DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI

ISSN: 2007-6363

Publicación Semestral Pädi Vol. 14 No. Especial (2026) 294-301

Control lineal y modelado virtual aplicado a un vehículo de tracción diferencial
Linear control and virtual reality applied to a differential-drive vehicle

C. Amaro de la Rosa [a], C. E. Gómez-Galindo [a], E. D. Hernández-Aquiahuatl [a], A. Roldán-Caballero [a,b,*], G. Texis-Texis [b]

[a] *Unidad Profesional Interdisciplinaria de Ingeniería Campus Tlaxcala, Instituto Politécnico Nacional, 90000, Tlaxcala, Tlaxcala, México.*
[b] *Facultad de Ciencias de la Electrónica, Benemérita Universidad Autónoma de Puebla, 72000, Puebla, Puebla, México.*

Resumen

En este trabajo se presenta el desarrollo de un sistema de control para un robot móvil con tracción diferencial, basado en su modelo cinemático. Se utilizó una técnica de linealización entrada-salida para diseñar un controlador capaz de realizar el seguimiento de trayectorias, validando su desempeño a través de simulaciones y pruebas físicas. La implementación física se llevó a cabo mediante una placa Arduino, empleando encoders para obtener las velocidades del sistema y retroalimentar el controlador. Se construyó un modelo virtual en Simscape Multibody con el fin de visualizar el comportamiento cinemático del prototipo. Los resultados obtenidos muestran un seguimiento preciso de las trayectorias, lo que confirma la efectividad del enfoque propuesto.

Palabras Clave: Robot móvil, seguimiento de trayectorias, controlador, Simscape Multibody

Abstract

This work presents the development of a control system for a differential-drive mobile robot, based on its kinematic model. An input-output linearization technique was used to design a controller capable of performing trajectory tracking, with its performance validated through simulations and physical tests. The physical implementation was carried out using an Arduino board, employing encoders to obtain the velocities of the physical system and provide feedback to the controller. A virtual model was built in Simscape Multibody to visualize the kinematic behavior of the prototype. The results demonstrate accurate trajectory tracking, confirming the effectiveness of the proposed approach

Keywords: Mobile robot, trajectory tracking, controller, Simscape Multibody

1. Introducción

El sistema de dirección de un automóvil, como la mayoría de los sistemas automotrices de la actualidad, es controlado por medio de dispositivos electrónicos. En este contexto, proporcionar una comunicación entre los diferentes elementos que lo conforman, teniendo el menor número de perturbaciones ha sido un tema de investigación frecuente.

La posición y velocidad son parámetros fundamentales en varios procesos, especialmente en aquellos que tienen perturbaciones generadas por factores externos (Vesentini, Rigo, Sanssonetto, Di Persio, & Muradore, 2024). Los sistemas dinámicos están especialmente influenciados por perturbaciones externas como irregularidades u obstáculos en el terreno, además, la imprecisión natural de las mediciones influye en la representación de los sistemas. (Phogat & Chang, 2020). Trasladando esto a un robot con tracción diferencial, reconocer el terreno en el que se desempeña es uno de los primeros pasos para su movimiento adecuado, en el ámbito de la dinámica, reconocer factores externos ayuda al funcionamiento adecuado del sistema (Martínez, Murrieta-Cid, M. Becerra, & Becerra, 2024). Debido a los factores anteriormente descritos, se debe saber que, para diferentes comportamientos del robot, debe de haber diferentes respuestas del sistema (Jardine, Kogan, Givigi & Yousefi, 2019). Esto genera que el reconocimiento de los estudios de un prototipo de automóvil con tracción diferencial sea útil para automóviles de tamaño real, esto debido a que su análisis puede llegar a ser escalable a tamaño real (Ali, Shen, & Hashim, 2024).

*Autor para la correspondencia: aroldanc@ipn.mx
Correo electrónico: camarod2100@alumno.ipn.mx (Christian Amaro de la Rosa), cgomezg2101@alumno.ipn.mx (Cristian Emiliano Gómez-Galindo), ehernandeza2107@alumno.ipn.mx (Erick David Hernández-Aquiahuatl), aroldanc@ipn.mx (Alfredo Roldan-Caballero), gtexist@ipn.mx (Gerardo Texis-Texis)

Historial del manuscrito: recibido el 15/07/2025, última versión-revisada recibida el 27/12/2025, aceptado el 16/12/2025, publicado el 25/03/2026. **DOI:** https://doi.org/10.29057/icbi.v14iEspecial.15522

El primer paso para comprender el movimiento de un robot con tracción diferencial es observar el origen de este movimiento, este se encuentra en las ruedas del automóvil. A su vez, junto con la tracción, son esencialmente las causas de la dinámica del automóvil. Hacer el análisis primario del movimiento a través de las ruedas se recomienda como primer paso, previo a generar un modelo matemático general del automóvil. El objetivo general de este artículo es controlar el comportamiento de las dos ruedas del robot. De esta manera, a partir de los datos recabados, se puede controlar su comportamiento.

