UACM

Universidad Autónoma
de la Ciudad de México

Nada humano me es ajeno

COLEGIO DE CIENCIA Y TECNOLOGÍA

LICENCIATURA EN INGENIERÍA EN SISTEMAS ELECTRÓNICOS INDUSTRIALES

“Diseño y comparación de un controlador PID digital convencional, un PID difuso y un PID neurodifuso para el control de posición de un motor aplicado a un robot móvil”

TRABAJO RECEPCIONAL
PARA OBTENER EL TÍTULO DE LICENCIATURA EN
INGENIERÍA EN SISTEMAS ELECTRÓNICOS INDUSTRIALES

PRESENTA:

C. KARINA GRANADOS ALBARRÁN

Director del trabajo recepcional

Ing. Amaranto de Jesús Dávila Jáuregui

México, D.F. Noviembre 2013.

UACM
Universidad Autónoma de la Ciudad de México
Nada humano me es ajeno

CA Coordinación Académica

SISTEMA BIBLIOTECARIO DE INFORMACIÓN Y DOCUMENTACIÓN

UNIVERSIDAD AUTÓNOMA DE LA CIUDAD DE MÉXICO
COORDINACIÓN ACADÉMICA

RESTRICCIONES DE USO PARA LAS TESIS DIGITALES

DERECHOS RESERVADOS©

AGRADECIMIENTOS.

A dios.

Por haberme acompañado y guiado a lo largo de mi vida, por ser mi fortaleza en los momentos de debilidad y por brindarme una vida de aprendizajes, experiencias y felicidad, pero sobre todo por darme una excelente familia.

A mi madre.

Silvia por ser mi inspiración en la vida, por todos sus consejos, por enseñarme a nunca darme por vencida, por enseñarme a ver la vida con optimismo y principalmente por todo su amor.

A mi padre.

José del Carmen por apoyarme en todo momento, por ser un ejemplo de vida, por apoyarme en mi educación y por todo su cariño.

A mis hermanos.

Ricardo por su apoyo, por ser un ejemplo de desarrollo profesional y por todo su cariño. Luis Fernando por su apoyo, por su enorme alegría, por llenar mi vida de grandes momentos y por todo su cariño.

A mis asesores

El profesor Amaranto de Jesús Dávila Jáuregui y a la profesora Diana Aurora Cruz Hernández por darme la oportunidad de realizar mi trabajo recepcional, por su tiempo y dedicación.

A mis lectores.

La profesora Araceli Liliana Reyes Cabello, el profesor Mario Villafuerte Bante y el profesor Marcos Ángel González Olvera por aceptar ser mis lectores y por su paciencia al revisar mi trabajo recepcional.

A mis compañeros

Por su amistad, por formar parte de mi desarrollo educativo y por ser integrantes en mis equipos de trabajo en los periodos de certificación.

A la UACM

Por abrirme las puertas a una excelente educación académica y por darme la oportunidad de terminar mis estudios universitarios.
Por darme el apoyo económico para la impresión y empastado de mí trabajo recepcional.

¡ Gracias !

Contenido

CAPITULO 1.
Preliminares.

En este proyecto se diseñó un controlador PID digital convencional, un controlador PID difuso y un controlador PID neurodifuso, la implementación de cada uno de estos controladores se hizo en un microcontrolador para el control de posición del motor de dirección de un robot móvil en configuración triciclo. Se realizó una comparación del desempeño de los tres controladores.

1.1. Introducción.

La robótica ha jugado un papel importante durante el desarrollo de la humanidad y su evolución ha sido por el deseo de crear robots que faciliten el trabajo humano. El término robot proviene de la palabra checa *robota*, que significa servidumbre o trabajo forzado.

Un robot está constituido por mecanismos físicos y sistemas virtuales de software. Existen varios tipos de robots, sin embargo este trabajo aborda solo los robots móviles.

La principal característica de un robot móvil es la capacidad de moverse o desplazarse en su área de trabajo, siendo una de estas una superficie plana mediante varios mecanismos, encontrándose entre los más usuales ruedas, cadenas o patas, y entre ellos el más común es a través de ruedas, por ser el más sencillo de construir y controlar.

Para controlar la locomoción de un robot móvil es necesario utilizar control automático. El control automático es una disciplina de la ingeniería que tiene como objetivo mantener el equilibrio de los sistemas dinámicos. De acuerdo con

[1], más de la mitad de los controladores industriales que se usan hoy en día utilizan esquemas de un controlador PID o PID modificado.

Un controlador PID (Proporcional Integral Derivativo) es un mecanismo de control por retroalimentación que calcula el error entre un valor medio y el valor deseado. La estructura de un PID considera tres acciones proporcional, integral y derivativo.

El control proporcional determina la reacción con respecto al error actual, el control integral genera una corrección proporcional a la integral del error, esto nos asegura que aplicando un esfuerzo de control suficiente, el error de seguimiento se reduce a cero y finalmente el control derivativo determina la reacción del tiempo en el que el error se produce.

La acción proporcional aumenta la ganancia para lograr que el error en estado estacionario se aproxime a cero, la acción integral actúa cuando hay una desviación entre la variable y el punto de consigna, integrando esta desviación en el tiempo, con el propósito de disminuir y eliminar el error en estado estacionario y la acción derivativa mantiene el error al mínimo corrigiéndolo proporcionalmente con la misma velocidad que se produce; de esta manera evita que el error se incremente.

El controlador PID es lineal, sin embargo existen otros tipos de controladores que son no lineales como el controlador difuso y el controlador neuronal.

El control difuso es una estrategia de control basada en la lógica difusa, la cual se inspira en el procesamiento de información de forma cuantitativa. Esto con el objetivo de emular la experiencia y el conocimiento del experto en un sistema.

El control neuronal es una técnica de control que utiliza la inteligencia artificial por medio de las redes neuronales artificiales, el objetivo de las redes neuronales es el diseño de un sistema con elementos neuronales que emule el comportamiento de los sistemas neuronales de los animales.

La explicación de los tres controladores (PID, difuso y neuronal) se ampliará más adelante.

1.2. Planteamiento del problema.

Los robots móviles que utilizan como medio de locomoción ruedas tienen la posibilidad de seguir distintas trayectorias y navegar en distintos terrenos, sus aplicaciones son: automatización de procesos, bodegaje, vigilancia, limpieza doméstica, limpieza de desechos tóxicos, exploración minera, exploración planetaria, entre otras; por lo cual es necesario diseñar un controlador que garantice estabilidad, repetitividad y fidelidad al seguimiento de trayectorias y a la navegación en distintos terrenos.

Uno de los aspectos más importantes en la navegación de un robot móvil con configuración triciclo es el control exacto de la llanta de dirección, situación que en ocasiones se vuelve más compleja al considerar la fricción asociada por las distintas superficies sobre las cuales se desplaza el terreno, lo que nos exige obtener un controlador que sea robusto a las diferentes fricción asociadas por estas superficies.

1.3. Hipótesis.

Es posible diseñar un controlador PID digital convencional, un controlador PID difuso y un controlador PID neurodifuso, e implementar cada controlador en un microcontrolador de 8 bits para controlar la posición de la llanta de dirección de un robot móvil y comparar el desempeño de los tres controladores.

1.4. Justificación.

En los últimos años se ha registrado un creciente interés en la investigación de robots móviles debido a la infinidad de aplicaciones de ayuda al ser humano que tienen estos sistemas. Debido al desarrollo que ha alcanzado el área de mecanismos y sensores se ha logrado la implementación de sistemas más complejos, obteniendo robots con un alto grado de interacción con el medio, que extraen información fiel de su entorno como: detección de gases, fugas, toma de muestras, señales de video, etc. Por esta razón es necesario desarrollar metodologías de control que permitan controlar adecuadamente cada aplicación.

