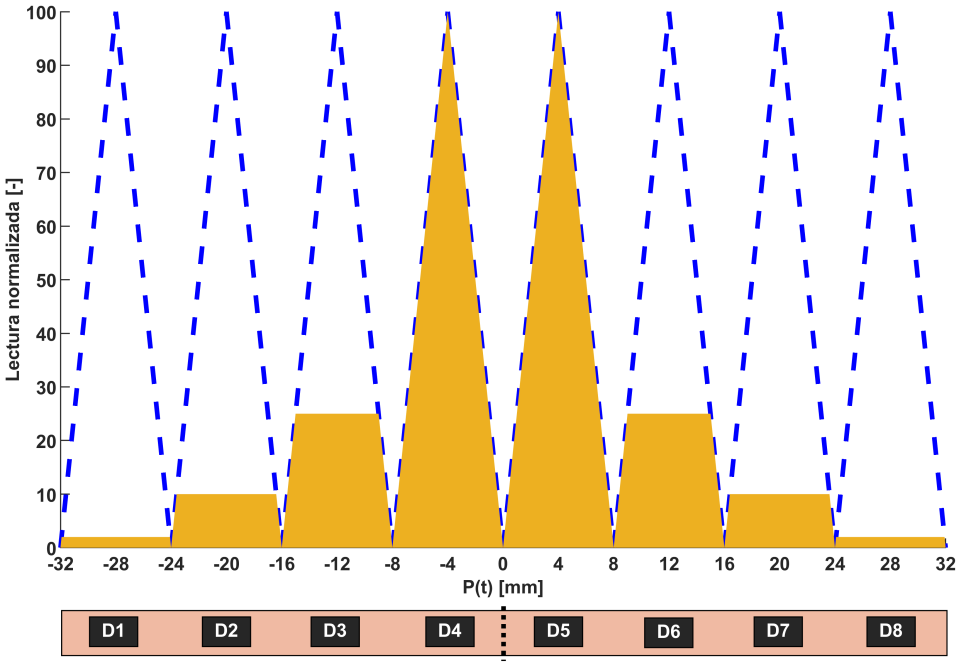


Figura 5.12: Representación del modelo de ponderación para el arreglo de sensores IR

Para construir el modelo, se definen las posiciones de cada sensor a partir del centro del arreglo. El centro de cada sensor, se encuentra a una distancia de 8 *mm* respecto del centro de los sensores laterales (vecinos). Para medir las variaciones en la posición de trayectoria, se definen funciones triangulares, donde el nivel de activación lo determina la lectura normalizada de cada sensor. Esto permite que el modelo sea sensible a pequeñas variaciones en la posición de la trayectoria. la expresión (5.4) muestra el modelo matemático de la función triangular usada.

$$Ft_i(p_i) = \begin{cases} \frac{100}{4}(p_i(t) + 32 * i) & si \quad p_i(t) \in [-32 + 8(i-1);\ -28 + 8(i-1)][mm], \\ & i = 1, 2, .., 8. \\ 100 - \frac{100}{4}(p_i(t) + 28 * i) & si \quad p_i(t) \in [-28 + 8(i-1);\ -24 + 8(i-1)][mm], \\ & i = 1, 2, .., 8. \end{cases} \tag{5.4}$$

$p_i(t)$ corresponde a una posición local, definida para cada sensor. La posición "global" de la trayectoria, se puede calcular como un promedio ponderado del área bajo la curva de cada función Ft_i, truncada hasta el valor de lectura normalizado de cada sensor, donde el factor de ponderación, se define como la posición espacial del i-ésimo sensor. Lo anterior, se puede observar en la Figura 5.12. Para calcular cada una de las áreas resultantes se considera el caso para un solo sensor, y luego, se aplica el mismo procedimiento para el resto de sensores. La Figura 5.13 muestra un ejemplo de medición y el área formada para el sensor D1.

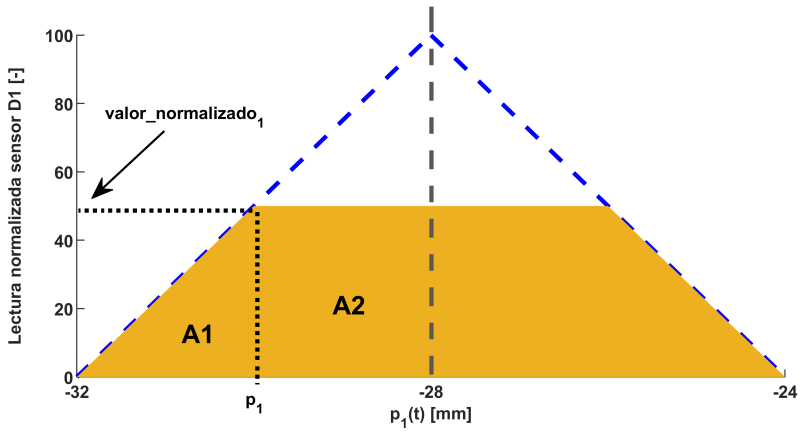


Figura 5.13: Ejemplo de lectura y área generada por el sensor D1

A partir de la figura se identifica que existe una simetría en la función triangular, de la cual, se puede calcular el área de la mitad derecha y luego multiplicar por 2 para obtener el área total At_1. Además, el área resultante se puede dividir en 2 tipos de áreas conocidas, el área de un triángulo $A1$ y el área de un rectángulo $A2$. Lo anterior se puede expresar como:

$$A1 = \frac{(p_1(t) + 32) \cdot valor\_normalizado_1}{2} \tag{5.5}$$

$$A2 = -(p_1(t) + 28) \cdot valor\_normalizado_1 \tag{5.6}$$

luego el área total queda determinada por:

$$\begin{aligned} At_1 &= 2(A1 + A2) \\ &= -valor\_normalizado_1(p_1(t) + 24) \end{aligned} \tag{5.7}$$

Donde $p_1(t)$ se obtiene a partir de la expresión (5.4).

$$Ft_1(p_1) = valor\_normalizado_1 = \frac{100}{4}(p_1(t) + 32) \tag{5.8}$$

$$p_1(t) = \frac{4}{100} \cdot valor\_normalizado_1 - 32 \tag{5.9}$$

Con esto se expresa el área total formada por el sensor D1 en función de su medición según:

$$\begin{aligned} At_1 &= -valor\_normalizado_1(\frac{4}{100} \cdot valor\_normalizado_1 - 32 + 24) \\ &= 8 \cdot valor\_normalizado_1 - \frac{4}{100} \cdot valor\_normalizado_1^2 \end{aligned} \tag{5.10}$$

como esta expresión solo depende del valor normalizado que mide el sensor D1, se puede extender este modelo para el resto de sensores.

$$At_i = 8 \cdot valor\_normalizado_i - \frac{4}{100} \cdot valor\_normalizado_i^2 \tag{5.11}$$

Finalmente, la posición de la trayectoria respecto del centro del arreglo de sensores se expresa como:

$$P(t) = \frac{\sum_{i=1}^{8}(8i - 36)At_i}{\sum_{i=1}^{8} At_i} \tag{5.12}$$

por último

$$\theta(t) \approx \begin{cases} \dfrac{P(t)}{d1} & con \quad Arreglo\ frontal \\ \dfrac{P(t)}{d2} & con \quad Arreglo\ posterior \end{cases} \tag{5.13}$$

El algoritmo 4 presenta la rutina implementada para obtener la orientación del DDMR respecto de la trayectoria de navegación.

Algoritmo 4 Medición de orientación $\theta(t)$

Entrada: DDMR posicionado sobre la trayectoria

1: Seleccionar el arreglo de sensores a utilizar (frontal o posterior)
2: Realizar una lectura normalizada de los sensores utilizando el algoritmo 3
3: Obtener la orientación a partir de la expresión (5.13)

5.3.2. Sensor de velocidad

Para medir la velocidad lineal $v(t)$ del DDMR, se dispone de un encoder de cuadratura. Cada encoder posee dos sensores de efecto hall (estáticos) y un disco magnético de 7 pares de polos (N-S) acoplado al eje del motor DC por su parte posterior (lado opuesto al de acoplamiento de la rueda). Los sensores de efecto hall utilizados poseen una salida de voltaje digital, la cual, alterna su valor lógico (de 0 a 1 y viceversa) en función del cambio en el sentido del campo magnético, producido por el giro del disco. La ubicación de los sensores permite que las señales digitales que produce cada uno (por el efecto de la rotación del eje del motor), estén desfasadas en 90°, esta característica permite identificar el sentido de giro del motor, poniendo atención en el orden en que ocurren los cambios de nivel en las señales de salida de los sensores, y define las cuentas por revolución (CPR) en 28 (7 pares de polos y desfase 90° entre las salidas del encoder). Además, se puede contar el número de pulsos generados por el encoder para determinar la posición angular del motor, del mismo modo, se puede aproximar la velocidad de giro del motor, dividiendo el número de pulsos acumulados en un periodo de tiempo, por el mismo periodo.

Las cuentas de pulsos de cada encoder que se producen por el movimiento del robot, se llevan a cabo por el MCU, para lo cual, se utiliza la librería *Encoder Library* [27], esta implementa las funciones necesarias para contar los pulsos desde un encoder de cuadratura. A continuación se explica su uso aplicado a este trabajo.

- *Encoder mi_encoder(pin1, pin2);*

  Esta función crea un objeto del tipo Encoder, utiliza dos pines provenientes de los sensores. Es posible crear múltiples objetos, cada uno definido por los dos pines mencionados anteriormente. Ambos pines deben ser aptos para realizar lecturas digitales, el rendimiento mejora cuando los pines utilizados tienen la capacidad de trabajar con interrupciones. La relación entre los encoders del DDMR y los objetos de la librería se muestra en la Tabla 5.2.

Tabla 5.2: Relación entre los encoders y los objetos de la librería

Hardware			Software		
Encoder	Salida 1	Salida 2	Objeto	*pin1*	*pin2*
Derecho	C1	C2	*encoder_der*	5	23
Izquierdo	C1	C2	*encoder_izq*	18	19

Es necesario aclarar que los pines definidos en el software, consideran el hecho que la orientación entre la ubicación de los motores es de 180° (se invierte el sentido de giro).

- *mi_encoder.read();*

  Esta función retorna la posición acumulada (cuentas), el valor puede ser positivo o negativo.

- *mi_encoder.write(nuevo_valor);*

  Esta función permite definir la posición acumulada (cuentas), a través del valor definido en la entrada *nuevo_valor*.

A partir de las funciones definidas en la librería se desarrollan los modelos para determinar algunas de las magnitudes físicas de interés.

$$\dot{\phi}_m(t) \approx cuentas \cdot \frac{2\pi}{CPR} \cdot \frac{1}{\Delta t} \left[\frac{rad}{s}\right] \tag{5.14}$$

CPR corresponde al parámetro cuentas por revolución (determinado por las características del encoder). La velocidad de giro de los motores DC depende del número de cuentas acumuladas durante un intervalo de tiempo Δt segundos. Con esto se puede expresar la velocidad de cada motor del DDMR según:

$$\dot{\phi}_{mr}(t) \approx encoder\_der.read() \cdot \frac{2\pi}{CPR} \cdot \frac{1}{\Delta t} \tag{5.15}$$

$$\dot{\phi}_{ml}(t) \approx encoder\_izq.read() \cdot \frac{2\pi}{CPR} \cdot \frac{1}{\Delta t} \tag{5.16}$$

Luego se utilizan las expresiones (3.8), (3.12), (3.15) y (3.17), presentadas en el Capítulo de modelado, para determinar las velocidades de interés en función de las velocidades de los motores DC del DDMR.

$$\dot{\phi}_r(t) = \dot{\phi}_{mr}(t) \cdot N \tag{5.17}$$

$$\dot{\phi}_l(t) = \dot{\phi}_{mr}(t) \cdot N \tag{5.18}$$

$$v(t) = \frac{N \cdot r}{2}(\dot{\phi}_{mr}(t) + \dot{\phi}_{ml}(t)) \tag{5.19}$$

$$\dot{\theta}(t) = \frac{N \cdot r}{2}(\dot{\phi}_{mr}(t) - \dot{\phi}_{ml}(t)) \tag{5.20}$$

Finalmente, se presenta el algoritmo 5 que implementa la rutina en el MCU para medir las velocidades de interés.

Algoritmo 5 Medición de las velocidades del DDMR

Entrada: Incluir librería *Encoder Library*; Inicializar los objetos encoder para cada sensor, con lectura inicial 0; Definir una variable que mida el tiempo actual; Definir un intervalo de medición Δt

1: Medir el tiempo transcurrido utilizando un *Timer* del MCU y el tiempo actual
2: **si** el tiempo transcurrido es mayor o igual a Δt **entonces**
3: Calcular la velocidad de los motores según (5.15) y (5.16)
4: Definir las cuentas de los encoders en 0
5: Actualizar el tiempo actual utilizando el *Timer*
6: **fin si**
7: Actualizar el valor del resto de las velocidades según (5.17), (5.18), (5.19) y (5.20)

5.3.3. Sensor de distancia

Cada DDMR está equipado con un sensor de distancia *Time of Flight* (ToF), VL53L0X [4] . Este estima la distancia lineal entre la parte frontal del sensor y un objeto reflectante ubicado en frente, a partir del tiempo que tarda un haz de luz emitido por el sensor en reflejarse en el objeto. El propósito de este sensor es medir la distancia entre un robot y su predecesor (otro robot en frente). El sensor utiliza el protocolo de comunicación I2C para interactuar con otros dispositivos.

Para utilizar este dispositivo con el MCU, se utilizan las librerías *Wire* [14] y *VL53L0X* [23], la primera permite al MCU comunicarse con dispositivos mediante el bus I2C a través de los pines *SDA* (datos)

y *SCL* (reloj), mientras que la segunda, entrega una serie de funciones que permiten interactuar con el sensor. Para efectos de este trabajo, se presenta una breve descripción de la librería, enfocada en las funciones básicas utilizadas.

- Wire.begin();

  Esta función pertenece a la librería *Wire* y permite habilitar la comunicación I2C en el MCU.

- VL53L0X *mi_sensor*;

  Permite crear un objeto del tipo VL53L0X

- void *mi_sensor*.setTimeout(uint16_t *timeout*);

  Establece un período de tiempo de espera en milisegundos después del cual se anularán las operaciones de lectura si el sensor no está listo. Un valor 0 deshabilita el tiempo de espera.

- bool *mi_sensor*.init(bool *io_2v8* = *true*);

  Inicializa y configura el sensor. Si el argumento es *true* (valor predeterminado si no se especifica), el sensor se configura para el modo 2V8 (E/S de 2.8 V), si es $false$, el sensor se deja en modo 1V8. La función retorna un valor booleano que indica si la inicialización se completó correctamente.

- bool *mi_sensor*.setSignalRateLimit(float *limit_Mcps*);

  Establece el límite de velocidad de señal de retorno al valor definido en unidades de MCPS (mega cuentas por segundo). Esta es la amplitud mínima de la señal reflejada desde el objetivo y recibida por el sensor necesario para que informe de una lectura válida. Establecer un límite inferior aumenta el rango potencial del sensor, pero también aumenta la probabilidad de obtener una lectura inexacta debido a las reflexiones de objetos distintos del objetivo previsto. Este límite se inicializa en 0.25 MCPS de forma predeterminada. El valor devuelto es un booleano que indica si el límite solicitado es válido.

- bool *mi_sensor*.setMeasurementTimingBudget(uint32_t *budget_us*);

  Establece el tiempo de actualización de muestras por el valor dado en microsegundos. Un tiempo más largo permite mediciones más precisas. El valor predeterminado es de aproximadamente 33000 μs, o 33 ms, siendo 20 ms el mínimo valor permitido. El valor devuelto es un booleano que indica si el tiempo de muestreo solicitado era válido.

- void *mi_sensor*.startContinuous(uint32_t *period_ms* = *0*);

  Inicia mediciones continuas. Si el argumento es 0 (valor predeterminado si no se especifica), el sensor toma las mediciones tan a menudo como sea posible, si es distinto de cero, se utiliza el modo de tiempo continuo, donde el periodo entre medidas queda definido por el argumento de la función en milisegundos.

- uint16_t *mi_sensor*.readReg16Bit(uint8_t *reg*)

  Lee un registro de sensor de 16 bits y devuelve el valor leído.

- *mi_sensor*.RESULT_RANGE_STATUS + 10

  Corresponde al registro donde se almacena la última muestra válida en unidades de milímetros.

La distancia en centímetros entre la parte frontal del DDMR y su predecesor se puede obtener como:

$$d(t) \approx \text{sensor.readReg16Bit(sensor.RESULT\_RANGE\_STATUS} + 10) \cdot 0{,}100 - delta \tag{5.21}$$

donde el factor 0,100 convierte la medición obtenida, de milímetros a centímetros, y *delta* corresponde a una constante que permite calibrar la medición de distancia, esto considerando que el sensor posee una zona muerta, en aquellas distancias próximas a este, y que el sensor se ubica atrás de la estructura frontal del robot (ver apéndice **??**).