2. Modelado y diseño de controlador

Para lograr un seguimiento de trayectorias en WMRs (por sus siglas en inglés wheeled mobile robots), se necesita determinar un modelo que describa el comportamiento del sistema (Siegwart et al., 2011; Dudek y Jenkin, 2010). En esta sección se presenta el desarrollo del modelo cinemático, así como el diseño de un controlador basado en la técnica de linealización entrada-salida

2.1. Modelo cinemático del sistema

Con el modelo cinemático se describe el desplazamiento y la orientación del robot respecto a un sistema de referencia fijo (x,y)

Para la obtención del modelo cinemático se situó el centro de masa en el eje que conecta las ruedas motrices del vehículo

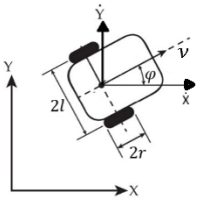

Figura 1 Diagrama del robot.

y se consideraron los parámetros: a) Velocidad general angular (ω), b) Radio de las ruedas (r), c) Distancia entre las ruedas (2ι), d) Velocidad angular de cada una de las ruedas (ω_r y ω_l) y e) Velocidad lineal (v). El modelo cinemático se obtuvo mediante el análisis de la *Figura 1*.

$$\begin{aligned}\dot{x} &= v\cos(\varphi)\\ \dot{y} &= v\,sen(\varphi)\\ \dot{\varphi} &= \omega\end{aligned} \tag{1}$$

Tanto v y ω dependen de las velocidades angulares de cada rueda, las ecuaciones que modelan estos parámetros son:

$$v = \frac{\omega_l r + \omega_r r}{2} \tag{2}$$

$$\omega = \frac{\omega_r r + \omega_l r}{2l}$$

Al sustituir las ecuaciones desarrolladas de v y ω se obtiene un sistema con ω_r y ω_l como entradas y se mantiene x,y y φ como estados, resultando:

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\varphi}\end{pmatrix} = \begin{pmatrix}\frac{r}{2}\cos(\varphi) & \frac{r}{2}\cos(\varphi)\\ \frac{r}{2}\text{sen}(\varphi) & \frac{r}{2}\text{sen}(\varphi)\\ \frac{r}{2l} & -\frac{r}{2l}\end{pmatrix}\begin{pmatrix}\omega_r\\ \omega_l\end{pmatrix} \tag{3}$$

2.2. Diseño de controlador

La técnica de control empleada en este trabajo se basa en el procedimiento de linealización entrada-salida propuesto por Silva-Ortigoza et al (2013). Dicho método permite eliminar la naturaleza no lineal del sistema simplificando las ecuaciones que describen el comportamiento del robot y así controlar su trayectoria. Este proceso se enfoca en la relación entre las entradas y las salidas del sistema, lo que permite definir nuevas entradas:

$$\begin{pmatrix}\omega_r\\ \omega_l\end{pmatrix} = \frac{1}{r}\begin{pmatrix} l\dot{\varphi} + \dot{x}\cos(\varphi) + \dot{y}\text{sin}(\varphi)\\ -l\dot{\varphi} + \dot{x}\cos(\varphi) + \dot{y}\text{sin}(\varphi)\end{pmatrix} \tag{4}$$

Se propuso un control para las entradas de la forma:

$$\begin{pmatrix}\omega_r\\ \omega_l\end{pmatrix} = \begin{pmatrix}\frac{1}{r}(l(\dot{\varphi}^* - k_\varphi(\varphi-\varphi^*)) + \frac{\dot{x}^* - k_x(x-x^*)}{\cos(\varphi)}\\ \frac{1}{r}(-l(\dot{\varphi}^* - k_\varphi(\varphi-\varphi^*)) + \frac{\dot{x}^* - k_x(x-x^*)}{\cos(\varphi)}\end{pmatrix} \tag{5}$$

En donde $\varphi^*, \dot{\varphi}^*, x^*, \dot{x}^*$ corresponden a las variables deseadas. Al sustituir en el modelo del sistema (Ecuación 3) se obtiene:

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\varphi}\end{pmatrix} = \begin{pmatrix}\dot{x}^* - k_x(x-x^*)\\ \tan(\varphi)\,[\dot{x}^* - k_x(x-x^*)\,]\\ \dot{\varphi}^* - k_\varphi(\varphi-\varphi^*)\end{pmatrix} \tag{6}$$

Este sistema en lazo cerrado nos permite controlar directamente dos estados (x, φ), realizando el análisis del error se observa que para todo $k_x > 0$ y $k_\varphi > 0$, se lograra que $x \to x^*$ y $\varphi \to \varphi^*$, por lo que el sistema tiende a seguir la trayectoria deseada. Mientras, $y(0) = y^*(0)$ se asegura que $y \to y^*$.

2.3. Implementación numérica

El sistema mostrado en la Figura 2 es la implementación del controlador, el cual cuenta con tres “MATLAB Function” necesarios para el funcionamiento del sistema.