La IFR (Federación Internacional de Robótica), está llevando a cabo una intensa labor de apoyo al desarrollo y aplicación de distintos tipos de robots, en las aplicaciones de servicio, denominándolos robot de servicio [2].

En México ya se están comercializando los robots de vigilancia que tienen incorporadas cámaras de video, detectores de ruido y de gases, radar y otros sistemas de vigilancia. Estos robots actualmente realizan tareas de vigilancia en centros comerciales, oficinas, laboratorios, fábricas automovilísticas, estadios de fútbol e incluso en domicilios particulares.

Debido a que el campo de la robótica móvil promete tener un crecimiento acelerado en los próximos años con repercusiones innovadoras en todos los sectores económicos, es importante que México desarrolle esta tecnología, para evitar una costosa dependencia a futuro y que no pierda la oportunidad de integrarse con éxito en este campo.

1.5. Objetivos.

En este trabajo se plantean los objetivos generales y los objetivos particulares.

1.5.1. Objetivos generales.

Diseñar un controlador PID digital convencional, un controlador PID difuso y un controlador PID neurodifuso.

Implementar los tres controladores en una microcomputadora de 8 bits que sean capaces de controlar la posición de la llanta de dirección de un robot móvil de manera eficiente.

1.5.2. Objetivos particulares.

Realizar investigaciones sobre el controlador PID digital, PID difuso y PID neurodifuso.

Diseñar los controladores PID digital, PID difuso y PID neurodifuso para manipular la dirección de la llanta de un robot móvil.

Implementar los controladores diseñados en una arquitectura RISC de 8 bits.

Comparar la eficiencia de los tres controladores en diferentes superficies.

1.6. Metodología.

- Investigar sobre el estado del arte en robótica móvil.
- Investigar sobre los controladores, PID, difuso y neuronal.
- Diseño del controlador PID
  - Sintonización del controlador PID.
- Implementación del controlador PID.
  - Calcular el error
  - Calcular la acción proporcional
  - Calcular la acción integral
  - Calcular la acción derivativa
  - Sumar las tres acciones
- Aplicar el controlador PID digital al robot móvil.
- Realizar pruebas del funcionamiento del controlador PID, colocando al robot en tres diferentes superficies.
- Investigar sobre el controlador PID difuso.
- Diseño del controlador PID difuso.
- Implementación del controlador PID difuso
  - Calcular el error
  - Calcular la acción proporcional
  - Calcular la acción integral
  - Calcular la acción derivativa
  - Sumar las tres acciones
- Aplicar el controlador PID difuso al robot móvil.
- Realizar pruebas del funcionamiento del controlador PID difuso, colocando al robot en tres diferentes superficies.
- Investigar sobre el controlador PID neurodifuso.
- Diseño del controlador PID neurodifuso.

- Implementación del controlador PID difuso
  - Calcular el error
  - Calcular la acción proporcional
  - Calcular la acción integral
  - Calcular la acción derivativa
  - Sumar las tres acciones
- Aplicar el controlador PID neurodifuso al robot móvil.
- Realizar pruebas del funcionamiento del controlador PID neurodifuso.
- Comparar el desempeño de los tres controladores.

1.7. Alcances y limitaciones.

Se diseñó un controlador PID digital convencional, un PID difuso y un controlador PID neurodifuso y realizó la implementación de cada controlador en un microcontrolador de 8 bits.

Este trabajo no contempla el diseño del robot móvil, solo abarca el diseño e implementación de los controladores mencionados y las pruebas sobre el robot móvil. Se realizaron pruebas básicas sobre el posicionamiento de la llanta de dirección del robot móvil sobre tres diferentes superficies y se comprobó su desempeño.

CAPITULO 2.
Antecedentes y marco teórico.

Aunque existe una amplia variedad de robots, este trabajo aborda el área de la robótica móvil, específicamente los robots que utilizan como medio de locomoción ruedas. En este capítulo se presenta la definición de un robot móvil, los diferentes tipos de locomoción de los robots móviles y la descripción del sistema a controlar.

2.1 Robots Móviles.

Un robot móvil es un sistema electromecánico con movilidad, que trata de emular las capacidades humanas o animales, y buscan desarrollar múltiples tareas enfocadas a la vigilancia, limpieza doméstica, automatización de procesos, entre otros, haciéndole la vida al hombre más práctica.

Los robots móviles fueron creados para realizar diferentes tareas que para el hombre eran arriesgadas como son: la exploración en otros planetas, la transportación de materiales pesados, entre otros.

La robótica móvil comenzó en los años 70 como banco de pruebas para estudiar técnicas de Inteligencia Artificial. En los años 80 se tiene el despliegue definitivo debido al abaratamiento y mejores prestaciones de los ordenadores [4].

Un robot móvil está compuesto por:

- Una estructura mecánica móvil.
- Sensores y actuadores
- Un sistema de control.

Estructura mecánica. La estructura mecánica de un robot móvil dependerá del tipo de aplicación a la que será destinado, por lo que para lograr una estructura mecánica eficiente deberá realizarse un análisis detallado de las condiciones de trabajo.

En general la estructura está compuesta por una serie de elementos físicos unidos entre sí, que proporcionan una base para los subsistemas electromecánicos y electrónicos que componen al robot y que también sirven de anclaje para el sistema de tracción del mismo, estos elementos pueden ser ruedas, cadenas, patas, u otros.

Sensores y actuadores. Un sensor es un dispositivo que permite al robot recibir y percibir información del entorno, es decir los sensores suelen asociarse por analogía como los sentidos del robot.

Existen diferentes tipos de sensores dependiendo de la cantidad física que se desea medir, de los cuales algunos se muestran en la tabla 2.1.

Cantidad física.	**Tipo de sensor.**
Temperatura	• Termistor • Termopar • RTD
Luz	• Fotodiodo • Fotorresistencia • Fototransistor • Célula fotoeléctrica
Posición lineal y angular	• Potenciómetro • Encoder
Desplazamiento y deformación	• Galga extensiométrica • LVT
Velocidad lineal y angular	• Encoder

	• Giróscopio
Aceleración	• Acelerómetro
Presión	• Piezoeléctricos • Membranas
Visión artificial	• Cámara de video
Sensores de presencia	• Sensor capacitivo • Sensor inductivo • Sensor fotoeléctrico

Tabla 2.1. Tipos de sensores.

Los actuadores son dispositivos físicos que convierten la energía eléctrica, hidráulica y neumática en la ejecución de una acción. Existen diferentes tipos de actuadores, mostrados algunos en la tabla 2.2.

Tipos de actuadores.	
Eléctricos.	• Pistón eléctrico • Motor CD. • Motor AC • Motor a pasos. • Servomotores
Hidráulicos.	• Cilindro hidráulico. • Motor Hidráulico
Neumático	• Cilindro neumático • Pistón neumático • Motor neumático

Tabla 2.2. Tipos de actuadores.

Sistemas de control. El diseño de un sistema de control cosiste en generar un algoritmo que pueda ser implementado en el “cerebro” del robot, el cual es un sistema implementado en un microprocesador que basado en la información recibida por los sensores, tome decisiones y ejecute las acciones necesarias a los actuadores para lograr que el robot realice una tarea bajo ciertos criterios de desempeño.