La secuencia necesaria para utilizar correctamente el sensor con el MCU debe seguir las siguientes etapas:

Algoritmo 6 Configuración de sensor ToF

Entrada: Incluir librería *Wire* y *VL53L0X*; Definir el objeto VL53L0X (sensor ToF)

1: Iniciar la comunicación I2C en el MCU
2: Inicializar el sensor
3: Configurar parámetros de medición del sensor

Algoritmo 7 Medición de la distancia entre el DDMR y su predecesor

Entrada: Algoritmo 6

1: Obtener la muestra actual de la distancia según (5.21)

5.4. Actuadores

El movimiento del robot depende del voltaje aplicado a cada uno de los motores DC. Para controlar el voltaje desde el MCU, se debe utilizar una interfaz de potencia, en este caso se ha escogido un driver puente H para cada motor DC. La interconexión entre los dispositivos fue presentada en el Capítulo 2 (ver Figura 2.14).

El módulo posee 2 canales (2 drivers puente H), los cuales, permiten controlar cada motor del robot de forma independiente. Si se escogen señales de disparo de tipo PWM, se puede controlar la velocidad de cada motor DC variando el ciclo de trabajo de cada una de las señales. Además, se puede controlar el sentido de giro manipulando las señales de control del módulo.

Los pines de control del módulo puente H se relacionan con los pines de salida del MCU según:

Tabla 5.3: Relación entre los pines de control del módulo puente H y los pines del MCU

Driver puente H - $vcc = 3v3$, $VM = 8v4$			
Canal	Pin PWM	Pin de control 1	Pin de control 2
A	GPIO17	GPIO27	GPIO16
B	GPIO14	GPIO13	GPIO12

Nuevamente, se aclara que los pines definidos en el software, consideran el hecho que la orientación entre la ubicación de los motores es de 180° (se invierte el sentido de giro). Para generar una señal PWM a través de los pines del MCU, se utiliza la librería *ESP32 AnalogWrite* [13], la cual, está basada en las funciones LEDC del MCU utilizado (ESP32). Las principales funciones utilizadas en este trabajo son:

- analogWriteFrequency(double *frequency*);

  Permite definir la frecuencia de la señal PWM

- analogWrite(uint8_t *pin*, uint32_t *value*, uint32_t *valueMax*);

  Asigna una señal PWM a un *pin* de salida GPIO del MCU. El ciclo de trabajo se controla a través del argumento de entrada *value*, el cual, puede tomar valores ente cero y un valor máximo definido por el argumento *valueMax*.

Para la implementación, se consideran las variables **pwm_r** y **pwm_l**, que controlan el ciclo de trabajo y signo (polaridad) de la señal PWM de disparo del driver. El rango de valores definido para estas variables es [-1023, 1023]. Se definen los pines de control del módulo puente H como se indica en la Tabla 5.4, y se utilizan las funciones:

```
analogWrite(17, abs(pwm_r), 1023);
analogWrite(14, abs(pwm_l), 1023);
```

Tabla 5.4: Relación entre el signo de las variables **pwm_r** y **pwm_l**, con los pines de control del módulo puente H

Variable pwm_r	**Canal A**		
	AIN1: GPIO27	AIN2: GPIO16	Salida PWM
>= 0	H	L	Positiva
< 0	L	H	Negativa
Variable pwm_l	**Canal B**		
	BIN1: GPIO13	BIN2: GPIO12	Salida PWM
>= 0	H	L	Positiva
< 0	L	H	Negativa

Con esto se controla el ciclo de trabajo y el signo de la señal PWM aplicada a cada motor del DDMR. Si se escoge una frecuencia de operación adecuada, se puede considerar que el voltaje aplicado a cada motor DC corresponde al valor medio de la señal PWM correspondiente, puesto que, como se determinó en el Capítulo 3 dedicado al modelo matemático, el sistema DDMR, actúa como un filtro pasa bajos, por lo que, el voltaje aplicado a cada motor DC queda definido por las variables en el MCU según:

$$u_{mr}(t) \approx \bar{u}_{mr}(t) = 8{,}4 \cdot \frac{\text{pwm\_r}}{1023} \tag{5.22}$$

$$u_{ml}(t) \approx \bar{u}_{ml}(t) = 8{,}4 \cdot \frac{\text{pwm\_l}}{1023} \tag{5.23}$$

La señal de voltaje que es aplicada a cada motor DC corresponde a una señal tipo PWM que posee la misma frecuencia y ciclo que trabajo que la señal de disparo, pero su amplitud está determinada por el voltaje aplicado al pin **VM** y del nivel lógico definido para los pines de control del módulo puente H. Para efectos de la implementación, se consideran las señales de control $\bar{u}_{mr}(t)$, $\bar{u}_{ml}(t)$, correspondientes al valor medio de la señal PWM.

El algoritmo 8 muestra la rutina que permite controlar el voltaje aplicado a los motores DC del DDMR.

Algoritmo 8 Accionamiento de motores DC

Entrada: Incluir librería *ESP32 AnalogWrite*; Definir una frecuencia de operación (señal PWM); Definir las variables de operación del módulo puente H

1: Solicitar valor para las variables **pwm_r** y **pwm_l** $\in [-1023, 1023]$ (definen de forma indirecta la velocidad y el sentido de giro de cada motor)
2: Configurar los pines del módulo puente H según la tabla 5.4
3: Actualizar las señales PWM utilizando la función *analogWrite()* (el voltaje aplicado a cada motor se puede aproximar según (5.22) y (5.23))

5.5. Comunicación

Una de las características más importantes de la plataforma experimental, es la capacidad que poseen los robots para comunicarse entre sí, y con el entorno. Esto, debido a las capacidades del MCU utilizado, el que incorpora hardware dedicado para la comunicación inalámbrica, permitiendo el uso de WiFi y Bluetooth de baja energía (BLE).

Las funciones de comunicación, se implementan sobre la **red** de comunicación WiFi, haciendo uso del protocolo de comunicación UDP [1] (en inglés: User Datagram Protocol), el cual, es un protocolo del nivel de transporte, basado en el intercambio de datagramas. Permite el envío de datagramas a través de

la **red**, sin la necesidad de establecer una conexión previa entre los dispositivos, puesto que, el datagrama contiene toda la información referente al direccionamiento en su cabecera. Este protocolo no garantiza el orden en que se reciben los paquetes (puede que no suceda a menudo en la práctica, por lo general sucede en topologías de red más grandes), ni tampoco se sabe si ha llegado correctamente, ya que no hay confirmación de entrega o recepción (esto puede no ser un problema en redes locales pequeñas), lo anterior permite que la velocidad de transmisión sea mucho más alta.

La red de comunicación se establece sobre un punto de acceso local (red local WiFi), por lo que cada dispositivo dentro de la plataforma experimental debe tener acceso a la red. Lo anterior se muestra en la Figura 5.14.

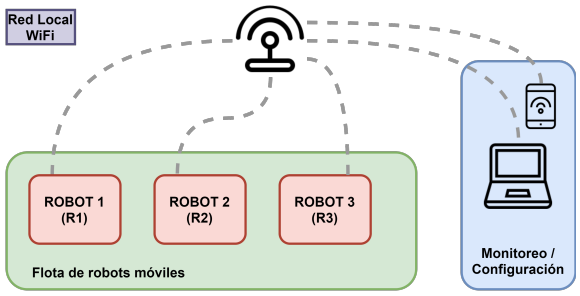


Figura 5.14: Esquema de conexión de la plataforma experimental a la red Local

5.5.1. Comunicación entre robots

La comunicación entre los robots de la plataforma experimental permite formar estructuras más complejas (flotas), capaces de cooperar entre sí, según la topología de red implementada. En la Figura 5.15 se muestra la topología de comunicación implementada. conocida como “cascada”. De esta forma, cada robot recibe la información de interés de su predecesor, al mismo tiempo que envía periódicamente sus variables locales al robot sucesor.

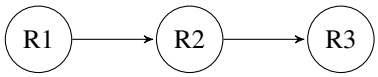


Figura 5.15: Topología de comunicación en cascada - comunicación entre robots

Para implementar esta topología en el MCU de cada DDMR se utiliza la librería *WiFi library* [15], esta implementa las funciones necesarias para establecer la conexión de los robots a una red WiFi, como también las funciones que permiten enviar y recibir paquetes UDP. Las principales funciones utilizadas son:

- WiFi.begin(*ssid*, *pass*);

  Inicializa la configuración de la red WiFi y entrega el estado actual. La información de la red se ingresa a través de los argumentos *ssid*, y *pass*, nombre de la red y contraseña, respectivamente.

- WiFi.mode(*mode*);

  Define el modo de conexión WiFi para el MCU, a través del argumento *mode* el cual puede ser: “STA”, “AP”, “STA+AP”. En nuestro caso se configura como “STA” que corresponde a una estación, para conectarla a una red WiFi, mediante de un punto de acceso.

- WiFi.localIP();

  Retorna la dirección IP local del MCU conectado a una red WiFi.

- WiFiUDP udp;

  Crea una clase UDP, que puede ser usada para enviar y recibir mensajes.

- udp.begin(*puerto*);

  Inicializa la configuración del protocolo UDP. Comienza la recepción de datos a través de un puerto local, definido por el argumento *puerto*.

- udp.remoteIP();

  Obtiene la dirección IP de una conexión remota.

- udp.remotePort();

  Obtiene el puerto de salida de una conexión remota.

- udp.beginPacket(*IP_remota*, *puerto_remoto*);

  Inicia la dirección de escritura UDP hacia una conexión remota definida por *IP_remota*, *puerto_remoto*.

- udp.printf(*mensaje*); Este finaliza la escritura UDP y envía el paquete a la dirección definida. Define el mensaje UDP a enviar, este se define como caracteres a través del argumento *mensaje*

- udp.endPacket();

A partir de las funciones señaladas anteriormente, se implementa el algoritmo 9, que permite a cada robot conectarse a una red WiFi, e inicializar la comunicación UDP.

Algoritmo 9 Inicializar conexión WiFi y comunicación UDP

Entrada: Incluir librería *WiFi Library*

1: Configurar el módulo WiFi
2: Conectarse a la red local **ssid**, **pass**
3: Inicializar protocolo UDP (definir un puerto local para la recepción de datos)

La información que se envían los robots tiene la siguiente estructura:

$$\text{V/variable de interés/valor} \tag{5.24}$$

Luego el algoritmo 10, permite a cada robot enviar sus variables de interés obtenidas localmente, a su robot sucesor, por ejemplo: R1 comunica sus variables a R2, y R2 comunica sus variables a R3.

Algoritmo 10 Envío de variables locales de interés a robot sucesor

Entrada: Algoritmo 9; Definir la dirección IP y el puerto del robot sucesor (dirección remota); Definir una variable que mida el tiempo actual; Definir el periodo de envío Δt

1: Medir el tiempo transcurrido utilizando un *Timer* del MCU y el tiempo actual
2: **si** el tiempo transcurrido es mayor o igual a Δt **entonces**
3: Iniciar la escritura de datos UDP hacia la dirección remota (robot sucesor)
4: Definir el mensaje correspondiente de acuerdo con la expresión (5.24)
5: Construir el datagrama (dirección+mensaje) y enviar hacia la dirección remota (robot predecesor)
6: Actualizar el tiempo actual utilizando el *Timer*
7: **fin si**

Por último, el algoritmo 11 define la rutina que permite a cada robot, recibir las variables de interés enviadas por su robot predecesor, por ejemplo: R2 recibe las variables locales de R1, y R3 recibe las variables locales de R2.

Algoritmo 11 Recepción de variables locales del robot predecesor

Entrada: Algoritmo 9

1: Verificar si hay paquetes UDP disponibles en el puerto local de recepción
2: **si** hay paquetes disponible **entonces**
3: Leer el dato disponible
4: Decodificar el mensaje según la expresión (5.24)
5: Almacenar las variables del predecesor, para su uso
6: **fin si**

5.5.2. Monitoreo

El monitoreo de señales permite obtener los datos de cada robot en línea. Aprovechando la capacidad de cada DDMR de poder enviar información de forma remota a través del protocolo de comunicación UDP, se implementa la topología de red en "estrella" unidireccional de la Figura 5.16, esta permite, a cada robot, compartir sus variables internas a un dispositivo externo (computador), conectado a la misma red local. La lectura y obtención de los datos se realiza a través de un script en Python, que permite abrir el puerto local, recibir los datos en tempo real de todos los robots, y almacenar los datos en un archivo de valores ***.csv***. El

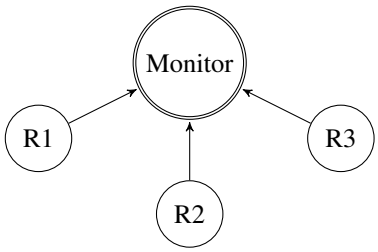


Figura 5.16: Topología de comunicación en estrella - monitoreo de señales

mensaje enviado por cada robot tiene la siguiente estructura:

$$\text{carácter identificador, variable 1, variable 2, ..., variable j} \tag{5.25}$$

donde el primer carácter, permite identificar los datos de cada robot, seguido por una coma (","), se concatena el valor obtenido por la variable 1, hasta la j-ésima variable (se han realizado pruebas hasta con 15 variables). El algoritmo 12 muestra la rutina que permite a cada robot enviar sus variables más importantes (velocidad, distancia respecto su predecesor, orientación, referencias, señales de control, etc.).

Algoritmo 12 Envío de datos para monitoreo

Entrada: Algoritmo 9; Definir la dirección IP y el puerto del dispositivo monitor (ej. Computador)

1: Iniciar la escritura de datos UDP hacia el dispositivo monitor (IP + puerto remota)
2: Definir el mensaje con los datos de interés, según la expresión (5.25)
3: Construir el datagrama (dirección-mensaje) y enviar hacia la dirección remota (dispositivo monitor)

5.5.3. Configuración de parámetros remota

Para facilitar la operación y configuración de los parámetros de los robots de la plataforma experimental, se implementa una topología de comunicación tipo "estrella" desde un dispositivo externo conectado a la red local (puede ser el mismo usado para el monitoreo), hacia cada uno de los robots (usando su dirección IP y puerto), y viceversa (bidireccional), tal como se indica en la Figura 5.17.

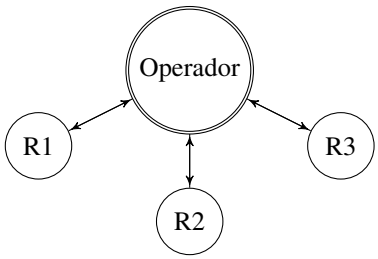


Figura 5.17: Topología de comunicación en estrella - configuración de parámetros

La estructura del los mensajes válidos que puede enviar el operador se muestra en (5.26). Alguno de los comandos implementado se presentan en la Tabla 5.5.

$$\text{E/variable/valor} \tag{5.26}$$

Tabla 5.5: Listado de comandos definidos para el operador de la plataforma experimental

Comando	**Descripción**
E/co_p/valor	Define la ganancia proporcional del controlador de orientación
E/co_i/valor	Define la ganancia integral del controlador de orientación
E/co_d/valor	Define la ganancia derivativa del controlador de orientación
E/cv_p/valor	Define la ganancia proporcional del controlador de velocidad
E/cv_i/valor	Define la ganancia integral del controlador de velocidad
E/cv_d/valor	Define la ganancia derivativa del controlador de velocidad
E/cv_ref/valor	Define la referencia de velocidad
E/cd_p/valor	Define la ganancia proporcional del controlador de distancia
E/cd_i/valor	Define la ganancia integral del controlador de distancia
E/cd_d/valor	Define la ganancia derivativa del controlador de distancia
E/cd_ref/valor	Define la referencia de distancia
E/calibrar/valor	Permite calibrar los arreglos de sensores IR si valor=1
E/parar/valor	Permite iniciar el ciclo de control (valor=si), o detenerlo (valor=no)

A continuación se presenta el algoritmo que permite a cada robot, recibir instrucciones de un operador.

Algoritmo 13 Recepción de instrucciones del operador

Entrada: Algoritmo 9

```
1:  Verificar si hay paquetes UDP disponibles en el puerto local de recepción
2:  si hay paquetes disponible entonces
3:      Almacenar la dirección del emisor (IP + puerto del operador)
4:      Leer el dato disponible
5:      Decodificar el mensaje según (5.26)
6:      si la variable y el valor son válidos entonces
7:          Asignar el valor recibido a la variable indicada
8:          Envía al operador el mensaje "ok"
9:      si no
10:         Envía al operador el mensaje "incorrecto", o, "datoIncorrecto"
11:     fin si
12: fin si
```

5.6. Controladores

En esta Sección se presentan los detalles de la implementación de los esquemas de control diseñados en el Capítulo anterior. Cada uno de los esquemas diseñados funciona de forma independiente, pero resuelven el mismo problema, el cual, consiste en otorgar a un robot la capacidad de navegar sobre una trayectoria definida en una superficie, manteniendo una distancia predeterminada respecto a otro agente predecesor. Cada esquema se implementa de igual manera en cada uno de los robots de la plataforma experimental.

