COMPARACIÓN DE DESEMPEÑO ENTRE DOS ESTRATEGIAS DE CONTROL AUTOMÁTICO BASADAS EN UN ENFOQUE CLÁSICO Y UN ENFOQUE FUZZY LOGIC PARA UN ROBOT MÓVIL TIPO SEGUIDOR DE LÍNEA

Daniel Felipe Bonilla Mera

Corporación Universitaria Comfacauca

Facultad de Ingeniería

Popayán – Cauca

2024

COMPARACIÓN DE DESEMPEÑO ENTRE DOS ESTRATEGIAS DE CONTROL AUTOMÁTICO BASADAS EN UN ENFOQUE CLÁSICO Y UN ENFOQUE FUZZY LOGIC PARA UN ROBOT MÓVIL TIPO SEGUIDOR DE LÍNEA

Daniel Felipe Bonilla Mera

Trabajo de grado presentado a la Facultad de Ingeniería de la Corporación Universitaria de Comfacauca

Director
M.Sc. Carlos Iván Vásquez Valencia
Codirector
M.Sc. Saúl Eduardo Ruiz Sarzosa

Corporación Universitaria Comfacauca
Facultad de Ingeniería
Popayán-Cauca
2024

Nota de aceptación

_____

_____

_____

_____

_____

Director: _____

M.Sc. Carlos Iván Vásquez Valencia

Firma del Jurado _____

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todos aquellos que contribuyeron de alguna manera a la realización de este trabajo de grado. Ha sido un viaje lleno de desafíos y aprendizajes, y no podría haberlo logrado sin el apoyo y la orientación de tantas personas increíbles.

En primer lugar, quiero agradecer a mis padres, Gerardo Bonilla y Yolanda Mera y mi hermano Juan David Bonilla Mera por su amor incondicional, apoyo emocional y constante motivación. Gracias por creer en mí y por enseñarme el valor del trabajo duro y la perseverancia.

A mi esposa y mi hija, por su comprensión y apoyo en los momentos más difíciles. Gracias por ser mi fuente de inspiración y por animarme siempre a seguir adelante.

A mi amigo Daniel Castellanos por su compañía, paciencia y por las risas compartidas. Gracias por ser parte de este viaje y por brindarme su apoyo incondicional.

Igualmente quiero agradecer a mi director de tesis, Ing. Carlos Vásquez por su invaluable orientación y apoyo a lo largo de todo el proceso. Sus conocimientos, paciencia y dedicación fueron fundamentales para dar forma a este trabajo y llevarlo a término.

También quiero expresar mi gratitud a mis profesores y asesores académicos, por su orientación y sus comentarios constructivos que me ayudaron a mejorar proyecto investigativo.

Agradezco profundamente a mis amigos y familiares por su constante apoyo emocional durante los momentos más desafiantes. Sus palabras de aliento y ánimo fueron una fuente de inspiración para mí.

Además, quiero reconoce a la Corporación Universitaria Comfacauca, por brindarme los recursos y el ambiente propicio para llevar a cabo este proyecto de investigación.

Este trabajo de grado no solo representa el esfuerzo individual, sino también el resultado del trabajo en equipo y la colaboración de muchas personas. Estoy profundamente agradecido por todas las contribuciones que hicieron posible este logro.

¡Muchas gracias a todos! **Daniel Felipe Bonilla M**

Contenido

Resumen

La robótica es una ciencia multidisciplinaria que combina conocimientos de ingeniería, electrónica, informática, física entre otras especialidades, con el objetivo de crear máquinas programables capaces de imitar comportamientos humanos o animales. Su impacto abarca desde la automatización industrial hasta la exploración espacial y otros ambientes extremos. En la actualidad, las actividades extracurriculares que buscan ampliar los conocimientos teóricos de los estudiantes universitarios y profesionales aficionados, junto con el fomento del pensamiento computacional y las actividades STEM en los niños, han llevado a la popularización de competencias de robótica. Entre estas competencias se encuentra el desafío del seguidor de línea, donde los robots equipados con sensores, motores y tarjetas de control compiten por seguir un circuito demarcado por una línea en el menor tiempo posible. En este contexto, es común que se utilicen algoritmos de control continuo, como el controlador PID tradicional, sin embargo, la literatura y en eventos de robótica se ha demostrado la viabilidad de otras estrategias, como los controladores difusos o fuzzificados. Por lo tanto, este proyecto tiene como objetivo diseñar y construir un robot móvil seguidor de línea para competencias, modelar matemáticamente su cinemática de locomoción y aplicar dos algoritmos de control: el PID tradicional y el nuevo controlador PID fuzzificado. Se realizará una comparación de su desempeño en términos de velocidad, tiempo y otras características para determinar cuál estrategia de control es más efectiva y bajo qué circunstancias.

Palabras claves: pensamiento computacional, STEM, competencias de robótica, seguidor de línea, robot móvil, cinemática, locomoción, algoritmo de control, controlador PID, controlador difuso, controlador fuzzificado.

Índice de figuras

Índice de tablas

1. CAPÍTULO ASPECTOS GENERALES DEL PROYECTO

1.1 Planteamiento del problema

En la actualidad, la navegación autónoma de robots móviles se ha convertido en un área de investigación crucial debido a su aplicabilidad en diversas industrias, desde la manufactura hasta los transportes de encomiendas, la asistencia a humanos, la agricultura, entre otras. En el contexto específico de robots seguidores de línea, el desafío radica en desarrollo de estrategias de control efectivas y eficientes que permitan seguir de manera precisa una trayectoria de referencia predefinida la cual se demarca normalmente en suelo, para que el robot se desplace a traves de esta marca para cumplir una labor bien sea de logistica, almacenamiento en bodegas para la industria o en competiciones de aficionados. [1]

El robot seguidor de líneas utiliza un sensor infrarrojo de entre 8 y 32 canales para detectar la posición de una línea negra sobre un fondo blanco, donde no hay reflexión óptica. Este sensor permite al robot calcular la desviación de la línea con respecto a su centro, lo que facilita la corrección de su trayectoria. El microcontrolador está programado con un algoritmo de control retroalimentado, comúnmente de tipo PID (Proporcional, Integral y Derivativo), que ajusta la velocidad de los motores para corregir la orientación del robot y mantenerlo sobre la línea. Hoy en día, se han implementado técnicas avanzadas de control y dispositivos adicionales, como turbinas de succión o alerones, que mejoran la tracción, estabilidad y velocidad del robot en la pista, optimizando su desempeño en condiciones de alta competencia.

Existen,entre otros,dos enfoques que han sido utilizados para abordar este problema de seguimiento de trayectorias demarcadas; el control clásico PID y el control basado en lógica difusa. El control clásico utiliza técnicas matemáticas para calcular modelos precisos para diseñar controladores, es común hablar de estrategias PID, realimentacion de estado, entre otras técnicas. Mientras que la lógica difusa proporciona un marco flexible para lidiar con la incertidumbre y la variabilidad inherentes a los modelos dinámicos imprecisos que pueden plantearse para estos robots, los cuales poseen dinámicas complejas de modelar matematicamente. [2] [3]

Sin embargo, la comparación directa entre estos dos enfoques en el contexto de robots

seguidores de línea aún presenta vacíos en la literatura, pues hasta ahora no existe una comparación entre estas estrategias, que permita indicar cual sobresale respecto a la otra. Se necesita una evaluación cuantitativa del desempeño en términos de error de seguimiento para comprender mejor las fortalezas y debilidades de cada estrategia de control.

No obstante, en el artículo "Journal of Electrical Engineering" se destaca la importancia del control de velocidad del motor DC en estos robots y la prevalencia de los controladores PID clásicos en más del 95% de las aplicaciones de control debido a su simplicidad, de igual manera. Se concluye que los controladores de velocidad difuso y neuro-difuso, ofrecen un mejor rendimiento (estabilidad, precisión y velocidad) que el controlador de velocidad PID, No hace falta decir que los métodos de computación suave son más robustos y eficientes para sistemas no lineales. [4]

Por otra parte, Toumpa, concluye que el seguimiento de línea representa un desafío en competiciones de robots para probar la funcionalidad y dinámica de los robots móviles con ruedas. Además, dado que los vehículos guiados automáticos están ganando cada vez más popularidad, el uso de robots seguidores de línea se está extendiendo en aplicaciones comerciales. La capacidad de los robots para navegar con precisión y fiabilidad a lo largo de un camino marcado en el suelo puede utilizarse para transportar productos en almacenes o para transferir materiales entre puntos de fabricación. [5]

Respecto al modelado, "El modelo matemático muestra cómo el error de seguimiento depende de varios factores, como la dirección de la trayectoria deseada, la orientación actual del robot y las velocidades angulares y lineales**"**. [6]El control PID es uno de los controladores más comunes en la industria en general, por su facilidad de implementación, robustez en la mayoría de los casos y en muchas de sus aplicaciones. Estas características mencionadas hacen que sea uno de los controladores más conocidos y utilizados, por tanto, es muy importante conocer su desempeño dentro de diversas ramas de la ingeniería, por ejemplo, las investigaciones en aplicaciones fotovoltaicas que se ha llevado a cabo recientemente en todo el mundo, donde existen numerosos avances

interesantes en varios aspectos complementarios para maximizar la eficiencia y extraer la máxima potencia de los sistemas fotovoltaicos, con ayuda sistemas de control [7]**.** Así mismo, Ferreyra habla de la implementación de controladores PID en hardware basados en lógica difusa, se motiva por su capacidad para capturar estrategias cualitativas de control y adaptarse a condiciones cambiantes impredecibles, como cambios ambientales o desgaste físico en los componentes del sistema. Esto permite una mayor flexibilidad en el comportamiento del control, lo que puede ser beneficioso para sistemas complejos y dinámicos. [8]

Hoy en día existen evidencias de implementación que han evaluado las estrategias de control automático basadas en un enfoque clásico y un enfoque fuzzy logic ,pero hasta ahora, no se ha logrado determinar con precisión cuál de estas estrategias de control prevalece una sobre la otra, por tanto, este proyecto se basa en establecer una evaluación cuantitativa, comparando el rendimiento de robots seguidores de línea bajo estrategias de control clásico y lógica difusa, evaluando el desempeño en términos de error de seguimiento, con el fin de determinar las fortalezas y debilidades de cada estrategia de control. Se debe agregar que, se pretende llevar a cabo pruebas experimentales mediante un prototipo de robot seguidor de línea, el cual se diseñó bajo ciertos parametros relacionados en el reglamento de esta clase de competencias, por ejemplo,dimensiones menores o iguales a 20x30 de ancho por largo y sin limite de altura y peso, además se intenta construir un robot con minimo peso, del orden de los 50 gramos a 300 gramos, de tal manera que los motores electricos de tracción puedan trasladar el propio peso del robot al tiempo que lo hacen en el menor tiempo posible. Se desea medir el error acumulado de seguimiento en diversas condiciones del robot prototipo y trayectorias de seguimiento, de manera que, se espera determinar cual estratgia de control es sobresaliente con respecto a la otra bajo ciertas circunstancias, contribuyendo así al avance de la investigación en el campo de la navegación autónoma de robots móviles.

Teniendo en cuenta el contexto anterior, se plantea la siguiente pregunta de investigación: En términos del error de seguimiento en control automático, ¿cuál estrategia de control presenta un mejor rendimiento en robots seguidores de línea para competición: el control

PID clásico o el control basado en lógica difusa?

1.2 Justificación

El desarrollo de sistemas de control automático para robots móviles tipo seguidor de línea es de gran importancia, toda vez que se encuentran en diversas áreas, como la industria,

la investigación y la robótica. Estos robots tienen la capacidad de seguir una línea trazada en la superficie, lo que lo convierte en una herramienta valiosa para la logística automatizada y exploración de entornos desconocidos, de esta manera, la elección de la estrategia de control automático adecuada juega un papel significativo en la eficiencia del robot**.** [9]

El propósito de este trabajo de investigación es realizar una comparación del desempeño entre dos estrategias de control automático para un robot móvil tipo seguidor de línea, la primera estrategia de control se basa en un enfoque clásico, y la segunda estrategia se basa en un enfoque de lógica difusa. Aquí vale la pena decir, que ambos enfoques tienen sus propias ventajas y desventajas, por tanto, es importante evaluar el rendimiento de cada uno de estos enfoques en un ambiente controlado, con el fin de determinar cuál es más adecuado en términos de precisión, velocidad, robustez y adaptabilidad a diferentes condiciones ambientales. Por lo que se refiere al enfoque clásico de control automático, este se basa en modelos matemáticos y técnicas de control convencionales, como el controlador proporcional-integral-derivativo (PID), el cual es ampliamente utilizado y comprendido en la teoría de control. Por otro lado, el enfoque de lógica difusa permite modelar el comportamiento humano y la incertidumbre de manera más flexible y tolerante, lo que puede ser beneficioso en situaciones donde las condiciones de operación son cambiantes [10]. De lo anterior, resulta imprescindible la implementación experimental de una y otra estrategia de control en un entorno real, esta investigación permitirá obtener un conocimiento más profundo de las capacidades y limitaciones de cada una de estas estrategias.

En resumen, este trabajo de investigación contribuirá al entendimiento en el ámbito de la robótica móvil al proporcionar una evaluación detallada y estructurada del rendimiento entre dos estrategias de control automático para robots seguidores de línea, los resultados obtenidos servirán como base para tomar decisiones fundamentadas respecto a la selección y diseño de sistemas de control automático para robots móviles seguidores de línea en diferentes aplicaciones y entornos.

1.3 Objetivos

1.3.1 **Objetivo general**

Evaluar el desempeño en términos de error de seguimiento para un robot móvil tipo

seguidor de línea, aplicando estrategias de control PID clásico y control de lógica difusa.

1.3.2 Objetivos específicos

- Calcular el modelo matemático que represente la cinemática del robot a partir de técnicas de modelado simplificado para robots móviles.
- Implementar dos estrategias de control para un prototipo de robot seguidor de línea, la primera estrategia basada en control PID clásico, mientras que la segunda se basa en un enfoque de lógica difusa.
- Comparar estrategia de control PID y control lógico difuso, para un prototipo de robot móvil seguidor de línea, mediante un método de pruebas de cálculo de error cuadrático medio acumulado.

1.4 Marco teórico

1.4.1 Robótica móvil

Los robots móviles enfrentan desafíos considerables en entornos extensos y no

estructurados debido a la incertidumbre en la percepción de posición e identificación de objetos. La incertidumbre es tan marcada que el desplazamiento de un punto “A” a un punto “B” es arriesgado para un robot móvil, a diferencia de ser una tarea trivial para un manipulador industrial. A pesar de las mayores incertidumbres ambientales, no se espera que los robots móviles sigan trayectorias o alcancen su destino con la misma precisión que los manipuladores industriales. Las diferencias en los índices de desempeño se deben a prioridades de investigación diferentes, con enfoque en áreas de censado y razonamiento para dichos robots. [11]

“El principal desafío para este tipo de robots es la generación de trayectorias y la conducción de su movimiento, basado en la información de sensores externos. Esto permite el desplazamiento seguro evitando colisiones, requiriendo sistemas de control de trayectorias en distintos niveles jerárquicos para garantizar la máxima autonomía. El robot móvil autónomo se distingue por la integración inteligente entre percepción y acción, lo que le permite alcanzar objetivos en entornos inciertos. Su autonomía está vinculada a la capacidad de abstraer el entorno y transformar la información en comandos para los actuadores, destacándose por estas características frente a otros vehículos las cuales son:

1. Percepción: El robot móvil debe poseer la habilidad de determinar su relación con el entorno de trabajo mediante su sistema sensorial integrado. La capacidad perceptiva del robot móvil se manifiesta en la síntesis de la información recopilada por los sensores, generando mapas globales y locales del entorno conforme a distintos niveles de control.
2. Razonamiento: El robot móvil debe ser capaz de tomar decisiones sobre las acciones requeridas en cada momento, basándose en su estado y el del entorno, con el fin de alcanzar sus objetivos predefinidos. La capacidad de razonamiento del robot móvil se traduce en la planificación de trayectorias globales seguras y la capacidad de modificarlas en presencia de obstáculos imprevistos (control local de trayectoria), permitiendo al robot lograr los objetivos asignados de manera efectiva.” [12], [13].

En todas estas aplicaciones, la justificación primordial para la implementación de la robótica radica en la dificultad o imposibilidad de intervención humana, ya sea directa o tele operada. El término "autónomo" se refiere a la capacidad del robot para percibir,

modelar, planificar y actuar para alcanzar objetivos sin la intervención, o con mínima intervención, de supervisores humanos. Esto establece la distinción entre los robots móviles y los vehículos o máquinas tele operadas, donde un operador humano realiza tareas de forma remota, de esta manera no se consideran robots móviles los llamados vehículos autoguiados, que siguen caminos preestablecidos (pintados en el suelo, bandas magnéticas, etc.) o actúan de manera repetitiva, como los carros filo-guiados comunes en fábricas y entornos industriales. [14]

Las dificultades para llevar a cabo estas tareas surgen debido a la naturaleza dispar de los procesamientos requeridos en cada una de ellas. En la estimación de la posición y la construcción (o actualización) del mapa del entorno, se priorizan características como precisión, resolución espacial y alcance. Por otro lado, en la detección de obstáculos, se requiere un tiempo entre observaciones mucho menor, siendo crucial el procesamiento rápido de información ya que las características mencionadas anteriormente no son prioritarias. [15]

A lo largo de la historia, ha existido una constante inclinación hacia el diseño y construcción de máquinas capaces de ejecutar tareas propias de los seres humanos. Este impulso ha llevado al desarrollo de robots destinados a enfrentar ambientes peligrosos o situaciones que podrían afectar adversamente la calidad de vida de las personas, desempeñando actividades intrínsecas a la naturaleza humana. [13].