En este tipo de sistemas en ocasiones se emula el comportamiento del cerebro humano para resolver los problemas a los que se enfrenta el robot, por medio de técnicas y algoritmos.

2.1.1. Locomoción con ruedas.

Se define a la capacidad de un robot de moverse de un lugar a otro como locomoción.

Existen varias formas de hacer mover a un robot en una superficie siendo las más comunes ruedas, cadenas y patas.

Robot con ruedas.

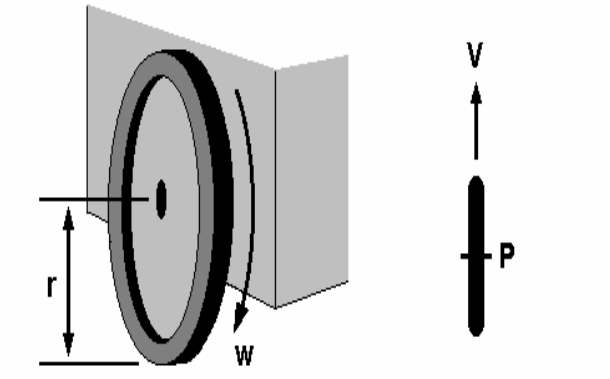
Los robot que utilizan ruedas suelen ser más fáciles de construir, ya que generalmente y dependiendo de la configuración solo se debe de suministrar energía al eje de las ruedas. La figura 2.1 muestra un ejemplo de un robot con ruedas.

Fuente: http://roboticaiquique.blogspot.mx/2010/11/robot-rover.html

Figura 2.1.Robot con ruedas fijas.

Existen varios tipos de ruedas.

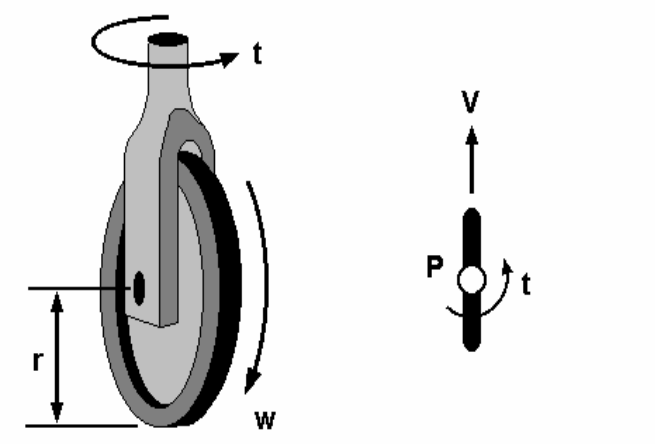
1. **Rueda fija**. . Éstas solo giran en torno a su propio eje. El movimiento se produce en la dirección de la rueda, como se observa en la figura 2.2.



Fuente: http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf

Figura 2.2. Rueda fija.

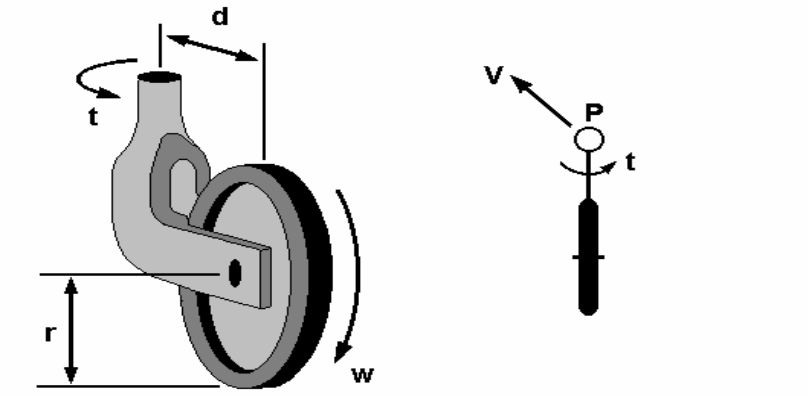
2. **Rueda orientada centrada**. Como se aprecia en la figura 2.3, se tienen dos movimientos, uno es el giro t y el otro alrededor del eje vertical que está dirigido al centro de la rueda.



Fuente: http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf

Figura 2.3. Rueda orientada centrada.

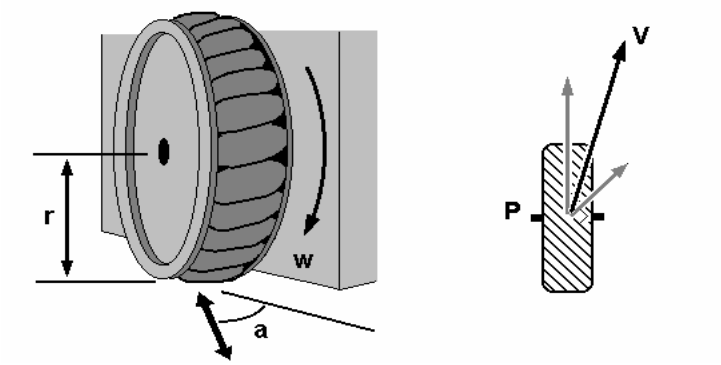
3. **Rueda orientada descentrada (rueda loca o de castor)**. Son ruedas orientables no controlables, como se ilustra en la figura 2.4.



Fuente: http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf

Figura 2.4. Rueda orientada descentrada.

4. **Rueda sueca (rueda omnidireccional)**. Se mueve en la dirección de la rueda y también se mueve en dirección perpendicular a la dirección de la rueda, como se muestra en la figura 2.5. Con dos componentes de desplazamiento.



Fuente: http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf

Figura 2.5. Rueda sueca.

Dentro de la locomoción con ruedas existen diferentes tipos de configuración.

- Diferencial.
- Triciclo.
- Ackermen.
- Síncrona.
- Omnidireccionales.

2.1.1.1. Configuración Diferencial.

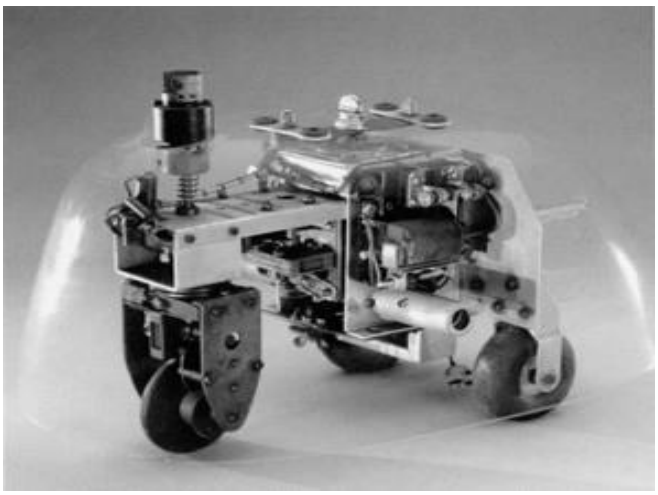
Esta configuración es una de las más sencillas, básicamente consiste en dos ruedas en un eje común, donde cada rueda se controla independientemente, lo que ocasiona que el radio de giro dependa de la diferencia entre las velocidades de las ruedas. Un ejemplo de un robot con configuración diferencial se ilustra en la figura 2.6.

Fuente: http://www.suallabs.com/index.php?route=product/product&product_id=69

Figura 2.6. Robot de configuración diferencial.

2.1.1.2. Configuración Triciclo.

Esta configuración consiste en tres ruedas. Dos ruedas traseras pasivas que no son controladas por ningún motor y sirven únicamente como soporte. La rueda delantera proporciona la dirección y la tracción del robot. Esta configuración no permite giros de 360° sobre el centro del robot a diferencia de la tracción diferencial, como se muestra en la figura 2.7.