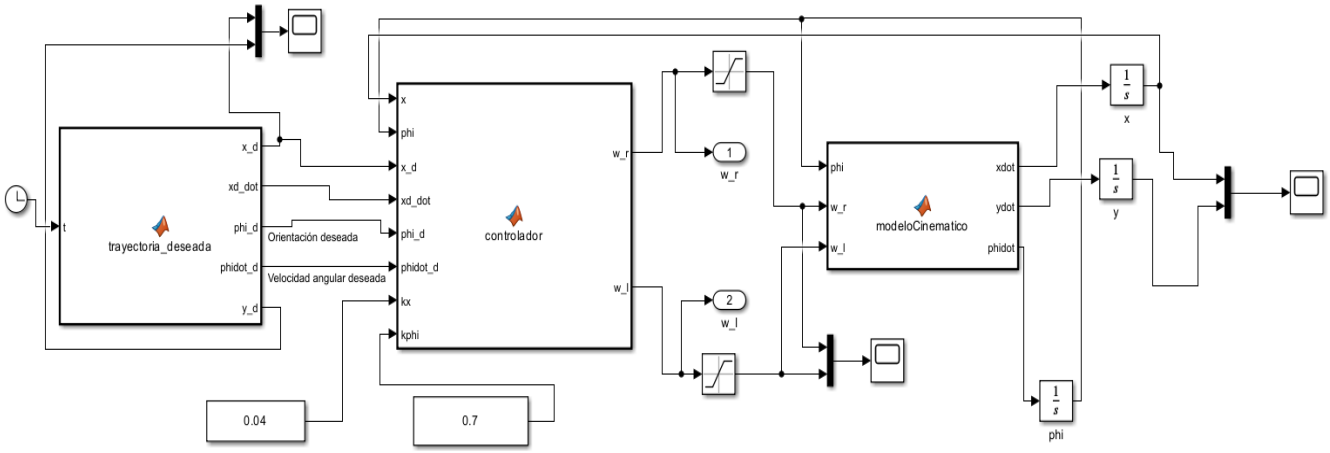

Figura 2 Bloques de control.

En el primer bloque se configuró una trayectoria deseada, la única entrada de este bloque es el tiempo y como salidas se obtiene $\boldsymbol{x_d, \dot{x}_d, \varphi_d, \dot{\varphi}_d, y_d}$. El segundo bloque es el controlador, el cual recibe estos parámetros, así como las constantes de $\boldsymbol{k_x}$ y $\boldsymbol{k_\varphi}$, entregando las velocidades $\boldsymbol{\omega_r}$ y $\boldsymbol{\omega_l}$ para el seguimiento de nuestra trayectoria deseada. El tercer bloque es el modelo cinemático del robot cuyas salidas retroalimentaran al controlador.

El controlador recibe cuatro parámetros de nuestro sistema deseado que serán calculados a través de razones trigonométricas: a) Posición en x deseada ($\boldsymbol{x_d}$), b) Velocidad en x deseada ($\boldsymbol{\dot{x}_d}$), c) Orientación del robot deseada ($\boldsymbol{\varphi_d}$), y d) Velocidad angular deseada ($\boldsymbol{\dot{\varphi_d}}$) y dos parámetros que se retroalimentan del sistema real: a) Posición en x real ($\mathbf{x}$) y b) Orientación del robot real ($\boldsymbol{\varphi}$).

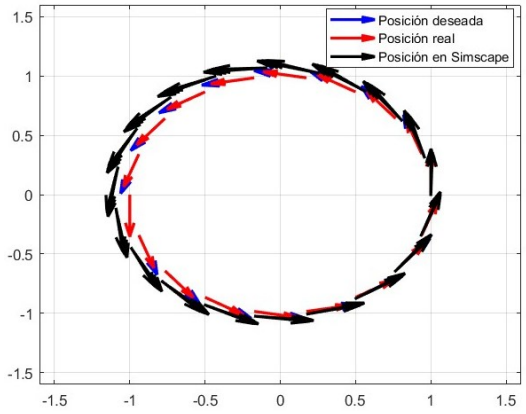

Figura 3 Gráfica de comportamiento del prototipo en Simscape

En la *Figura 3* se muestra las gráficas de las trayectorias real y deseada (x, y), así también, se muestra la gráfica del resultado de Simscape. Para la simulación de la trayectoria real, se configuran parámetros que permiten garantizar el control del sistema, tales como: a) la posición inicial en x del robot móvil coincide con el punto de inicio de la trayectoria deseada, b) la orientación inicial del robot es de $\frac{\pi}{2}$ radianes y c) la posición inicial real del robot en y coincide con su posición deseada.

3. Modelado Virtual en Simscape Multibody

Para el modelado y simulación del prototipo del robot móvil de tipo diferencial se hará uso de la herramienta Simscape Multibody de MATLAB. Este entorno virtual permite representar el comportamiento dinámico de mecanismos, donde es posible diseñar, modelar y simular sistemas o equipos.

3.1. Modelado y simulación del prototipo en Simscape Multibody

Al abrir un modelo nuevo de Simscape Multibody, existen tres bloques que configuran el sistema: Solver Configuration, World Frame y Mechanism Configuration. El bloque Solver Configuration especifica los parámetros que el modelo necesita para realizar la simulación de la estructura topológica, mientras que el bloque World Frame permite el acceso a un sistema de coordenadas, o punto de referencia global, que es base de todos los demás puntos de referencia que se definen con respecto a este sistema de coordenadas global, siendo este inercial y estando en reposo absoluto, Figura 4. Y, por último, el bloque Mechanism Configuration, Figura 5, especifica la gravedad y parámetros de simulación.

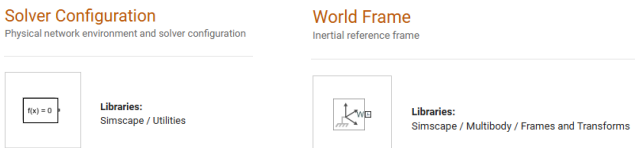

Figura 4 Bloques Solver Configuration y World Frame de Simscape Multibody.

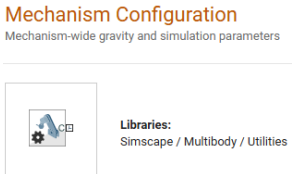

Figura 5 Bloque Mechanism Configuration de Simscape Multibody.