1.4.2 **Robótica de competencia en la educación**

En la actualidad, se impulsa el fomento del desarrollo de habilidades de programación desde una etapa temprana en la educación, con el objetivo de que los niños asuman un papel activo y creativo en el uso de tecnologías.

La creciente relevancia y evolución continua de la tecnología en la sociedad contemporánea ha llevado a que la tecnología misma se integre como una parte esencial del proceso educativo para la niñez y la juventud. En este contexto, es imperativo desarrollar propuestas que ofrezcan a los niños y jóvenes la oportunidad de familiarizarse

con las nuevas tecnologías, mediante el uso de herramientas de software y hardware, como prototipos robóticos y programas especializados con objetivos pedagógicos. [16]

1.4.3 Habilidades STEM

No podemos predecir cuáles serán los empleos del futuro, pero es posible imaginar qué competencias serán necesarias basándose en los cambios acontecidos en las décadas recientes. La ciencia, la tecnología y los nuevos valores sociales configuran el nuevo perfil esperado donde las habilidades STEM (Ciencia, Tecnología, Ingeniería y Matemáticas) deben desarrollarse desde la infancia para formar a mejores profesionales futuros. Para ello, la robótica educativa está mostrando excelentes resultados en el desarrollo de estas competencias. [17]

1.4.4 Sistema y Control automático

La historia del control automático abarca varios siglos y ha experimentado una evolución significativa desde los primeros mecanismos manuales hasta los sistemas digitales complejos de la actualidad. Iniciando en 1868 con el "regulador" inventado por William Mather, que mantenía la velocidad constante de un motor sin intervención humana, desde entonces se sentaron las bases para sistemas más avanzados de control automático. A lo largo del siglo XX, los sistemas digitales se expandieron en diversos sectores industriales, regulando temperatura, presión y guiando vehículos aéreos no tripulados. En las últimas décadas, con el avance tecnológico en robótica, inteligencia artificial, el control automático ha experimentado un crecimiento exponencial. [18]

De esta manera se puede afirmar que el control automático es una disciplina de la ingeniería que se enfoca en el diseño, análisis y aplicación de sistemas que regulan automáticamente el comportamiento de otros sistemas. Estos sistemas utilizan técnicas desarrollados en el campo de los sistemas de control para manipular variables de entrada y salida, con el objetivo de mantener ciertas variables en un estado deseado o seguir una trayectoria especifica. [19]

Por otra parte, los sistemas de control son un conjunto de componentes interrelacionados que trabajan juntos para manipular variables de entra y salida en un sistema o proceso. [20]

Para resumir el control automático se basa en los principios y técnicas desarrollados en el campo de los sistemas de control para diseñar e implementar sistemas automáticos que a su vez regulen el comportamiento de otros sistemas de manera eficiente y efectiva.

1.4.5 Control PID

Sin duda alguna, desde su introducción en 1940, los controladores PID (proporcional integral derivativo), son la opción más utilizada en las diferentes aplicaciones de control de procesos. Su éxito se debe principalmente, a la sencillez de su estructura (tres parámetros a sintonizar) y funcionamiento, que le permiten al ingeniero de control un mejor y fácil entendimiento, comparado con otras técnicas de control avanzadas. Este hecho ha motivado los continuos esfuerzos de investigación orientados a encontrar enfoques alternativos "de diseño y nuevas reglas de sintonía", con el fin de mejorar el rendimiento de los lazos de control con base en controladores PID. [21] [22]

En la industria, especialmente en procesos industriales, el controlador Proporcional, Integral y Derivado (PID) destaca como la principal herramienta de control. Este módulo PID se utiliza como un bloque de construcción para la regulación y el rechazo de perturbaciones en diversos esquemas de control, como bucle único, cascada, bucle múltiple y sistemas de múltiples entradas y salidas. A lo largo de las décadas, la tecnología de control PID ha evolucionado considerablemente, y hoy en día, el controlador PID puede existir como una rutina de utilidad estándar en el software del sistema de supervisión, una unidad controladora de proceso de hardware dedicada o un módulo de entrada-salida en un sistema electrónico programable. Con una tecnología industrial tan avanzada, no sorprende que los académicos también estén interesados en explorar nuevas capacidades y enfoques para el control PID. A pesar de las reglas clásicas de Ziegler-Nichols y los experimentos tradicionales, el campo del control PID sigue evolucionando. [23]

1.4.6 Lógica difusa

La lógica difusa, reconocida por su amplia variedad de aplicaciones, desde el control de procesos industriales hasta la construcción de dispositivos de deducción automática, ha experimentado un rápido crecimiento en la expedición de patentes industriales en todo el mundo. Aunque se atribuye su concepto a Lotfi A. Zadeh en 1965 en la Universidad de California en Berkeley, las lógicas difusas son, en realidad, lógicas multivaluadas que extienden las lógicas clásicas, las cuales se basan únicamente en valores de veracidad; verdadero o falso (true o false), representado matemáticamente en el conjunto {0,1} [24] Estas lógicas difusas buscan crear aproximaciones matemáticas para resolver problemas con datos imprecisos, siendo particularmente útiles en aplicaciones electrónicas y computacionales. El término "difuso" se refiere a la incertidumbre asociada con los valores de verdad no deterministas utilizados en estas lógicas. Se caracterizan por permitir grados de veracidad o falsedad más allá de los simples "verdadero" o "falso". Las lógicas difusas han encontrado aplicaciones relevantes en el procesamiento electrónico de datos, asociando valores de verdad con grados de veracidad o falsedad. Estos sistemas difusos se utilizan para describir y controlar procesos, y algunos parecen tener capacidades de aprendizaje. [2] [25]

Desde una perspectiva tecnológica, las lógicas difusas forman parte de la Inteligencia Artificial y han dado lugar a sistemas expertos y sistemas de control automático.

El concepto de conjuntos difusos, que difiere de la noción tradicional de conjuntos al asociar a cada elemento del universo un grado de pertenencia entre 0 y 1. Este grado de pertenencia indica cuánto un elemento está en el conjunto [26], siendo 1 para una pertenencia completa y 0 para ninguna. Un conjunto difuso se describe como una función cuyo dominio es el universo y su codominio es el intervalo [0, 1]. Se presentan ejemplos de conjuntos difusos relacionados con el concepto de empleado, mostrando cómo diferentes criterios, como horas trabajadas, tiempo empleado y salario, pueden definir grados de pertenencia variables. También se introduce la ponderación de tiempo e ingreso como una combinación lineal de grados de pertenencia anteriores. Además, se sugieren interpretaciones del grado de pertenencia, como la proporción en la que se posee un atributo o la probabilidad de un evento. [24]

La lógica difusa se utiliza en la programación de robots seguidores de línea, para interpretar información ambigua o imprecisa y llegar a conclusiones razonables.

De esta manera, para que el controlador difuso funcione, se siguen tres fases principales:

Fuzzycación: Es un proceso que permite asociar valores reales (numéricos) a valores lingüísticos, asignando un grado de pertenencia a cada uno de los términos lingüísticos (variables de entrada), a partir de la función de pertenencia del conjunto difuso. Este proceso es implementado por software a partir del conocimiento que es brindado por el razonamiento humano ya prestablecido. [27], es decir se convierten los datos a conjuntos difusos.

Inferencia Difusa: Razonamiento aproximado que permite derivar conclusiones desde un conjunto de reglas difusas tipo SI-ENTONCES y conjuntos difusos de entrada con valores de pertenencia, este resultado es asignado a un conjunto difuso (salida) con cada decisión de las reglas previamente evaluadas. [28].El método de inferencia es aquel que traduce matemáticamente las proposiciones por medio de técnicas de razonamiento, donde se encuentra el método de inferencia Mandani y el método de inferencia Takagi-Sugeno siendo los más usados en la actualidad. [27], de esta manera se obtiene el resultado a partir de una base de conocimiento.

Defuzzyficación: Encargado de convertir los valores difusos obtenidos del motor de inferencia difusa a un valor real (numérico) y es el representativo de todo el conjunto difuso, que posteriormente será el resultado del sistema experto, para que se lleve a cabo este componente debe utilizar métodos matemáticos. [27], Se interpretan los resultados difusos en datos que el robot pueda entender.

En el caso de los robots seguidores de línea, el controlador difuso utiliza etiquetas lingüísticas para establecer intervalos a los conjuntos difusos. Esto permite enviar señales de control menos bruscas a los motores, lo que reduce el deterioro y desgaste de las baterías y escobillas.

El **Error Cuadrático Medio (MSE, por sus siglas en inglés)** es una métrica ampliamente utilizada para evaluar el desempeño de sistemas, especialmente en **sistemas de control automático** y en áreas como aprendizaje automático, ingeniería, y procesamiento de señales. El MSE mide la **discrepancia promedio** entre los valores reales y los valores predichos o esperados.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(yi - \hat{y}i)^2 \quad (1)$$

Donde:

- yi : son los valores reales (o deseados).
- $\hat{y}i$: son los valores calculados o predichos por el sistema.
- n: es el número total de observaciones.

En términos prácticos, **cuantifica el error promedio elevado al cuadrado**, penalizando errores grandes de manera más significativa que errores pequeños.

En el contexto de los sistemas de control automático, el MSE se utiliza para evaluar el rendimiento del controlador al intentar minimizar las diferencias entre el **valor de referencia** (setpoint) y la **salida del sistema**. Por ejemplo, estrategias como el control PID, el MSE puede ser usado como criterio para ajustar los parámetros Kp, Ki, y Kd para lograr un comportamiento óptimo.

1.5 Estado del arte

En este estado del arte, se presenta una revisión de la literatura relacionada con el diseño y control de robots seguidores de línea. Se analizan diversas estrategias de control automático utilizadas en estos robots, con un enfoque particular en la comparación entre el control PID tradicional y los enfoques basados en lógica difusa.

Se llevó a cabo una búsqueda exhaustiva de artículos científicos, libros y tesis relacionados con la robótica, específicamente en el ámbito de los robots seguidores de línea. Se seleccionaron aquellos documentos relevantes que abordaban el diseño,

modelado y control de estos robots y similares, así como estudios que comparaban diferentes estrategias de control.

La historia de la robótica se remonta a civilizaciones antiguas como la griega, donde se ideaban mecanismos movidos por poleas y bombas hidráulicas. Sin embargo, el concepto moderno de robot empezó a tomar forma en la civilización árabe, donde se inició a dar sentido a estos mecanismos para facilitar la vida humana, marcando así el inicio de la evolución de la robótica. A lo largo del tiempo, se desarrollaron numerosas figuras con partes móviles, pero fue en la década de 1920 cuando el escritor checo Capek acuñó el término "robot" en su obra "Rossum's Universal Robots" (R.U.R.). Posteriormente, se introdujo el término "robótica" en su obra "I, Robot", junto con las tres leyes de la robótica. Hoy en día, los robots han evolucionado más allá de la mera imitación de los seres humanos, adoptando diversas formas para satisfacer diversas necesidades de manera óptima. La aparición de la computadora y los avances en la integración de circuitos en la década de 1940 permitieron los primeros intentos de desarrollo de robots verdaderamente autónomos. Este documento se enfoca en revisar la evolución y estado actual de los robots móviles para competición, así como en analizar los métodos utilizados para el seguimiento de trayectorias y la evasión de obstáculos [29].

La primera patente de un aparato robótico fue solicitada en Reino Unido. Sin embargo, en Estados Unidos (EE.UU.) es donde se dieron los desarrollos y se establecieron las bases del robot industrial que se utiliza hoy en día. Los primeros desarrollos en (EE.UU.) fueron hechos por el ingeniero George C. Devol, quien creó un dispositivo de transferencia de artículos programado. Devol da a conocer su idea a Joseph F. Engelberger, director de ingeniería de la división aeroespacial de la empresa Manning Maxwell y Moore en Estados Unidos. Devol y Engelberger comienzan trabajando en el uso industrial de sus máquinas, conforman una sociedad e instalan su primera máquina Unimate (1960) el cual realizaba soldadura automática, obedeciendo a ordenes almacenados en un tambor magnético en la fábrica de General Motors de Trenton, Nueva Jersey. Al mismo tiempo grandes empresas, comenzaron la construcción de máquinas similares. [30]

Los robots móviles surgieron de la curiosidad y capacidad humana de imitar su entorno, buscando crear máquinas que se comporten de manera similar a los humanos. Inicialmente, las máquinas industriales simplemente seguían órdenes, pero la robótica móvil persigue objetivos más ambiciosos: dotar al robot de un sistema locomotor y un sistema de control que le permitan desenvolverse en su entorno sin intervención humana.

Desde el punto de vista de la autonomía, los robots móviles tienen como antecesores a los dispositivos electromecánicos creados desde los años treinta, diseñados para realizar funciones inteligentes como descubrir caminos en laberintos. Un ejemplo notable es la tortuga de Walter, presentada en 1948, que podía reaccionar ante obstáculos, subir pendientes y dirigirse a una posición de recarga cuando su fuente de energía comenzaba a agotarse. [30]

En un enfoque general, “En el mundo de la robótica, cuando hablamos de un robot seguidor de línea se piensa en un dispositivo que, utilizando fotodiodos, va siguiendo una línea negra sobre un fondo blanco. La reacción de estos fotodiodos es muy rápida y, por tanto, si se dispone de un buen dispositivo de control, el robot se mueve rápidamente por el mapa siguiendo ágilmente la línea. Este tipo de robots son muy populares y se pueden realizar verdaderas maravillas utilizando pocos recursos.” [31]

“Con el gran avance de la robótica en el presente siglo, en el año 2008 salió al mercado electrónico un arreglo de sensores detectores de línea en forma de regla, Este dispositivo no era más que los antiguos sensores ópticos en una nueva presentación en montaje superficial, disminuyendo el tamaño y peso. En la actualidad, los LFR (Line Follower Robot) son estéticamente más profesionales, con una topología estandarizada e implementando controladores digitales, dejando de lado la lógica combinacional.” [32]

El presente trabajo, titulado, “Diseño y construcción de un robot seguidor de línea velocista”, detalla el diseño y desarrollo de un Robot Seguidor de Línea llamado Sprinter, destinado a competencias de robótica. Inicialmente, el diseño se determina a partir de un

análisis mecánico, cinemático y electrónico, siguiendo los lineamientos establecidos en los torneos de robótica. Además, se analiza el método de programación, el cual se basa en las ecuaciones cinemáticas derivadas para una configuración de motores diferencial. Esto permite definir el controlador PID encargado de mantener el robot centrado con respecto a la línea negra sobre una superficie blanca. [31]

Por otra parte, Jaime Gutiérrez, analiza sobre el desempeño de los controladores clásicos P, PD, PID y un controlador Difuso tipo Proporcional para resolver el problema de seguimiento de pared con un robot móvil con locomoción diferencial autónomo. El objetivo principal de emplear diversos sistemas de control radica en comparar el rendimiento al realizar la tarea de desplazarse de un punto (A), a un punto (B) y regresar al punto A siguiendo las paredes laterales. Este enfoque se utiliza comúnmente en robots de laberinto. Así, en el diseño de los controladores Difuso y Clásico, se consideró la distancia entre el robot móvil y la pared, medida mediante un sensor ultrasónico ubicado al frente y otros dos sensores adicionales, uno a cada lado del robot. Las distancias obtenidas son almacenadas en una memoria microSD, permitiendo analizar y comparar el comportamiento del robot móvil en forma gráfica para las diferentes acciones de **control propuestas al realizar la tarea seleccionada."** [33]**.**

González Acevedo, en su "Estudio comparativo entre tres técnicas de navegación para robots móviles", realiza un análisis comparativo entre algoritmos genéticos, redes neuronales y lógica difusa para la generación de trayectorias de robots móviles, revelando distintas ventajas y limitaciones. Los algoritmos genéticos son idóneos para entornos estructurados, donde una ecuación de aptitud previamente definida guía el proceso. Sin embargo, su incapacidad para explorar entornos desconocidos limita su utilidad en este contexto. Por otro lado, las redes neuronales ofrecen una eficiencia notable, superando el 70% en los resultados y requiriendo un menor costo computacional en comparación con los algoritmos genéticos. En cuanto a la lógica difusa, destaca su capacidad para incorporar la intuición humana en el control del robot, simplificando la sintonización de parámetros y no demandando cálculos complejos. No obstante, las simulaciones realizadas hasta el momento no consideran parámetros importantes como las constantes

de inercia del robot y las limitaciones mecánicas, lo que sugiere la necesidad de implementar algoritmos más robustos para mejorar la precisión de la navegación del robot y reducir el error de movimiento. [34]

En este estudio, se ha observado un creciente interés en el desarrollo de algoritmos de control que permitan a los robots navegar de manera autónoma en entornos dinámicos y desconocidos, se han evidenciado diferentes estudios que han abordado la importancia de calcular la distancia y la orientación del robot con respecto a obstáculos, en diferentes estudios se han destacado los desafíos asociados con la precisión de los sensores y la presencia de ruido en las señales. Sin embargo, a pesar de estos avances, persisten limitaciones en la precisión y la efectividad de los controladores, como lo señala Xavier Aguas. [35] Por lo tanto, existe una necesidad continua de desarrollar nuevas metodologías y técnicas de control que permitan a los robots móviles operar de manera más eficiente y robusta en una variedad de entornos y condiciones.