Los controladores diseñados para tiempo continuo, se aproximan a tiempo discreto usando la representación en diferencias, conocida como diferencias hacia atrás (*Backward differences*) [31, 17], para la derivada de primer orden, tal como se muestra a continuación:

$$\frac{dy(t)}{dt} \approx \frac{y[t_k] - y[t_{k-1}]}{h} \tag{5.27}$$

donde t_k corresponde al instante de muestreo actual, y h el periodo de muestreo, este último se define como:

$$h = t_k - t_{k-1}$$

De manera similar, se puede aproximar la integral definida a continuación,

$$I(t) = \int_0^t e(s)ds$$

siguiendo la expresión:

$$\frac{dI(t)}{dt} = e(t) \tag{5.28}$$

Luego, se aproxima esta ecuación usando diferencias hacia atrás,

$$\frac{I[t_k] - I[t_{k-1}]}{h} = e[t_k]$$

lo cual, nos lleva a una representación recursiva para la integral, de la forma:

$$I[t_k] = I[t_{k-1}] + he[t_k] \tag{5.29}$$

El texto "Astrom, K.J & T. Hagglund (1995). *PID Controllers: Theory, Design and Tunning,* 2^{nd} Ed, ISA" [32], aborda con mayor detalle aspectos de implementación y operación de controladores digitales, secuencia de operación, muestreo, discretización, junto con técnicas de control avanzadas aplicadas a controladores PID. En algunas librerías del entorno de programación Arduino IDE, que implementan controladores digitales, se emplean las expresiones (5.27) y (5.29) para aproximar controladores del tipo PID [10], y controladores en espacio de estado con acción integral [26]. También implementan algunas de las prácticas más comunes para la operación de los controladores, como el cambio paramétrico suave, y anti-enrollamiento de la acción integral.

Cabe señalar que esta forma de implementar controladores digitales a partir del diseño en tiempo continuo, no es la única forma de hacerlo, incluso la aproximación en diferencias hacia atrás (5.27), tampoco es el único método existente. Por lo tanto, es necesario aclarar que el objetivo de esta parte del trabajo no busca ser la mejor implementación para los esquemas de control diseñados, sino, desarrollar un controlador aplicable al hardware del DDMR, a fin de evaluar sus capacidades.

5.6.1. Esquema LQI

El controlador diseñado en el Capítulo 4.1 corresponde a un esquema basado en el regulador LQR de tiempo continuo, que además incorpora acción integral. Con esto se consigue llevar las salidas a un valor deseado a través de una ley de control que depende de los estados del sistema.

El esquema diseñado en tiempo continuo se implementa en el MCU, el cual, corresponde a un sistema que opera en tiempo discreto. El algoritmo 14 muestra la rutina que implementa la aproximación en tiempo discreto del controlador LQI diseñado en tiempo continuo. Para facilitar la implementación se utiliza la librería *Basic Linear Algebra* [25], la cual, permite definir vectores y matrices y realizar operaciones algebraicas entre estos elementos.

Algoritmo 14 Rutina de control de seguimiento de trayectoria y predecesor basada en regulador LQR

Entrada: Incluir librería *Basic Linear Algebra*; Definir una variable que mida el tiempo actual; Definir un periodo de muestreo; Inicializar sensores y actuadores

1: Leer los sensores
2: Calcular el vector de estados **x**
3: Medir el tiempo transcurrido y el tiempo actual, utilizando un *Timer* del MCU
4: **si** el tiempo transcurrido es mayor o igual al periodo de muestreo **entonces**
5: Calcular las señales u_{mr} y u_{ml} según la ley de control **u = -kx**
6: Actualizar el tiempo actual utilizando el *Timer*
7: **fin si**
8: Actualizar las salidas utilizando el algoritmo 8, con **pwm_r =** u_{mr} y **pwm_l =** u_{ml}

5.6.2. Esquema PID

El esquema diseñado en el Capítulo 4.2, se plantea para tiempo continuo. En el MCU se implementa una aproximación del esquema diseñado, al igual que el controlador LQI presentado anteriormente. El algoritmo 15 muestra la rutina utilizada para implementar el esquema de control PID propuesto.

Algoritmo 15 Rutina de control de seguimiento de trayectoria y predecesor basada en controladores PID

Entrada: Definir una variable que mida el tiempo actual; Definir un periodo de muestreo; Inicializar sensores y actuadores

1: Medir la orientación, velocidad y distancia, utilizando los algoritmos 4, 5 y 7 respectivamente
2: Medir el tiempo transcurrido utilizando un *Timer* del MCU y el tiempo actual
3: **si** el tiempo transcurrido es mayor o igual al periodo de muestreo **entonces**
4: Computar el controlador PID de distancia, con **d_ref** y **d(t)**
5: Computar el controlador PID de velocidad, con **vel_ref =** u_d y **v(t)**
6: Computar el controlador PID de orientación, con θ**_ref** y $\theta(t)$
7: Actualizar el tiempo actual utilizando el *Timer*
8: **fin si**
9: Actualizar las salidas utilizando el algoritmo 8, con **pwm_r =** u_v**+**u_θ y **pwm_l =** u_v**-**u_θ

5.6.3. Esquema PID y comunicación R2R

Hasta ahora, se han desarrollado esquemas de control altamente descentralizado, es decir, que se ejecutan localmente por cada robot. Gracias a la capacidad de los robots de comunicarse de forma remota, y a la topología de comunicación en "cascada" presentada anteriormente, cada robot tiene acceso a la información de su robot predecesor. En algunos trabajos como [22, 18], se presenta un esquema de control para el seguimiento de agentes (restringido al desplazamiento longitudinal), basado en las mediciones de distancia y velocidad de los vehículos. En estos se presenta un esquema de control que incorpora la velocidad del agente líder como una referencia de velocidad local para cada vehículo, de la forma:

$$v_{ref}(t) = v_0(t) + u_d(t) \tag{5.30}$$

Donde la referencia de velocidad cada vehículo $v_{ref}(t)$, depende de la velocidad del agente líder $v_0(t)$, y el esfuerzo de control $u_d(t)$, calculado localmente en función del error de seguimiento de distancia.

Este tipo de controlador, se puede implementar en la plataforma experimental, pero al igual que los trabajos citados, el funcionamiento del esquema queda restringido a escenarios de operación, en donde los robots se desplacen en línea recta, debido a que, como el DDMR solo puede medir distancias en una sola dirección, durante una trayectoria curva, el comportamiento del sensor de distancia provocaría perturbaciones muy altas en la referencia de velocidad (5.30), esto podría ocasionar colisiones entre los agentes, comportamientos oscilatorios, y pérdida de la capacidad de los robots para mantenerse en la trayectoria.

Un esquema de control similar al descrito anteriormente, se propone para incorporar la información de velocidad del agente predecesor en el lazo de control local, y mejorar el comportamiento del seguimiento de agente predecesor en trayectorias cerradas con curvas. Para este esquema, la velocidad de referencia de los robots de una flota (excepto del robot líder, que solo sigue referencias de velocidad y la trayectoria de navegación), queda definida como:

$$v_{ref}(t) = sat(u_d(t)) \tag{5.31}$$

Este esquema está definido a partir del segundo robot en la flota, suponiendo que el primer robot corresponde al líder, el cual, puede navegar a una velocidad deseada, sin preocuparse de la medición de distancia. En este modelo se describe que la velocidad de referencia de un robot, depende de la acción de control local proveniente del controlador de distancia, siempre y cuando la actuación $u_d(t)$ se encuentre dentro de la zona lineal de la función *sat*(). Esta función se propone como:

$$sat(u_d) = \begin{cases} v_{Rp}(t) + \epsilon & \text{, para} \quad u_d(t) >= v_{Rp}(t) + \epsilon \\ u_d(t) & \text{, para} \quad -v_{Rp}(t) - \epsilon < u_d(t) < v_{Rp}(t) + \epsilon \\ -v_{Rp}(t) - \epsilon & \text{, para} \quad u_d(t) <= -v_{Rp}(t) - \epsilon \end{cases} \tag{5.32}$$

siendo $v_{Rp}(t)$ la velocidad del robot predecesor y ϵ un valor positivo, cercano a cero. De esta forma, la velocidad de referencia tomará el valor calculado localmente por el controlador de distancia, siempre y cuando este no supere la velocidad del robot predecesor. Escoger el parámetro ϵ ligeramente mayor que cero, permite que los robots puedan seguir una referencia de distancia, aun cuando, su robot predecesor esté detenido.

5.6.4. Automatización basada en estados

En este apartado, se presenta un diagrama de estados implementado para activar y desactivar distintas funciones del DDRM (ver Figura 5.18). A partir de los comandos “E/calibrar/valor” y “E/parar/valor”, definidos para el operador (ver Tabla 5.5), se puede controlar remotamente el funcionamiento de cada robot. El diagrama de estados indica que inicialmente (al energizar), el robot se encuentra en un estado “Detenido”, desde este estado, se puede realizar una rutina de calibración de los sensores del DDMR, enviando desde un dispositivo externo el comando “E/calibrar/1”. La rutina de calibración está automatizada, y al completarse realiza la asignación “calibrar=0;”, y vuelve al estado “Detenido”. Luego se puede ejecutar el ciclo de control a partir del comando “E/parar/no”. Por último, se puede volver al estado detenido, siempre y cuando, la variable “parar==si”, lo cual, ocurre si el operador lo indica, o, si el DDRM se pierde de la trayectoria.

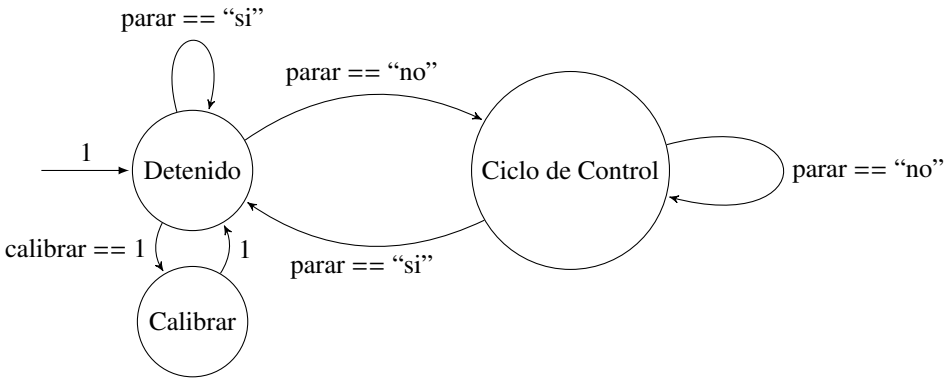


Figura 5.18: Descripción de estados de operación del DDMR

5.7. Resultados

En el siguiente link, se puede observar la operación remota de la plataforma experimental https://youtu.be/nR3WdcdglEQ

5.7.1. Resultados de navegación con controlador LQI

5.7.1.1. Resultados en trayectoria recta - Controlador LQI

- Prueba 1[2]: Cambios de referencia a los robots R1, R2 y R3 con un obstáculo fijo en frente de R1.

  Para esta prueba, los robots siguen una formación: R3→ R2→ R1, y comienzan con una referencia de distancia de 15 cm entre cada robot (ver Figura 5.19). Primero, se realiza un cambio en la referencia de distancia del robot R1 a 8 cm, La cual, es alcanzada rápidamente por el robot. El desplazamiento del robot R1 hacia su nueva referencia, provoca que el robot R2 deba moverse para corregir su formación, lo que a su vez afecta al robot R3, el cual, debe moverse para cumplir con su referencia de seguimiento. Se realizan otros dos cambios de referencia en la distancia para el robot R1, y se observa como este cambio se propaga como una perturbación hacia los demás robots (ver Figura 5.20). Luego se realizan cambios de referencia de distancia para el robot R2, manteniendo fijo al robot R1, el robot R2 se desplaza para cumplir con las referencias definidas, lo que, provoca perturbaciones en la distancia del robot R3. El robot R3 logra compensar las perturbaciones, para mantener su referencia. Finalmente, se realizan cambios de referencia en la distancia del robot R3, las cuales, son seguidas por el robot.

[2]video de la prueba: https://youtu.be/m4ZePHtsVhk

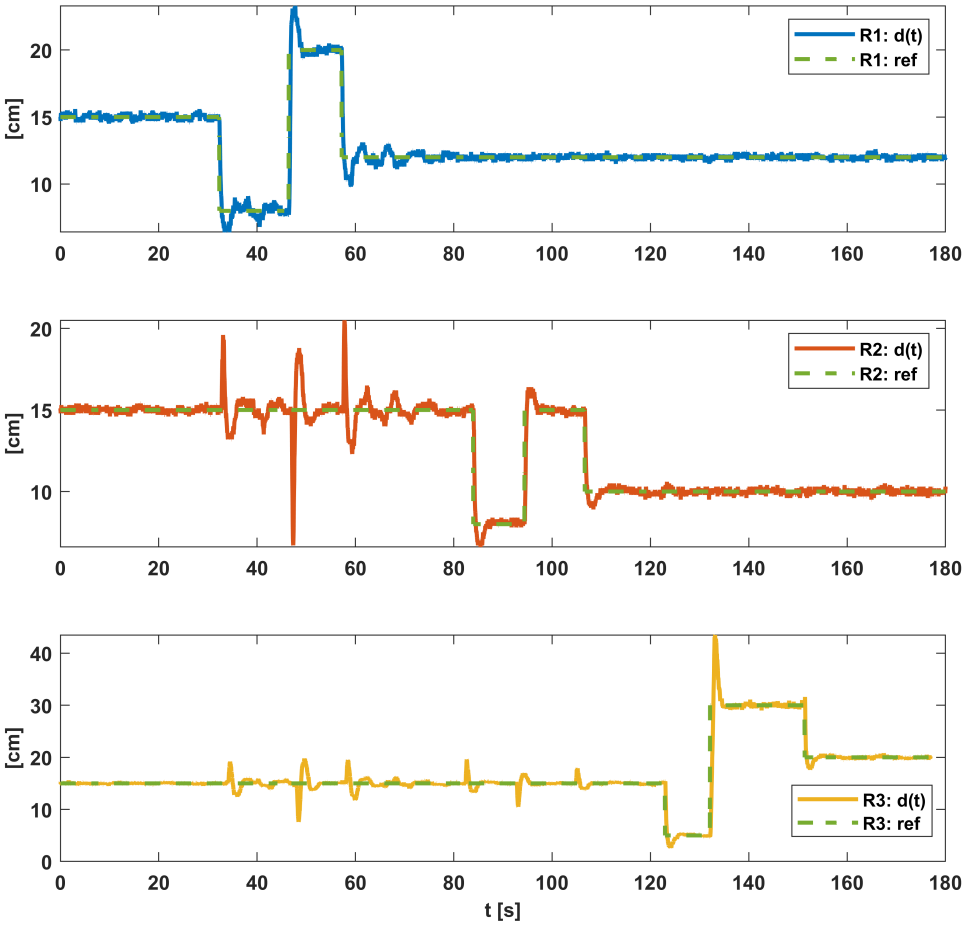


Figura 5.19: Seguimiento de distancia con controlador LQI - cambios de referencia - trayectoria recta

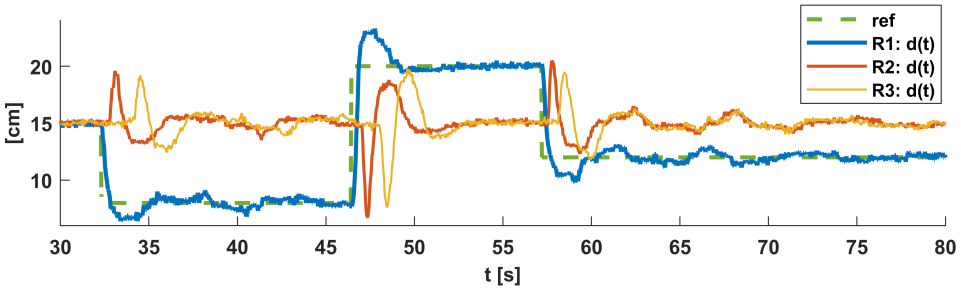


Figura 5.20: Acercamiento de la respuesta temporal del seguimiento de distancia de la Figura 5.19

La orientación de cada robot respecto de la trayectoria de navegación, se mantiene prácticamente en 0 rad (ver Figura 5.21), se observa la aparición de algunos valores de mayor amplitud, los cuales aparecen justo en los instantes donde se cambia la referencia de distancia. Esto nos indica que el desempeño del controlador de orientación tiene mejores resultados cuando el robot está en movimiento.

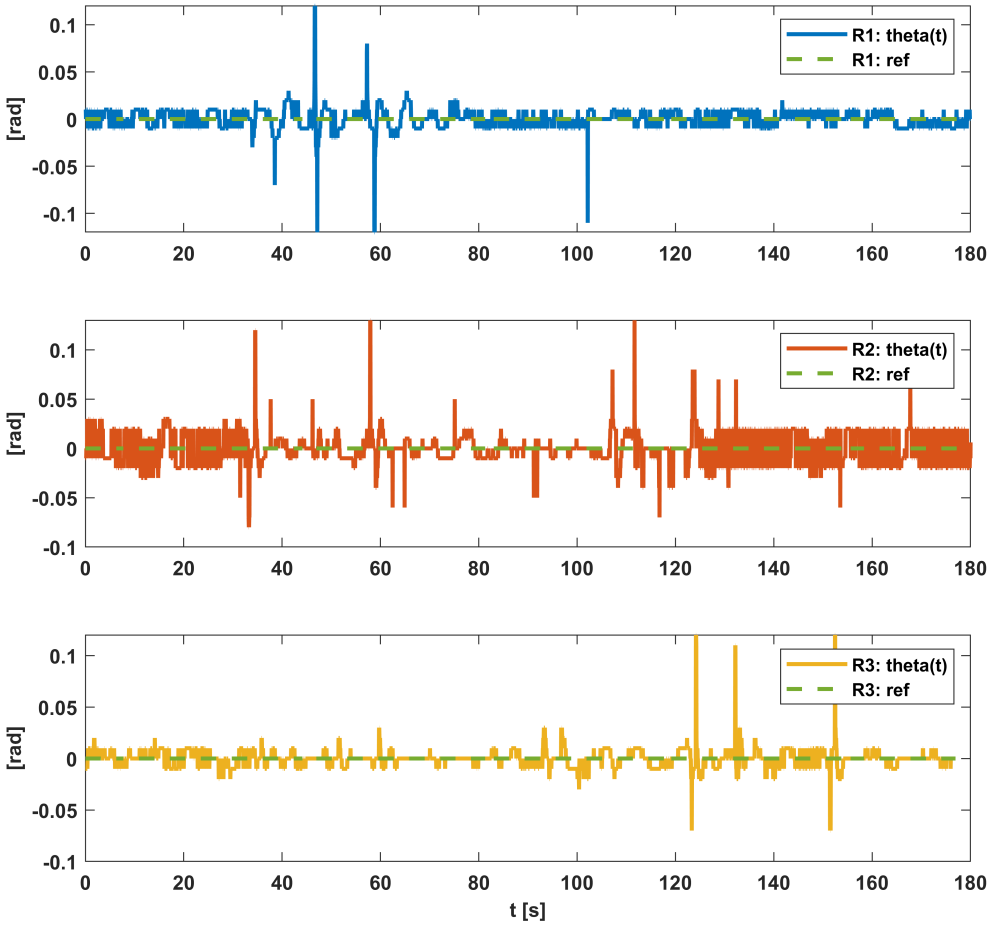


Figura 5.21: Seguimiento de orientación con controlador LQI - trayectoria recta - prueba 1

- Prueba 2[3] : Perturbaciones en la posición del obstáculo, para referencia de distancia fija.