3.2. Modelado de sólidos

Los cuerpos son componentes principales de un mecanismo, por lo que deben modelarse para ensamblar y definir un sistema.

Antes de modelar el prototipo, se considera una base donde se desplazará. Se conforma del bloque Infinite Plane, Figura 6, que genera un plano infinito utilizado para modelar las fuerzas de contacto entre el plano y los sólidos del prototipo, con dimensiones de 40 m en el eje X y 40 m en el eje Y.

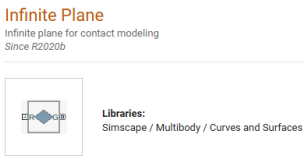

Figura 6 Bloque Infinite Plane de Simscape Multibody.

Para modelar el prototipo del robot móvil con tracción diferencial, se diseña un cuerpo simple, que cumple con las dimensiones de la implementación física. Utilizando el bloque Brick Solid, Figura 7, centrado en el punto cero del sistema de coordenadas y con parámetros de dimensión geométrica, similares al prototipo físico, de 12 cm en el eje X, 20 cm en el eje Y, y 0.05 cm en el eje Z.

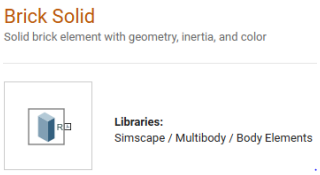

Figura 7 Bloque Brick Solid de Simscape Multibody.

Se implementan las cuatro ruedas del prototipo, utilizando el bloque Cylindrical Solid, Figura 8. Para cada una de las ruedas se le definen parámetros de radio y longitud de 3 cm y 4 cm, respectivamente.

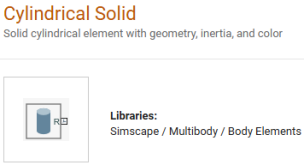

Figura 8 Bloque Cylindrical Solid de Simscape Multibody.

3.3. Creación de puntos de referencia personalizados

Cada modelo consta de puntos de referencia. Los puntos de referencia sirven para definir la posición y orientación de los cuerpos. Cada punto de referencia se conforma de tres ejes de coordenadas perpendiculares que se cruzan en un punto, el cual se denomina **origen**. Este punto indica la ubicación de los elementos y proporciona los puntos de unión para las articulaciones entre cuerpos. Para identificar cada eje, se tiene un código de colores, donde el eje X se representa en rojo, el eje Y en verde y el eje Z en azul, como se muestra en la Figura 9.

Figura 9 Código de colores de un punto de referencia.

De forma predeterminada, cada sólido cuenta con un punto de referencia, cuyo origen coincide con su centro de masa. Para el modelado del prototipo, se crean nuevos puntos de referencia con el fin de conectar las articulaciones, Figura 10. Estos puntos de referencia personalizados permiten establecer uniones entre sólidos mediante articulaciones.

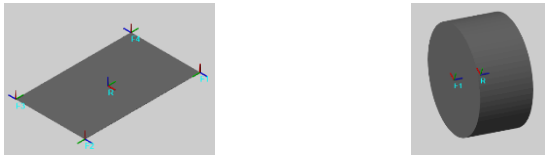

Figura 10 Nuevos puntos de referencia.

3.4. Conexión de articulaciones

En un modelo, las articulaciones representan las principales restricciones cinemáticas que determinan el movimiento relativo entre cuerpos. Estas articulaciones se definen mediante el uso de Joint Blocks. Cada tipo de bloque representa diferente clase de articulación, según el movimiento que se desea permitir entre los cuerpos.

Para unir la base del prototipo con las ruedas antes modeladas, mediante una articulación, se agrega el bloque Revolute Joint, Figura 11. Este bloque de articulación tiene un grado de libertad rotacional, la cual restringe el movimiento entre dos puntos de referencia alrededor de un eje común donde se define uno como base **Base** y otro como seguidor **Follower**. El eje de rotación está alineado con el eje Z del punto de referencia base de la articulación, por lo que los puntos de referencia del sólido base y seguidor comparten un origen y eje Z común. Esto permite que el punto de referencia seguidor rote alrededor del eje Z. Para ello, se conecta la articulación Revolute Joint a el bloque Brick Solid como base, y al bloque Cylindrical Solid como seguidor, Figura 12.

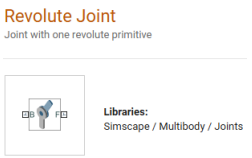

Figura 11 Bloque Revolute Joint de Simscape Multibody.

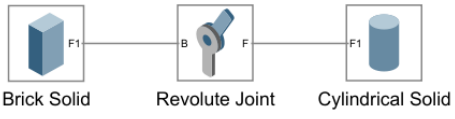

Figura 12 Conexión de Revolute Joint.

3.5. Entradas de movimiento a las articulaciones

Las articulaciones incluyen un apartado denominado Actuation, que permite definir entradas Torque y Motion, del cual depende el comportamiento de la articulación según la configuración seleccionada. Dado que se busca simular el comportamiento cinemático del sistema, se define el modo de actuación como Unactuated Motion, configurando los dos parámetros de entrada. Para el parámetro Torque, se selecciona la opción None, que establece fuerza/torque a cero, aunque la articulación puede moverse libremente durante la simulación. En el parámetro Motion, se selecciona Provided by Input para las dos ruedas frontales, donde la entrada es un objeto de serie temporal que especifica la trayectoria, y dicha trayectoria es la coordenada de posición a lo largo del eje, dada en función del tiempo.