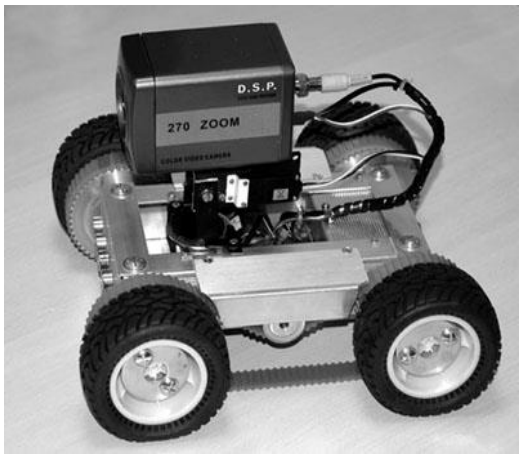


Fuente: http://www.aadeca.org/pdf/CP_monografias/monografia_robot_movil.pdf

Figura 2.7. Robot de configuración triciclo.

2.1.1.3. Configuración Ackerman.

Esta configuración consiste de cuatro ruedas. Dos ruedas traseras que son las de tracción y dos ruedas delanteras que son las de dirección. Esta configuración es muy usada en la industria del automóvil. La figura 2.8 muestra la configuración Ackerman.



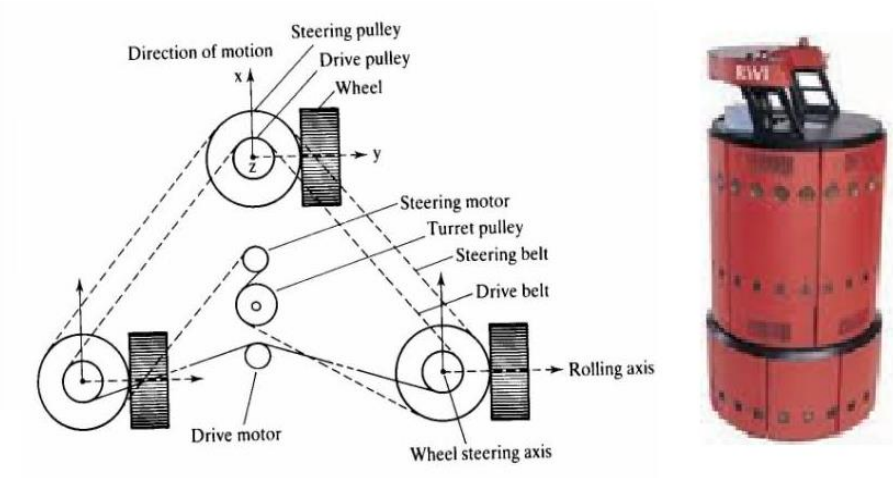
Fuente: http://www.aadeca.org/pdf/CP_monografias/monografia_robot_movil.pdf

Figura 2.8. Robot de configuración Ackerman.

2.1.1.4. Configuración Síncrona.

Esta configuración consiste de tres o más ruedas. Todas las ruedas están acopladas mecánicamente de tal forma que todas rotan a la misma dirección y a la misma velocidad. La orientación de los ejes de rotación del robot pueden ser controlados.

La sincronización de las ruedas por lo general se hace mecánicamente, sin embargo en ocasiones se hace electrónicamente colocando a cada rueda un motor. Un ejemplo de la configuración síncrona se muestra en la figura 2.9.



Fuente: http://www.esi2.us.es/~vivas/ayr2iaei/LOC_MOV.pdf

Figura 2.9. Robot de configuración síncrona.

2.1.1.5. Configuración Omnidireccional.

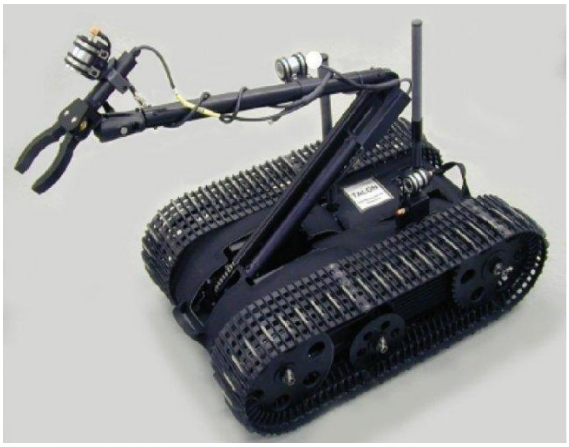
En esta configuración el robot es construido con ruedas omnidireccionales. Las ruedas omnidireccionales tienen rodillos adicionales, cuyo eje de giro es perpendicular a la dirección normal de avance. Como se observa en la figura 2.10, la configuración omnidireccional permite que el robot se pueda mover en cualquier dirección.

Fuente: http://www.elotrolado.net/hilo_review-robot-rovio_1518110

Figura 2.10. Robot de configuración omnidireccional.

2.1.2. Locomoción con cadenas (orugas).

Los robot que utilizan cadenas (orugas) se construyen uniéndole una cadena a las ruedas delanteras y traseras del robot como se muestra en la figura 2.11, con el fin de aumentar la fricción y conseguir rebasar mayores obstáculos.

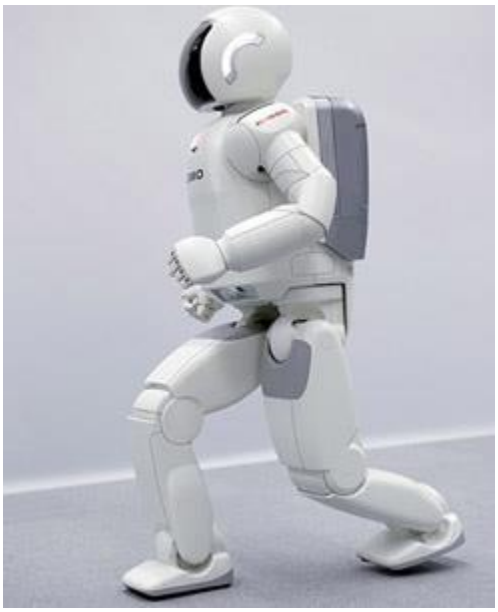


Fuente: http://www.xatakaciencia.com/robotica/robots-moviles-i

Figura 2.11. Robot oruga.

2.1.3. Locomoción con patas.

Los robots con patas son los que imitan los movimientos de los humanos o de los animales. Si el movimiento de un robot es con patas se debe de tener en cuenta los factores físicos que involucra esto como son la posición, la velocidad y el equilibrio. Los robot que utilizan patas suelen ser los más difíciles de construir, modelar y controlar. La figura 2.12 muestra un robot con patas.



Fuente: http://www.robotic-lab.com/blog/2007/10/16/el-nuevo-asimo-debuta-en-barcelona/

Figura 2.12. Robot con patas (ASIMO).

2.2 Control Automático.

Un sistema de control automático es un conjunto de elementos que interactúan entre sí de forma autónoma para obtener un resultado deseado. Los sistemas de control automático pueden ser en lazo cerrado o en lazo abierto.

Un sistema de control en lazo cerrado es cuando hay una retroalimentación al controlador con la señal de error. La señal de error es la diferencia entre el valor medio y el valor deseado.

En un sistema de control de lazo abierto, la salida no afecta al controlador, es decir no se mide la salida ni hay una realimentación que compare la señal de salida con la señal de entrada. Para que un sistema de control en lazo abierto tenga un buen desempeño es necesario conocer la relación entre la entrada y la salida, y que el sistema no tenga perturbaciones tanto externas como internas.