  Los robots mantienen una referencia de distancia fija de 10 cm. Se realizan cambios manualmente en la posición del objeto frente al robot 1, esto provoca que los robots intenten seguir el movimiento del objeto y a su robot predecesor respectivamente, para mantener su formación inicial. En la Figura 5.22 se puede apreciar el comportamiento de la distancia de los 3 robots, las cuales, se mantienen en torno a la referencia respectiva durante toda la prueba. Es posible apreciar como el error de seguimiento del robot 1, se propaga hacia los robots 2 y 3, y como este se va amplificando (ver Figura 5.23).

[3] video de la prueba: https://youtu.be/uM7gtMpJ4DY

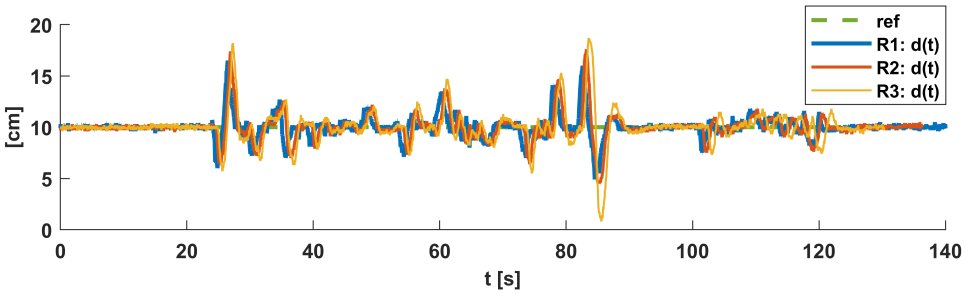


Figura 5.22: Seguimiento de distancia con controlador LQI - perturbaciones con formación fija - trayectoria recta

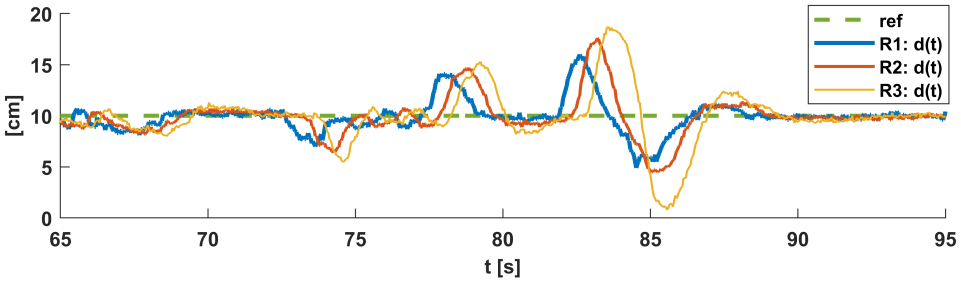


Figura 5.23: Acercamiento de la respuesta temporal del seguimiento de distancia de la Figura 5.22

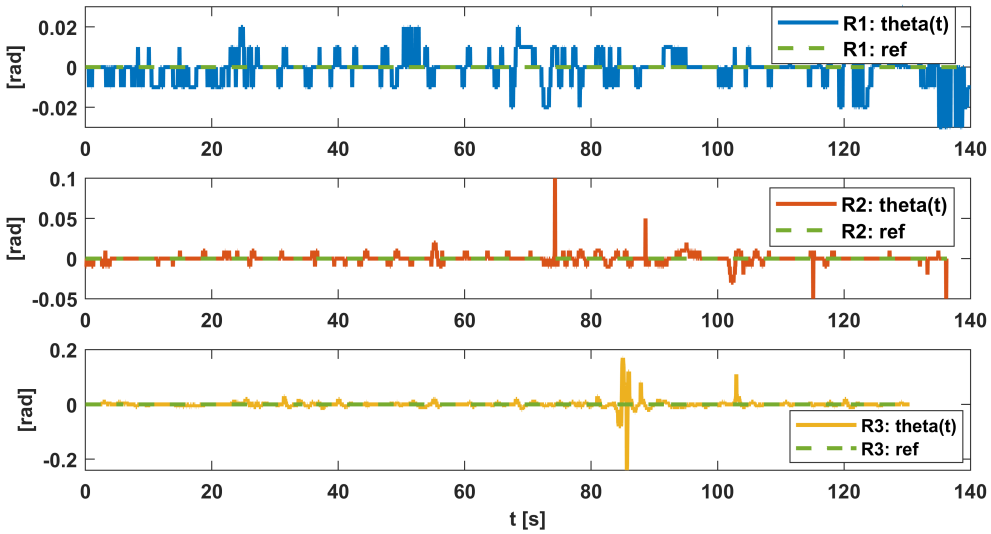


Figura 5.24: Seguimiento de orientación con controlador LQI - trayectoria recta - prueba 2

En la Figura 5.24 se puede observar la medición de la orientación de los robots, la cual, se mantiene en torno a los 0 rad, también se puede observar algunas perturbaciones en la orientación de los robots 2 y 3 en los instantes de tiempo cercanos a los 80 s, las cuales, ocurren producto del cambio en el sentido de la velocidad de desplazamiento de cada robot. Cuando los robots se encuentran a una distancia mayor a la referencia, estos deben moverse hacia adelante, para lo cual, usan el arreglo de sensores frontal para medir la orientación. Cuando la distancia de los robots es menor a la referencia, estos comienzan a retroceder, y utilizan el arreglo de sensores posterior para medir la orientación. En aquella transición

del sentido de la velocidad lineal, ocurren las perturbaciones en la orientación, ya que, además, el robot debe invertir su lógica de navegación (al igual que manejar un automóvil en reversa).

5.7.1.2. Resultados en trayectoria cerrada con curvas - Controlador LQI

Prueba 3[4] : Seguimiento de trayectoria cerrada con curvas, para referencia de distancia fija.

Para esta prueba, se implementa un esquema de saturación para la acción de control que produce el movimiento lineal, y, para la acción de control que produce el movimiento rotacional. Los niveles de saturación se definen en distintos valores, para forzar que el DDMR pueda seguir una trayectoria curva cuando el sensor de distancia queda fuera del rango de detección.

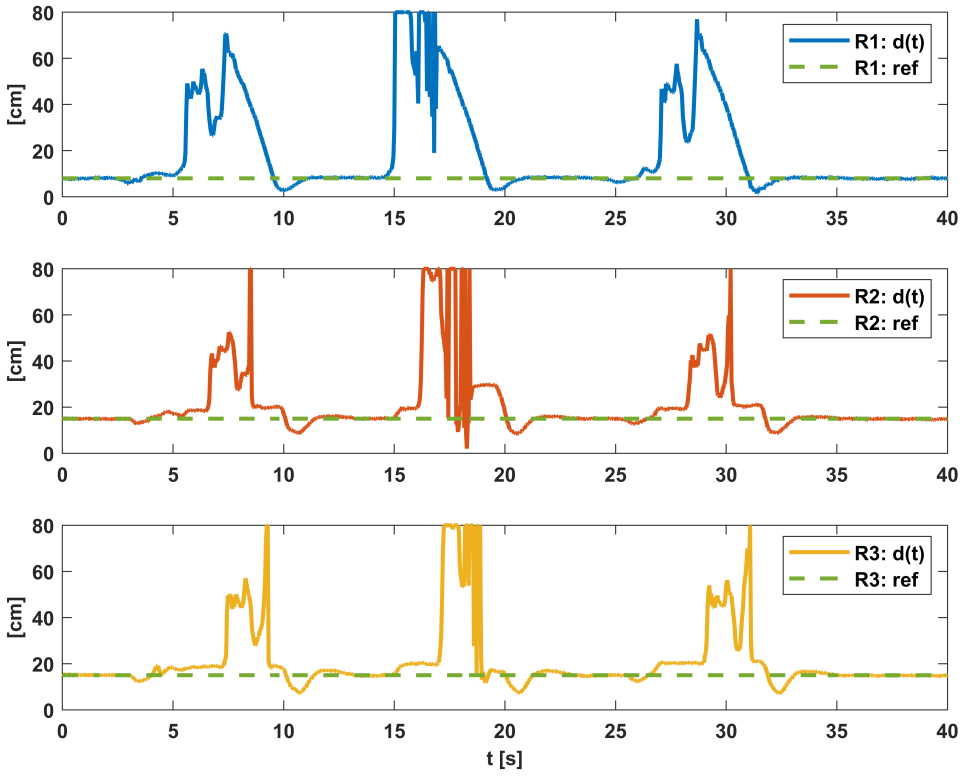


Figura 5.25: Seguimiento de distancia con controlador LQI - perturbaciones con formación fija - trayectoria con curvas

La Figura 5.25 muestra el comportamiento de la medición de distancia cuando se permite que los robots sigan una trayectoria con curvas. Inicialmente, los robots mantienen una formación fija, donde el robot 1 se mantiene a una distancia respecto de un objeto. luego, se posiciona el objeto en una zona recta de la trayectoria de navegación, la cual, se encuentra posterior a una curva. Producto de la ausencia de un objeto frente al robot 1, la medición de distancia aumenta rápidamente por sobre la referencia, esto provoca que el robot 1 comience a moverse alcanzando rápidamente una velocidad máxima, definida por el esquema de saturación. Cuando el robot 1 sale de la curva, la distancia de seguimiento comienza a disminuir progresivamente producto de la presencia del objeto, el robot avanza hasta llegar a la referencia de distancia definida inicialmente y ocurre un pequeño sobrepaso, producto de la capacidad de integración del controlador LQI. Este sobrepaso en la distancia de seguimiento, se atenúa rápidamente por acción del controlador, y

[4]video de la prueba: https://youtu.be/q-zFbPaVuFQ

debido a un esquema de anti-enrollamiento que satura la componente integral del controlador. Los Robots 2 y 3, se comportan de forma similar al robot 1, pero la medición de distancia cambia bruscamente cuando salen de la curva, ya que inmediatamente se encuentran con su robot predecesor.

La posición del objeto se cambia 3 veces durante la prueba. Esto permite identificar un comportamiento no deseado del sensor de distancia, cuando se encuentra fuera de rango. En la segunda vuelta, durante la curva (sensor de distancia fuera de rango), el sensor de distancia del robot R2 entrega de mediciones erróneas que aparecen de forma aleatoria, algunos de los valores que el sensor entrega de forma errada, se encuentran dentro del rango de medición (ver Figura 5.25, en t cercano a 17 s, para el robot R2), esto provoca que el controlador de distancia responda como si el robot estuviese cerca de la referencia, o incluso, como si el robot se hubiese acercado demasiado a su predecesor, lo mismo ocurre con el robot 3 en t cercano a 19 s.

Los robots son capaces de medir su orientación respecto de la trayectoria de navegación, la cual, cambia principalmente cuando el robot navega por sobre una curva. El controlador logra que la orientación se mantenga cercana a los -0.4 rad en las zonas curvas, y que se retorne a 0 rad de forma suave cuando sale de la curva y entra a una recta (ver Figura 5.26).

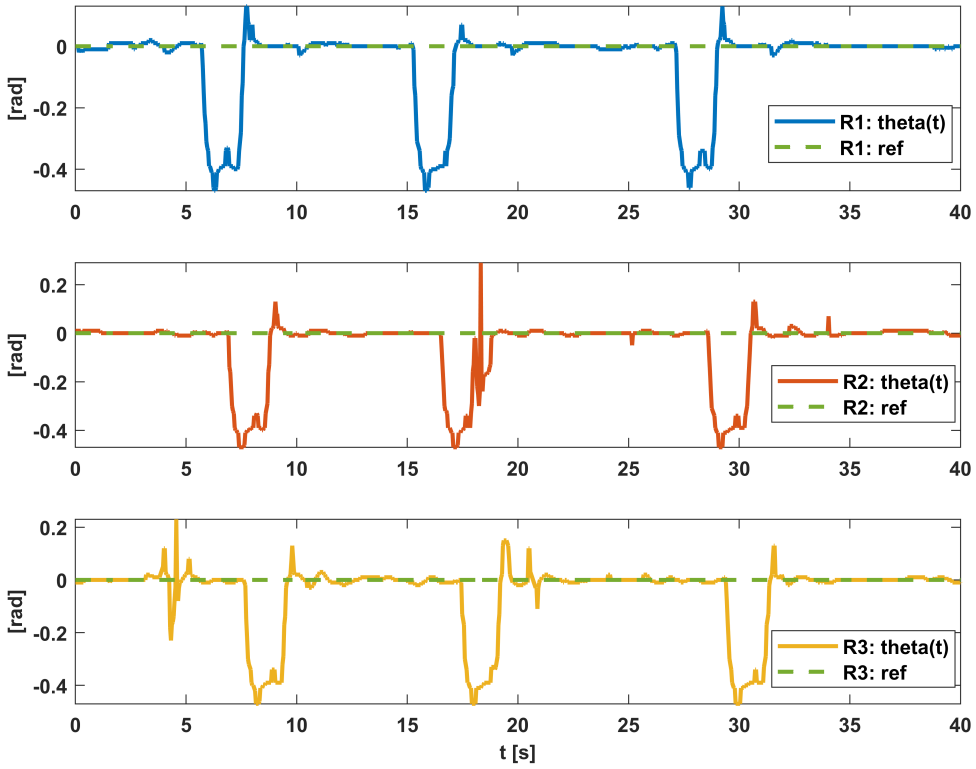


Figura 5.26: Seguimiento de orientación con controlador LQI - trayectoria con curvas - prueba 3

5.7.2. Resultados de navegación con controlador PID

5.7.2.1. Resultados en trayectoria recta - Controlador PID

- Prueba 4[5]: Cambios de referencia a los robots R1, R2 y R3 con un obstáculo fijo en frente de R1.

[5] video de la prueba: https://youtu.be/PFPsYpaMkwI

Para esta prueba los robots parten con una formación fija. El robot R1 se mantiene siguiendo una distancia de referencia respecto a un objeto fijo. Se realizan cambios en la referencia de distancia que mantiene cada robot respecto a su predecesor. En la Figura 5.28, se muestra el comportamiento de las mediciones de distancia de cada robot y su respectiva referencia, donde se puede observar como los robots son capaces de seguir una referencia deseada y como eso afecta al resto de los robots.