Para generar esta entrada se utiliza el bloque PS-Simulink Converter, Figura 13, tomando una señal de velocidad angular extraída del control, la cual se integra para obtener su posición angular y se deriva para obtener la aceleración angular, Figura 14. Estas tres magnitudes se combinan en un único vector de señales físicas que varían en el tiempo, utilizando un bloque PS-Simulink Converter. Al recibir este vector en el puerto de entrada Motion en la articulación Revolute Joint, fuerza su comportamiento cinemático a seguir la trayectoria.

Figura 13 Bloque PS-Simulink Converter de Simscape Multibody.

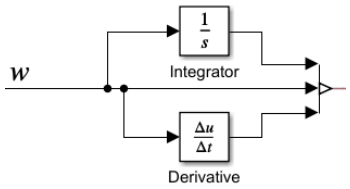

Figura 14 Conexión de bloque integrador y derivativo al bloque PS-Simulink Converter de Simscape Multibody.

3.6. Fuerza de contacto espacial

Para modelar el contacto entre las ruedas y el plano donde se desplaza el prototipo, se agrega el bloque Spatial Contact Force, Figura 15, dado que únicamente modela contactos entre dos geometrías, se agrega uno en cada rueda.

El bloque tiene dos puertos de geometría, definidos como base **Base** y otro como seguidor **Follower**. El puerto **Base** conecta al plano y el puerto **Follower** conecta a la rueda como se muestra en la Figura 16.

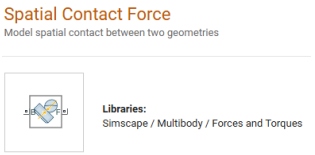

Figura 15 Bloque Spatial Contact Force de Simscape Multibody.

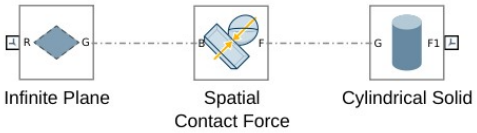

Figura 16 Conexión del bloque Spatial Contact Force.

3.7. Visualización de la simulación

Mechanics Explorers, es una herramienta de Simscape Multibody, donde se puede explorar la simulación mediante un panel de visualización interactivo que permite manipular la perspectiva del modelo mediante vistas isométricas, frontales, laterales y superiores, Figura 17. En la Figura 18, se muestra el resultado del modelo virtual del prototipo del robot móvil con tracción diferencial utilizando la herramienta Simscape Multibody de MATLAB para modelar y simular el comportamiento cinemático.

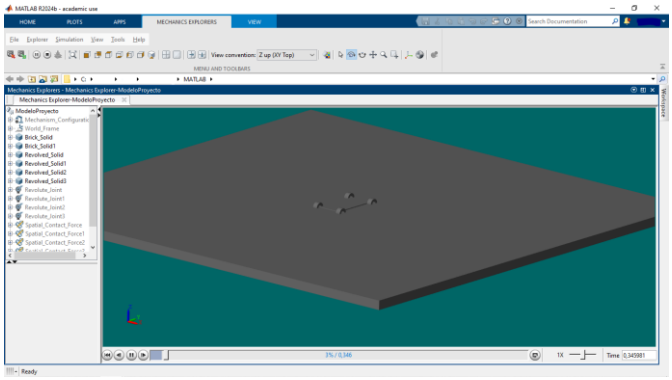

Figura 17 Simulación en Mechanics Explorers del modelo virtual del prototipo.

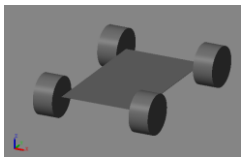

Figura 18 Resultado del modelo virtual del prototipo de robot móvil de tracción diferencial.

4. Resultados experimentales

4.1 Parámetros considerados.

En este caso, se consideran las velocidades de las ruedas y la posición del prototipo en el espacio (orientación) como variables principales del modelo.

La trayectoria de referencia establecida corresponde a una trayectoria circular, utilizada con el fin de evaluar la

precisión del controlador para seguir un movimiento continuo y cerrado. Dicha trayectoria se describe mediante las siguientes ecuaciones paramétricas:

$$\begin{cases} x_d(t) = R\cos(\omega t) \\ y_d(t) = R\sin(\omega t) \end{cases}$$

donde:

- R es el radio de la circunferencia,
- v es la velocidad lineal deseada,
- $\omega = \frac{v}{R}$ es la velocidad angular del movimiento circular,
- y $t$el tiempo de simulación, definido en un principio a 20 segundos

De manera adicional, se puede decir que los factores externos que generan el movimiento (como fricción, deslizamiento o perturbaciones externas) son, por el momento, ignorados, ya que se realiza un análisis cinemático y no uno dinámico.

4.2 Implementación física.

Para la implementación física del control fue necesario realizar una comunicación serial. Esta se logró con una placa de Arduino, configurada de manera que su salida sirviera de retroalimentación al control.