Comparando ambos sistemas se tiene que para el control en lazo cerrado es de suma importancia la estabilidad, mientras que para el control en lazo abierto la estabilidad no es tan importante, también se tiene que para el sistema de control en lazo cerrado las perturbaciones no son mucho problema, a diferencia del sistema de control en lazo abierto que es muy sensible a las perturbaciones.
Un ejemplo de un sistema de control en lazo cerrado es el controlador PID, el cual considera tres acciones de control: proporcional, integral y derivativa, cada acción requiere de sintonizar el controlador PID para encontrar los parámetros Kp, Ki y Kd adecuados que garanticen un buen diseño del controlador. El controlador PID es el más común en las industrias para resolver problemas de control de robots industriales.

El diseño de controladores no lineales es un problema de suma importancia en la Ingeniería de Control, esto debido a las dificultades encontradas en el diseño. Para diseñar un controlador no lineal se deben de hacer suposiciones válidas acerca de la estructura del controlador y del sistema a controlar. Sin embargo si estas suposiciones no son las correctas nos llevan frecuentemente al desarrollo de controladores ineficientes.

Existen varios controladores no lineales, no obstante para el propósito en este trabajo se utilizaron dos el controlador difuso y el controlador por redes neuronales por ser los que facilitan más el diseño del sistema que se quiere controlar.

El control difuso se basa en la lógica difusa que trabaja con conjuntos difusos, que están definidos por sus funciones de pertenencia, las cuales expresan el grado de membresía.
Un conjunto difuso se define matemáticamente como la asignación a cada posible objeto que existe en el universo de discurso, un valor que representa su grado de pertenencia o membresía en el conjunto difuso.

El control por redes neuronales artificiales busca controlar sistemas complejos, por medio de la evolución de sistemas de computación inspirados en el cerebro humano.

2.3 Descripción del sistema a controlar.

En este trabajo se diseñó e implementó tres controladores para controlar la posición de la rueda de tracción y dirección de un robot móvil. Se probó y comparó el desempeño de los tres controladores (PID digital, PID difuso y PID neurodifuso) con el fin de obtener un controlador que garantice un mejor control de posición. Los controladores deberán controlar eficientemente la posición del motor, para lograr que el robot móvil siga una trayectoria determinada.

La estructura mecánica del robot móvil a controlar está compuesta por una base de madera con dos ruedas fijas traseras y una rueda orientada centrada delantera. El robot móvil tiene una configuración triciclo, donde las dos ruedas fijas son pasivas y la rueda orientada centrada delantera es la que proporciona la dirección y la tracción, como se muestra en la figura 2.13. La dirección y la tracción están controladas por dos motores de cd de 12V.

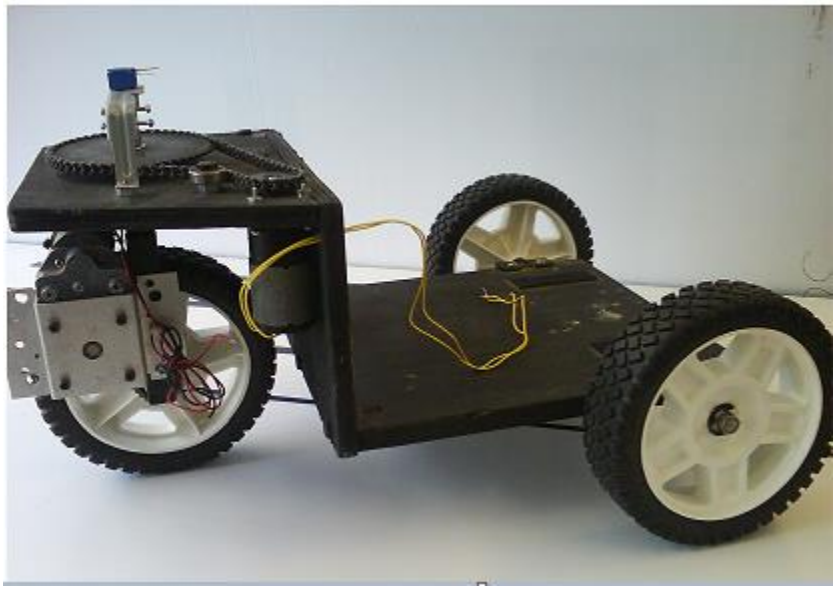


Figura 2.13. Estructura mecánica del robot móvil a controlar.

El sistema a controlar está compuesto por la rueda delantera del robot móvil, el motor de dirección y un potenciómetro de precisión de 10KΩ. Como se muestra en la figura 2.14 el motor de cd se acopló mecánicamente al potenciómetro y a la rueda por medio de dos engranes uno pequeño de 17 dientes y otro grande de 52 dientes y una cadena. Este acoplamiento es con el fin de que cuando el motor gire, la rueda y el potenciómetro también. El potenciómetro se conectó a una batería de 5V, de tal forma que cuando gire el motor la resistencia del potenciómetro varié y en consecuencia el voltaje también en el rango de 0V a 5V.
Se eligió esta construcción para facilitar el control, ya que la configuración triciclo es la más sencilla de controlar porque solo se necesita controlar un motor para determina la dirección del robot.

Figura 2.14. Acoplamiento mecánico del sistema a controlar.

El motor de cd es una máquina que convierte la energía eléctrica en mecánica provocando un movimiento rotatorio. El motor de cd que se utilizó para el control de la posición de la llanta del robot móvil es de la marca maxon con número de serie 21 40 934 22112 050 y sus características principales se muestran en la tabla 2.3.

Características	
Voltaje nominal	12.0 V
Velocidad sin carga	4080 rpm
Corriente sin carga	12.3 mA
Velocidad nominal	2430 rpm
Torque nominal	13.9 mNm
Corriente nominal	0.508 A
Corriente de arranque	1.23 A
Máxima eficiencia	82 %

Tabla 2.3. Características del motor de cd maxon.

El programa del sistema de control se realizó con la ayuda del software mikroC y se implementó en un microcontrolador PIC16F887. El software mikroC es un compilador de C para programar microcontroladores. El microcontrolador es un circuito integrado complejo que está constituido por una unidad central de procesamiento, por memorias y por puertos de entrada y salida y otros periféricos. El microcontrolador es el encargado de recibir información del sensor, tomar decisiones y mandar la señal necesaria al motor de cd.

El PIC16F887 es un microcontrolador de 8 bits, que tiene un oscilador interno, tres temporizadores, un convertidor analógico digital de 10 bits, una memoria EEPROM, una memoria Flash, un módulo de PWM con una resolución de 10 bits.

Para controlar el sentido de giro del motor se utilizó un puente h (l298). Un puente h es un circuito electrónico que permite acoplar el microcontrolador con el motor.

CAPITULO 3.
Diseño del controlador PID digital.

Los controladores PID son suficientes para resolver el problema de control de muchas aplicaciones en la industria, particularmente cuando la dinámica del proceso lo permite y los requerimientos de desempeño son sencillos.
El microprocesador ha tenido una influencia enorme sobre el desarrollo del controlador PID; ha permitido ofrecer nuevas oportunidades para implementar funciones adicionales como el ajuste automático de parámetros y los cambios de modos de control.