Soto Mayor y Sánchez, proponen el uso de un controlador borroso para controlar las velocidades del robot móvil, permitiendo un guiado eficaz en entornos desconocidos y la evasión de obstáculos. Los robots móviles, antes objeto de estudio, han evolucionado para convertirse en dispositivos prácticos utilizados en diversas aplicaciones, desde limpieza doméstica hasta tareas industriales. Para su desplazamiento seguro y eficiente, estos robots requieren sistemas computacionales capaces de realizar funciones de guiado y navegación. Existen diferentes métodos para capturar información del entorno y generar trayectorias, desde el uso de cámaras de tiempo de vuelo hasta algoritmos de control como los controladores PID y los enfoques basados en lógica difusa. Los resultados de simulación en entornos virtuales muestran un desempeño satisfactorio, lo que demuestra el potencial de esta técnica en aplicaciones prácticas. [36]

Este artículo, llamado “Control hibrido pid-difuso en robot seguidor de línea no holonómico” describe la fabricación de un robot seguidor de línea que utiliza un algoritmo de control híbrido basado en lógica difusa. Se implementó un controlador PD+I difuso para el control de la posición del robot y un controlador PI con realimentación

odométrica para el control de la velocidad. El robot estudiado es de tipo tracción diferencial, que desde un punto de vista matemático se considera un robot no holonómico, lo que significa que sus ecuaciones cinemáticas, dinámicas y de control son bastante complejas. En este contexto, la lógica difusa surge como una alternativa viable para desarrollar controladores basados en inteligencia artificial y adaptados al proceso a controlar [37].

En los últimos años, se han explorado diversas estrategias de control para robots, especialmente en el ámbito de los manipuladores industriales. Aunque el control PID es común en la robótica industrial, la complejidad de los brazos manipuladores y las incertidumbres asociadas a sus parámetros han llevado a investigar soluciones alternativas. Este estudio llamado, "Evaluación del desempeño de diversos controladores avanzados aplicados a un robot manipulador tipo scara", compara varios enfoques de control, incluyendo el control por par calculado, el control robusto, el control predictivo y el control por modos deslizantes, aplicados a un robot manipulador industrial. El objetivo es ofrecer alternativas al clásico control PID, mostrando su desempeño en diferentes trayectorias industriales y su robustez frente a incertidumbres. Este trabajo busca ampliar el conocimiento sobre opciones de control para manipuladores industriales, más allá del PID, que podrían ser útiles en aplicaciones industriales en Colombia y en otros lugares. [38]

De esta misma manera, José Ignacio Cazalilla Morenas se sumerge en el mundo de la ingeniería del software basada en componentes para dar vida a controladores avanzados destinados a un robot paralelo de 3 grados de libertad, mientras traza el camino hacia una innovadora aplicación para la rehabilitación de miembros inferiores. A través de un exhaustivo análisis del estado del arte en el desarrollo de componentes de software y controladores avanzados, se desvela una metodología que engloba desde el diseño de controladores dinámicos, adaptativos y híbridos hasta la creación de una aplicación completa para la rehabilitación. Estas estrategias no solo abordan problemas inherentes en la implementación de controladores para sistemas robóticos, sino que también alumbran una aplicación con un propósito noble y transformador. [39]

Según Alonso Álzate, Los controladores difusos representan una aplicación fundamental de la teoría difusa. A diferencia de los controladores convencionales, estos trabajan de manera distinta, utilizando el conocimiento experto en lugar de ecuaciones diferenciales para describir el comportamiento del sistema. Este conocimiento se expresa de manera natural mediante variables lingüísticas, las cuales son descritas a través de conjuntos difusos. La lógica difusa se ha convertido en una herramienta invaluable en el desarrollo de técnicas de control en robótica, dado su manejo efectivo de la incertidumbre inherente a las mediciones de los sensores. El uso de la lógica difusa ha experimentado un crecimiento significativo en diversas aplicaciones de control de agentes autónomos debido a sus características, que incluyen el tratamiento robusto de información imprecisa, la facilidad para interpolar mediciones de sensores y la flexibilidad en la definición de reglas para control no lineal. En este artículo se presenta un modelo del movimiento de la plataforma móvil y de la trayectoria a seguir. Se introduce un modelo difuso basado en las lecturas de los sensores, los cuales se caracterizan por un novedoso enfoque estadístico. [40]

2. DISEÑO Y CONSTRUCCIÓN DEL ROBOT PROTOTIPO PARA EXPERIMENTACIÓN

El diseño y construcción de un robot seguidor de línea es un reto que anualmente se actualiza en robótica móvil, utilizado tanto en entornos educativos como en aplicaciones industriales. Este tipo de robot sigue una línea predefinida en el suelo mediante el uso de sensores, materiales y algoritmos de control, ajustando su trayectoria en tiempo real.

En este capítulo, se detalla el proceso completo de creación de un robot seguidor de línea, abarcando desde la definición de requerimientos y la selección de materiales, hasta el ensamblaje, programación y pruebas del robot.

2.1 Materiales y diseño mecánico

En esta sección se abordan los aspectos técnicos de diseño como dimensiones, peso, forma, como la selección de materiales y componentes, para la construcción del robot, A continuación, se detallan los elementos clave de cada área.

2.1.1 **Chasis:**

Materiales: El chasis ha sido diseñado en SolidWorks y fabricado en fibra de vidrio, con un grosor de 2 mm, ofreciendo ligereza y resistencia. Esta estructura presentada en la ilustración 1 reduce el peso total del robot, lo que mejora tanto su maniobrabilidad como su eficiencia energética.

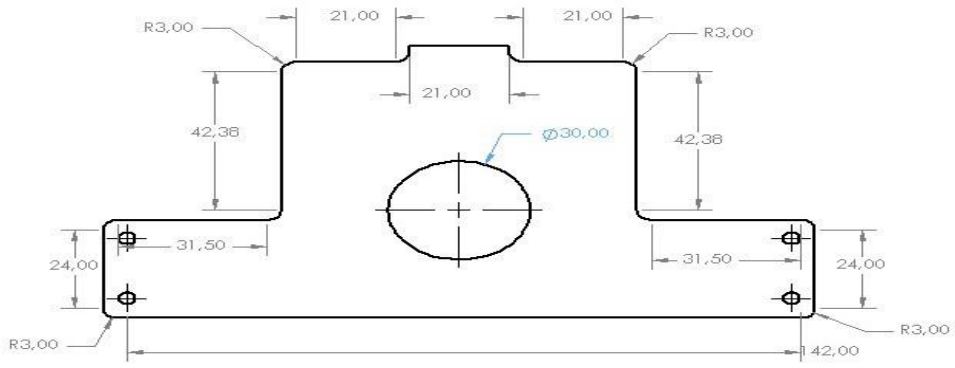


Ilustración 1.Diseño Chasis, Fuente: Elaboración propia

2.1.2 Soporte:

Materiales: El soporte del sensor como se muestra en la ilustración 2 está diseñado en SolidWorks, posteriormente se realizó la impresión en PLA (Ácido poliláctico), el cual es un termoplástico de alta rigidez y precisión dimensional. La base diseñada con las dimensiones precisas del sensor, proporciona un montaje firme y seguro.

2.1.2.1 Unión mediante barras de fibra de carbono: La fibra de carbono se emplea en las uniones entre el soporte del sensor, la placa de control y los motores gracias a su excelente relación entre peso y resistencia.

2.1.2.2 Diseño: El soporte presentado en la Ilustración 2 se diseñó en SolidWorks, esto con el fin de ajustar todos los componentes, incluyendo sensores, actuadores, y la batería, asegurando una distribución equilibrada del peso para una mejor estabilidad y maniobrabilidad.

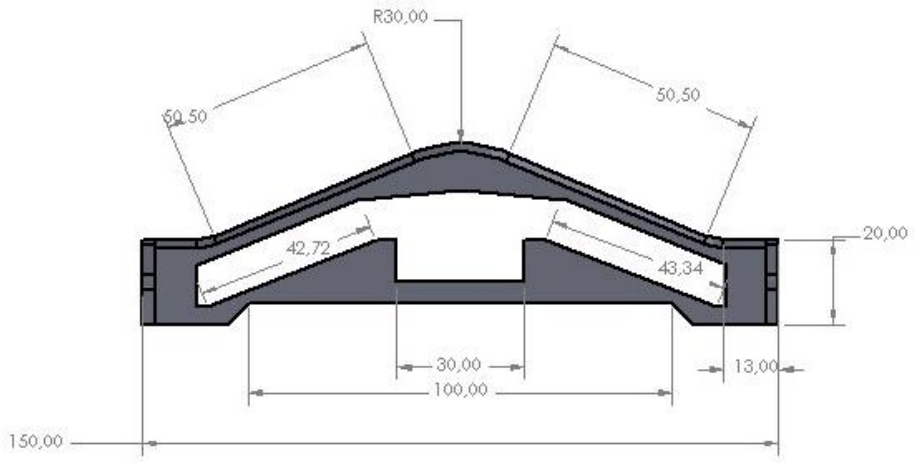


Ilustración 2.Diseño Soporte, Fuente: Elaboración propia

2.1.3 Locomoción:

Para la locomoción Se seleccionaron ruedas Jabots con rin de PLA de 15 mm de diámetro exterior y 12 mm de diámetro interior, una longitud de 40 mm, y un diámetro total de 23 mm (rin + llanta) como se muestra en la ilustración 3. El recubrimiento es de caucho de silicona con dureza 10A para mayor agarre en superficies lisas. Donde El diámetro del eje es de 3 mm (+/- 0.02 mm).

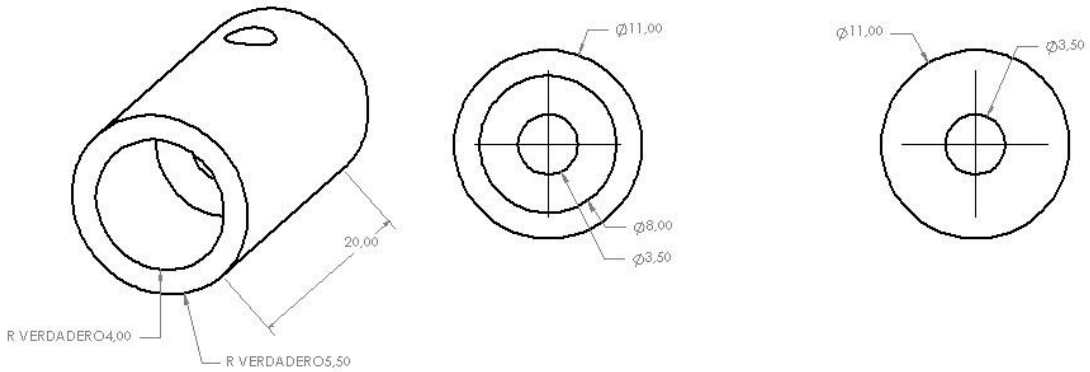


Ilustración 3.Ruedas Jabots y Diseño del Rin, Fuente: Elaboración propia

2.1.4 Motores:

Se seleccionaron 2 motores N20 de la marca Pololu como se muestra en la ilustración 4, debido a sus características y su eficiencia. Estos motores operan con un voltaje nominal de 6V. y cuentan con una relación de velocidad de 1:10, que proporcionan un equilibrio optimo entre potencia y control, con una velocidad máxima es de 3000 RPM (revoluciones por minuto), cada motor tiene unas dimensiones aproximadas de 25mm de largo,10mm de diámetro, y un peso cercano a 20g, lo que los hace compactos y ligeros, ideales para situaciones con restricciones de espacio.

Ilustración 4. Motores N20, Fuente: Elaboración propia

2.1.5 Batería:

La batería de 7.4V de la marca Rline tiene una capacidad de 550 mAh (miliamperios hora) y una tasa de descarga de 95C como se muestra en la ilustración 5, lo que proporciona una alta capacidad de descarga rápida. Sus dimensiones son aproximadamente 64 mm de largo, 34 mm de ancho y 16 mm de alto, con un peso de alrededor de 70 g. Esta batería utiliza un conector XT30, que es común en aplicaciones de robots para asegurar una conexión segura.

Ilustración 5. Batería Rline, Fuente: Elaboración propia

2.1.6 Turbina EDF27:

Esta turbina funciona con una batería lipo de 2 celdas, que proporciona un voltaje de 7.4V a 8.5V, Con un diámetro interior de 27 mm y un diámetro exterior de 29 mm como se muestra en la ilustración 6, su peso es de aproximadamente 8 g. Y requiere un controlador de tipo ESC de al menos 10A para su correcto funcionamiento.

Ilustración 6. Turbina EDF27, Fuente: Elaboración propia

2.1.7 Soporte Motor N20

El soporte para motor N20 de la marca Pololu está diseñado para sujetar micromotores Pololu como se muestra en la ilustración 7, compatible con los modelos HP y HPCB. Fabricado en plástico negro, este soporte protege la caja de engranajes del motor y facilita su ajuste a la base. Con un peso de solo 0.7 g, es ideal para aplicaciones como seguidores de línea y motorreductores.

Ilustración 7. Soporte Motor N20, Fuente: Elaboración propia

2.1.8 Barras de carbón:

Las barras de fibra de carbono de 1.5 mm x 50 cm son ideales para unir el soporte del motor con el chasis base de los robots, brindando una conexión ligera y resistente entre la parte de control y los sensores como se muestra en la ilustración 8. Estas barras, ampliamente utilizadas en robots seguidores de línea, son apreciadas por su durabilidad y bajo peso, con un rango de peso entre 1.5 g, 2 g y 6 g, dependiendo del modelo.

Ilustración 8. Barras de carbón, Fuente: Elaboración propia.

2.1.9 Tornillos:

Los tornillos M2 de cabeza plana avellanada, fabricados en acero inoxidable, tienen un largo de 10 mm, diámetro nominal de 2 mm, ancho de cabeza de 3.5 mm, y diámetro de tuerca de 2.02 mm como se muestra en la ilustración 9. Cuentan con una cabeza hexagonal tipo Bristol, lo que facilita su ajuste y asegura una sujeción firme. Con métrica M2 y un recubrimiento de color negro, estos tornillos son resistentes a la corrosión y proporcionan una sujeción confiable y duradera para todas las sujeciones en el robot.

Ilustración 9. Tornillos M2, Fuente: Elaboración propia.

2.1.10 Distribución de Componentes

La distribución equilibrada y ordenada de los componentes del robot seguidor de línea es esencial para optimizar su rendimiento y estabilidad como se ve en la ilustración 10. Por lo tanto, se garantiza una fácil accesibilidad para el mantenimiento y ajustes, al mismo tiempo que se preserva la solidez estructural y funcional del robot.

2.1.10.1 Distribución

Para asegurar un buen equilibrio y un centro de gravedad bajo, las baterías y los motores, que son los componentes más pesados del robot, se colocaron cerca del

centro formado por los ejes de los motores. Esto proporciona mayor tracción y evita que las ruedas se resbalen, mejorando la estabilidad durante el movimiento. Además, el microcontrolador se ubicó de manera estratégica como centro de conexiones para todos los demás componentes, lo que reduce el cableado y facilita su accesibilidad para la programación. Esta disposición no solo optimiza el espacio, sino que también permite un mantenimiento rápido y sencillo sin necesidad de desmontar completamente el robot.

Los motores se montaron en la parte trasera del chasis y se conectan directamente a las ruedas motrices. Se usaron soportes plásticos para fijar los motores, reduciendo la vibración y asegurando rígidamente el motor al chasis, Además, los soportes protegen completamente la caja reductora de engranajes de los motores evitando que ingrese suciedad a la piñonearía, lo cual malogra los motores.

El cableado se organizó con canaletas para mantener un interior ordenado y reducir el riesgo de interferencias y enredos.

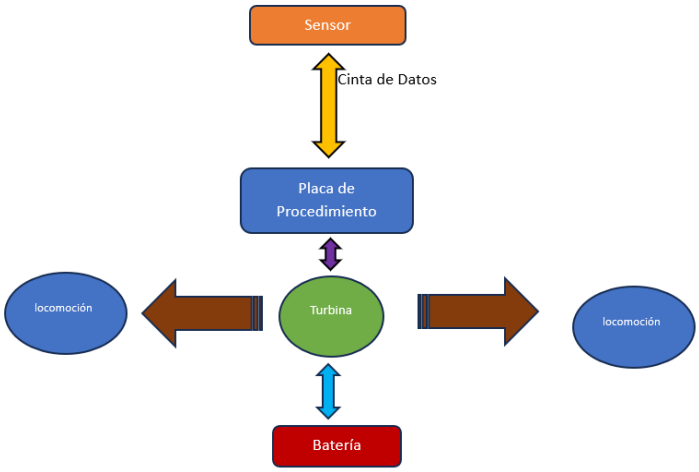


Ilustración 10. Distribución de Componentes, Fuente: Elaboración propia

2.2 Diseño Electrónico

En términos eléctricos y electrónicos el robot consta de un microcontrolador ESP32 como elemento de control y programación el cual a su vez recibe información de sensores y manipula actuadores para que el robot se mantenga dentro del circuito de línea a seguir como se muestra en la ilustración 11. El robot consta de un sensor de 16 canales infrarrojos el cual

posee un multiplexor para seleccionar mediante una dirección binaria uno de los 16 canales a ser leído a través del pin analógico del microcontrolador. El sensor entrega una lectura digital comprendida en una resolución de 10 bits es decir un rango de palabra digital de cero a 1024, este dato es pasado a través de la ecuación del promedio ponderado para el sensor al igual que el valor entregado por los otros 15 canales de este. Finalmente, este valor del promedio ponderado es transformado a un valor de entre -1.0 a +1.0 tomando variaciones porcentuales alrededor del centro de la línea cero.

El robot consta de 2 drivers tipo puente h los cuales se conectan cada uno a una pareja de pines del microcontrolador, el primer pin es un pin de acción digital que indica uno el avance y cero el retroceso del motor, mientras que el otro pin es configurado con funcionalidad de modulación de ancho de pulso PWM y permite variar la velocidad del motor.
En robot posee una turbina para generar el efecto suelo sobre el mismo robot, esta turbina se manipula en velocidad mediante PWM y un módulo adicional ESC12A para motores brushless.

La fuente de alimentación del robot es una batería de lipo de 7.4 voltios y 550 miliamperios hora. El microcontrolador y en general la tarjeta poseen comunicación inalámbrica tipo Bluetooth y Wi-Fi.