La placa de Arduino recibió los datos iniciales del bloque de control; acto seguido esta información se enlazó a la entrada del modelo cinemático a través de un módulo L298N. El uso de este módulo facilita el control de los dos motorreductores de manera independiente. Para que la comunicación de datos sea exacta se deben tomar en cuenta varios factores, así como especificaciones que permitan una comunicación con el menor número de perturbaciones influenciando el comportamiento del control. Dentro de estas especificaciones están el uso de una velocidad de datos de 9600 baudios, el cuál es un estándar en la mayoría de los sistemas embebidos debido a la estabilidad que presenta gracias a su baja velocidad. Para el muestreo de datos se necesitó consultar la frecuencia de cada motorreductor. Las especificaciones del fabricante indican que a 6V los motorreductores giran a 207 RPM, para convertir esto a hercios hacemos uso de la siguiente ecuación:

$$\frac{207\ RPM}{60} = 3.45\ Hz$$

Se hace uso del teorema de Nyquist, que explica que para reconstruir una señal continua se debe muestrear por lo menos al doble de su frecuencia máxima significativa. Haciendo el cálculo:

$$3.45 \times 2 = 6.9\ Hz$$

Convirtiendo esto a segundos para establecer el tiempo de muestreo con la siguiente fórmula:

$$T = \frac{1}{6.9\ Hz} = 0.144\ s$$

Siendo este el valor necesario para reconstruir la señal, sin embargo, se estableció un margen de seguridad para evitar comportamientos erráticos de la medición. Este margen consistió en redondearlo a la centésima más cercana previa a la existente. De esta manera el muestreo se tomó en 0.13 s o lo equivalente 7.7 Hz.

Una vez realizada la comunicación, el Arduino enviaba la señal al módulo L298N, generando movimiento en los motorreductores, este movimiento queda registrado gracias al uso de encoders rotatorios de salida cuadrática. La selección de este instrumento se basó en la facilidad que tenía en comparación a dispositivos analógicos. Los pulsos por revolución de cada encoder se convierten a rad/s por medio de la siguiente ecuación.

$$\omega = \frac{2\pi}{t}$$

Entonces:

$$\text{Ángulo por cada pulso} = \frac{2\pi}{PPR},$$

$$\Delta\theta = pulsos\ \frac{2\pi}{PPR},$$

$$\omega = \frac{2\pi * p}{PPR * \Delta t},$$

en donde:

p=pulsos en el intervalo de tiempo
PPR= Pulsos Por Revolución física del encoder
Δt= Variación del tiempo

Esta conversión se realiza para facilitar el análisis de la velocidad angular del prototipo.

Los únicos parámetros que se tomaron en cuenta para la retroalimentación del sistema fueron las velocidades de las llantas, debido a que se está controlando únicamente la cinemática del sistema.

La señal digital de los encoders sirve como retroalimentación a la función de controlador, esto permite corregir y analizar la trayectoria del prototipo. Se hace uso de un bloque de saturación, con un rango de 0 a 255, el cual limita la salida del controlador. Este rango se usa por dos razones:

- El rango de 0 a 255 representa el formato de datos que espera el PWM.
- El Arduino interpreta ese rango como un ciclo de trabajo de 8 bits. En donde 0 representa 0% del ciclo de trabajo y 255 el 100% del ciclo de trabajo.

Por otra parte, al terminar la comunicación se deben regresar los datos de PWM a rad/s para mantener la unidad original establecida como medida de la velocidad angular.

Se usa un muestreo de datos discreto para facilitar el traspaso de la señal, sin embargo, la salida de la función "Modelo Cinemático" tiene bloques de datos continuos (integration). Por esta razón el método de solución que se usó para el diagrama de bloques fue el de Bogacki-Shampine, el cual permite tener bloques discretos y continuos en un mismo sistema.

La trayectoria deseada en el diagrama de bloques obedece a una geometría circular pero no sobre su propio eje. Esta trayectoria se ve como una función senoidal de rango 1 en una gráfica de posición contra tiempo.

El prototipo se conectó de la siguiente manera para seguir el comportamiento deseado:

Los motores derecho e izquierdo se conectaron al módulo L298N por medio del Arduino haciendo uso de los puertos PWM que dispone la placa, conectándose ENA al puerto 9 y ENB al puerto 10.

Los pulsos de cada encoder se registraron conectándolos directamente al Arduino. De manera que los pulsos de cada

giro se registren en puertos 2 y 4. Mientras que su sentido (data) se registra en los puertos 18 y 19 (para interrupciones).

El diagrama de la Figura 19 explica de manera gráfica las conexiones y la comunicación necesaria para el control.

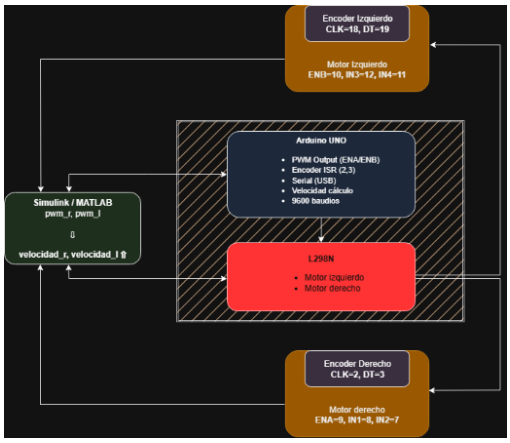

Figura 19 Diagrama de conexiones del prototipo del automóvil.