3.1. Controlador PID.

El funcionamiento del controlador PID consiste en la lectura de dos señales que son: la señal de referencia que es el valor donde se quiere llegar, y la señal del sensor que representa el valor medido. Estas dos señales tienen que ser de la misma naturaleza.
El controlador PID resta las dos señales y así es como obtiene la señal de error. La señal de error es utilizada por los tres parámetros (proporcional, integral y derivativo) del controlador, para lograr tres señales que sumadas componen la señal de salida. La señal de salida es la que va a controlar algún actuador, sin embargo esta señal debe de ser modificada para ser compatible con el actuador.

La figura 3.1 muestra el diagrama de bloques en paralelo del controlador PID.

Donde:

R(s) = señal de referencia.
E(s) = error
Kp = constante proporcional
Ki = constante integral

Kd = constante derivativa

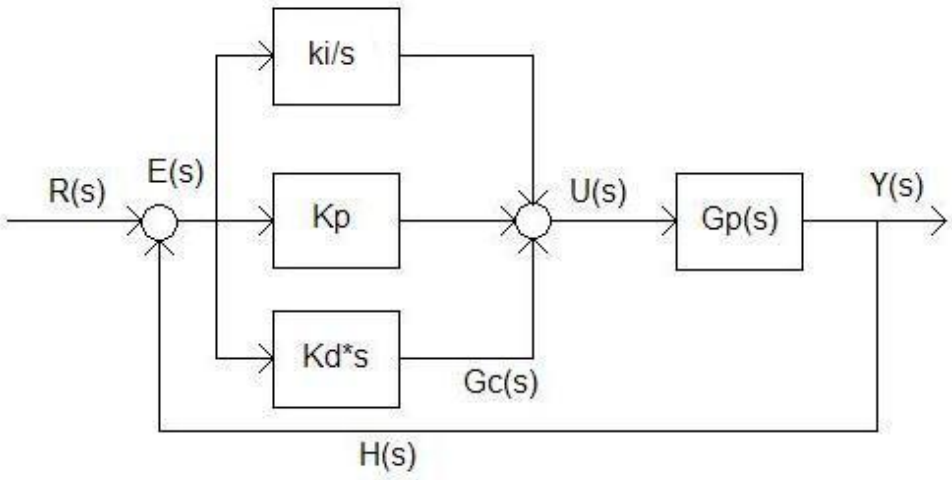
Gc(s) = función de transferencia del controlador

Gp(s) = función de transferencia del sistema o planta

U(s) = entrada de la planta

Y(s) = salida de la planta

H(s) = retroalimentación.



Fuente: http://control-pid.wikispaces.com/

Figura 3.1. Diagrama de bloques en paralelo de un controlador PID.

La parte proporcional del controlador PID consiste en la multiplicación entre la señal de error y la constante proporcional (Kp), esto con el fin de lograr que el error en estado estacionario del sistema sea casi nulo. Sin embargo se debe tener cuidado al asignarle algún valor a Kp para no llegar a una sobreoscilación. La sobreoscilación ocurre cuando el sistema alcanza valores superiores a los deseados.

La parte proporcional, denotada por $up(t)$, está definida por la expresión:

$$up(t) = Kpe(t) \qquad (3.1)$$

La parte integral del controlador PID consiste en integrar el error, que tiene la función de sumar el error en un periodo determinado, después el resultado se multiplica por la constante integral (Ki).

La ganancia se puede ajustar para conducir al error a cero en el tiempo requerido; un aumento demasiado alto en la ganancia puede causar oscilaciones y una ganancia demasiado baja puede resultar en una respuesta lenta.

La parte integral, denotada por $ui(t)$, está dada por la expresión:

$$ui(t) = Ki \int_0^t e(\tau)\, d\tau \qquad (3.2)$$

La parte derivativa del controlador PID consiste en derivar el error y luego multiplicarlo por la constante derivativa (Kd).

La parte derivativa corrige el error en proporción a la velocidad en que este se produce. La manifestación de la parte derivativa se da solo cuando el error no es constante.

Si la ganancia es demasiado alta el sistema puede oscilar, y si la ganancia es demasiado baja la respuesta puede ser lenta.

La parte derivativa, denotada por $ud(t)$, está dada por la expresión:

. $$ud(t) = Kd\ \frac{de(t)}{dt} \qquad (3.3)$$

La relación entrada-salida de un controlador PID se puede expresar como:

$$u(t) = Kpe(t) + Ki \int_0^t e(\tau)d\tau + Kd\ \frac{de(t)}{dt} \qquad (3.4.)$$

Cabe aclarar que Ki y Kd están dadas por:

$$Ki = \frac{Kp}{Ti} \qquad Kd = KpTd$$

Kp es la constante que determina el nivel de amplificación del elemento de control.
Ki es una constante que determina la velocidad de respuesta del sistema de control y **Ti** es el tiempo integral que controla la acción integral del sistema.
Kd es la constante derivativa y **Td** es el tiempo derivativo o de adelanto y controla la acción derivativa del sistema, es decir es la medida de la rapidez con que compensa un controlador PD un cambio en la variable regulada.

La función de transferencia del controlador PID es la siguiente:

$$\frac{U(S)}{E(S)} = kp + \frac{ki}{S} + kds \qquad (3.5)$$

Para implementar el controlador PID utilizando una computadora digital tenemos que convertir la ecuación 3.4, de un proceso continuo a un proceso discreto. Existen varios métodos para hacer esto, pero el más simple es el uso de la aproximación trapezoidal, este método consiste en hacer una aproximación de la integral y de la derivada en forma continua a una discreta.

$$\int_0^t e(t)dt \sim \sum_{k=1}^{n} T\, e\,(kT) \qquad \frac{de(t)}{dt} \sim \frac{e(kT) - e(kT-T)}{T}$$

Donde k es el número de muestras y T es el tiempo de muestreo.
Sustituyendo las aproximaciones en la ecuación 3.4 tenemos:

$$u(kT) = kpe(kT) + ki \sum_{k=1}^{m} Te(kT) + kd \frac{e(kT)-e(kT-T)}{T} \qquad (3.6).$$

3.2. Sintonización del controlador PID.

Para el diseño de un controlador PID es necesario modelar la planta para determinar los parámetros del controlador; sin embargo en ocasiones la planta es complicada de modelar, es por esto que se recurre a métodos experimentales para la sintonización de los controladores PID.
El ajuste o sintonización de un controlador PID consiste en seleccionar los valores de los parámetros del controlador Kp, Ti y Td con base a la respuesta de un escalón experimental. Hay muchas técnicas para el ajuste de un controlador.

Una de estas técnicas es la de Ziegler – Nichols, la cual es útil cuando se desconoce la función de transferencia de la planta. Las reglas que propusieron Ziegler – Nichols para determinar los valores de Kp, Ti y Td se basan en las características de la respuesta transitoria de una planta específica.
Hay dos métodos para la sintonización de Ziegler – Nichols, en los dos métodos se pretende lograr un 25% de sobrepaso máximo en la respuesta al escalón unitario.

En el primer método la respuesta al escalón unitario se obtiene en lazo abierto. La respuesta debe de tener la forma de una S como se muestra en la figura 3.2, si no se logra obtener este resultado este método no se puede aplicar. La respuesta en forma de S tiene tres parámetros que son la constante de ganancia K, el tiempo de retardo L y la constante de tiempo T, estos parámetros se determinan dibujando una recta tangente en el punto de inflexión de la curva con forma de S y determinando las intersecciones de esta tangente con el eje del tiempo. Ziegler - Nichols sugirieron establecer los valores de Kp, Ti y Td de acuerdo con la tabla 3.1. En este caso, la función de transferencia se aproxima mediante un sistema de primer orden.

$$G(s) = \frac{K\, e^{-Ls}}{Ts+1} \qquad (3.7)$$

Donde K es la constante de ganancia, L es el tiempo de retardo y T es la constante de tiempo.