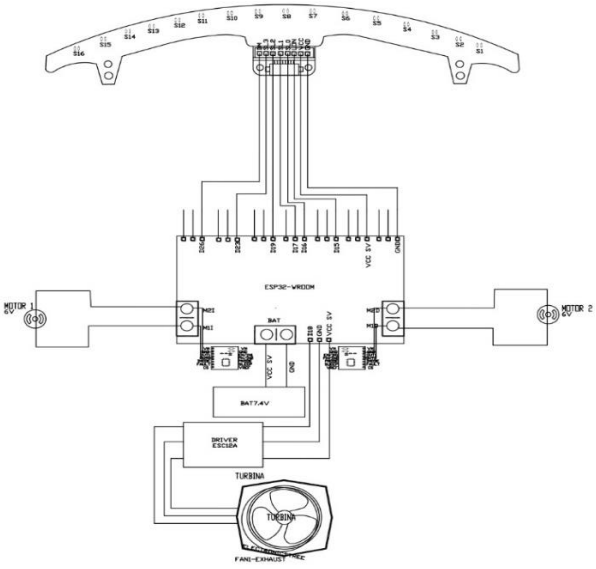


Ilustración 11.Diseño Eléctrico, Fuente: Elaboración propia

2.2.1 Microcontrolador

Un microcontrolador es el componente central de muchos sistemas de control en robótica, especialmente en robots móviles tipo seguidor de línea. Este dispositivo es esencial para manejar múltiples entradas y salidas y ejecutar algoritmos de control en tiempo real. Los microcontroladores populares incluyen Arduino y Raspberry Pi, cada uno con sus características, y ventajas y desventajas.

2.2.2 Sensor

2.2.2.1 Sensores de Línea

Se utilizó una barra de sensores infrarrojos alineados en la parte frontal, a 2 mm de la superficie, optimizando la detección de la línea como se muestra en la ilustración 12. Los sensores miden la cantidad de luz reflejada, generando una señal que el microcontrolador procesa para ajustar dirección y velocidad. La barra de sensores se conectó a la tarjeta Sumo Force, y se calibraron los umbrales de detección a través de un ADC, que escala las mediciones entre 0 y 15,000 unidades, donde 0 indica línea a la izquierda, 15,000 a la derecha y 7,500 en el centro.

Ilustración 12.Sensor XL 16 Barrera.ino, Fuente: Elaboración propia

2.2.3 Driver controlador de turbina:

El controlador ESC12A es un regulador de velocidad electrónico (ESC) que soporta hasta 12A de corriente continua y 15A de ráfaga durante 10 segundos, siendo ideal para motores brushless como la turbina EDF27 como se muestra en la ilustración 13. Compatible con baterías LiPo de 2 celdas (7.4V). Este ESC cuenta con un modo BEC lineal con salida de 1A/5V, es programable y tiene dimensiones de 25 mm x 20 mm x 7 mm con un peso aproximado de 7 g. El cual es adecuado para aplicaciones en seguidores de línea.

Ilustración 13.Driver controlador de turbina, Fuente: Elaboración propia

2.2.4 Placa de procesamiento

La placa de procesamiento Sumo Force está basada en el microcontrolador ESP32, que cuenta con conectividad Bluetooth y Wi-Fi para una comunicación inalámbrica eficiente como se muestra en la ilustración 14. Dispone de 34 pines de entrada/salidas digitales, 16 pines PWM para control preciso de motores, y hasta 18 pines analógicos gracias a su ADC integrado. Además, incluye interfaces para sensores y motores, facilitando la programación directa y el control en tiempo real. La tarjeta es compacta y robusta, diseñada para aplicaciones de robótica como robots seguidores de línea, con protección contra sobrecargas y compatibilidad con varias fuentes de alimentación.

Ilustración 14. Placa de procesamiento, Fuente: Elaboración propia

2.2.5 Módulo de inicio

Él módulo de inicio es compatible con señales de 5V y 3.3V como se muestra en la ilustración 15, requiriendo solo un pin de entrada en Arduino o cualquier microcontrolador. Sus dimensiones compactas (10.5 mm x 12.5 mm x 4 mm) lo hacen ideal para robots de alta competición. Dispone de cuatro pines de conexión: VCC, GND, Start y Stop, y su diseño de código y hardware abiertos facilita su uso y programación.

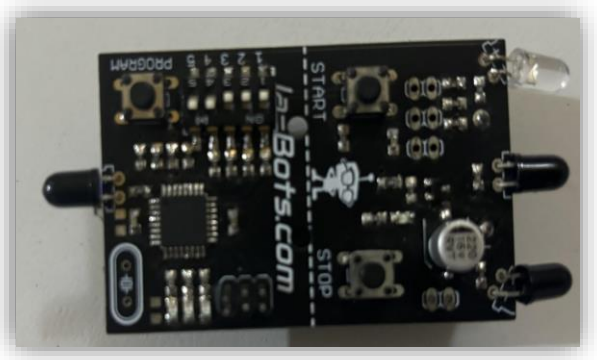


Ilustración 15. Módulo de inicio, Fuente: Elaboración propia

2.2.6 Driver DVR 8876:

Es un controlador avanzado para motores que funciona con un voltaje de 3-10 VDC y proporciona una corriente de salida máxima de 1.5 A, con picos de hasta 2 A como se muestra en la ilustración 16. Incorpora protección contra sobrecorriente, cortocircuito y sobrecalentamiento, y cuenta con un modo sleep de bajo consumo. Sus dimensiones son aproximadamente 18.1 x 15.6 mm. Además, su modo de corriente constante y control PWM (modulación de ancho de pulso) permiten ajustar la velocidad de los motores con precisión, garantizando una gestión eficiente de la potencia y una respuesta rápida a las variaciones en la carga para un rendimiento estable y fiable.

DRV8876 Single Brushed DC Motor Driver Carrier (top view).

Ilustración 16.Driver DRV8876, Fuente: Elaboración propia

2.2.7 Cinta de datos sensor:

La cinta de conexión bus entre sensores y la tarjeta de procesamiento, conocida como cable FFC (Flexible Flat Cable) como se muestra en la ilustración 17, es un circuito impreso flexible compuesto por una película plana de plástico con conductores de cobre. Su diseño incluye un tensor que refuerza la superficie, facilitando la conexión con el conector. Este cable tiene una longitud de 20 cm, una distancia entre pines de 1.0 mm y una cantidad de 16 vías, además de soportar temperaturas de hasta 80 °C, lo que lo convierte en una solución compacta y flexible para conexiones electrónicas.

Ilustración 17. Cinta de datos sensor, Fuente: Elaboración propia

A continuación, se presenta el prototipo final del seguidor de línea como se muestra en la ilustración 18, con todos sus componentes electrónicos organizados y conectados adecuadamente.

Ilustración 18. Robot seguidor de línea, Fuente: Elaboración Propia

3.MODELO CINEMÁTICO DEL ROBOT SEGUIDOR DE LÍNEA

Un sistema es un conjunto de elementos o partes funcionales que se conectan entre sí, porque han sido diseñados con un objetivo común. Los sistemas pueden tener elementos de diferente tipo; eléctricos, electrónicos, hidráulicos, mecánicos, neumáticos. En cuanto al modelo cinemático de un robot seguidor de línea, este es una representación matemática que describe cómo se mueve el robot en función de sus entradas y la geometría de su diseño. En este caso, el robot seguidor de línea utiliza un sistema diferencial (dos ruedas motrices independientes) para seguir un trayecto marcado por una línea en el suelo.

3.1 Modelo Matemático de un Sistema

Es la expresión matemática por medio de ecuaciones diferenciales que describe el comportamiento dinámico de un sistema. Se usan para estudiar la relación entre los elementos del sistema. Son base para el análisis y diseño de sistemas de control automático, en todas las áreas tecnológicas que rodean al ser humano; industria, medicina, deporte, servicios, milicia.

- **Cinemática de robots**

  La cinemática de robots estudia el movimiento de los robots, enfocándose en la posición, desplazamiento y velocidad. No considera las fuerzas que originan el movimiento, el peso, los coeficientes de fricción ni la gravedad. Se limita al estudio de la trayectoria del robot en el espacio a través del tiempo, utilizando velocidades, aceleraciones y la geometría del robot.

Tipos de cinemática de robots-

- **Cinemática directa**

  Permite obtener la posición y la orientación del robot, en función de las velocidades y posiciones de las ruedas, de igual manera permite estimar donde se encontrará tras un cierto tiempo en función de las velocidades de las ruedas.
- **Cinemática inversa** Permite obtener las velocidades y posiciones de las ruedas que

hacen que la posición y orientación del robot sea la deseada, así mismo permite generar velocidades en las ruedas para alcanzar una localización deseada.

El modelo cinemático de un robot seguidor de línea es fundamental para comprender y predecir su comportamiento en función de las entradas del sistema. Toda vez que describe la relación entre las velocidades y posiciones de las ruedas del robot y su movimiento en el plano. [41]

3.2 Modelado cinemático de robot móvil tipo diferencial

Un robot móvil tipo diferencial es un tipo de robot que se desplaza sobre un plano de movimiento bidimensional definido por las coordenadas XX & YY. La posición del robot en este plano se describe mediante las coordenadas (x,y)(x,y), que indican la ubicación del eje del robot en el plano.

El robot puede rotar sobre su propio eje, y este ángulo de rotación se denota como ϕ. La velocidad a la que el robot rota sobre su propio eje se llama velocidad angular y se representa con w.

El movimiento del robot se controla mediante las velocidades de giro de sus llantas, denotadas como Wi para la llanta izquierda y Wd para la llanta derecha. La distancia entre las dos ruedas del robot se indica como 2L. Finalmente, la velocidad de avance del robot, que es la velocidad a la que el robot se desplaza hacia adelante en el plano, se denota como V como se puede observar en la ilustración 19. Esta velocidad de avance está relacionada con las velocidades de giro de las llantas y la geometría del robot.

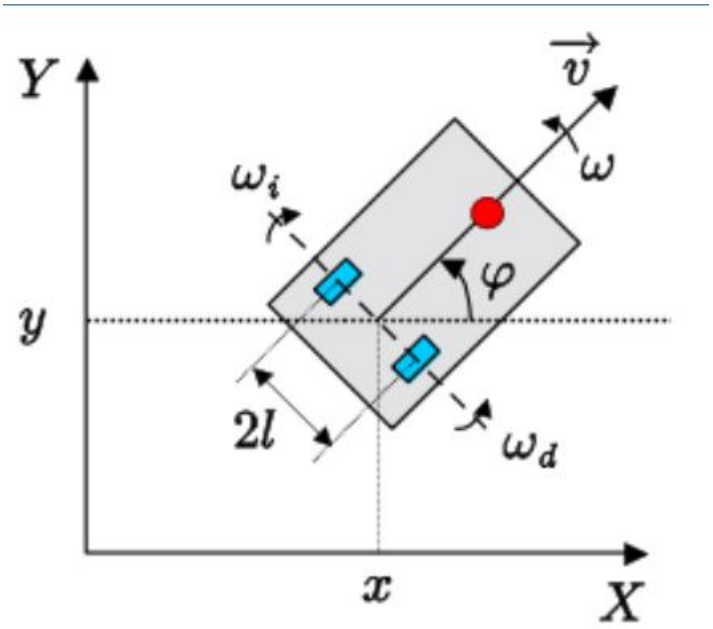


Ilustración 19. Plano de Movimiento, Fuente: [42]

Así miso el robot se desplaza usando dos ruedas accionadas de manera independiente. Estas ruedas pueden girar a diferentes velocidades, permitiendo que el robot se mueva en distintas direcciones y gire sobre su propio eje.

- **Radio de las llantas "r"**

  El radio de las llantas, es la distancia desde el centro de la llanta hasta su borde exterior. Este parámetro es fundamental porque afecta cómo las velocidades de las llantas se traducen en movimientos del robot.

3.2.1 Arco o distancia recorrida por la llanta

La distancia que una llanta recorre al girar se puede calcular utilizando el radio de la llanta y el ángulo de rotación. Si una llanta gira un ángulo θ (medido en radianes), la distancia recorrida por un punto en el borde de la llanta, conocida como el arco, se puede calcular con la fórmula:

$$s = r\theta \qquad (2)$$

Donde:

- S es la distancia recorrida por la llanta.
- r es el radio de la llanta.
- θ es el ángulo de rotación de la llanta en radianes.

El radio de las llantas r y la fórmula S=rθ son esenciales para entender cómo las rotaciones de las llantas del robot móvil tipo diferencial se traducen en desplazamientos lineales como lo muestra la ilustración 20, lo cual es crucial para controlar el movimiento del robot.

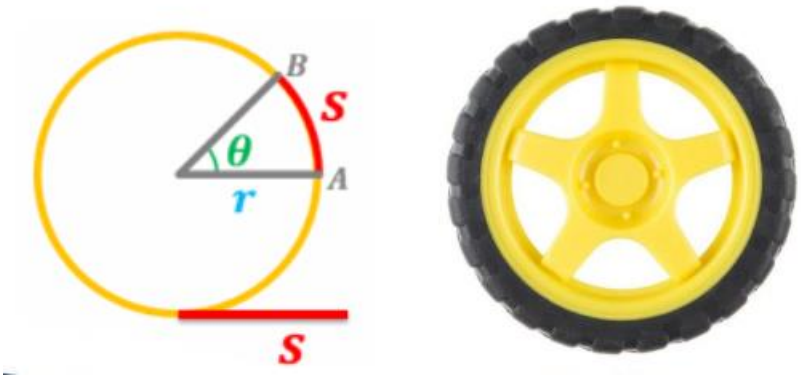


Ilustración 20. Radio de las llantas y Arco recorrido por llanta, Fuente: [42]

3.2.2Cinemática Diferencial

El robot seguidor de línea típicamente se basa en un sistema de tracción diferencial, donde dos ruedas accionadas independientemente controlan la dirección y la velocidad del robot. El modelo cinemático diferencial se expresa mediante las siguientes ecuaciones:

- **Velocidad lineal V**

  Es la velocidad a la que el robot avanza en una dirección recta. Se mide en unidades de distancia por unidad de tiempo (por ejemplo, metros por segundo).

- **Velocidad angular W**

  Es la velocidad a la que el robot gira alrededor de su propio eje. Se mide en unidades de ángulo por unidad de tiempo (por ejemplo, radianes por segundo).

- **Velocidad lineal de avance de las llantas**

  Para un robot móvil tipo diferencial, la velocidad lineal de avance de cada llanta puede calcularse a partir de la velocidad angular de cada una y el radio de la llanta.

- **Velocidad lineal de la llanta derecha Vd**

  La velocidad lineal de la llanta derecha Vd se puede calcular con la fórmula:

$$Vd = r * Wd \quad (3)$$

Donde:

- Vd es la velocidad lineal de la llanta derecha.
- r es el radio de la llanta.
- Wd es la velocidad angular de la llanta derecha.

- **Velocidad lineal de la llanta izquierda Vi**

  La velocidad lineal de la llanta izquierda Vi se puede calcular así:

$$Vi = r * wi \quad (4)$$

Donde:

- Vi es la velocidad lineal de la llanta izquierda.
- r es el radio de la llanta.
- wi es la velocidad angular de la llanta izquierda.

3.2.3 Velocidad total instantánea de avance V del robot

La velocidad total instantánea de avance del robot V es la media de las velocidades de avance de las llantas derecha e izquierda. Esto se debe a que el movimiento del robot en línea recta es el resultado combinado de ambas llantas.

La fórmula para calcular la velocidad total instantánea de avance del robot es:

$$v = \frac{vd+v_i}{2} = \frac{r}{2} * (wd + wi) \quad (5)$$

Esto significa que la velocidad total instantánea de avance del robot depende directamente de la suma de las velocidades angulares de ambas llantas como se muestra en la ilustración 21, multiplicada por el radio de las llantas y dividida por dos. Esta fórmula permite determinar qué tan rápido se mueve el robot en línea recta en función de las velocidades de sus llantas.

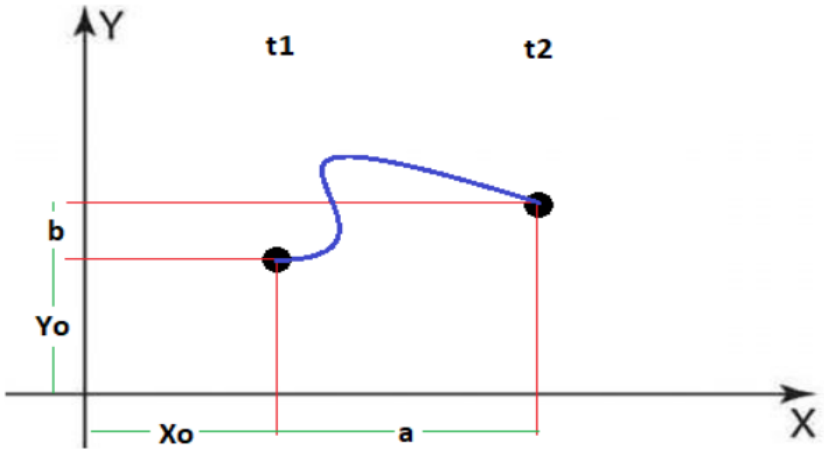


Ilustración 21.Posición (x, y), Fuente: [42]

3.2.4 Posición del robot en el plano (x, y)

En un sistema de coordenadas bidimensional (x,y), la posición del robot en cualquier instante se puede describir por sus coordenadas (Xpos, Ypos) como lo muestra la ilustración 22. Estas coordenadas representan la ubicación del centro del eje del robot en el plano.

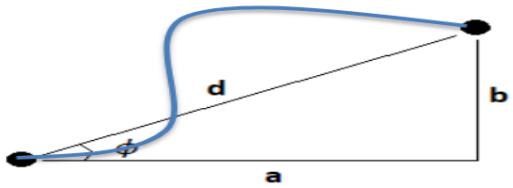


Ilustración 22.Trayectoria de un robot móvil entre dos puntos, Fuente: [42]

- Determinación de la posición (x, y) del robot

  La nueva posición en el plano (x, y) puede calcularse considerando la distancia d recorrida y el ángulo de orientación ϕ.