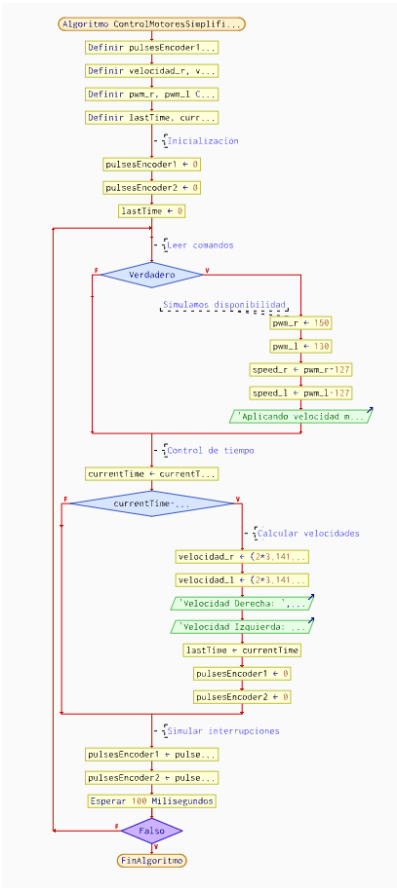

Figura 20 Diagrama de flujo que describe el funcionamiento del control.

El diagrama de flujo de la Figura 20 representa un sistema de control en lazo cerrado para dos motores DC con capacidad de retroalimentación mediante encoders, implementado en una placa Arduino. se configuran los pines del encoder de manera que puedan ser interpretados por los motores a través del Arduino (utilizando ENA, IN1, IN2 para el motor derecho y ENB, IN3, IN4 para el izquierdo) y se configuran los pines de los encoders como entradas. Se establece en el Arduino las interrupciones necesarias para registrar los pulsos del encoder. Al terminar de configurarse el sistema, se establece un bucle en el que se siguen cuatro pasos importantes:

1. Verificar los datos disponibles desde Simulink (por lo menos dos bytes libres para los valores de PWM).
2. Se envían los datos recabados a una función que define la velocidad y dirección del motor.
3. De manera simultánea, el sistema cuenta con un temporizador para tomar muestras cada 100 milisegundos, pasado este tiempo mide las velocidades reales. De manera que calcula las velocidades angulares en radianes por segundo usando la fórmula descrita anteriormente.
4. Se convierten las velocidades a un rango de 0 a 255 para su transmisión y las envía de vuelta a Simulink, completando así el ciclo de comunicación bidireccional.

Cada vez que los encoders generan un pulso las rutinas de interrupción de cada encoder se ejecutan. Todo este sistema opera en tiempo real, respondiendo inmediatamente a los comandos recibidos mientras mantiene una tasa constante de muestreo y reporte de 7.7 Hz, permitiendo tanto el control de los motores como el monitoreo de su desempeño real.

Usando esta programación se puede implementar el control en un prototipo funcional (Figura 21).

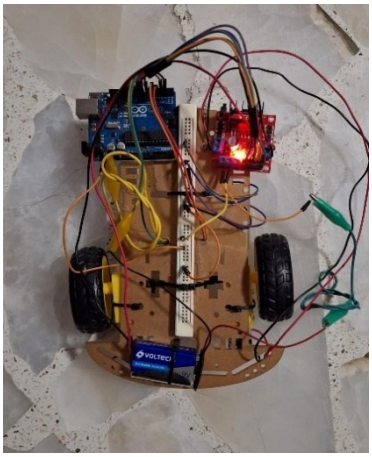

Figura 21 Prototipo armado del automóvil.

El desfase observado entre la trayectoria deseada y la trayectoria real del prototipo puede atribuirse a factores no incluidos en el modelo teórico, como pequeñas imperfecciones en la medición, retardos o variaciones en las velocidades de las ruedas. Para cuantificar de forma objetiva esa desviación se emplea el Error Cuadrático Medio (MSE), que mide la diferencia promedio al cuadrado entre la referencia y la respuesta del sistema. Está descrito por la siguiente ecuación:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_d(i) - y_r(i))^2$$

donde $y_d(i)$ es la señal deseada (referencia) en la muestra i, $y_r(i)$ es la señal real medida o estimada en la misma muestra, y N es el número total de muestras.

Si la trayectoria se expresa por coordenadas (x, y) del centro del robot en el sistema de referencia inercial, el término de error por muestra se define como la distancia al cuadrado entre la posición deseada y la real:

$$s_i = (x_d(i) - x_r(i))^2 + (y_d(i) - y_r(i))^2,$$

y el MSE en el plano se describe con la siguiente ecuación:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} s_i \;=\; \frac{1}{N}\sum_{i=1}^{N}[(x_d(i) - x_r(i))^2 + (y_d(i) - y_r(i))^2].$$

En este caso y_d o (x_d, y_d) representa la trayectoria de referencia, es decir, la posición deseada del centro del chasis con origen en la pose inicial del robot

Por su parte y_r o (x_r, y_r) corresponde a la posición real del centro del robot obtenida por los pulsos del encoder. Ambas señales deben estar expresadas en el mismo sistema de referencia para que la diferencia tenga significado físico.

En la prueba se tomaron muestras cada 0.01 s durante 10 s, por lo que $N = 1001$. Evaluando la sumatoria de errores al cuadrado se obtuvo la suma total

$$S = \sum_{i=1}^{1001}[(x_d(i) - x_r(i))^2 + (y_d(i) - y_r(i))^2] \approx 0.0043023$$

Por tanto,

$$MSE = \frac{S}{N} = \frac{0.0043023}{1001} \approx 4.298 \times 10^{-6}$$

y el RMSE (raíz cuadrática media), que tiene las mismas unidades que la posición (metros) y resulta más intuitivo, es:

$$RMSE = \sqrt{MSE} \approx \sqrt{4.298 \times 10^{-6}} \approx 0.00207 \text{ m}$$

El valor obtenido de $RMSE \approx 0.00207$ m indica que, en promedio, el centro del prototipo se desvía aproximadamente 2 mm de la trayectoria deseada durante todo el recorrido. Dado que el modelo físico del vehículo tiene una longitud característica cercana a 0.4 m, el error medio representa menos del 1 % del tamaño del robot, lo que indica un seguimiento sumamente preciso.