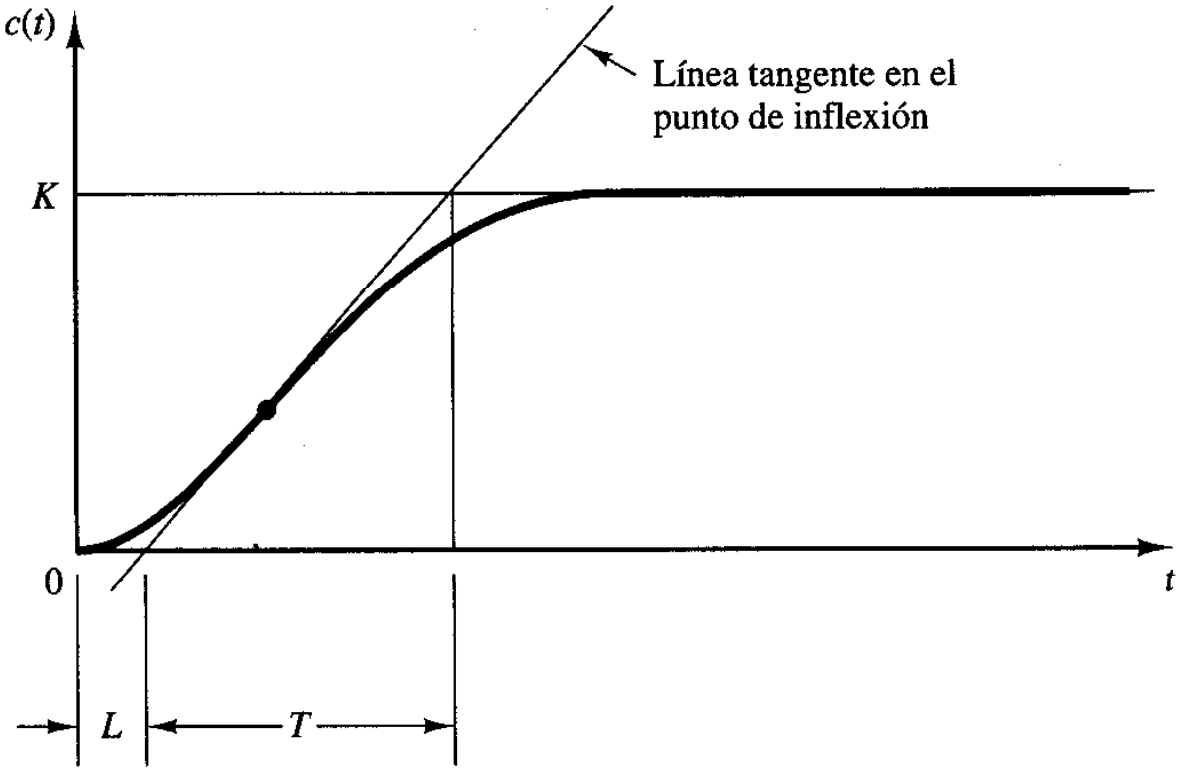


Figura 3.2. Respuesta a una entrada escalón

Tipo de controlador	Kp	Ti	Td
P	$\frac{T}{L}$	∞	0
PI	$\frac{T}{L}$ 0.9	$\frac{L}{0.3}$	0
PID	$\frac{T}{L}$ 1.2	2L	0.5L

Tabla 3.1. Regla de sintonización de Ziegler – Nichols basada en la respuesta de una entrada escalón.

El segundo método es en lazo cerrado, primero se asignan T i= 0 y Td = 0, esto ocasiona que solo se use la acción del control proporcional, posteriormente se va incrementando Kp de 0 hasta un valor crítico Kcr, este incremento debe de ocasionar que la respuesta de la salida oscile como se muestra en la figura 3.3, sí

esto no sucede este método no se puede aplicar. Con este método encontramos experimentalmente la ganancia crítica Kcr y el periodo Pcr como se muestra en la figura 3.3.

Para este método Ziegler – Nichols sugirieron establecer los valores de los parámetros Kp, Ti y Td de acuerdo con la tabla 3.2.

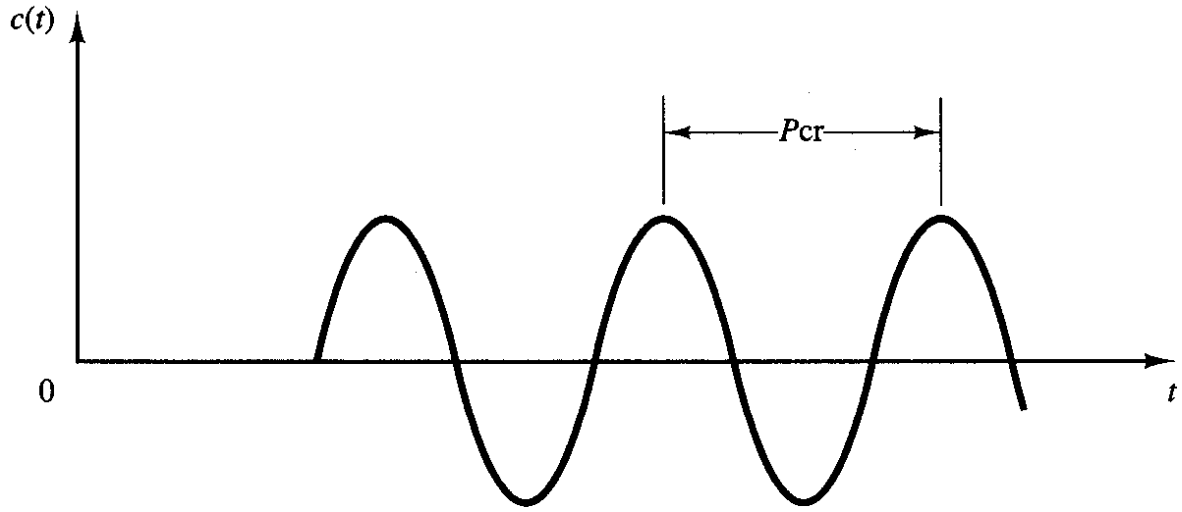


Figura 3.3. Respuesta se salida oscilatoria.

Tipo de controlador	Kp	Ti	Td
P	0.5Kcr	∞	0
PI	0.45Kcr	$\frac{1}{1.2}$ Pcr	0
PID	0.6Kcr	0.5 Pcr	0.125Pcr

Tabla 3.2. Regla de sintonización de Ziegler – Nichols basada en la ganancia critica.

En este caso la sintonización del controlador pid digital se realizó con el método en lazo abierto de Ziegler – Nichols que se explicó anteriormente.

El primer paso del método en lazo abierto de Ziegler – Nichols consiste en aplicar una entrada escalón al sistema que se quiere controlar; el sistema a controlar es un motor de cd de 12 V.

Para aplicar una entrada escalón al motor se realizó un programa en lenguaje C usando el software PIC C Compiler, el programa se ejecutó en el microcontrolador PIC16F887. El programa consistió en activar tres pines del microcontrolador de acuerdo a la tabla 3.3, con la finalidad de controlar la dirección del motor, hacia la derecha o izquierda, los pines del microcontrolador fueron conectados a un puente h (L298), el cual sirve para acoplar al microcontrolador con el motor de cd.

La figura 3.4 muestra la respuesta del motor de cd a una entrada escalón.