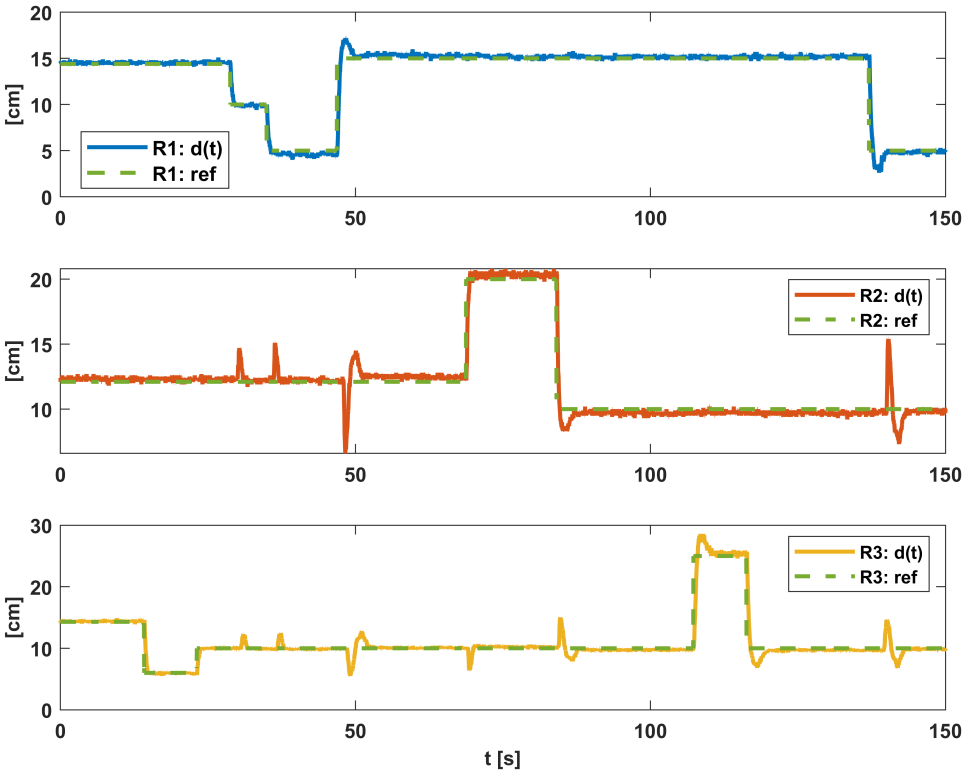


Figura 5.27: Seguimiento de distancia con controlador PID - cambios de referencia - trayectoria recta

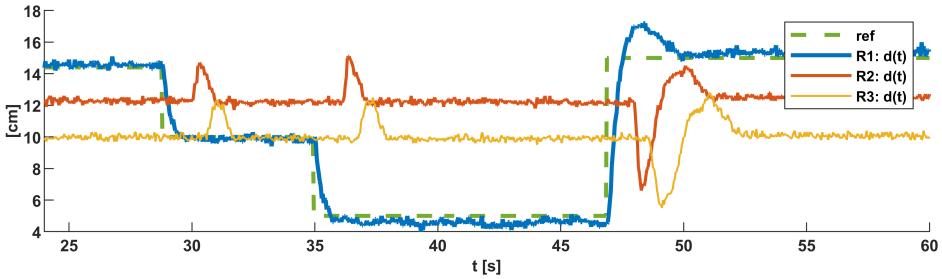


Figura 5.28: Acercamiento de la respuesta temporal del seguimiento de distancia de la Figura 5.27

La Figura 5.28, muestra una sección de los resultados presentados en la Figura 5.27, entre los instantes t=24 s y t=60 s, acá se puede observar con mayor detalle el seguimiento de referencia para el robot R1 y como su movimiento modifica la distancia de seguimiento para el robot R2, el cual, se comienza a desplazar para mantener su referencia de distancia respecto al robot R1. El movimiento del robot R2, provoca que el error de seguimiento para el robot R3 comience a aumentar, lo que produce el

movimiento de este para corregir su formación. En esta prueba no se observa una amplificación del error a medida que se propaga a los demás robots, es decir, el sistema en conjunto es estable.

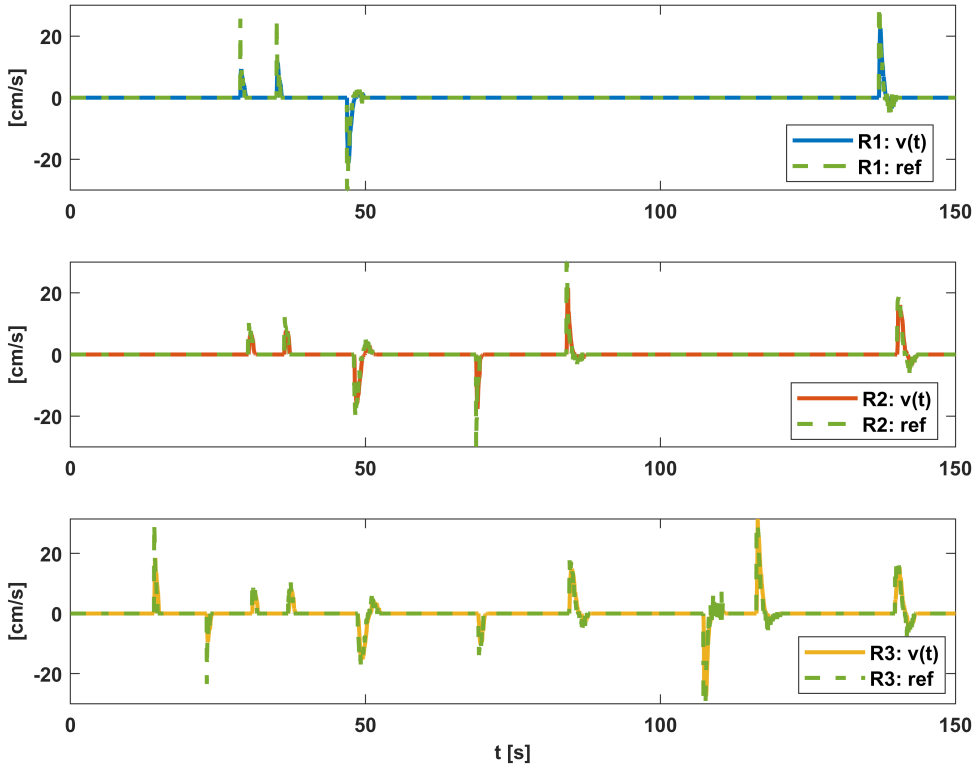


Figura 5.29: Seguimiento de velocidad con controlador PID - trayectoria recta - prueba 4

La Figura 5.29 muestra la medición de la velocidad lineal y su respectiva referencia. Dado que el esquema PID implementado, es una estructura en cascada, la referencia de velocidad la define la salida del controlador de distancia, la cual, corresponde a una señal interna del esquema. A partir de la medición de velocidad se observa directamente la capacidad de los robots de desplazarse hacia adelante y hacia atrás. Durante toda la prueba, se puede observar la capacidad de los robots de seguir referencias de velocidad.

La Figura 5.30, muestra la medición de orientación de los 3 robots, la cual, se mantiene la mayor parte del tiempo en torno a los 0 rad. En algunas ocasiones la medición de orientación alcanza valores cercanos a los 0.05 rad, lo cual, ocurre principalmente para velocidades cercanas a 0, donde el robot puede pasar de moverse hacia adelante o hacia atrás. De todas maneras, el valor máximo alcanzado por la medición de orientación sigue siendo pequeño, lo que implica que el DDMR se encuentra sobre la trayectoria de navegación durante toda la prueba.

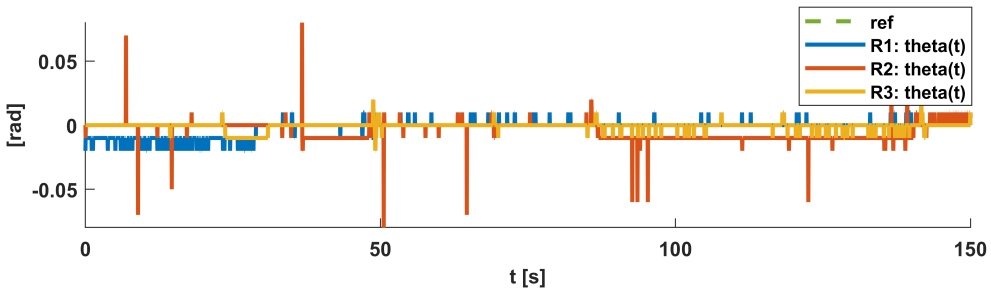


Figura 5.30: Seguimiento de orientación con controlador PID - trayectoria recta - prueba 4

- Prueba 5[6]: Cambios de referencia en la velocidad de navegación del robot R1 (controlador de distancia apagado), con los robots R2 y R3 a una referencia de distancia fija respecto a su robot predecesor.

  En esta prueba, se utiliza la capacidad del esquema de control PID (en cascada), para implementar un lazo de velocidad, apagando el controlador de distancia y definiendo remotamente una velocidad de referencia. Solo el robot R1 navega por la trayectoria siguiendo referencias de velocidad, mientras que los robots R2, y R3, utilizan el controlador de distancia para seguir a su robot predecesor. Como esta prueba se realiza en una trayectoria recta, se definen referencias de velocidad positivas y negativas, para que los robots se mantengan dentro de la trayectoria (ver Figura 5.31). También, se puede observar la velocidad para los robots R2 y R3, pero en este caso, su referencia corresponde a una señal interna del lazo de control, la cual, es regulada por la salida de sus controladores de distancia local.

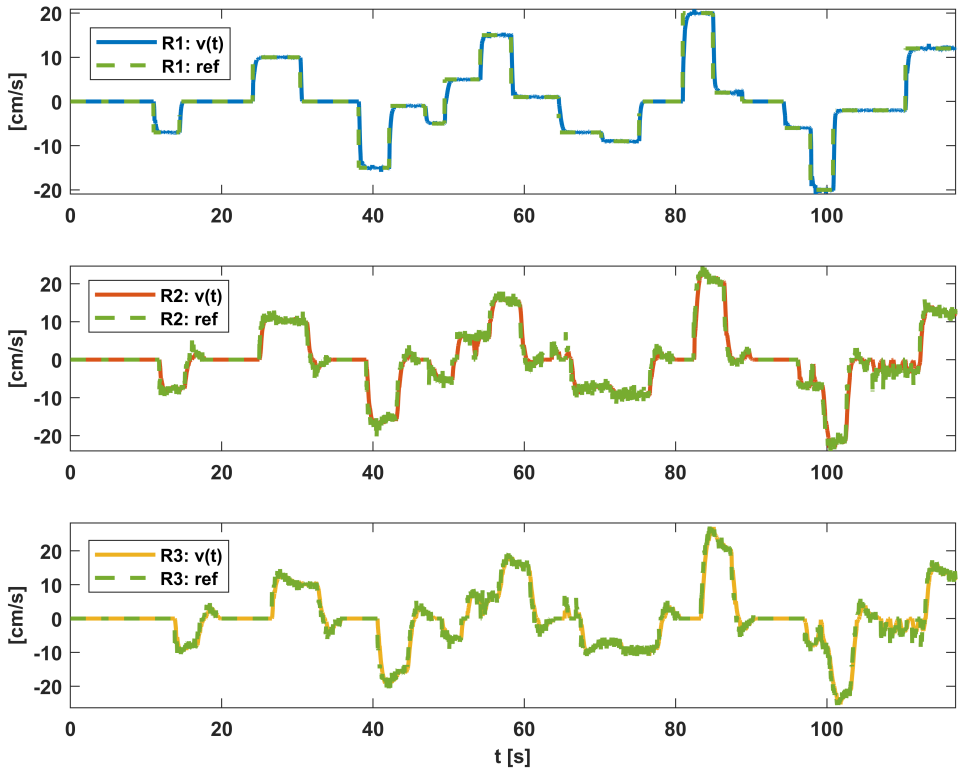


Figura 5.31: Seguimiento de velocidad con controlador PID - cambios de referencia para el robot R1 - trayectoria recta

[6]video de la prueba: https://youtu.be/18n6NMefTAs

Las señales de velocidad de los robots R2 y R3 toma una forma parecida a la velocidad alcanzada por el robot R1, se puede ver como esta se va distorsionando un poco entre cada robot, producto de la respuesta propia del esquema de control descentralizado. Estas velocidades permiten que la distancia de seguimiento para los robots R2 y R3, se mantengan cercanas al valor deseado (ver Figura 5.32). Del mismo modo que en las pruebas anteriores, el robot R2 mantiene su referencia de distancia respecto de R1, para lo cual, debe "imitar" el movimiento de este, al mismo tiempo, el robot R3 actúa en función de la distancia respecto al robot R2, por lo que necesariamente debe desplazarse para mantener su formación. En este caso existe una pequeña propagación del error de seguimiento del robot R2 hacia el robot R3, lo cual, se puede concluir de las amplitudes que alcanza la medición de distancia para cada robot. Dado que el robot R1 actúa como líder, siguiendo referencias de velocidad (definidas por el operador de la plataforma experimental), no es necesario mostrar sus mediciones de distancia.

La orientación de los robots se mantiene cercana a los 0 rad, lo que implica que los robots se encuentran dentro de la trayectoria de navegación, durante toda la prueba (ver Figura 5.33).

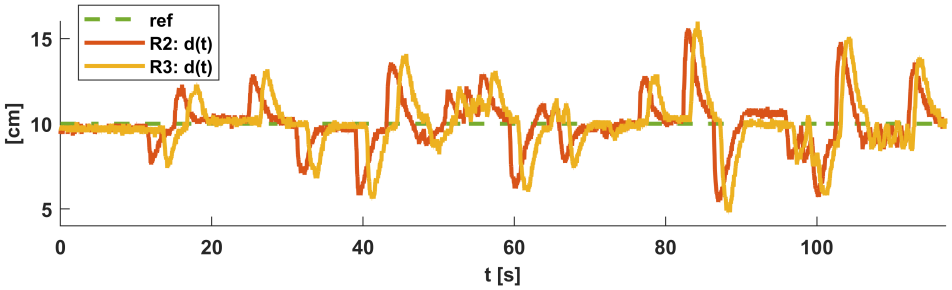


Figura 5.32: Seguimiento de distancia con controlador PID - R1 líder - trayectoria recta

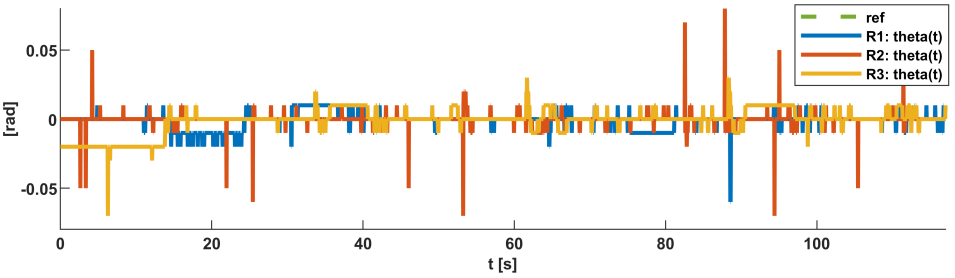


Figura 5.33: Seguimiento de orientación con controlador PID - trayectoria recta - prueba 5

5.7.2.2. Resultados en trayectoria cerrada con curvas - Controlador PID

Prueba 6 [7]: Seguimiento de trayectoria cerrada con curvas, para referencia de distancia fija.

En esta prueba, se evalúa la capacidad de los robots de seguir una trayectoria de navegación cerrada con curvas, para el esquema de control PID. El robot R1 puede navegar a través de la trayectoria cerrada, siguiendo referencias de velocidad, de este modo, los robots son capaces de seguir la trayectoria de navegación, mientras tratan de mantener una formación fija.

Inicialmente, se define una referencia de velocidad para el robot líder de 22 cm/s, mientras que los robots R2 y R3 mantienen una referencia de distancia de 15 cm respecto de su predecesor. En la Figura 5.34, durante los primeros 30 s, se puede observar, como el robot R1 alcanza rápidamente la referencia de velocidad deseada. El robot R2 comienza a aumentar su velocidad, para tratar de mantener su formación. Cuando el robot R1 entra a la curva, el sensor de distancia del robot R2 queda fuera del rango de medición, lo

[7] video de la prueba: https://youtu.be/iG-xn3Ssy3U

que provoca que su referencia de velocidad aumente hasta saturarse. El robot R2 entra a la curva, por lo que, el controlador de orientación genera una actuación en cada motor del DDRM que permite que este se pueda mantener sobre la trayectoria de navegación. El resultado es que una de las ruedas se mantiene girando a la velocidad de saturación para tratar de mantener la velocidad lineal, mientras que la otra rueda del robot debe disminuir su velocidad para provocar el giro del robot. Cuando el robot R2 sale de la curva, inmediatamente el sensor de distancia detecta al agente predecesor, y debido a la integración del error durante la curva, el robot, debe disminuir rápidamente su velocidad, para no colisionar. Lo mismo ocurre para el robot R3.

Luego se realiza un cambio de referencia de la velocidad del robot R1 a 25 cm/s, entre los instantes de t=30 s y t=60 s. El comportamiento del seguimiento de distancia para los robots R2 y R3 comienza a mejorar, lo cual, se puede observar en la Figura 5.35 entre los instantes de tiempo mencionados anteriormente, donde la medición de distancia de ambos robos, cuando salen de la curva, es cada vez más próxima a la referencia, en la medida que la velocidad del robot líder se acerca a los límites de velocidad máxima definida para los robots.

Cuando se define una velocidad de referencia para el robot R1, de 30 cm/s, se puede observar como el controlador es capaz de llegar a esa referencia, pero solo en las secciones rectas de la trayectoria (ver Figura 5.34 entre los instantes t=60 s y t=110 s), puesto que como se dijo anteriormente, en las zonas curvas, una de las ruedas disminuye su velocidad, y la otra no puede aumentar más producto de la saturación de la señal de control. El robot 2 comienza a alejarse progresivamente de la referencia (ver Figura 5.35 entre los instantes t=60 s y t=110 s), puesto que el sensor en algunas ocasiones introduce mediciones erróneas que provocan que la velocidad del robot R2 disminuya. El robot 3 por su parte, puede seguir sin mayor problemas la referencia de distancia respecto a su predecesor, y su formación al salir de la curva es próxima a la referencia.