Si la trayectoria deseada es de forma circular con un radio de 1 m, el RMSE equivale a solo el 0.2 % del radio, lo cual demuestra una excelente coincidencia entre la posición deseada y la real.

En este rango, el error puede considerarse dentro de un margen ideal para sistemas de control de trayectoria bien calibrados y correctamente compensados.

En consecuencia, el RMSE obtenido se encuentra en un rango óptimo, ya que la desviación promedio es mucho menor que las dimensiones físicas del vehículo y del radio de la trayectoria. Esto sugiere un control con alta precisión posicional, sin errores significativos atribuibles a dinámica no modelada, fricción o discretización. No obstante, en implementaciones reales se recomienda conservar márgenes de seguridad que mantengan el error por debajo del 5 % del radio de trayectoria o del 10 % de la longitud del robot, para compensar posibles perturbaciones externas.

5. Conclusiones

El método de retroalimentación de un control por medio de encoders demostró ser una técnica adecuada para el control de la trayectoria de un prototipo de vehículo, y la implementación con Arduino facilitó la adquisición de datos en comparación con microprocesadores más complejos para el público en general. El modelado del prototipo para un robot móvil de tipo diferencial en Simscape Multibody permitió visualizar el comportamiento cinemático en un entorno virtual, facilitando la validación de los requerimientos de diseño. Lo anterior abre la posibilidad de extender el modelo con sensores, algoritmos de navegación o simulaciones en distintos terrenos, generando un prototipado virtual óptimo. Para la implementación virtual, se considera el modelo cinemático, se desprecian las condiciones dinámicas como la fricción, deslizamiento o inercia, por lo que puede afectar en un entorno físico real. El modelado y simulación se hicieron en condiciones controladas, con terreno plano y un entorno sin perturbaciones. La simulación en Simscape Multibody valida la implementación del controlador basado en la técnica de linealización entrada-salida siguiendo la trayectoria deseada. En un entorno similar a las condiciones reales implica ajustes que compensan la diferencia entre el modelo ideal y el comportamiento físico del robot para tener un mejor desempeño en el análisis dinámico del modelo. El error cuadrático medio comprueba que, en una implementación futura, se deben tomar en cuenta factores ya descritos para evitar desviaciones muy grandes de trayectoria. El proyecto establece una base sólida para simulaciones más avanzadas y estudios de optimización, impulsando el desarrollo e innovación en la robótica móvil.

Referencias

A. M. Ali, C. Shen, and H. A. Hashim, “A linear MPC with control barrier functions for differential drive robots,” *IET Control Theory and Applications*, vol. 18, no. 18, pp. 2693–2704, Aug. 2024, doi: 10.1049/cth2.12709.

G. Dudek and M. Jenkin, *Computational principles of mobile Robotics*. 2010. doi: 10.1017/cbo9780511780929.

L. E. S. Guzmán, M. A. M. Villa, and E. L. R. Vásquez, “Seguimiento de trayectorias con un robot móvil de configuración diferencial,” *Ingenierías USBmed*, vol. 5, no. 1, pp. 26–34, Jun. 2014, doi: 10.21500/20275846.298.

P. T. Jardine, M. Kogan, S. N. Givigi, and S. Yousefi, “Adaptive predictive control of a differential drive robot tuned with reinforcement learning,” *International Journal of Adaptive Control and Signal Processing*, vol. 33, no. 2, pp. 410–423, Apr. 2018, doi: 10.1002/acs.2882.

E. Martinez, R. Murrieta-Cid, H. M. Becerra, and I. Becerra, “Feasible minimum distance feedback-based-navigation for a differential drive robot in an environment with obstacles,” *Journal of the Franklin Institute*, vol. 361, no. 18, p. 107253, Sep. 2024, doi: 10.1016/j.jfranklin.2024.107253.

G. H. Millán, L. H. R. Gonzales, and M. B. López, “Implementation of a position and movement controller for a differential mobile robot,” *DOAJ (DOAJ: Directory of Open Access Journals)*, Jun. 2016, doi: 10.14483/udistrital.jour.tecnura.2016.2.a09.

K. S. Phogat and D. E. Chang, “Invariant extended Kalman filter on matrix Lie groups,” *Automatica*, vol. 114, p. 108812, Jan. 2020, doi: 10.1016/j.automatica.2020.108812.

R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, “Introduction to autonomous mobile robots,” *Choice Reviews Online*, vol. 49, no. 03, pp. 49–1492, Nov. 2011, doi: 10.5860/choice.49-1492.

S. G. Tzafestas, *Introduction to mobile robot control*. 2014. doi: 10.1016/c2013-0-01365-5.

F. Vesentini, D. Rigo, N. Sansonetto, L. Di Persio, and R. Muradore, “Minimum-energy switching geometric filter on lie groups for differential-drive wheeled mobile robots,” *European Journal of Control*, vol. 80, p. 101101, Aug. 2024, doi: 10.1016/j.ejcon.2024.101101.

R. Silva-Ortigoza *et al.*, “Construction of a WMR for Trajectory Tracking Control: Experimental results,” *The Scientific World JOURNAL*, vol. 2013, no. 1, Jan. 2013, doi: 10.1155/2013/723645.