  Supongamos que el robot empieza en una posición inicial (X0, Y0) después de moverse una distancia d con un ángulo de orientación ϕ, los desplazamientos a y b se calculan como:

$$a = d * cos(\phi) \quad (6)$$

$$b = d * sin(\Phi) \quad (7)$$

A continuación, se explica cómo se relacionan la velocidad, la distancia recorrida y el tiempo en el contexto de un robot móvil tipo diferencial, y cómo estas fórmulas se aplican para calcular la posición del robot.

3.2.5Velocidad

la velocidad se va medir con la siguiente formula;

$$v = \frac{distancia\ d}{tiempo\ t} \quad (8)$$

Donde:

- v es la velocidad
- d es la distancia recorrida.
- t es el tiempo que se tarda en recorrer esa distancia.

Esta fórmula indica que la velocidad es la razón entre la distancia recorrida y el tiempo que se tarda en recorrer esa distancia.

Distancia, esta se va medir de la siguiente manera;

$$d = velocidad * tiempo \quad (9)$$

Donde:

- d es la distancia recorrida.
- v es la velocidad.
- t es el tiempo.

- Distancia, Es la longitud del camino recorrido por el robot. Se mide en unidades de longitud, como metros (m).
- Velocidad, Es la rapidez con la que el robot se mueve. Se mide en unidades de longitud por unidad de tiempo, como metros por segundo (m/s).
- Tiempo, Es el intervalo durante el cual el robot se está moviendo. Se mide en unidades de tiempo, como segundos (s).

$$d = \frac{vd+vi}{2} * t \quad (10)$$

- *v*: Es la velocidad total instantánea de avance del robot.
- *vd*: Es la velocidad lineal de avance de la llanta derecha.
- *vi:* Es la velocidad lineal de avance de la llanta izquierda.

Las velocidades de las llantas derecha e izquierda pueden ser diferentes. Esta diferencia en las velocidades es lo que permite al robot girar. La velocidad total instantánea de avance del robot se calcula como el promedio de las velocidades de las dos llantas.
La razón para tomar el promedio es que el centro del robot se mueve a una velocidad que es la media de las velocidades de las dos llantas, ya que una llanta puede estar avanzando más rápido o más lento que la otra.

$$a = \frac{vd+vi}{2} * t * \cos(\phi) \quad (11)$$

Donde:

- *a:* Es el desplazamiento en el eje X
- *vd*: Es la velocidad lineal de avance de la llanta derecha.
- *vi:* Es la velocidad lineal de avance de la llanta izquierda.
- *t*: Es el tiempo durante el cual el robot se mueve.
- *cos:* Es el coseno del ángulo de orientación del robot.

De esta manera se calcula el desplazamiento en el eje X (denotado como *a*) del robot, después de un cierto tiempo, considerando las velocidades de sus llantas y su ángulo de orientación ϕ.

- **Velocidad media:** es la velocidad total instantánea de avance del robot, calculada como el promedio de las velocidades de las dos llantas.
- **Desplazamiento lineal en el tiempo**, multiplicando la velocidad media por el tiempo, obtenemos la distancia total recorrida por el robot en ese tiempo.
- **Componente en el eje X**, multiplicando la distancia total recorrida por el coseno del ángulo obtenemos el componente de esa distancia en la dirección del eje X.

$$b = \frac{vd+vi}{2} * t * \sin(\phi) \quad (12)$$

Donde:

- *b*: Es el desplazamiento en el eje Y.
- *vd:* Es la velocidad lineal de avance de la llanta derecha.
- *vi*: Es la velocidad lineal de avance de la llanta izquierda.
- *t*: Es el tiempo durante el cual el robot se mueve.
- *sin*: Es el seno del ángulo de orientación del robot

Con esta fórmula se calcula el desplazamiento en el eje Y (denotado como b) del robot, después de un cierto tiempo, considerando las velocidades de sus llantas y su ángulo de orientación.

$$Xpos = x_o + \frac{vd+vi}{2} * t * cos(\phi) \quad (13)$$

Donde:

- $Xpos$: Es la posición final en el eje X del robot.
- x_o: Es la posición inicial en el X del robot.
- vd : Es la velocidad lineal de avance de la llanta derecha.
- vi : Es la velocidad lineal de avance de la llanta izquierda.
- t: Es el tiempo durante el cual el robot se mueve.
- cos: Es el diseño del ángulo de orientación del robot.

De esta manera, es posible calcular la nueva posición en el eje X del robot, después de un cierto tiempo, considerando las velocidades de sus llantas y su ángulo de orientación.

$$Ypos = y_o + \frac{vd+vi}{2} * t * sin(\phi) \quad (14)$$

Donde:

- $Ypos$: Es la posición final en el eje Y del robot.
- y_o: Es la posición inicial en el eje Y del robot.
- vd: Es la velocidad lineal de avance de la llanta derecha.
- vi: Es la velocidad lineal de avance de la llanta izquierda.
- t: Es el tiempo durante el cual el robot se mueve.
- sin: Es el seno del ángulo de orientación del robot.

De este modo, se puede determinar la nueva posición en el eje Y del robot después de un cierto tiempo considerando las velocidades de sus llantas y su ángulo de orientación.

$$Xpos = X_o + \frac{r(wd+wi)}{2} * t * cos(\phi) \quad (15)$$

Donde:

- $Xpos$: Es la nueva posición en el eje X del robot.
- X_o: Es la posición inicial en el eje X del robot.
- r: Es la radio de las llantas.
- wd: Es la velocidad de giro de la llanta derecha.
- wi: Es la velocidad de giro de la llanta hacia la izquierda.
- t: Es el tiempo transcurrido.
- cos: Es el ángulo de orientación del robot.

3.2.6Orientación

Velocidad angular del robot

$$w = \frac{v_r - vi}{2L} = \frac{r(wd - wi)}{2L} \quad (16)$$

Donde:

- *w:* Es la velocidad angular del robot.

- *vr*: Es la velocidad lineal de la llanta derecha.
- *vi:* Es la velocidad lineal de la llanta izquierda.
- *L*: Es la mitad de la distancia entre las dos ruedas del robot.
- *r*: Es el radio de las llantas del robot.
- *wd*: Es la velocidad angular de la llanta derecha.
- *wi* Es la velocidad angular de la llanta izquierda.

La Velocidad angular media se calcula como la diferencia entre las velocidades lineales de las llantas derecha e izquierda, divididas por dos veces la longitud entre las ruedas (2L) o como la diferencia entre las velocidades angulares de las llantas derecha e izquierda, divididas por dos veces la longitud entre las ruedas (2L).

Posición angular del robot

$$\phi = \phi_0 + w * t \tag{17}$$

Donde:

Esta fórmula se utiliza para calcular la posición angular (orientación) del robot en un momento determinado.

- ϕ: Es la posición angular (orientación) actual del robot.
- ϕ_0 : Es la posición angular inicial del robot.
- *w*: Es la velocidad angular del robot.
- *t:* Es el tiempo transcurrido.

Además de la orientación del robot, es posible calcular su posición angular utilizando la siguiente ecuación:

$$\phi = \phi_0 + \frac{r(wd-wi)}{2L} * t \tag{18}$$

- ϕ :Es la posición angular (orientación) actual del robot.
- ϕ_0: Es la posición angular inicial del robot.
- *r*: Es el radio de las llantas del robot.

- *wd:* Es la velocidad angular de la llanta derecha.
- *wi:* Es la velocidad angular de la llanta izquierda.
- *L*: Es la mitad de la distancia entre las ruedas del robot.
- *t*: Es el tiempo transcurrido.

3.2.7Posición y orientación

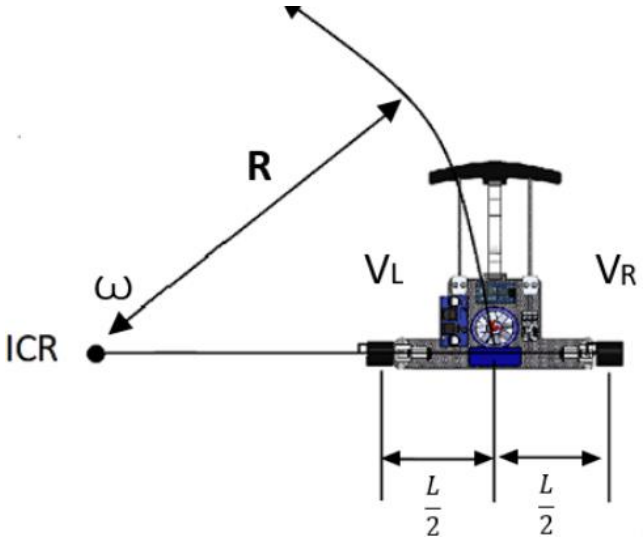


Ilustración 23.Posición y orientación del robot, Fuente: [36]

$$\dot{x} = \frac{r(Wd*Wi)}{2} * cos(\Phi) \quad (19)$$

Esta fórmula se utiliza para calcular la velocidad del robot en la dirección del eje X en un instante dado.

Donde:

- *x*: Es la velocidad en la dirección del eje X.
- *r:* Es el radio de las llantas del robot.
- *wd:* Es la velocidad angular de la llanta derecha.
- *wi*: Es la velocidad angular de la llanta izquierda.
- ϕ: Es la orientación angular del robot en ese instante.

Con la formula anterior, se busca calcular la velocidad instantánea del robot en la dirección del eje X, teniendo en cuenta las velocidades angulares de sus llantas y su orientación angular.

$$\dot{y} = \frac{r(Wd*Wi)}{2} * sin(\Phi) \quad (20)$$

Con esta fórmula se calcula la velocidad del robot en la dirección del eje Y en un instante dado.

Donde:

- *y*: Es la velocidad en la dirección del eje Y.
- *r:* Es el radio de las llantas del robot.
- *wd*: Es la velocidad angular de la llanta derecha.
- *wi*: Es la velocidad angular de la llanta izquierda.
- ϕ*:* Es la orientación angular del robot en ese instante.

Velocidad angular del robot

$$w = \frac{vr-vi}{2L} = r\frac{(wd-wi)}{2L} \quad (21)$$

Posición angular del robot

$$\emptyset = \emptyset_0 + W * t \quad (22)$$

$$\emptyset = \emptyset_0 + \frac{r\,(Wd+Wi)}{2L} * t \quad (23)$$

Ecuaciones finales de cinemática directa

$$\dot{X} = \frac{r\,(Wd+Wi)}{2} * cos(\emptyset) \quad (24)$$

$$\dot{Y} = \frac{r\,(Wd+Wi)}{2} * sin(\emptyset) \quad (25)$$

$$\dot{\emptyset} = = \frac{r\,(Wd+Wi)}{2L} \quad (26)$$

Con esta ecuación se calcula la velocidad angular instantánea del robot, para determinar qué tan rápido está girando el robot alrededor de su eje central.

Donde:

- ϕ: Es la velocidad angular del robot, es decir, la tasa de cambio de la orientación del robot respecto al tiempo.
- *r*: Es el radio de las llantas del robot.
- *wd:* Es la velocidad angular de la llanta derecha.
- *wi*: Es la velocidad angular de la llanta izquierda.
- *L*: Es la mitad de la distancia entre las dos ruedas del robot.

Modelo en espacio de Estados matricial del sistema de tipo no lineal.

El modelo en espacio de estados, es una representación matemática que describe el comportamiento dinámico de un sistema mediante un conjunto de ecuaciones diferenciales de primer orden. En el caso de un sistema robótico, este modelo es crucial para analizar y diseñar controladores que permitan a los robots realizar tareas específicas de manera eficiente y estable.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix}_{3x1} = \begin{bmatrix} \cos(\phi) & 0 \\ 0 & Sen(\phi) \\ 1/2 & 0 \end{bmatrix}_{3x2} * \begin{bmatrix} \gamma/2 & \gamma/2 \\ \gamma/2 & \gamma/2 \end{bmatrix}_{2x2} \begin{bmatrix} \omega d \\ \omega I \end{bmatrix}_{2x1} \quad (27)$$

Las ecuaciones 24 a 26 representan la cinemática del robot seguidor de línea específicamente la velocidad en el eje X , velocidad en el eje Y y la velocidad de orientación del mismo robot, Las cuales se han unificado en un mismo arreglo matricial para mostrar qué estás representan en conjunto el movimiento en el plano cartesiano del robot para diferentes velocidades angulares de sus motores. Esta representación además permite evidenciar la relación matemática no lineal entre las velocidades lineales de los ejes XY con respecto a la orientación Φ del robot.

4. ESTRATEGIAS DE CONTROL PARA ROBOT SEGUIDOR DE LÍNEA

4.1 Algoritmo PID clásico

El código está diseñado para un sistema embebido que controla un robot utilizando un controlador PID, sensores de reflectancia, motores y una turbina, además de incluir comunicaciones bluetooth.

Ilustración 24.Diagrama de flujo, código control PID clásico, Fuente: Elaboración propia

.

A continuación, se describen los componentes:

4.1.1 Librerías:

- **QTRSensors**: Controla los sensores de reflectancia (QTR8A) que se usan para el seguimiento de líneas.
- **Servo**: Maneja la turbina utilizando señales PWM para controlar su velocidad.
- **SoftwareSerial**: Permite comunicación serial en pines distintos al estándar, facilitando la conexión Bluetooth.
- **OrangutanMotors**: Controla los motores, ajustando sus velocidades y direcciones.

4.1.2 Variables

- **Tiempo de muestreo (10 ms)**: Es el intervalo entre cada ciclo de control, donde se actualizan las lecturas de los sensores y se ajusta el controlador PID.
- **Comunicación Bluetooth**: Utiliza una variable "char" para recibir datos seriales. Estos datos permiten ajustar los parámetros del controlador PID en tiempo real desde un dispositivo externo.
- **Función setup ():** La función "setup()" configura los sensores, la comunicación serial y la turbina del sistema.
- **Sensores QTR**: Se configura como sensores analógicos para leer la reflectancia y detectar líneas. Los sensores infrarrojos se controlan a través del pin 12. Se establece el tipo de sensor con "qtr.setTypeAnalog()", indicando que el sensor es analógico. Se definen los pines de los sensores mediante "qtr.setSensorPins()", asignando ocho pines analógicos (de A0 a A7) para leer los valores de reflectancia. El pin 12 se usa para controlar los emisores LED infrarrojos del sensor QTR con "qtr.setEmitterPin(12)".
- **Turbina**: Se configura el pin 9 para controlar la velocidad de la turbina mediante señales PWM.

4.1.3 Esfuerzo de Control:

La función “Esfuerzo Control” (float Control) ajusta la velocidad de los motores y la turbina en función del esfuerzo de control calculado por el controlador PID, el cual mide la desviación entre el valor deseado y el valor actual (o error) del sistema.

Este esfuerzo de control se convierte en señales PWM (modulación por ancho de pulso) para los motores y la turbina. Permitiendo ajustar su velocidad. El valor de la velocidad de la turbina (“Vel_Turb”), se calcula mediante la formula:

$$Vel\_Turb = offsetTurb + (150 - offsetTurb) * |Error| * k\_t \quad (23)$$

Donde:

- *“offsetTurb”*: es el valor base de la turbina.
- *“Error”,* es el valor absoluto del error actual entre la referencia y la salida.
- *“kt*”, es un factor de ajuste para la turbina.

El resultado, es un ajuste proporcional a la magnitud del error: cuanto mayor es el error, más rápido gira la turbina. Además, se utilizó “Turbina.attach(9)” para configurar el pin 9 como salida PWM para la turbina, y “Turbina.write (Vel_Turb)” para enviar el valor de velocidad calculado a la turbina, controlando su giro.

4.1.4 Sintonía por Bluetooth:

La función “Sintonia_Bluetooth” (void) permite modificar parámetros del controlador PID a través de la comunicación Bluetooth. Los caracteres enviados desde el dispositivo externo se leen y concatenan para formar una cadena con los valores de los parámetros.
Paso a paso:

- Lectura del carácter enviado: “cpp caracter = mySerial.read(); datos = datos + carácter;
- mySerial.read() “lee un carácter desde el puerto serie Bluetooth”
- El carácter leído se concatena al final de la cadena “datos”, que almacena la secuencia completa de caracteres que representan los valores de los parámetros.

Este código permite un control dinámico y ajustable del robot, asegurando que el comportamiento del sistema pueda modificarse en tiempo real. La comunicación Bluetooth es clave para hacer ajustes sin detener el robot, lo que optimiza su rendimiento en diferentes entornos.