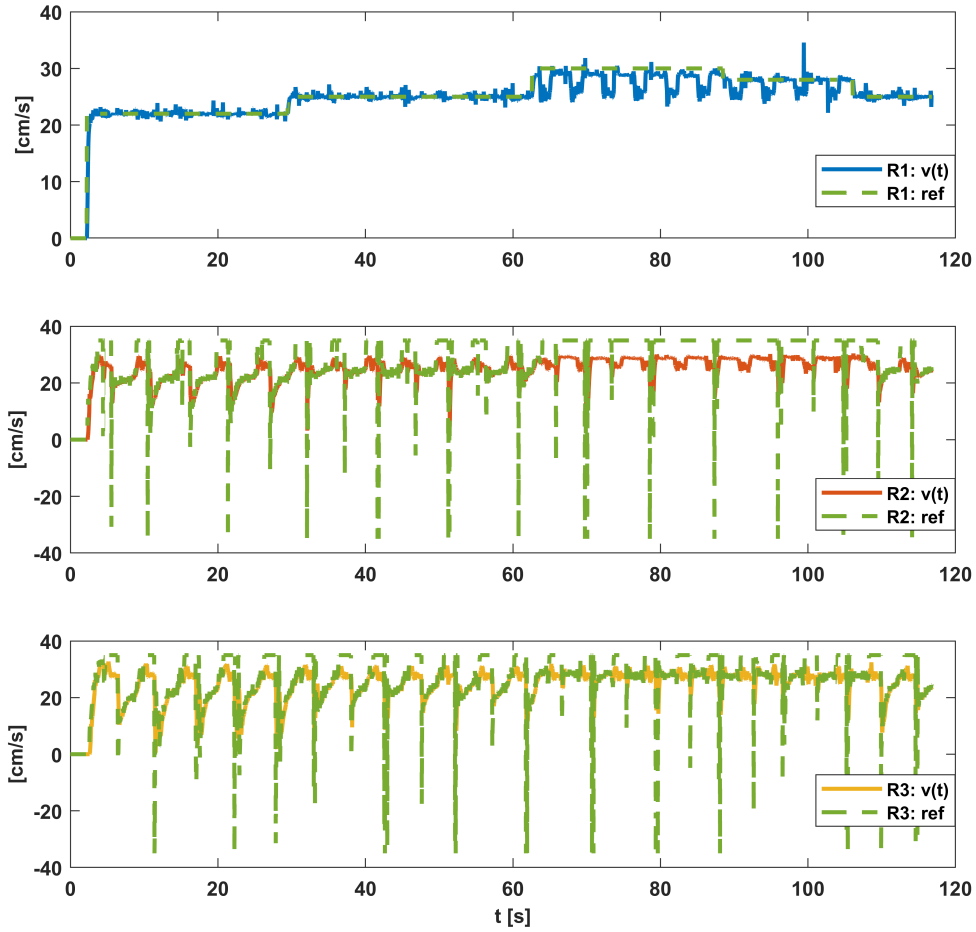


Figura 5.34: Resultados del seguimiento de velocidad con controlador PID

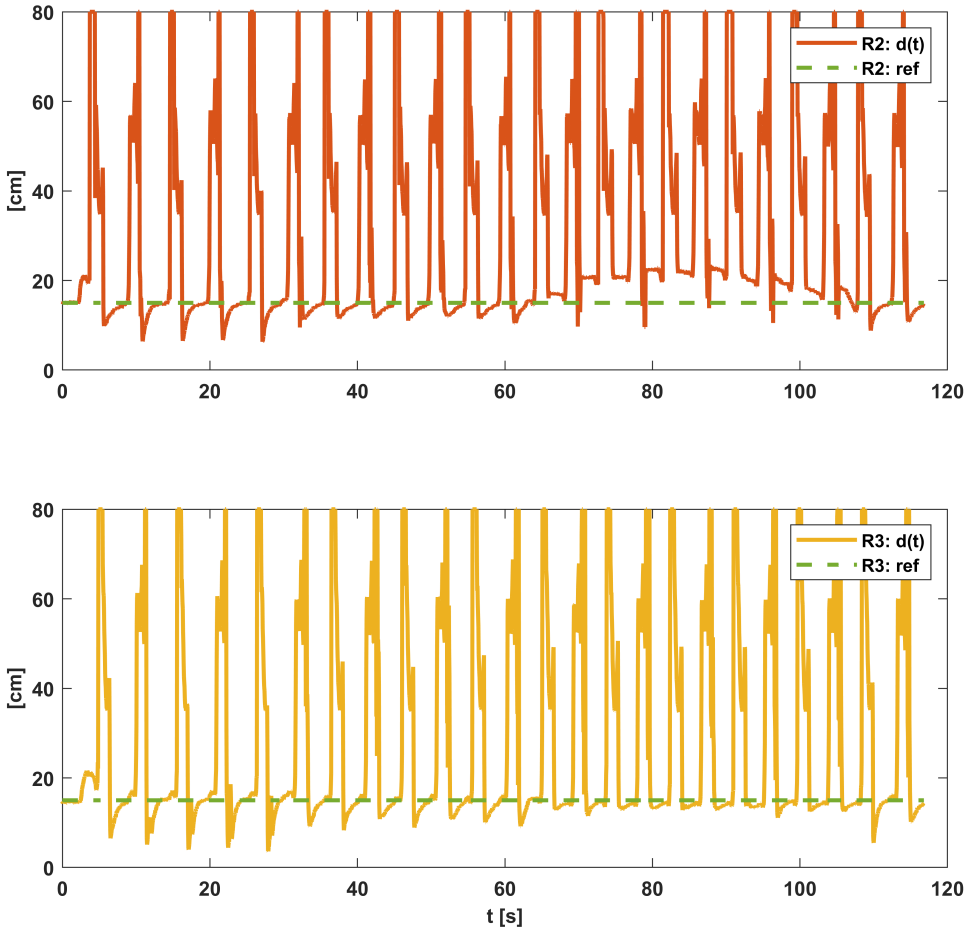


Figura 5.35: Resultados del seguimiento de distancia con controlador PID

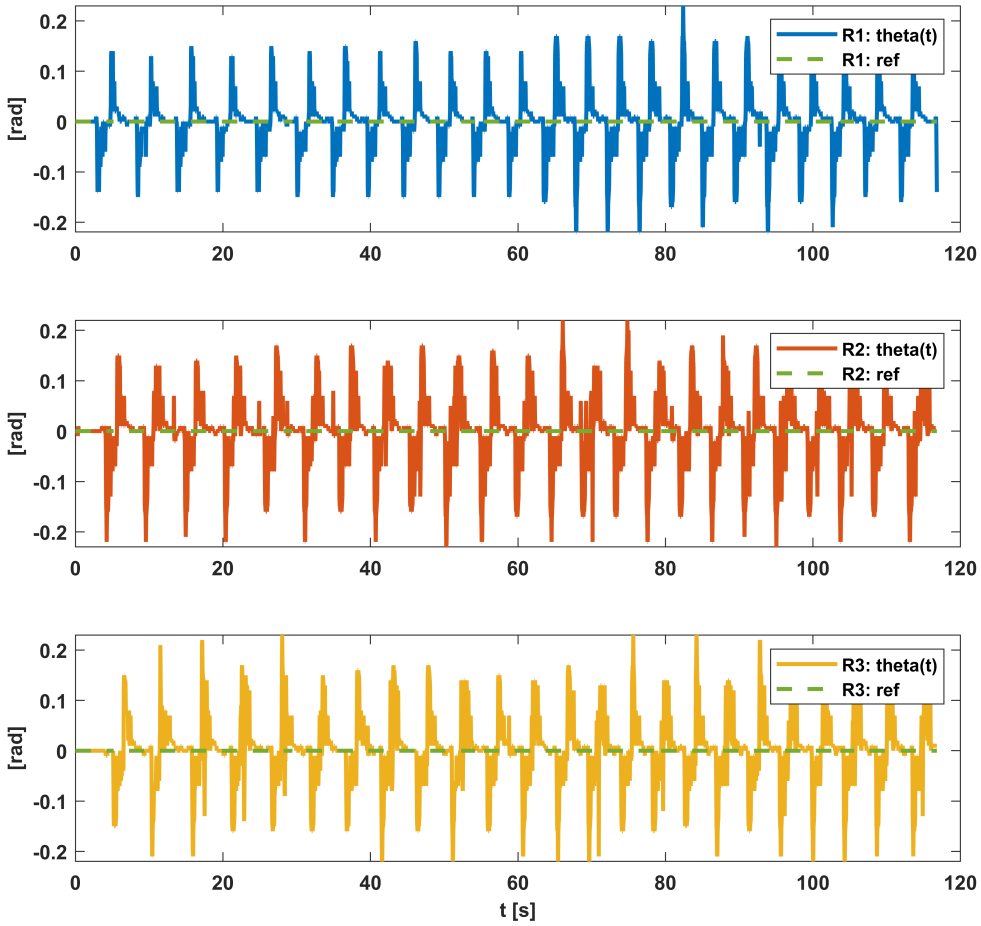


Figura 5.36: Resultados del seguimiento de orientación con controlador PID

5.7.3. Resultados de navegación con controlador PID y comunicación R2R

5.7.3.1. Resultados de seguimiento de velocidad, con distancia constante entre robots - PID y comunicación R2R

Prueba 7 [8]: Seguimiento cooperativo de trayectoria cerrada con curvas, manteniendo formación inicial constante.

En esta prueba, se define una distancia de seguimiento inicial de 15 cm, para los robots R2 y R3 (formación inicial), luego se define una velocidad de referencia para el robot R1. Durante la prueba se realizan cambios de referencia de velocidad del robot R1 (ver Figura 5.37), además cada robot recibe la velocidad de su robot predecesor, lo cual, es utilizado localmente para ayudar a mantener la formación inicial a lo largo de la trayectoria de navegación.

Como primera observación, el uso de la información compartida de forma inalámbrica entre los robots permite extender la navegación de la flota de robots a través de la trayectoria cerrada a velocidades más bajas, puesto que, el esquema de control usa la velocidad del robot predecesor para saturar la actuación del

[8] video de la prueba: https://youtu.be/eNRIYkTCwRM

controlador local, esto permite que el robot no alcance velocidades mucho mayores que su robot predecesor y con esto mantener la formación, incluso cuando el sensor de distancia queda fuera del rango de medición. En pruebas anteriores, cada robot calcula de forma local la cantidad de voltaje aplicado a cada uno de sus motores para mantener una velocidad de referencia, sin tomar en consideración la velocidad de su robot predecesor, esto provoca que en las zonas donde el sensor de distancia queda fuera de rango, el robot calcule una velocidad mayor a la de su robot predecesor provocando oscilaciones para tratar de mantener la formación deseada, en ocasiones estas oscilaciones terminan en colisiones, o, en movimientos que sacan al robot de la trayectoria de navegación.

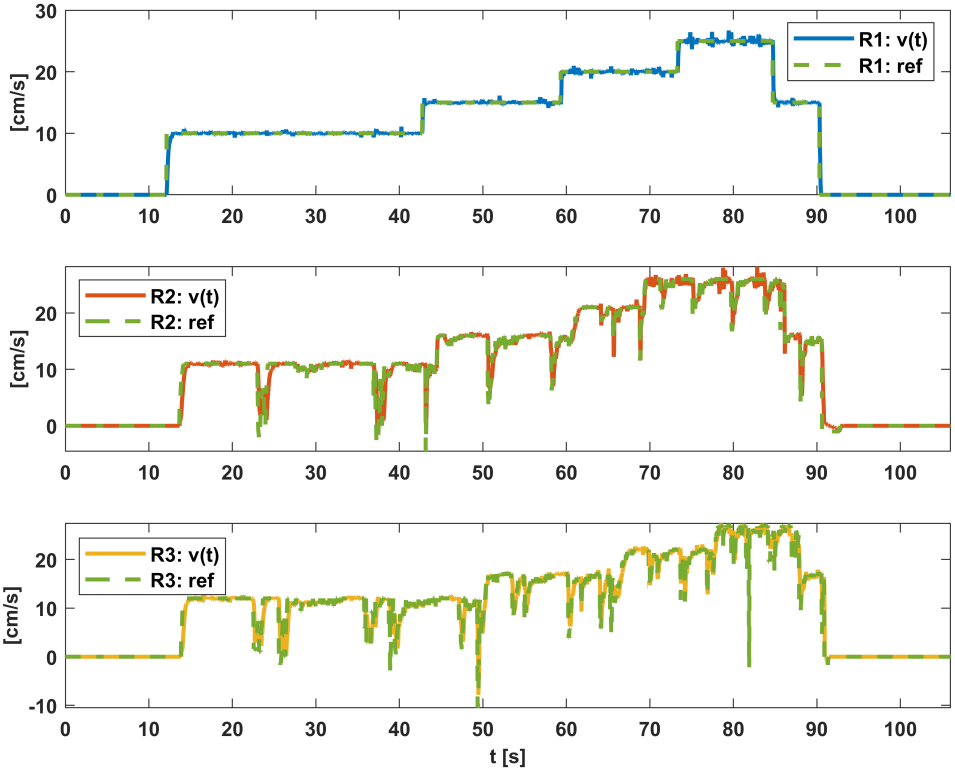


Figura 5.37: Resultados del seguimiento de velocidad con controlador PID

La velocidad máxima que alcanzan los robots R2 y R3 es similar a la velocidad del robot R1, incluso durante las zonas curvas de la trayectoria. Cuando el robot R2 sale de la zona curva y es capaz de medir la distancia respecto de su robot predecesor R1, puede calcular localmente la velocidad que debe tener para mantener la formación de referencia. En este caso se hace evidente que el robot necesita detenerse o disminuir la velocidad unos instantes para volver a la distancia de referencia. Este cambio en la referencia de velocidad en el robot R2 es comunicado al robot R3, por lo que en ese instante el robot no puede alcanzar velocidades mayores y se mantiene a una menor velocidad hasta que el robot R2 alcanza nuevamente la formación de referencia. El resultado de no permitir a los robots alcanzar velocidades mucho mayores que la velocidad de su robot predecesor es que visualmente los robots son capaces de mantener una formación de referencia la mayor parte del tiempo, esto se puede comprobar a partir de las mediciones de distancia local de cada robot, pero es necesario tener en consideración que el sensor de distancia solo es capaz de medir esta magnitud en una sola dirección por lo que en las zonas curvas no podrá medir la distancia a la cual se encuentra su robot predecesor.

La Figura 5.38 muestra las mediciones de distancia de los robots R2 y R3. Los cambios repentinos en la medición que se saturan en los 80 cm están directamente relacionados con la entrada del robot predecesor

a una zona curva, por lo cual, la medición aumenta instantáneamente. En dicho instante el robot mantendrá una velocidad similar a la de su robot predecesor, lo que provoca que al salir de la zona curva, el sensor mide distancias cercanas a la referencia, luego el controlador es capaz de realizar las acciones de control necesarias para corregir dicha formación. Cuando la velocidad de navegación es menor, se aprecia en las mediciones fuera de rango, puesto que la duración es mayor debido a que la curva posee la misma longitud.

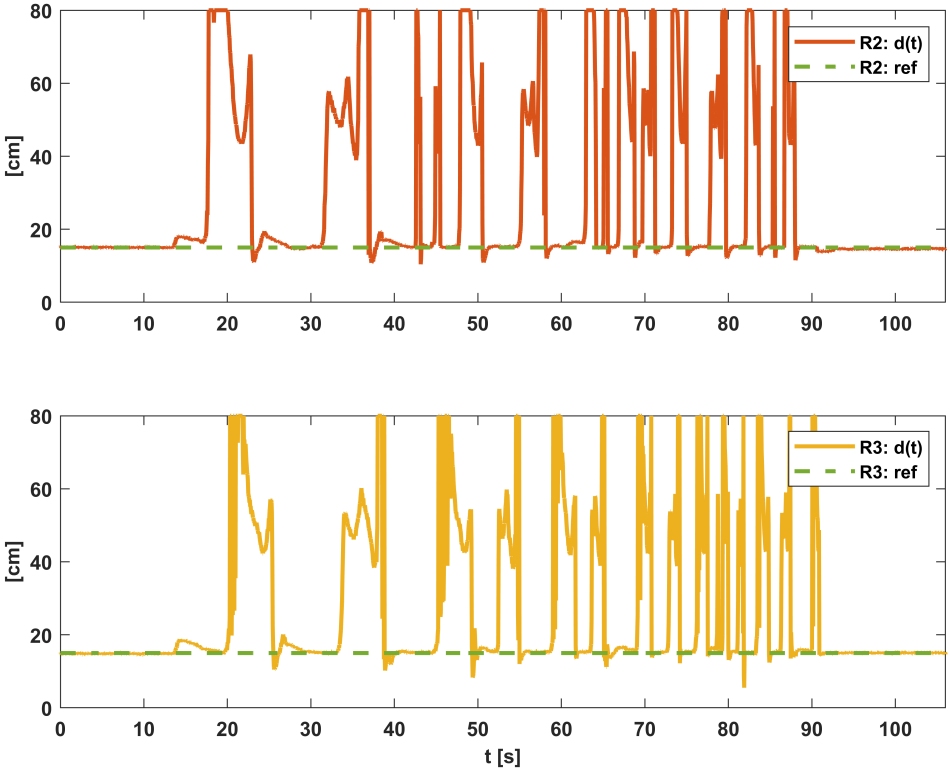


Figura 5.38: Resultados del seguimiento de distancia con controlador PID

La comparación del desempeño obtenido cuando los robots no cooperan entre sí, se puede observar a partir de los resultados anteriores, pero se aprecia directamente en la Figura 5.35, en la cual, la medición de distancia que registran los sensores de los robots R2 y R3 cuando estos salen de las zonas curvas, se encuentra más alejada de la referencia, al punto de casi colisionar (medición cercana a los 0 cm) en dichas zonas, cuando la velocidad de referencia del robot R1 es menor.

El comportamiento de la orientación de los robots respecto de la trayectoria de navegación se muestra en la Figura 5.39, donde se puede observar el efecto del cambio de curvatura de la trayectoria como una perturbación de dicha orientación, la cual, se mantiene acotada en ± 0.2 rad.