4.2 Algoritmo PID Fuzzy

En este código se encuentran algunas variaciones puntales en ciertas líneas de código en comparación con el PID clásico. Por tanto, se implementa la lógica de un controlador difuso PID utilizando funciones de membresía simples para definir reglas difusas y determinar la salida de control. Esto incluye el cálculo de la diferencia entre la referencia y la salida actual del sistema, así como el cambio de error entre el ciclo actual y el anterior. También almacena la salida calculada por el controlador difuso. El cuál es la salida final del controlador después de aplicar reglas y escalarla. Estas serían las ganancias que controlan la intensidad de la acción proporcional y derivativa (Kp_fuzzy) y (Kd_fuzzy)

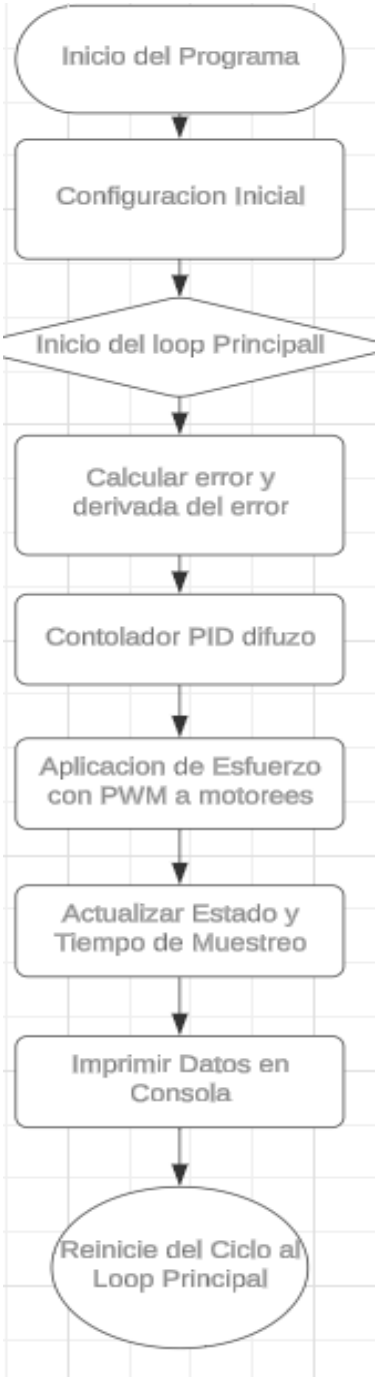


Ilustración 25.Diagrama de flujo, código control PID Fuzzy, Fuente: Elaboración propia

4.3 Validación del robot seguidor de línea

Para la validación del robot seguidor de línea, se tuvo en cuenta la siguiente metodología:

1. **Preparación de la Pista**: Organizar y limpiar la pista para asegurar condiciones óptimas de prueba.
2. **Configuración del Robot**:
    - Cargar el programa desde la computadora al robot seguidor de línea.
    - Limpiar las ruedas con cinta adhesiva para eliminar imperfecciones en las llantas.
3. **Calibración Inicial**: Encender el robot y moverlo sobre la cinta negra de la pista durante 5 a 8 segundos para calibrar la orientación y funcionamiento.
4. **Conexión Bluetooth**:
    - Establecer la intercomunicación por Bluetooth entre el robot y el teléfono móvil.
    - Configurar los parámetros de velocidad en el algoritmo.
5. **Iniciar la Prueba Cronometrada**:
    - Encender el cronómetro para medir el tiempo por vuelta del robot en la pista.
    - Realizar 10 pruebas, cada una de cinco vueltas.
6. **Mantenimiento entre Pruebas**:
    - Medir el voltaje de la batería del robot con el fin de garantizar condiciones similares entre cada prueba, limpiar la pista y ruedas antes de cada prueba para garantizar un buen agarre y desempeño.
7. **Pruebas en Pistas Diferentes**:
    - Pruebas en una pista fondo blanco línea negra con dimensiones de 120 x 180 cm, la cual se puede apreciar en la figura 32.
    - Validación adicional en una pista competitiva de 240 x 180 cm, la cual se presenta en la figura 31.
8. **Cálculo del Promedio**: Al finalizar todas las pruebas, se obtuvo un promedio de los tiempos con el fin de evaluar el rendimiento y el equilibrio del seguidor de línea.

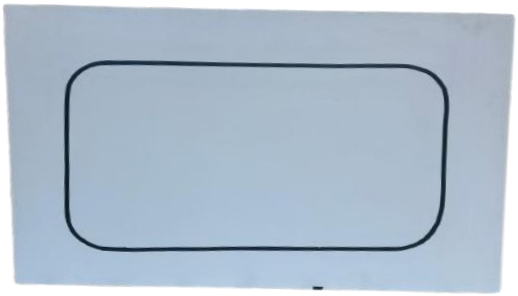
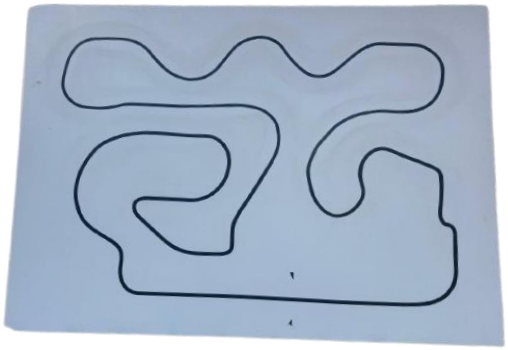


Ilustración 26. A Pista pequeña Ilustración



27. B Pista grande, Elaboración: Propia

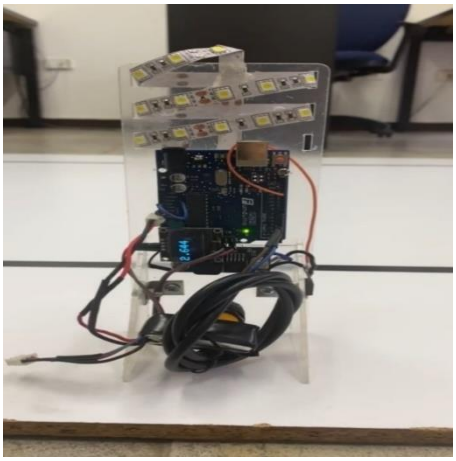


Ilustración 28. C Temporizador digital, Fuente: Propia

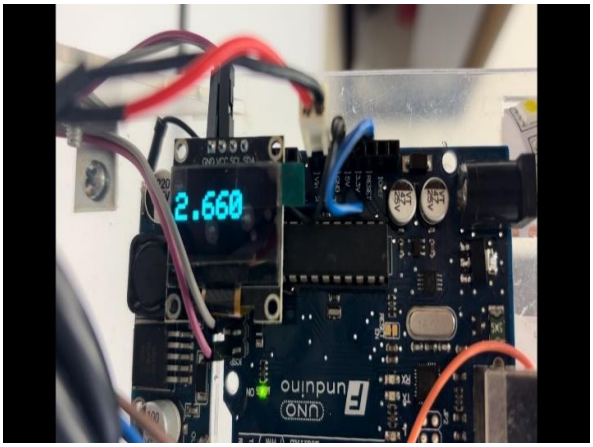


Ilustración 29. D Temporizador digital, Fuente: Propia

Este temporizador digital permite la toma de los tiempos con exactitud a cada una de las competencias realizadas en nuestros dos circuitos, dándonos unos tiempos precisos con un margen de error muy mínimo para así mismo nuestros resultados fueran acertados.

4.3 Validación controlador PID clásico

Tabla 1. Prueba seguidor de línea pista pequeña

PRUEBA DE SEGUIDOR DE LINEA PISTA PEQUEÑA / SEGUNDOS										
Vueltas	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Prueba 4**	**Prueba 5**	**Prueba 6**	**Prueba 7**	**Prueba 8**	**Prueba 9**	**Prueba 10**
1	2,697	2,614	2,597	2,668	2,622	2,625	2,585	2,696	2,641	2,662
2	2,731	2,623	2,678	2,698	2,683	2,693	2,673	2,661	2,714	2,644
3	2,761	2,739	2,668	2,626	2,683	2,625	2,632	2,674	2,717	2,742
4	2,82	2,652	2,689	2,674	2,629	2,671	2,681	2,726	2,701	2,688
5	2,742	2,691	2,625	2,641	2,628	2,631	2,666	2,698	2,695	2,686
Tiempo promedio	**2,6844**	**2,7502**	**2,6638**	**2,6514**	**2,6614**	**2,649**	**2,649**	**2,6474**	**2,691**	**2,6936**

En la tabla anterior se puede observar los resultados obtenidos de las pruebas de validación del robot seguidor de línea en la pista pequeña, del cual se obtuvo un tiempo promedio de 2,674 Segundos de las 10 pruebas realizadas.

Tabla 2.Prueba de seguidor de línea pista grande

PRUEBA DE SEGUIDOR DE LINEA PISTA GRANDE/ SEGUNDOS										
Vueltas	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Prueba 4**	**Prueba 5**	**Prueba 6**	**Prueba 7**	**Prueba 8**	**Prueba 9**	**Prueba 10**
1	8,231	8,338	8,414	8,343	8,417	8,925	8,474	8,402	8,371	8,422
2	8,302	8,431	8,687	8,443	8,541	9,061	8,719	8,447	8,431	8,438
3	8,371	8,466	8,778	8,441	8,588	9,019	8,748	8,588	8,449	8,458
4	8,397	8,481	8,844	8,466	8,688	8,956	8,783	8,483	8,489	8,404
5	8,371	8,463	8,994	8,496	8,756	9,024	8,786	8,538	8,477	8,434
Tiempo Promedio	**8,3344**	**8,4358**	**8,7434**	**8,4378**	**8,598**	**8,997**	**8,702**	**8,4916**	**8,4434**	**8,4312**

En la tabla anterior se puede observar los resultados obtenidos de las pruebas de validación

del robot seguidor de línea en la pista grande, del cual se obtuvo un tiempo promedio de 8,561 Segundos de las 10 pruebas realizadas.

4.4 Validación controladora Fuzzy

Tabla 3.Prueba seguidor de línea en pista pequeña

PRUEBA DE SEGUIDOR DE LINEA PISTA PEQUENA / SEGUNDOS										
Vueltas	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Prueba 4**	**Prueba 5**	**Prueba 6**	**Prueba 7**	**Prueba 8**	**Prueba 9**	**Prueba 10**
1	2.546	2.640	2.721	2.661	2.691	2.674	2.674	2.737	2.784	2.745
2	2.647	2.686	2.734	2.746	2.722	2.720	2.711	2.758	2.737	2.760
3	2.684	2.677	2.686	2.714	2.722	2.731	2.716	2.718	2.721	2.749
4	2.670	2.706	2.701	2.725	2.811	2.738	2.791	2.761	2.771	2.759
5	2.716	2.677	2.681	2.699	2.731	2.697	2.792	2.755	2.726	2.755
Tiempo Promedio	**2.653**	**2.677**	**2.705**	**2.709**	**2.735**	**2.712**	**2.737**	**2.746**	**2.748**	**2.754**

En la tabla anterior se puede observar los resultados obtenidos de las pruebas de validación del robot seguidor de línea en la pista pequeña con controlador Fuzzy, del cual se obtuvo un tiempo promedio de 2.717 Segundos de las 10 pruebas realizadas.

Tabla 4.Prueba Seguidor de línea pista grande

PRUEBA DE SEGUIDOR DE LINEA PISTA GRANDE / SEGUNDOS										
Vueltas	**Prueba 1**	**Prueba 2**	**Prueba 3**	**Prueba 4**	**Prueba 5**	**Prueba 6**	**Prueba 7**	**Prueba 8**	**Prueba 9**	**Prueba 10**
1	8.142	8.165	8.122	8.189	8.330	8.494	8.361	8.539	8.070	8.049
2	8.097	8.229	8.255	8.243	8.295	8.429	8.358	8.450	8.107	8.057
3	8.148	8.226	8.270	8.240	8.315	8.382	8.401	8.373	8.089	8.105
4	8.155	8.203	8.281	8.220	8.331	8.453	8.439	8.492	8.060	8.136
5	8.209	8.232	8.334	8.221	8.393	8.459	8.532	8.421	8.070	8.183
Tiempo promedio	**8.150**	**8.211**	**8.252**	**8.223**	**8.333**	**8.443**	**8.418**	**8.455**	**8.079**	**8.106**

En la tabla anterior se puede observar los resultados obtenidos de las pruebas de validación del robot seguidor de línea en la pista grande con controlador Fuzzy, del cual se obtuvo un tiempo promedio de 8.267 Segundos de las 10 pruebas realizadas

4.5 Análisis de validación de resultados

4.5.1 **Resultados**

El error Cuadrático medio es una métrica de evaluación que mide el promedio de los errores al cuadrado entre valores predichos o estimados y valores reales. Se utiliza para evaluar la precisión de un modelo en varios campos, como en este caso ver con exactitud el error que presenta el seguidor de línea en cada una de sus vueltas.

El error cuadrático se calcula utilizando la siguiente fórmula:

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(Refi - Youti)\,2 \qquad (24)$$

Donde:

- n: Número de datos
- $Refi$: Referencia o línea = 0
- $Youti$: Lectura del sensor

A continuación, se presenta el análisis de los resultados obtenidos de la validación del seguidor de línea en las dos pistas de prueba:

Tabla 5. Error cuadrático medio, seguidor de línea pista grande

ERROR CUADRÁTICO MEDIO PID CLASICO / SEGUNDOS									
Prueba 1	**Prueba 2**	**Prueba 3**	**Prueba 4**	**Prueba 5**	**Prueba 6**	**Prueba 7**	**Prueba 8**	**Prueba 9**	**Prueba 10**
0.16	0.17	0.15	0.17	0.17	0.16	0.16	0.17	0.16	0.17

En esta tabla se puede ver el resultado el error cuadrático medio de la prueba con el

controlador PID CLÁSICO donde el rango está entre el 0,15% y un 0,17%.

Tabla 6.Error cuadrático medio Fuzzy, seguidor de línea pista grande

ERROR CUADRÁTICO MEDIO FUZZY / SEGUNDOS									
Prueba 1	**Prueba 2**	**Prueba 3**	**Prueba 4**	**Prueba 5**	**Prueba 6**	**Prueba 7**	**Prueba 8**	**Prueba 9**	**Prueba 10**
0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.14

En esta tabla se puede ver el resultado el error cuadrático medio de la prueba con el controlador FUZZY donde el rango es de 0,14%.

```
close all;
clear all;
clc;
% code comparacion de tiempos pista grande
s1 = [8.3344 8.4358 8.7434 8.4378 8.598 8.997 8.702 8.4916 8.4434 8.4312];
s2 = [8.150 8.211 8.252 8.223 8.333 8.443 8.418 8.455 8.079 8.106];
boxplot([s1' s2']);
title('Prueba de Tiempo en Pista Grande',FontSize=16);
ylabel('Time [Seg]',FontSize=16);
xlabel('PID   vs   Fuzzy',FontSize=16);
```

Ilustración 30.ECM, Fuente: Elaboración propia

Estas líneas de código generan un gráfico de caja de Bigotes para hacer la comparación de dos conjuntos de datos relacionados con tiempos de ejecución de un seguidor de línea en una pista grande.

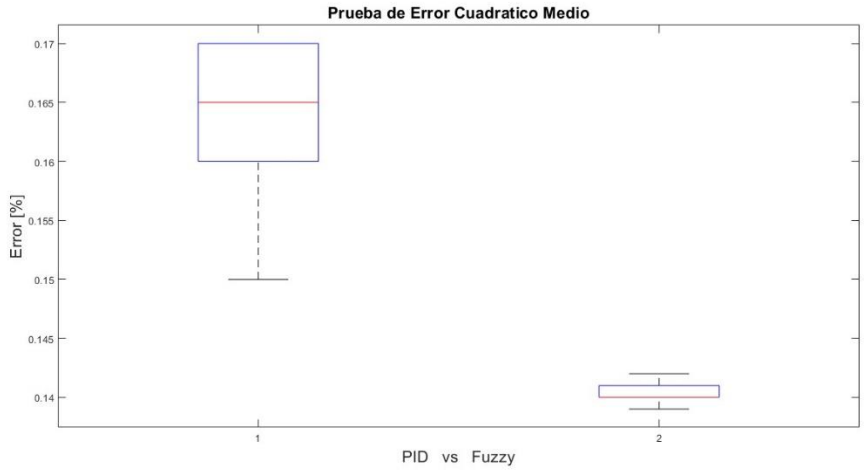


Ilustración 31.Caja de Bigotes, Error cuadrático medio, Fuente: Elaboración propia

Como se puede observar en esta caja de bigotes el método Fuzzy es superior al PID Clásico en términos de precisión (menor error cuadrático medio) y estabilidad (menor variabilidad).

```
close all;
clear all;
clc;
% code comparacion de tiempos pista pequeña
s1 = [2.6844 2.7502 2.6638 2.6514 2.6614 2.649 2.649 2.6474 2.691 2.6936];
s2 = [2.653 2.677 2.705 2.709 2.735 2.712 2.737 2.746 2.748 2.754];
boxplot([s1' s2']);
title('Prueba de Tiempo en Pista Pequeña',FontSize=16);
ylabel('Time [Seg]',FontSize=16);
xlabel('PID   vs   Fuzzy',FontSize=16);
```

Ilustración 32.Tiempos pista pequeña, Fuente: Elaboración propia

Este código permite tiempos de respuesta de dos métodos implementados PID Clásico y Fuzzy y comparar los tiempos de ambos métodos en una pista pequeña. Esto ayuda a evaluar cuál tiene mejor rendimiento en términos de rapidez y estabilidad.

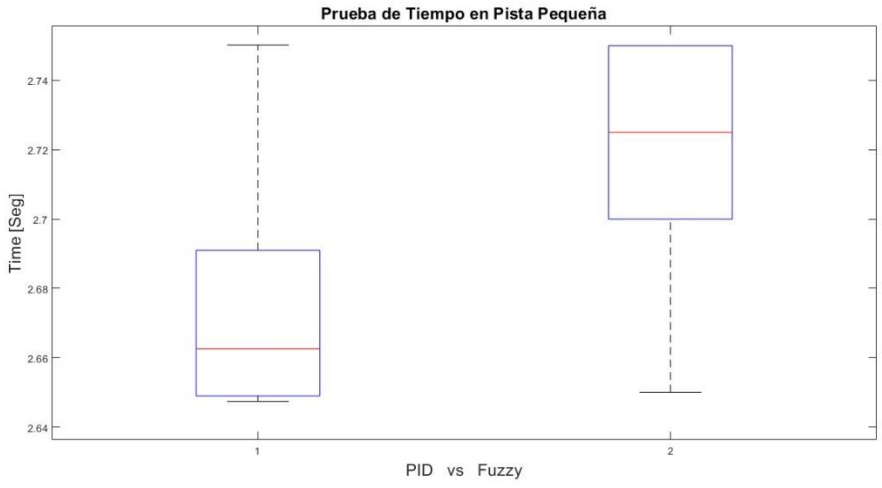


Ilustración 33.Prueba de tiempo pista pequeña, Fuente: Elaboración propia

Se puede observar que el controlador PID Clásico nos da una mejor opción para esta tarea específica, ya que combina rapidez y consistencia en pista pequeña. Mientras que el controlador Fuzzy se muestra menos adecuado para tareas en esta pista, dado que es más

lento y muestra mayor variabilidad en su desempeño.

```
close all;
clear all;
clc;
% code comparacion del error cuadratico  medio
s1=[0.16 0.17 0.15 0.17 0.17 0.16 0.16 0.17 0.16 0.17];
s2=[0.142 0.141 0.139 0.141 0.14 0.142 0.14 0.14 0.14 0.14];
boxplot([s1' s2']);
title('Prueba de Error Cuadratico Medio',FontSize=16);
ylabel('Error [%]',FontSize=16);
xlabel('PID   vs   Fuzzy',FontSize=16);
```

Ilustración 34Tiempo pista grande, Fuente: Elaboración propia

Este código permite hacer una comparación gráfica del error cuadrático medio entre dos métodos de control, en este caso el controlador PID Clásico y el controlador Fuzzy.