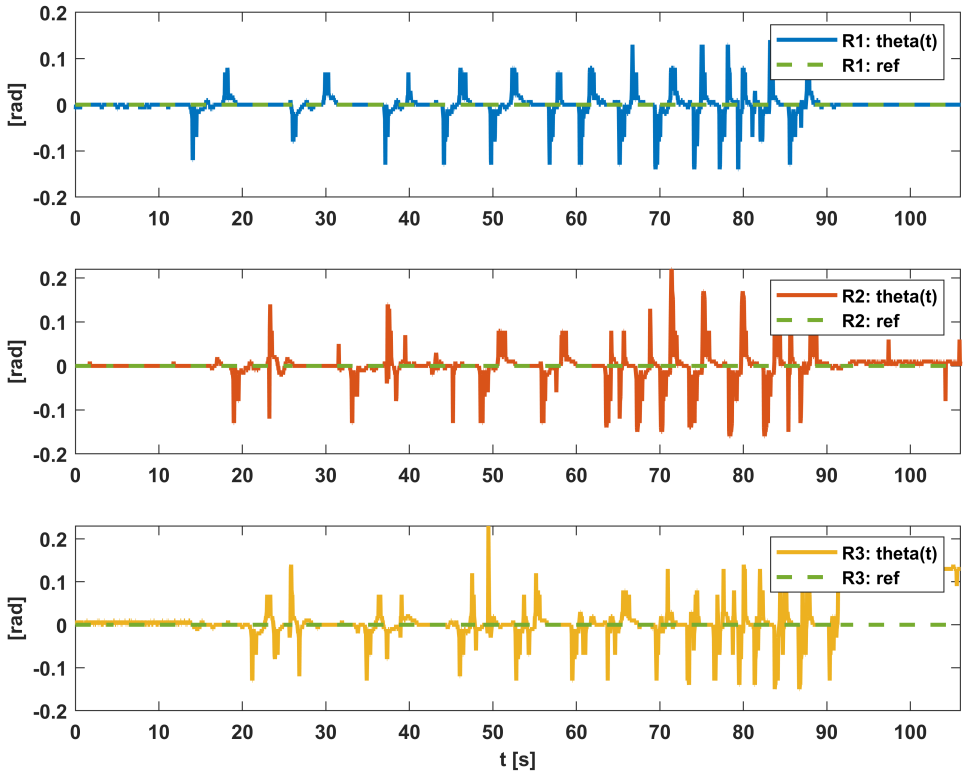


Figura 5.39: Resultados del seguimiento de orientación con controlador PID

5.7.3.2. Resultados de seguimiento de distancia, con velocidad de navegación constante- PID y comunicación R2R

Prueba 8 [9]: Seguimiento cooperativo de trayectoria cerrada con curvas, manteniendo la velocidad de navegación constante.

Durante esta prueba, se define una referencia de velocidad de 20 cm/s constante para el robot R1, y se inicia la prueba con los robots manteniendo una distancia de 8 cm entre ellos. Luego se realiza un cambio en la distancia de seguimiento del robot R3 de 20 cm, en aquel instante los robots R1 y R2 mantienen una distancia de 8 cm, mientras que los robots R2 y R3 se mantienen distanciados en 20 cm. Luego se define la referencia de distancia del robot R2 en 20 cm, con lo cual, los robots se mantienen a una distancia similar navegando a través de la trayectoria. Posteriormente, Se define una referencia de distancia para el robot R3 de 8 cm, de esta forma los robots R2 y R3 se mantienen cercanos, mientras que el robot R1 se mantiene más alejado del resto, pero manteniendo la formación definida. Finalmente, se define la referencia de distancia del robot R2 en 8 cm, con esto se consigue que los robots R2 y en consecuencia el robot R3 se acerquen progresivamente a la referencia definida, logrando mantener la formación deseada.

La Figura 5.40 muestra la velocidad de navegación de los robots R1, R2, y R3. La velocidad del robot R1 está controlada directamente, con el propósito de seguir una referencia de velocidad definida de forma externa (operador), esta se mantiene la mayor parte de la prueba en 20 cm/s, puesto que el objetivo de esta es evaluar la capacidad de los robots de seguir referencias de distancia al mismo tiempo que navegan a través de la trayectoria. Al final de la prueba la velocidad del robot R1 disminuye de forma escalonada, dicho cambio de referencia tiene el propósito de detener de forma ordenada a los robots.

[9] https://youtu.be/y85RmEPikzc

La velocidad del robot R2 se mantiene la gran parte del tiempo en torno a los 20 cm/s, se pueden observar disminuciones periódicas de la velocidad que son provocadas por la compensación introducida por el controlador para corregir la distancia de seguimiento cuando los robots salen de la zona curva de la trayectoria. También se puede observar como cambia la velocidad del robot R2 cuando se define una distancia de seguimiento mayor, la cual, disminuye rápidamente para llegar a la referencia deseada. Con el robot R3 ocurre algo similar, pero en este caso existen perturbaciones no deseadas, introducidas por el sensor de distancia cuando se encuentra fuera de rango de medición. Estas perturbaciones provocan un incremento indeseado en el error de seguimiento de distancia (ver Figura 5.41, y ocurren principalmente a mayor distancia de seguimiento, producto de que en este caso, el sensor de distancia permanece más tiempo fuera de rango.

En la Figura 5.42 se muestra la medición de orientación de cada robot respecto de la trayectoria de navegación. El comportamiento es similar al obtenido en resultados anteriores, donde se puede ver una señal periódica que varía en torno a los 0 rad, y representa la geometría de la trayectoria de navegación, alcanzando los valores máximos y mínimos al entrar y salir de las zonas curvas respectivamente.

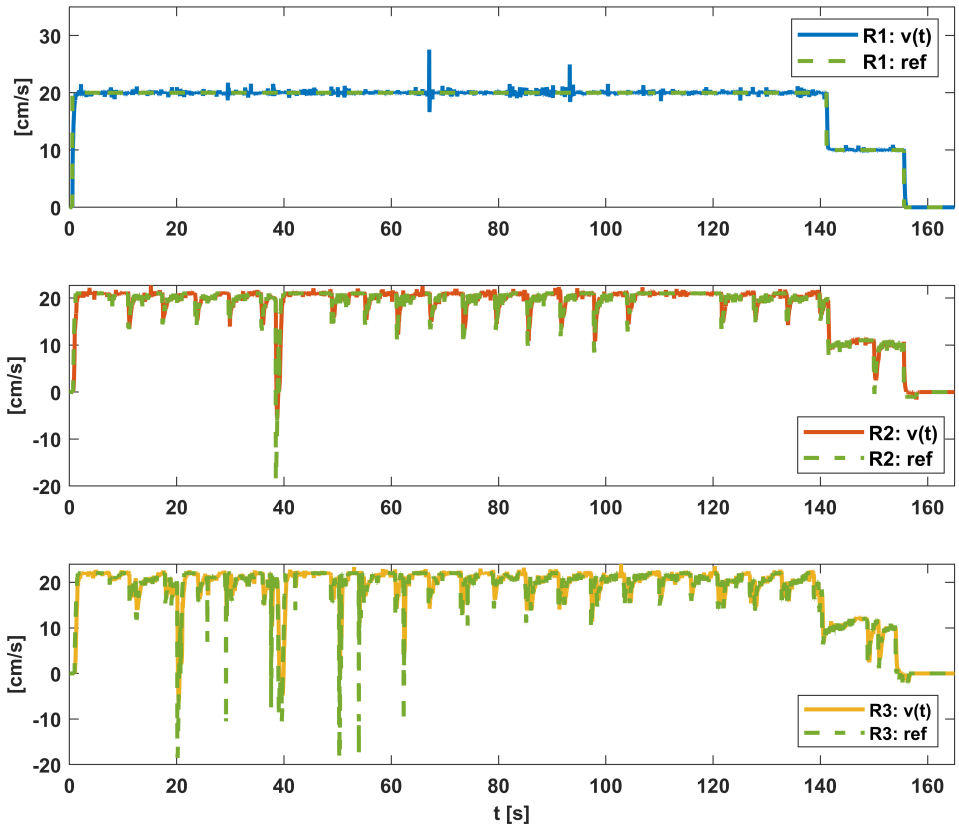


Figura 5.40: Resultados del seguimiento de velocidad con controlador PID

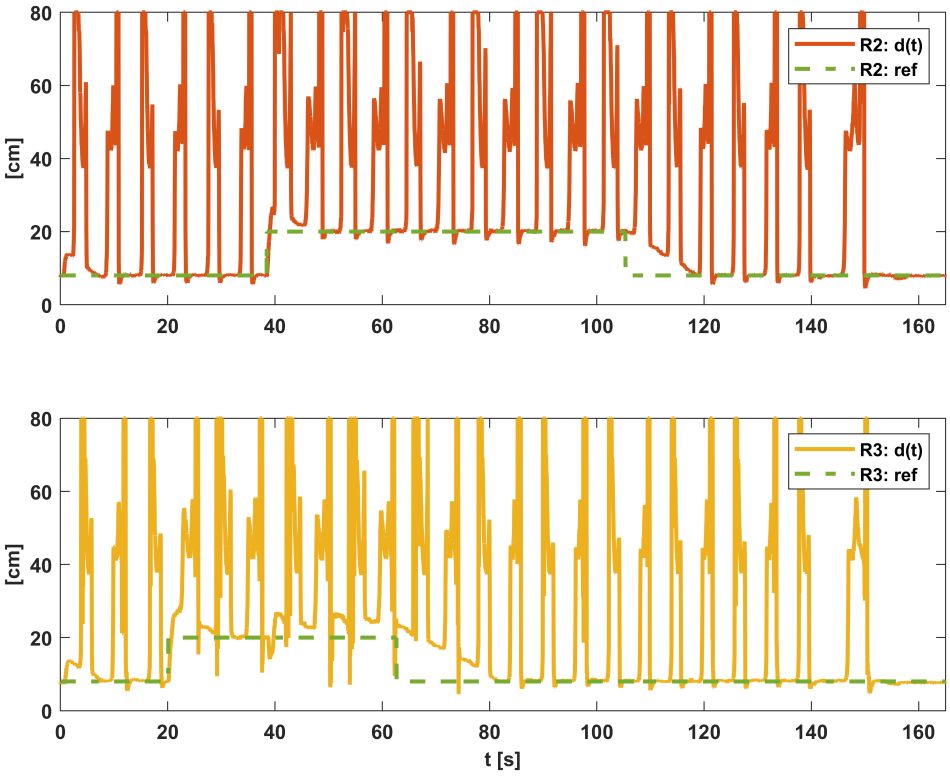


Figura 5.41: Resultados del seguimiento de distancia con controlador PID

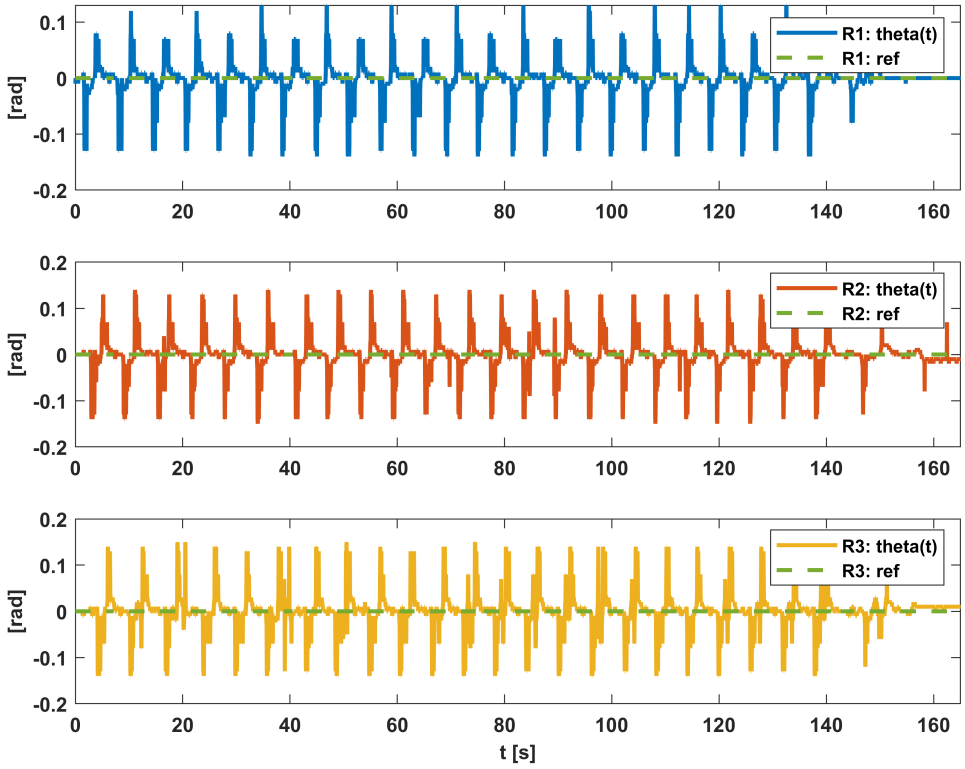


Figura 5.42: Resultados del seguimiento de orientación con controlador PID

6 | CONCLUSIONES

6.1. Resumen

En la práctica, el estudio de sistemas de control trae consigo desafíos de implementación, debido a las características y comportamiento no lineal de la mayoría de componentes (sensores, actuadores, comunicación, microcontrolador, baterías, etc.). Resolver cada uno de estos desafíos locales permite resolver problemas más complejos y garantizar un buen desempeño de un sistema de control.

Hoy en día las plataformas experimentales para el estudio de flotas de robots móviles generan un gran interés en académicos, investigadores y estudiantes de diversas áreas de la electrónica, en primer lugar debido a que estas otorgan un escenario sobre el cual se pueden implementar diversos esquemas de control, máquinas de estados, algoritmos, topologías de comunicación entre los robots, donde la implementación trae consigo problemas propios del entorno real, en el cual, existe ruido en las mediciones, comportamiento no lineal de sensores, pérdida de datos en la comunicación, velocidad de muestreo limitado, retardos en las mediciones, etc. En segundo lugar, este tipo de plataformas permite visualizar problemáticas reales referentes a la navegación autónoma y cooperativa de flotas de robots móviles, los que a su vez pueden representar automóviles navegando por una carretera, como también robots dedicados al transporte de materias primas en la industria, o emular cualquier escenario de navegación autónoma y/o cooperativa, para realizar estudios y/o desarrollar soluciones desde la plataforma experimental. Por último, este tipo de plataformas genera un impacto positivo en actividades de difusión y docentes, ya que, se pueden mostrar las principales áreas de la electrónica y su interacción analizando sus componentes, también se puede emplear en actividades prácticas o de laboratorio, a fin de complementar la formación de futuros profesionales del área.

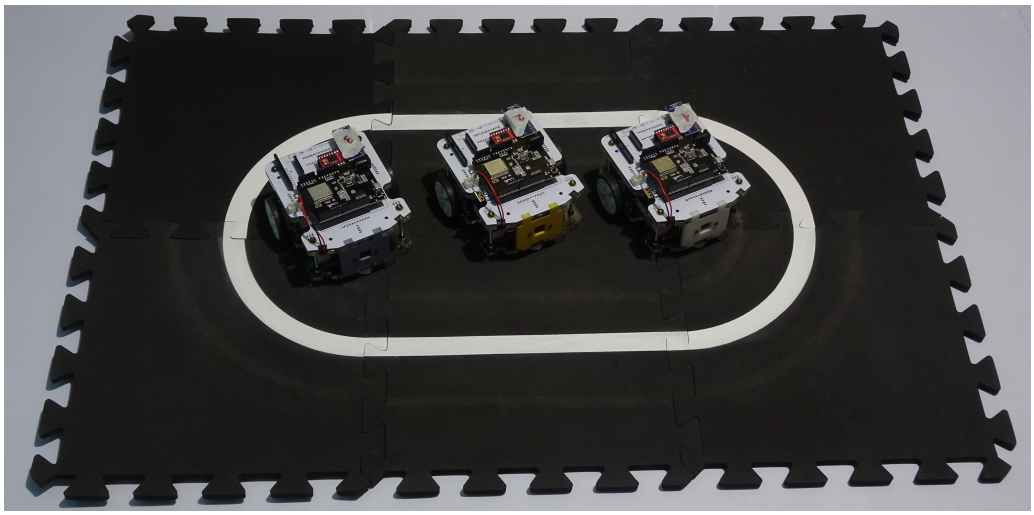


Figura 6.1: PMA: Plataforma multi-agentes

6.2. Conclusiones

La plataforma experimental presentada reúne las características y capacidades para cumplir con cada uno de los objetivos de este trabajo. En primer lugar, la superficie y trayectoria de navegación modular permite crear diversos escenarios de operación utilizando un número limitado de módulos, como también crear escenarios de mayor extensión y complejidad. Lo anterior posibilita que el escenario de navegación se adapte al entorno donde sea utilizada la plataforma experimental y facilita su traslado y almacenamiento. En segundo lugar, el diseño y selección de componentes del DDMR permite crear réplicas idénticas y fabricar la cantidad deseada de robots móviles diferenciales. Los componentes del DDMR le otorgan la capacidad de movimiento, detección de patrones en la superficie, medición de distancia de obstáculos o agentes en frente, medición de posición y velocidad angular de cada rueda, comunicación con otros robots, comunicación con el entorno, procesamiento de datos, implementación de automatismos y esquemas de control, entre otras.

El uso del software de diseño Autodesk Fusion 360 ayuda a diseñar el tamaño y forma del DDMR, importar y/o diseñar cada uno de sus componentes y también permite importar modelos CAD de tarjetas electrónicas (PCB) diseñadas en el software Autodesk Eagle. Estas características lo hacen una herramienta útil para el diseño del DDMR, donde las componentes estructurales se pueden exportar en archivos de fabricación *.svg* o *.stl*, para luego mecanizar estas partes utilizando servicios de corte láser o impresión 3D respectivamente.