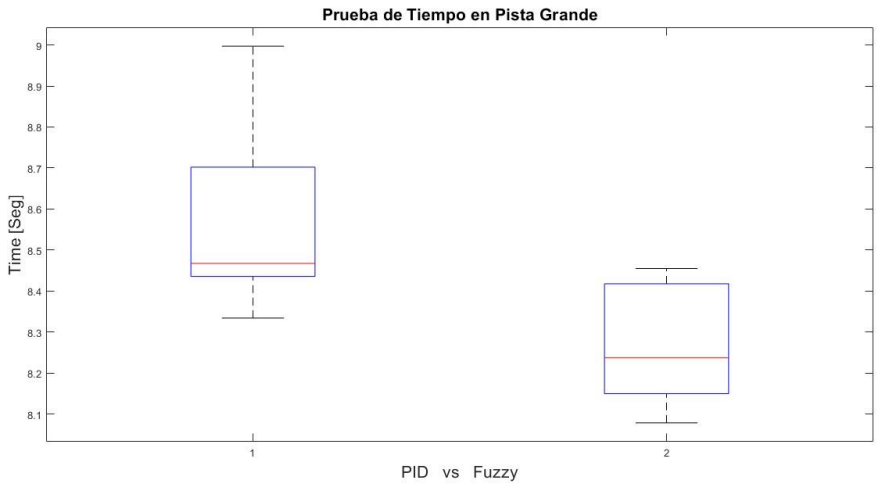


Ilustración 35.Prueba de tiempo pista grande, Fuente: Elaboración propia se puede evidenciar que en este caso el controlador fuzzy es más efectivo en cuestión de rapidez comparándolo con el controlador PID Clásico con una diferencia de 0,3 segundos.

5. CONCLUSIONES

El controlador PID clásico muestra un desempeño consistente en la pista pequeña, con un promedio por vuelta alrededor a 2.6 segundos. Los resultados se mantienen dentro de un rango estrecho, lo que indica que el controlador se adapta adecuadamente al entorno y mantiene el robot sobre la línea de la pista. En cuanto a la pista grande, los tiempos promedio son más altos, rodando los 8.3 segundos, lo cual es esperado debido a que la pista presenta mayor tamaño y complejidad debido a sus curvas. Sin embargo, los tiempos de las pruebas son relativamente consistentes, lo que indica que el controlador mantiene un rendimiento aceptable a pesar de las dificultades adicionales.

Entre las diferencias observadas en las pruebas realizadas en ambas pistas, se destaca que el controlador PID clásico muestra un rendimiento óptimo en circuitos más simples. Esto se evidencia en los tiempos más bajos y mayor agarre en las curvas.

En las pruebas realizadas en la pista grande**,** el controlador difuso, muestra un mejor rendimiento con tiempos cercanos a 8.2 segundos. Esto sugiere que el controlador difuso tiene una mayor capacidad para manejar la variabilidad y complejidad de las curvas, lo que permite un mejor seguimiento de la línea.

El controlador difuso presenta un desempeño más estable en ambas pistas, tanto en la pequeña como en la grande. Las variaciones entre las vueltas son significativamente menores en comparación con el controlador PID clásico, lo que indica que el controlador difuso es menos sensible a pequeños errores de medición o perturbaciones externas.

Según el análisis estadístico descriptivo del diagrama de caja y bigotes, el controlador PID clásico demuestra un mejor desempeño en la pista pequeña en comparación con el controlador PID Fuzzy. Esto se debe a que el PID clásico presenta una mediana más baja, lo que indica mejores tiempos promedio. Por tanto, se puede concluir que este controlador es más viable para este tipo de pistas. Por otro lado, en la pista grande, el controlador PID Fuzzy

destaca por su mayor fiabilidad, y sus tiempos son más bajos al tener una mediana menor. En contraste, el PID clásico muestra una mayor variabilidad en tiempos y menor estabilidad en este escenario, lo que podría resultar problemático en condiciones complejas.

El cálculo del Error Cuadrático Medio (ECM) en las pruebas realizadas demostró ser una herramienta clave para evaluar el desempeño de los controladores PID clásico y difuso en diferentes escenarios, también para cuantificar la efectividad de cada controlador, ante condiciones específicas en las que cada uno sobresale. Por ejemplo, el PID clásico en la pista pequeña, muestra que el ECM fue bajo, reflejando una capacidad del controlador para mantener el robot cerca de la trayectoria deseada. Sin embargo, en pistas más complejas, el ECM mostró un aumento notable debido a la dificultad del controlador clásico para adaptarse especialmente en curvas cerradas. El controlador basado en lógica difusa obtuvo un ECM significativamente menor en pistas complejas, evidenciando su capacidad superior para manejar dinámicas de sobre esfuerzo y trayectorias con alta variabilidad.

La capacidad de ajustar los parámetros PID en tiempo real mediante comunicación Bluetooth resulta altamente beneficiosa, ya que permite sintonizar el controlador y por ende el desempeño del robot en diversas condiciones sin necesidad de detener su funcionamiento.

Este estudio proporciona evidencia comparativa entre dos enfoques de control ampliamente utilizados, destacando las condiciones en las que cada uno puede ser más efectivo. Los resultados obtenidos sientan una base sólida para futuras investigaciones en robótica móvil y control automático.

6. RECOMENDACIONES Y TRABAJOS FUTUROS

Redactar y someter un artículo científico que detalle los hallazgos de esta investigación, enfatizando las fortalezas y limitaciones de los controladores PID clásico y PID difuso en diferentes escenarios. Esto contribuirá al desarrollo de nuevas estrategias en sistemas de control para robots móviles.

Utilizar esta misma plataforma robótica como prototipo experimental de pruebas para implementar otras estrategias de control en seguidores de línea, por ejemplo: control por realimentación de estados, control optimo LQR o control robusto.

Utilizar esta misma plataforma robótica como prototipo experimental de pruebas para implementar otros esquemas de control en seguidores de línea, por ejemplo: control feedforward, control en cascada o control por sobre posición.

Investigar y desarrollar mejoras en el diseño de las ruedas, utilizando materiales más ligeros y dimensiones optimizadas para incrementar el agarre y la estabilidad del robot, especialmente a velocidades más altas.

Ampliar la funcionalidad de comunicación Bluetooth, no solo para Android, si no incluir sistemas operativos iOS u otros dispositivos móviles, mejorando la accesibilidad y versatilidad del sistema.

Realizar pruebas en escenarios híbridos, combinando pistas con diferentes niveles de complejidad y condiciones ambientales, por ejemplo, altibajos o curvas de 90°, para evaluar la robustez de ambos controladores en entornos más complejos.

7. Referencias

[1] J. J. Craig, Introduction to Robotics, Mechanics and Control.Pearson, 2005.

[2] B. M. A. &. O. A. Kluge, Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I. IEEE Potentials,, 2002, pp. 17-20.

[3] B. &. K. O. Siciliano, Springer Handbook of Robotics, Springer, 2008.

[4] J. o. E. Engineering, Electronics, Control and Computer Science, 2022, pp. 49-60.

[5] A. K. Alexia Toumpa, Control de un robot seguidor de línea basado en estimación FSM, vol. 51, 2018, pp. 542-547.

[6] A. D. Asham, Análisis matemático de un robot seguidor de línea, diseño de un controlador estable utilizando el enfoque de Lyapunov y pruebas experimentales., vol. 06, 2022, p. 385–395.

[7] M. D. L. Á. C. IBÁÑEZ, "ANÁLISIS COMPARATIVO DE ALGORITMOS DE CONTROL PID, FUZZY Y PREDICTIVO APLICADOS A SISTEMAS SEGUIDORES DE LA POSICIÓN DEL SOL PARA LA CAPTACIÓN DE ENERGÍA SOLAR USANDO PANELES FOTOVOLTAICOS, GUAYAQUIL – ECUADOR, 2018.

[8] A. Ferreyra, ESTUDIO COMPARATIVO ENTRE CONTROL PID Y DIFUSO, 1999.

[9] J. A. Soto, Sistema de control y arquitectura de un robot seguidor de línea., Universidad Tecnológica de Chihuahua, (2016)..

[10] M. C. Romero, Sistema de control y arquitectura de un robot seguidor de línea, Universidad Tecnológica de Chihuahua, 2016.

[11] A. Rosales, Navegacion de robots moviles en entornos no estructurados, 2009.

[12] i. Bambino, Una Introducción a los Robots Móviles, 2008.

[13] J. G. G. M. Ph.D, Una Revisión Sobre la Evolución de la Robótica, Universidad Santo Tomás, 2021.

[14] N. Q. L. Y. C. &. Z. S. Zijie, «Fuzzy control strategy for course correction of omnidirectional mobile robot. International Journal of Control, Automation and Systems,2019

[15] T. e. I. Agencia de Noticias de Ciencia, Impacto de la robótica en la sociedad, 2022.

[16] A. G.-V. Muñoz, Robótica para desarrollar el pensamiento, Universidad España, 2018.

[17] Educrear, Cuál es la importancia de incorporar habilidades STEM, 2021.

[18] Maruja, Control Automático, 2020.

[19] UPV, Introducción al Control Automático,2016..

[20] Dr.JorgeJuanGilNobajas, CONTROLAUTOM´ ATICO DESISTEMASCONTINUOSYMUESTREADOS, 2010.

[21] R. Vilanova, Control PID robusto: Una visión panorámica, 2011.

[22] J. S. Han, Fuzzy gain scheduling PID control of a hybrid robot based on dynamic characteristics., 2023.

[23] M. A. Johnson, Pid Control, Springer, 2005.

[24] P. Pueyo, Conceptos basicos de la logica difusa, 2001.

[25] X. C. K. Z. Y. J. J. &. J. P. Jin, Simulation of hydraulic transplanting robot control system based on fuzzy PID controller., 2020.

[26] G. Z. X. Y. C. Y. S. L. B. &. J. C. Cao, «Fuzzy adaptive PID control method for multi-mecanum-wheeled mobile robot. Journal of Mechanical Science and Technology,2022.

[27] D. y. B. S. Bairon Adolfo Astaiza Sanchez, *Sistema de ayuda para toma de decisiones basado en lógica difusa aplicado a la gestión de planeación estratégica de la empresa caso de estudio EMTEL S.A. E.S.P.,* Popayán, 2020, p. 6.

[28] S. M. Hurtado, Estado de la cuestión acerca del uso de la lógica difusa en, 2006, p. 32.

[29] R. S. Ortigoza, «UNA PANORÁMICA DE LOS ROBOTS MÓVILES,» *Revista Electrónica de,* p. 2,2007.

[30] A. S. SILVERA, ROBOT MÓVIL CON APLICACIONES METODOLOGICAS PARA EL ESTUDIO DE LA ROBÓTICA MEDIANTE LEGO NXT, 2018

[31] D. O. Martínez, «ROBÓTICA PARA SEGUIMIENTO DE LÍNEAS,» 2016.

[32] Y. D. TAPIERO, «DISEÑO E IMPLEMENTACIÓN DE UN ROBOT SEGUIDOR DE LÍNEA DE COMPETENCIA PARA LA CATEGORÍA VELOCISTA,» 2019.

[33] J. F. Gutiérrez, «Desempeño de los controladores clásicos y difuso proporcional para el seguimiento,» 2022.

[34] H. GONZÁLEZACEVEDO, «ESTUDIO COMPARATIVO DE TRES TÉCNICAS DE NAVEGACIÓN PARA ROBOTS MOVILES,» 2007.

[35] X. Aguas, «Controlador PD-Difuso para seguimiento de pared de un robot móvil: validación experimental,» *Revista Digital Novasinergia,* 2019.

[36] S. J. y S. F, «Diseño de un Controlador Fuzzy para Guiado de un Robot Móvil,» 2014.

[37] E. Díaz, «CONTROL HIBRIDO PID-DIFUSO EN ROBOT SEGUIDOR DE LINEA NO HOLONOMICO,» 2018.

[38] E. Muñoz, «Evaluación del desempeño de diversos controladores avanzados aplicados a un robot manipulador tipo scara, 2011.

[39] J. I. C. Morenas, Diseño e implementación de un sistema de control de robots mediante la ingeniería del software basada en componentes, España, 2017.

[40] A. Alzate, Control difuso de una plataforma móvil para el seguimiento de trayectorias, 2007.

[41] O. F, Simulación Cinemática de un Robot Seguidor de Línea, 2017.

[42] J. T. González, "*Todo sobre robots móviles tipo velocistas",* [Diapositivas de PowerPoint]., 2023.

[43] J. Gutierrez, Desempeño de los controladores clásicos y difuso proporcional para el seguimiento de pared de un robot móvil, 2022.

[44] J. L. Martín, Analizando el desarrollo de las habilidades STEM a través de un proyecto ABP, Universidad de Madrid,2016

[45] R. J. LEÓN, GESTIÓN DE TENDENCIAS STEM EN EDUCACION SUPERIOR, 2021.

[46] M. C. Romero, Sistema de control y arquitectura de un robot seguidor de línea, 2017.

[47] Cistatac, QUÉ SON METODOLOGÍAS INTERACTIVAS, 2020.

[48] M. C. Romero, Sistema de control y arquitectura de un robot seguidor de línea, 2016.

[49] Unit, Cómo hacer tu Mini Robot Seguidor de líneas, 2020.

[50] mintforpeople, Controlador PID, 2022.

[51] S. Talisis, Habilidades STEM: ¿qué son y por qué están causando tanto revuelo?, 2022.

[52] J. A. C. S. G. A. A. G. Mariano Carrillo Romero, Sistema de control y arquitectura de un robot seguidor de linea, Universidad Tecnologica de Chihuahua,2017

[53] D. Ege Technical and Business Collage, Path Planning of line follower robot, IEEE Proceedings of the 5th European DSP education and research conference, 2012.

[54] S. I. F Kaiser, Line Follower Robot: Fabrication and accuracy, International Conference on Electrical Engineering and Information & Communication Techonology, 2014.

[55] J. A. C. Soto, Sistema de control y arquitectura de un robot seguidor de línea, Universidad Tecnológica de Chihuahua, 2016, p. 2.

[56] J. A. C. Soto, Sistema de control y arquitectura de un robot seguidor de línea, Universidad Tecnológica de Chihuahua, 2016, p. 2.

[57] J. o. Engineering, EngineElectronics, Control and Computer Science,2024

[58] Educrear, Cuál es la importancia de incorporar habilidades STEM, 2021.

[59] W. W. Royce, Managing the Development of Large Software Systems., 1970.

[60] A. N. K. Nasir, Comparación de rendimiento entre el controlador lógico difuso (FLC) y el controlador PID para un robot de equilibrio de dos ruedas altamente no lineal, 2011.