Analizar el modelo del DDMR desde un punto de vista cinemático resulta en ecuaciones que describen el movimiento del robot sin considerar las dinámicas de las fuerzas o torques que lo provocan, a partir de escenarios de operación simples y fáciles de comprender. Luego el modelo dinámico del movimiento del DDMR se aborda a partir de las ecuaciones de Euler-Lagrange, el cual, se construye a partir de la energía total del robot, y se puede extender a otro tipo de sistemas, teniendo en consideración la analogía que existe entre los diferentes sistemas físicos (mecánico lineal, mecánico rotacional, eléctrico, hidráulico, etc.). Las ecuaciones de E-L resultan de gran utilidad al momento de modelar un sistema en movimiento, ya que, son independientes del sistema de coordenadas. Al considerar la energía total del DDMR (mecánica y eléctrica) desde el sistema de coordenadas local definido en el centro de giro del robot, y considerando modelos lineales para el comportamiento de las componentes del sistema (resistencia, inductancia, masa, roce, inercia, etc.) se logra obtener un modelo dinámico lineal que describe la dinámica de la velocidad lineal y rotacional del robot en función del voltaje aplicado a cada uno de los motores. El modelo resultante se puede expresar en el espacio de estados reordenando las ecuaciones obtenidas y definiendo las matrices y vectores correspondientes, es necesario construir la ecuación de salida a partir de los estados definidos. Obtener una representación en espacio de estados permite analizar la interacción entre distintos modelos en espacio de estados de forma directa, puesto que en este se definen de forma explícita las entradas y salidas del sistema (a diferencia de E-L donde las salidas de interés están descritas de forma implícita). Además, resulta sencillo derivar el modelo obtenido en otros modelos que representen la dinámica de otras salidas del sistema real, cuando las nuevas salidas de interés dependen de combinaciones lineales de los estados del modelo inicial.

Utilizar instrumentos de medición como pie de metro, báscula, multitester, permite medir de forma directa algunos de los parámetros del DDMR, también es de gran utilidad utilizar el software de diseño CAD para estimar otros parámetros físicos como dimensiones y el centro de masa del robot. Analizar la respuesta a escalón de las señales de voltaje, corriente y velocidad en estado estacionario permite determinar los parámetros que relacionan a dichas magnitudes físicas, mientras que analizar la respuesta transitoria de aquellas señales entrega información de los modos del sistema, lo cuales, están directamente relacionados con los parámetros del modelo. Lo anterior se facilita mediante el uso de un osciloscopio, y analizando cada subsistema por separado.

El software MATLAB es una herramienta computacional que facilita y complementa el estudio de las propiedades de los sistemas dinámicos. También resulta útil al momento de diseñar controladores, y de observar de forma gráfica las respuestas temporales obtenidas de las señales de interés en un lazo de control. La programación mediante diagramas de bloques que ofrece el entorno Simulink ayuda a diseñar esquemas de control complejos a partir de bloques individuales que contienen información de los modelos, controladores, ganancias, operadores matemáticos, señales temporales, entre otros. También permite analizar la evolución

temporal de las señales de interés del lazo de control a través de bloques tipo *scope* que muestran de manera gráfica la respuesta temporal de las señales.

Utilizar aproximaciones a tiempo discreto de los modelos diseñados en tiempo continuo, para sensores, actuadores y controladores, permite implementar dichos modelos en el microcontrolador del DDMR, independiente del lenguaje de programación utilizado. Siendo esta una alternativa sencilla para explorar y evaluar las capacidades de la plataforma experimental. Concluyendo que los modelos implementados funcionan correctamente siempre y cuando los sistemas trabajen en un punto de operación adecuado y en el entorno para el cual fueron diseñados, para ejemplificar lo anterior, se puede considerar el sensor que permite medir el ángulo de orientación del DDMR respecto de la trayectoria de navegación, donde este medirá correctamente la orientación siempre y cuando se encuentre sobre la trayectoria de navegación. Lo mismo ocurre con el sensor que mide la distancia entre el DDMR y un agente en frente del mismo, el sensor obtendrá lecturas válidas siempre y cuando el agente externo se encuentre dentro del rango de medición del sensor. Por lo que la implementación de algoritmos para saturar las mediciones que se encuentran fuera del rango de operación, e implementar esquemas de histéresis para discriminar mediciones válidas, ayudan a mejorar el desempeño de los sistemas de control implementados. Los parámetros del controlador definen una dinámica determinada para el sistema en lazo cerrado y dependen en gran medida de los requerimientos del problema a resolver, donde el diseño de los parámetros está sujeto de forma general a un compromiso entre la velocidad de convergencia de las salidas de interés y el esfuerzo de las señales de control. El movimiento lineal y rotacional del DDMR dependen de una combinación lineal de las entradas del sistema, por lo que las dinámicas están acopladas entre sí. Cada uno de los controladores presentados permite definir en cierta medida las dinámicas de cada salida de interés (dado que las dinámicas están acopladas) y definir por ejemplo un comportamiento más rápido frente a cambios en la orientación del robot, y más lento para cambios en la distancia de seguimiento. Lo anterior es de gran utilidad, puesto que en trayectorias con curvas el sensor de distancia queda fuera del rango de detección cuando el robot predecesor entra a una curva desde una trayectoria recta, haciendo que el error de seguimiento aumente rápidamente, lo que implica que el desplazamiento lineal sea más agresivo que el movimiento rotacional y el robot no pueda seguir la trayectoria curva. La implementación de esquemas de saturación de las señales de control y anti-enrollamiento de la acción integral de los controladores, permite definir límites de operación para los controladores independiente de la dinámica definida por sus parámetros, esto ayuda a mejorar el desempeño del sistema de control, además si se definen distintos niveles de saturación para cada señal de control o para cada combinación lineal de las señales de control se puede lograr por ejemplo que el movimiento rotacional sea más agresivo que el movimiento lineal, sin sacrificar necesariamente la velocidad de convergencia.

Diseñar un esquema de control que incorpore la información de la velocidad lineal del DDMR en un controlador adicional conectado en cascada al controlador de distancia, permite que el robot tenga la capacidad de seguir una distancia respecto a un agente predecesor, o seguir una referencia de velocidad (*apagando* el controlador de distancia y definiendo una referencia de velocidad deseada). De esta forma, utilizando solo un esquema de control se puede definir un robot líder de la flota, el cual, navegará por la trayectoria a una velocidad determinada, mientras que el resto de robots de la flota navegarán siguiendo una distancia de referencia determinada.

Para un conjunto de parámetros de los controladores, que permitan un rápido seguimiento de referencia de distancia y orientación, la velocidad en que se desplace el robot líder sobre la trayectoria de navegación puede provocar que los robots de la flota colisionen o se pierdan en las zonas curvas de la trayectoria. Por lo que otorgar a cada robot la capacidad de obtener, mediante comunicación inalámbrica, la velocidad de navegación de su robot predecesor, permite utilizar esa información para regular la actuación del controlador de distancia de forma local, de esta forma se puede lograr que el desempeño del seguimiento de distancia y orientación sobre la trayectoria mejore y no dependa de la velocidad del robot líder (punto de operación).

El protocolo de comunicación UDP, utilizado para establecer la comunicación entre los robots, resulta ser un medio de comunicación efectivo para la topología de red implementada, donde cada robot recibe la información de interés desde su robot predecesor (cascada). Dado que el protocolo no requiere establecer conexiones entre los dispositivos (cliente-servidor) porque la dirección de envío está contenida en el paquete UDP, la tasa de transmisión de datos es lo suficientemente alta para enviar periódicamente la información de interés de un robot a otro y así compensar las posibles pérdidas de datos características de este protocolo.

Adicionalmente, el protocolo es utilizado para la obtención de datos de cada robot en línea, donde cada uno envía un conjunto local de variables cada vez que los controladores se actualizan, a un dispositivo externo (computador) conectado a la misma red WiFi. Gracias a esto se puede hacer procesamiento de los datos obtenidos, generar gráficos y analizar cuantitativamente el comportamiento de la flota de robots móviles. El protocolo también es utilizado para configurar de forma remota los parámetros de los controladores y algunas variables que permiten activar estados de operación de cada robot.

6.3. Trabajo Futuro

A fin de dar continuidad al esfuerzo invertido en el desarrollo de esta memoria, es necesario identificar las posibles líneas de trabajo futuro que se pueden derivar tanto del propio uso de la plataforma experimental desarrollada, como de los temas tratados durante el desarrollo de esta memoria. A continuación se presenta un listado con algunas de las principales líneas de investigación, además se sugieren algunos desarrollos específicos que complementen el uso de la plataforma experimental.

- Desarrollar una interfaz gráfica (HMI) para el monitoreo de señales y operación remota de la plataforma experimental.
- Desarrollar en mayor profundidad el diseño de controladores a fin de mejorar el desempeño y la elección de sus parámetros.
- Diseñar una ley de control que mejore individualmente el desempeño de los robots móviles en trayectorias curvas, utilizando la información disponible del robot predecesor y otras técnicas de control derivadas de: sistemas difusos, control adaptativo, redes neuronales, entre otras.
- Implementar algoritmos que permitan discriminar entre mediciones válidas e inválidas.
- Implementar otras topologías de red para la comunicación entre los robots.
- Desarrollar ejemplos demostrativos y experiencias de laboratorio haciendo uso de la plataforma experimental.
- Estudiar la extensión de las capacidades de la plataforma experimental, evaluando el uso de hardware con mayores capacidades.
- Desarrollar otras aplicaciones basadas en el cambio del escenario de navegación, por ejemplo: seguimiento de carril, búsqueda y exploración colaborativa, planificación de rutas, etc.
- Implementar estrategias de identificación de parámetros a partir de los datos obtenidos.

BIBLIOGRAFÍA

[1] (1980). User Datagram Protocol. RFC 768. doi:10.17487/RFC0768. URL https://rfc-editor.org/rfc/rfc768.txt. 5.5

[2] (2003). *High-Speed CMOS Logic 16-Channel Analog Multiplexer/Demultiplexer*. Texas Instruments. URL https://www.ti.com/lit/ds/symlink/cd74hc4067.pdf. 2.2.1.2, 5.3.1

[3] (2014). *TB6612FNG Driver IC for Dual DC motor*. TOSHIBA. URL https://toshiba.semicon-storage.com/info/docget.jsp?did=10660&prodName=TB6612FNG. 2.2.1.3

[4] (2016). *World's smallest Time-of-Flight (ToF) laser ranging sensor*. STMicroelectronics. URL https://www.st.com/resource/en/data_brief/vl53l0x.pdf. Rev. 3. 2.2.1.2, 5.3.3

[5] (2020). *ESP32 Datasheet*. Espressif. URL espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf. Rev. 3.4. 2.2.1.1, 5.3.1

[6] (2020). *Miniature Reflective Object Sensor*. ON Semiconductor. URL https://www.onsemi.com/pdf/datasheet/qre1113-d.pdf. Rev. 7. 2.2.1.2, 5.3.1

[7] (2020). *Self-Driving Car Research Studio Brochure*. Quanser. URL https://quanserinc.app.box.com/s/ks5qdxfnb2tsx0kjcao69dildmfycxvc. 1.2

[8] Alam, A., Mårtensson, J., and Johansson, K.H. (2015). Experimental evaluation of decentralized cooperative cruise control for heavy-duty vehicle platooning. *Control Engineering Practice*, 38, 11–25. doi:https://doi.org/10.1016/j.conengprac.2014.12.009. URL https://www.sciencedirect.com/science/article/pii/S0967066114002822. 1.1

[9] Ali, N. (2020). Fuzzy logic based real time go to goal controller for mobile robot. *International Journal of Computer Applications*, 176, 32–36. doi:10.5120/ijca2020920078. 1.1

[10] Beauregard, B. (2017). Arduino pid library. repositorio de GitHub. URL https://github.com/br3ttb/Arduino-PID-Library. [Consulta: 02 febrero 2021]. 5.6

[11] Cariou, C., Laneurit, J., Roux, J.C., and Lenain, R. (2020). Multi-robots trajectory planning for farm field coverage. In *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 351–356. doi:10.1109/ICARCV50220.2020.9305333. 1.1

[12] Dòria-Cerezo, A., Biel, D., Olm, J.M., and Repecho, V. (2019). Sliding mode control of a differential-drive mobile robot following a path. In *2019 18th European Control Conference (ECC)*, 4061–4066. doi:10.23919/ECC.2019.8796166. 1.1

[13] ERROUAGUY, A. (2019). Esp32 analogwrite. repositorio de GitHub. URL https://github.com/ERROPiX/ESP32_AnalogWrite. [Consulta: 05 marzo 2021]. 5.4

[14] Gochkov, H. (2020). Wire. repositorio de GitHub. URL https://github.com/espressif/arduino-esp32/tree/master/libraries/Wire. [Consulta: 05 marzo 2021]. 5.3.3

[15] Gochkov, H. (2021). Wifi. repositorio de GitHub. URL https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFi. [Consulta: 05 marzo 2021]. 5.5.1

[16] Hu, J., Bhowmick, P., and Lanzon, A. (2021). Group coordinated control of networked mobile robots with applications to object transportation. *IEEE Transactions on Vehicular Technology*, 70(8), 8269–8274. doi:10.1109/TVT.2021.3093157. 1.1

[17] K.J. Åström, B.W. (1997). *Computer-Controlled Systems - Theory and Design*. Prentice Hall, 3rd edition. 293-295, [Consulta: 02 marzo 2021]. 5.6

[18] Lád, M., Herman, I., and Hurák, Z. (2017). Vehicular platooning experiments using autonomous slot cars**i.h. was supported by the czech science foundation within the project gacr 16-19526s. *IFAC-PapersOnLine*, 50(1), 12596 – 12603. doi:https://doi.org/10.1016/j.ifacol.2017.08.2201. URL http://www.sciencedirect.com/science/article/pii/S2405896317328719. 20th IFAC World Congress. 1.1, 5.6.3

[19] Mu, J., Yan, X., Spurgeon, S.K., and Mao, Z. (2017). Nonlinear sliding mode control of a two-wheeled mobile robot system. *International Journal of Modelling, Identification and Control*, 27(2). URL https://kar.kent.ac.uk/60116/. 1.1

[20] Naidu, D.S. (2002). *Optimal Control Systems*. Electrical Engineering Series. CRC Press, 1 edition. 4.1

[21] Paull, L., Tani, J., Ahn, H., Alonso-Mora, J., Carlone, L., Cap, M., Chen, Y.F., Choi, C., Dusek, J., Fang, Y., et al. (2017). Duckietown: an open, inexpensive and flexible platform for autonomy education and research. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 1497–1504. IEEE. 1.2

[22] Peters, A.A., Vargas, F.J., Garrido, C., Andrade, C., and Villenas, F. (2021). Pl-toon: A low-cost experimental platform for teaching and research on decentralized cooperative control. *Sensors*, 21(6). doi:10.3390/s21062072. URL https://www.mdpi.com/1424-8220/21/6/2072. 1.2, 5.6.3

[23] Pololu (2020). Vl53l0x library for arduino. repositorio de GitHub. URL https://github.com/pololu/vl53l0x-arduino#readme. [Consulta: 05 marzo 2021]. 5.3.3

[24] Scheffe, P., Maczijewski, J., Kloock, M., Kampmann, A., Derks, A., Kowalewski, S., and Alrifaee, B. (2020). Networked and autonomous model-scale vehicles for experiments in research and education. *arXiv preprint arXiv:2004.08364*. 1.2

[25] Stewart, T. (2021). Basic linear algebra. repositorio de GitHub. URL https://github.com/tomstewart89/BasicLinearAlgebra. [Consulta: 10 agosto 2021]. 5.6.1

[26] Stewart, T. (2021). Statespacecontrol. repositorio de GitHub. URL https://github.com/tomstewart89/StateSpaceControl. [Consulta: 28 agosto 2021]. 5.6

[27] Stoffregen, P. (2021). Encoder library. repositorio de GitHub. URL https://github.com/PaulStoffregen/Encoder#readme. [Consulta: 05 marzo 2021]. 5.3.2

[28] Velasco-Villa, M., Cruz-Morales, R.D., Rodriguez-Angeles, A., and Domínguez-Ortega, C.A. (2021). Observer-based time-variant spacing policy for a platoon of non-holonomic mobile robots. *Sensors*, 21(11). doi:10.3390/s21113824. URL https://www.mdpi.com/1424-8220/21/11/3824. 1.1

[29] Wang, C., Liu, X., Yang, X., Hu, F., Jiang, A., and Yang, C. (2018). Trajectory tracking of an omni-directional wheeled mobile robot using a model predictive control strategy. *Applied Sciences*, 8(2). doi:10.3390/app8020231. URL https://www.mdpi.com/2076-3417/8/2/231. 1.1

[30] Xie, D., Wang, S., and Wang, Y. (2018). Trajectory tracking control of differential drive mobile robot based on improved kinematics controller algorithm. In *2018 Chinese Automation Congress (CAC)*, 2675–2680. doi:10.1109/CAC.2018.8623764. 1.1

[31] Zeltkevic, M. (2015). Forward and backward euler methods. página web. URL https://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node3.html. [Consulta: 06 abril 2021]. 5.6

[32] Åström, Karl J.; Hägglund, T. (1995). *PID Controllers - Theory, Design, and Tuning*. ISA, 2nd edition. 5.6