[61] Zambeca, Sensor de reflectancia QTR-8Aª,2024

8. ANEXOS

8.1 Código PID clásico

```
//Incluir Librerias
#include <QTRSensors16.h>
#include <BluetoothSerial.h>
//#include <ESP32Servo.h>

// Bluetooth configuracion
#if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
#error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
#endif

BluetoothSerial SerialBT;

//Modulo de Inicio
const byte MInit=34;
int Estado;

//TURBINA
//Creación del Objeto
//Servo myTurbina;

//PIN PARA EL CONTROL DE TURBINA
//const byte Tur=D6;

//Variables para sensores
#define NUM_SENSORS             16   // Numero de sensores usados
#define NUM_SAMPLES_PER_SENSOR  3    // Numero de muestras
#define IN_PIN                  25   // PIN de entrada analogico del multiplexor


// Inicialización del sensor, digitales D9,D10,D0,D1
QTRSensorsMux qtra((unsigned char[]) {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15},
NUM_SENSORS, NUM_SAMPLES_PER_SENSOR, (unsigned char) IN_PIN, (unsigned char[]){17,18,19,23} );
unsigned int sensorValues[NUM_SENSORS];

// Declaracion de Sub-Funciones
float  Lectura_Sensor(void);                                    // Función para lectura del modulo
QTR8A
float  Lectura_Referencia(void);                                // Función para lectura del valor de
referencia
float  Controlador(float, float);                               // Función para la ley de control
void   Esfuerzo_Control(float);                                 // Función para enviar el esfuerzo de
control en PWM
unsigned long int   Tiempo_Muestreo(unsigned long int);//,unsigned int);          // Función para
garantizar un tiempo de muestreo constante
void   Sintonia_Bluetooth(void);                                // Función de actualización de
parametros para la ley de control
void  Imprimir(void);
```

Ilustración 36Código Control PID Clásico Parte 1. Fuente: Elaboración propia

```
// Declaracion de variables del lazo cerrado
unsigned long int Tm = 10.0;                                    // Tiempo de muestreo en mili
segundos
float E_derivativo=0.0, E_integral=0.0;                         // Variables auxiliares usadas para
el calculo de accion integral y deravada
float Referencia=0.0, Control=0.0, Kp = 1.5, Ti = 0.0, Td = 0.05;  // Inicialización de constantes y
variables del controlador
float Salida=0.0, Error=0.0, Error_ant=0.0;                     // Inicializacion variables de
control
float offset = 1.0, Vmax = 0.0, s1 = 0.0, s2=0.0;                                  // Definición del
Offset y la velocidad Maxima
//int offsetTurb = 120, Vel_Turb=0;
unsigned long int Tinicio = 0;                                  // Inicialización tiempo de inicio


// Declaración de variables de la sintonización bluetooth
char caracter; String datos;                                    // Definicion variables para
almacenar los datos
int d1, d2, d3, d4;                                             // Indices para la separación de las
subcadenas
String S_Kp, S_Ti, S_Td, S_Vmax;                                // Definición variables string de
sintonizacion bluetooth,se ingresan desde el Celular

//CREACIÓN DE PWM
const uint16_t Frecuencia = 5000;
const byte Canales[] ={0,1};
const byte Resolucion = 10;

/*#define LEDC_TIMER_12_BIT  12
// use 5000 Hz as a LEDC base frequency
#define LEDC_BASE_FREQ     5000
// fade LED PIN (replace with LED_BUILTIN constant for built-in LED)
#define LED_PIN            5
*/
#define   PWM_D   26                                            // Definición Pin 6 PWM Motor
Derecho
#define   PWM_IZ  33
int  DirI = 32;
int  DirD = 27;
const int  Led  = 2;

void setup() {
  Serial.begin(9600);
  SerialBT.begin("Botnilla");
  Serial.println("El ESP32 está listo, puedes emparejarlo mediante Bluetooth");
  delay(1000);

  //   inicio calibracion
  pinMode(DirI,OUTPUT);
  pinMode(DirD,OUTPUT);
  pinMode(Led,OUTPUT);
  digitalWrite(Led,HIGH);
  Serial.println("-------------- Proceso de Calibracion de Los Sensores --------------" );
  Serial.println("Iniciando ......");
  //Calibración Inicial de Pines Sensor
  for (int i = 0; i < 400; i++){  // make the calibration take about 10 seconds
    qtra.calibrate();       // reads all sensors 10 times at 2.5 ms per six sensors (i.e. ~25 ms per
call)
  }
  digitalWrite(Led,LOW);
  delay(3000);
```

Ilustración 37. Código Control PID Clásico parte 2, Fuente: Elaboración propia

```
 //Inicializacion_turbina();     //Inicialización de la turbina

  ledcAttach(PWM_D, Frecuencia, Resolucion);
  ledcAttach(PWM_IZ, Frecuencia, Resolucion);
  /*
  ledcSetup(Canales[0],Frecuencia,Resolucion);
  ledcSetup(Canales[1],Frecuencia,Resolucion);
  ledcAttachPin(PWM_D,Canales[0]);
  ledcAttachPin(PWM_IZ,Canales[1]);
*/
}

void loop() {
  Estado=digitalRead(MInit);
  Estado=1;

  while(Estado){
    Tinicio    = millis();                                  // toma el valor en milisengundos
    if(SerialBT.available()) { Sintonia_Bluetooth();}
    Estado     = digitalRead(MInit);
    Tinicio    = millis();                                  // toma el valor en milisengundos
    Salida     = Lectura_Sensor();                          // funcion de lectura de la variable
salida del  proceso
    Control    = Controlador(Referencia,Salida);            // funcion de la ley de control
    Esfuerzo_Control(Control);                              // funcion encargada de enviar el
esfuerzo de control
    Tm = Tiempo_Muestreo(Tinicio);
    //Imprimir();
    //myTurbina.write(50);
    }
}

//Para leer el sensor
float Lectura_Sensor(void) {
  Salida = (qtra.readLine(sensorValues)/7500.0) - 1.0;
  return Salida;                                    // retorno la variable de salidad del
proceso normalizada entre 0-1, al olgoritmo de control
}

//Controlador para Motoresu
float Controlador(float Referencia, float Salida) {                         // Funcion para la ley de
control
  float E_derivativo;
  float E_integral;
  float Control;

  Error          = Referencia - Salida;
  //if(abs(Error)< 0.15){  Error = 0.0;}
  E_integral     = E_integral + (((Error*(Tm/1000.0)) + ((Tm/1000.0)*(Error - Error_ant)))/2.0);
  E_integral     = ( E_integral > 100.0) ? 100.0 :  (E_integral < -100.0 ) ? -100 : E_integral;
  E_derivativo   = (Error - Error_ant)/(Tm/1000.0);
  Control        = Kp*Error + Ti*E_integral + Td*E_derivativo ;
  Error_ant      = Error;

  //constrain(Control, 0.0, 1.0);
  if(Control > 1.5){  Control =  1.5;}                              // Limita el esfuerzo de control a un
máximo de 1.5
  if(Control < -1.5){ Control = -1.5;}                              // Limita el esfuerzo de control a un
mínimo de -1.5

  return Control;
}
void Esfuerzo_Control(float Control) {                              //envia el esfuerzo de control en forma
de PWM
  s1  = (offset - Control);
  s2  = (offset + Control);

  if( s1 <= 0.0 ){// Motor Derecho
    digitalWrite(DirD,HIGH);}
  else{digitalWrite(DirD,LOW);}

  if( s2 <= 0.0 ){ //Motor Izquierdo
    digitalWrite(DirI,HIGH);}
  else{digitalWrite(DirI,LOW);}

  ledcWrite(PWM_D,constrain(abs(s1), 0.0, 1.0)* Vmax);
  ledcWrite(PWM_IZ,constrain(abs(s2), 0.0, 1.0)* Vmax);
```

Ilustración 38. Código Control PID Clásico Parte 3, Fuente: Elaboración propia

```
unsigned long int Tiempo_Muestreo(unsigned long int Tinicio){//, unsigned int Tm){ // Funcion que asegura
que el tiempo de muestreo sea el mismo siempre
  unsigned long int T =millis()-Tinicio;
  return  T;
  }

void Imprimir (){
  Serial.print("Salida= ");
  Serial.print(Salida);
  Serial.print("   Error= ");
  Serial.print(Error);
  Serial.print("   Control= ");
  Serial.print(Control);
  Serial.print("   Tm= ");
  Serial.print(Tm);
  Serial.print("   pwmI= ");
  Serial.print(abs(s1));
  Serial.print("   pwmD= ");
  Serial.println(abs(s2));
  }

void Sintonia_Bluetooth(void){                                    // Recibe porbluetooth KP,Ki,KD y Vmax
del controlador.
  caracter = SerialBT.read();                                       // Guarda un caracter de los datos
seriales
  datos = datos + caracter;                                       // Almacena caracter a caracter en el
string datos
  if (caracter == '*') {                                          // cada caracter llega separado por una
coma, y toda la cadena finaliza con un *
    d1    = datos.indexOf(',');                                   // Guarda la posición del primer
caracter ',' en el string datos
    S_Kp  = datos.substring(0, d1);                               // Almacena la subcadena que contiene
constante Kp
    d2    = datos.indexOf(',', d1+1);                             // Guarda la posición del segundo
caracter ',' en el string datos
    S_Ti  = datos.substring(d1+1, d2);                            // Almacena la subcadena que contiene
la constante Ti
    d3    = datos.indexOf(',', d2+1);                             // Guarda la posición del tercer
caracter ',' en el string datos
    S_Td  = datos.substring(d2+1, d3);                            // Almacena la subcadena que contiene
la constante Td
    d4    = datos.indexOf(',', d3+1);                             // Guarda la posición del cuarto
caracter ',' en el string datos
    S_Vmax  = datos.substring(d3+1, d4);                          // Almacena la subcadena que contiene
la constante de la velocidad máxima

    datos = "";                                                   // Limpia el string datos para una
nueva recepcion de datos
    Kp = S_Kp.toFloat();  Ti = S_Ti.toFloat();  Td = S_Td.toFloat();  // Actualiza las constantes del
controlador PID, formato flotante
    Vmax = S_Vmax.toFloat();                                      // Actualiza la velocidad máxima, a
flotante
  }
}
```

Ilustración 39.Codigo Control PID Clásico Parte 4, Fuente: Elaboración propia

8.2 Código control fuzzy

```
/*#define LEDC_TIMER_12_BIT  12
// use 5000 Hz as a LEDC base frequency
#define LEDC_BASE_FREQ     5000
// fade LED PIN (replace with LED_BUILTIN constant for built-in LED)
#define LED_PIN            5
*/
#define   PWM_D   26                                          // Definición Pin 6 PWM Motor Derecho
#define   PWM_IZ  33
int  DirI = 32;
int  DirD = 27;
const int  Led  = 2;

// VARIABLES CONTROLADOR DIFUSO
float Error_fuzzy, Error_ant_fuzzy, Cambio_Error_fuzzy, Control_fuzzy;
float Defuzzificado;
// Definir variables para las ganancias del controlador PD difuso
float Kp_fuzzy = 1.0;  // Ganancia proporcional ajustable
float Kd_fuzzy = 0.5;  // Ganancia derivativa ajustable

// Funciones de membresía difusas ajustadas para P y D
float membresiaPositiva(float x) {
  return (x > 0) ? (1 - x) : 0;
}

float membresiaNegativa(float x) {
  return (x < 0) ? (1 + x) : 0;
}

float membresiaCero(float x) {
  return (abs(x) <= 0.1) ? (1 - abs(x)) : 0;
}
// aqui termina la modificacion difusa el resto de funciones es identico
// a excepcion de la rutina abajo llamada controlador() que tambien se modificó

void setup() {
  Serial.begin(9600);
  SerialBT.begin("Botnilla");
  Serial.println("El ESP32 está listo, puedes emparejarlo mediante Bluetooth");
  delay(1000);

  //   inicio calibracion
  pinMode(DirI,OUTPUT);
  pinMode(DirD,OUTPUT);
  pinMode(Led,OUTPUT);
  digitalWrite(Led,HIGH);
  Serial.println("-------------- Proceso de Calibracion de Los Sensores --------------" );
  Serial.println("Iniciando ......");
  //Calibración Inicial de Pines Sensor
  for (int i = 0; i < 400; i++){  // make the calibration take about 10 seconds
    qtra.calibrate();       // reads all sensors 10 times at 2.5 ms per six sensors (i.e. ~25 ms per
call)
  }
  digitalWrite(Led,LOW);
  delay(3000);
```

Ilustración 40. Código Control Fuzzy Parte 1, Fuente: Elaboración propia

```
//Inicializacion_turbina();     //Inicialización de la turbina

  ledcAttach(PWM_D, Frecuencia, Resolucion);
  ledcAttach(PWM_IZ, Frecuencia, Resolucion);
  /*
  ledcSetup(Canales[0],Frecuencia,Resolucion);
  ledcSetup(Canales[1],Frecuencia,Resolucion);
  ledcAttachPin(PWM_D,Canales[0]);
  ledcAttachPin(PWM_IZ,Canales[1]);
*/
}

void loop() {
  Estado=digitalRead(MInit);
  Estado=1;

  while(Estado){
    Tinicio    = millis();                                    // toma el valor en milisengundos
    if(SerialBT.available()) { Sintonia_Bluetooth();}
    Estado     = digitalRead(MInit);
    Tinicio    = millis();                                    // toma el valor en milisengundos
    Salida     = Lectura_Sensor();                            // funcion de lectura de la variable
salida del  proceso
    Control    = Controlador(Referencia,Salida);              // funcion de la ley de control
    Esfuerzo_Control(Control);                                // funcion encargada de enviar el
esfuerzo de control
    Tm = Tiempo_Muestreo(Tinicio);
    //Imprimir();
    //myTurbina.write(50);
    }
}

//Para leer el sensor
float Lectura_Sensor(void) {
  Salida = (qtra.readLine(sensorValues)/7500.0) - 1.0;
  return Salida;                                        // retorno la variable de salidad del
proceso normalizada entre 0-1, al olgoritmo de control
}

//Controlador PD difuso
float Controlador(float Referencia, float Salida) {                    // Funcion para la ley de
control
  // Calcular el error y el derivada del error
  Error_fuzzy = Referencia - Salida;
  Cambio_Error_fuzzy = Error_fuzzy - Error_ant_fuzzy;

  // Definición de las reglas difusas para el controlador PD
  float P = membresiaPositiva(Error_fuzzy);
  float N = membresiaNegativa(Error_fuzzy);
  float C = membresiaCero(Error_fuzzy);

  float DP = membresiaPositiva(Cambio_Error_fuzzy);
  float DN = membresiaNegativa(Cambio_Error_fuzzy);
```

Ilustración 41.Codigo Control Fuzzy Parte 2, Fuente: Elaboración propia

```
// Regla de inferencia difusa: Modificar con las ganancias Kp_fuzzy y Kd_fuzzy
  float Control_fuzzy_P = 0.0;
  float Control_fuzzy_D = 0.0;

  // Aplicar las reglas difusas con ganancias ajustables Kp_fuzzy y Kd_fuzzy
  if (P > 0.5) {
    Control_fuzzy_P = Kp_fuzzy * P; // Ganancia proporcional
  } else if (N > 0.5) {
    Control_fuzzy_P = -Kp_fuzzy * N; // Ganancia proporcional negativa
  }

  if (DP > 0.5) {
    Control_fuzzy_D = Kd_fuzzy * DP; // Ganancia derivativa positiva
  } else if (DN > 0.5) {
    Control_fuzzy_D = -Kd_fuzzy * DN; // Ganancia derivativa negativa
  }

  // El control total es la suma de los términos proporcional y derivativo
  Control_fuzzy = Control_fuzzy_P + Control_fuzzy_D;

  // Defuzzificación de los resultados
  Defuzzificado = Control_fuzzy * Vmax;

  // Limitar la salida de control
  if (Defuzzificado > 1.5) {
    Defuzzificado = 1.5;
  } else if (Defuzzificado < -1.5) {
    Defuzzificado = -1.5;
  }

  // Guardar el error actual para el siguiente ciclo
  Error_ant_fuzzy = Error_fuzzy;

  return Defuzzificado;
}//Aqui termina el controlador PD difuso

void Esfuerzo_Control(float Control) {                          //envia el esfuerzo de control en forma
de PWM
  s1  = (offset - Control);
  s2  = (offset + Control);

  if( s1 <= 0.0 ){// Motor Derecho
    digitalWrite(DirD,HIGH);}
  else{digitalWrite(DirD,LOW);}

  if( s2 <= 0.0 ){ //Motor Izquierdo
    digitalWrite(DirI,HIGH);}
  else{digitalWrite(DirI,LOW);}

  ledcWrite(PWM_D,constrain(abs(s1), 0.0, 1.0)* Vmax);
  ledcWrite(PWM_IZ,constrain(abs(s2), 0.0, 1.0)* Vmax);

  /*ledcAnalogWrite(PWM_D, constrain(abs(s1), 0.0, 1.0)* Vmax);
  ledcAnalogWrite(PWM_IZ, constrain(abs(s2), 0.0, 1.0)* Vmax);
  analogWrite(PWM_D,  constrain(abs(s1), 0.0, 1.0)* Vmax);      // Envia el esfuerzo de control en PWM al
motor derecho
  analogWrite(PWM_IZ, constrain(abs(s2), 0.0, 1.0)* Vmax);      // Envia el esfurezo de control en PWM al
motor izquierdo
  */
}
```

Ilustración 42. Código Control Fuzzy Parte 3, Fuente: Elaboración propia

```
unsigned long int Tiempo_Muestreo(unsigned long int Tinicio){//, unsigned int Tm){ // Funcion que asegura
que el tiempo de muestreo sea el mismo siempre
  unsigned long int T =millis()-Tinicio;
  return  T;
  }

void Imprimir (){
  Serial.print("Salida= ");
  Serial.print(Salida);
  Serial.print("   Error= ");
  Serial.print(Error);
  Serial.print("   Control= ");
  Serial.print(Control);
  Serial.print("   Tm= ");
  Serial.print(Tm);
  Serial.print("   pwmI= ");
  Serial.print(abs(s1));
  Serial.print("   pwmD= ");
  Serial.println(abs(s2));
  }

void Sintonia_Bluetooth(void){                                    // Recibe porbluetooth KP,Ki,KD y Vmax
del controlador.
  caracter = SerialBT.read();                                       // Guarda un caracter de los datos
seriales
  datos = datos + caracter;                                       // Almacena caracter a caracter en el
string datos
  if (caracter == '*') {                                          // cada caracter llega separado por una
coma, y toda la cadena finaliza con un *
    d1    = datos.indexOf(',');                                   // Guarda la posición del primer
caracter ',' en el string datos
    S_Kp  = datos.substring(0, d1);                               // Almacena la subcadena que contiene
constante Kp
    d2    = datos.indexOf(',', d1+1);                             // Guarda la posición del segundo
caracter ',' en el string datos
    S_Ti  = datos.substring(d1+1, d2);                            // Almacena la subcadena que contiene
la constante Ti
    d3    = datos.indexOf(',', d2+1);                             // Guarda la posición del tercer
caracter ',' en el string datos
    S_Td  = datos.substring(d2+1, d3);                            // Almacena la subcadena que contiene
la constante Td
    d4    = datos.indexOf(',', d3+1);                             // Guarda la posición del cuarto
caracter ',' en el string datos
    S_Vmax  = datos.substring(d3+1, d4);                          // Almacena la subcadena que contiene
la constante de la velocidad máxima

    datos = "";                                                   // Limpia el string datos para una
nueva recepcion de datos
    Kp = S_Kp.toFloat();  Ti = S_Ti.toFloat();  Td = S_Td.toFloat();  // Actualiza las constantes del
controlador PID, formato flotante
    Vmax = S_Vmax.toFloat();                                      // Actualiza la velocidad máxima, a
flotante
  }
}
```

Ilustración 43.Codigo Control Fuzzy Parte 4, Fuente: Elaboración propia