PIN1(ENABLE)	PIN2	PIN3	Dirección
1	0	1	Derecha
1	1	0	Izquierda

Tabla 3.3. Control de dirección de un motor.

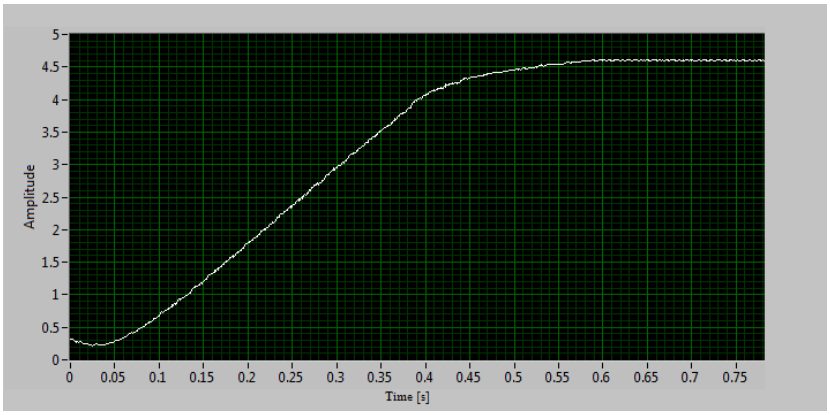


Figura 3.4. Respuesta del motor de cd a una entrada escalón.

Como se observa en la figura 3.4 la respuesta al escalón tiene la forma de una S por lo tanto si se puede aplicar el método en lazo abierto de Ziegler – Nichols; dibujando una recta tangencial en el punto de inflexión de la curva con forma de S de la figura 3.4 y determinando las intersecciones de esta tangente con el eje del tiempo se obtuvieron los parámetros de K=4.51, L = 0.1 y T=0.3 como se muestra en la figura 3.5.

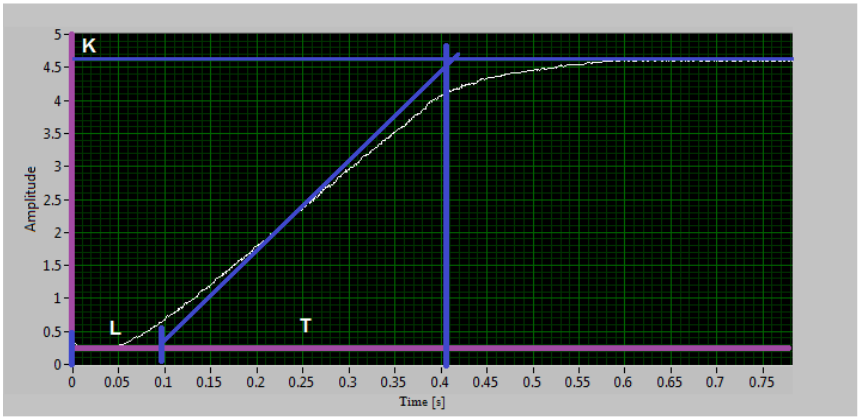


Figura 3.5. Respuesta del motor de cd a una entrada escalón, con recta tangencial en el punto de inflexión.

Teniendo los valores de L y T con la ayuda de la tabla 3.1 se calcularon los parámetros de Kp, Ti, Ki, Td y Kd.

Controlador PID.

$$\mathrm{Kp} = 1.2\frac{T}{L} = 1.2\frac{0.3}{0.1} = 3.6, \quad \mathrm{Ti} = 2\mathrm{L} = 2(0.1) = 0.2, \quad \mathrm{Td} = 0.5\mathrm{L} = 0.5(0.1) = 0.05$$

$$\mathrm{Ki} = \frac{Kp}{Ti} = \frac{3.6}{0.2} = 18, \qquad \mathrm{Kd} = \mathrm{KpTd} = 3.6(0.05) = 0.18$$

Después de calcular los parámetros (Kp, Ki y Kd) del controlador se sustituyen en la ecuación 3.5 para obtener la función de transferencia del controlador.

$$Gc(s) = \frac{0.18\,s^2+3.6s+18}{s} \qquad (3.8)$$

3.3. Realización del controlador PID.

Antes de la implementación física del controlador PID se recomienda simularlo para verificar que los parámetros obtenidos en la sintonización fueron los adecuados.

La simulación se realizó con la ayuda del software matlab en simulink, para hacer la simulación en simulink fue necesario tener la función de transferencia de la planta, para esto nos basaremos en la función de transferencia de un motor de cd de la literatura del autor Benjamín C. Kuo [9].

$$G(s) = \frac{ki}{(LaJm)s^3+(RaJm+BmLa)s^2+(KbKi+RaBm)s} \qquad (3.9)$$

La función de transferencia es de tercer orden, no obstante si se factoriza a *s* el sistema se puede reducir a un sistema de segundo orden con un integrador.

$$G(s) = \frac{ki}{(LaJm)s^2+(RaJm+BmLa)s+(KbKi+RaBm)} \ \frac{1}{s} \qquad (3.10)$$

Donde:

G(s) = Función de transferencia de un motor de cd

La = inductancia de la armadura.

Jm = inercia del motor.

Ra = resistencia de armadura.

Bm = coeficiente de fricción viscosa.

Kb = constante de la fuerza contraelectromotriz.

Ki = constante del par.

La ecuación 3.9 representa el modelo matemático del motor, pero se desconocen los valores de cada parámetro, como la obtención de los parámetros es complicada, el sistema de segundo orden se determinara con el primer método de sintonización de Ziegler – Nichols de acuerdo a la ecuación 3.7.

$$G(s) = \frac{K\, e^{-Ls}}{Ts+1}$$

Para obtener el sistema de segundo orden de la ecuación 3.7 es necesario reacomodarla con algunos pasos algebraicos.

$$e^{-Ls} = \frac{1}{1+sL}$$

$$\frac{C(s)}{U(s)} = \frac{K\frac{1}{1+sL}}{Ts+1} = \frac{K}{(1+sL)(Ts+1)}$$

$$\frac{C(s)}{U(s)} = \frac{K}{TLs^2+(T+L)s+1} \qquad (3.11)$$

Sustituyendo los valores K, L y T obtenidos anteriormente, en la ecuación 3.8, se obtiene el sistema de segundo orden.

$$\frac{C(s)}{U(s)} = \frac{4.5}{0.03s^2+0.4s+1} \qquad (3.12)$$

De acuerdo a la ecuación 3.10 la función de transferencia del motor es un sistema de segundo orden con un integrador, por lo tanto la función de transferencia del motor es la siguiente:

$$\frac{C(s)}{U(s)} = \frac{4.5}{0.03s^2+0.4s+1}\ \frac{1}{s} = \frac{4.5}{0.0.3s^3+0.4s^2+s} \qquad (3.13)$$

Analizando la función de trasferencia se puede determinar el comportamiento del sistema como es la estabilidad, el tipo y su respuesta al escalón unitario.
Para determinar la estabilidad y el tipo del sistema es necesario calcular los polos del sistema (ecuación 3.13), con la ayuda de matlab los valores de los polos son:

0 -3.33 y -10, de acuerdo al criterio de estabilidad de Routh – Hurwitz, que establece que si cualquiera de los polos de la ecuación característica está en el semiplano derecho del plano s el sistema es estable, por lo tanto el sistema del motor de cd cumple las condiciones de estabilidad.

Otro método que determina la estabilidad de un sistema es el lugar geométrico de las raíces que consiste en determinar el lugar geométrico de los polos y ceros de una función de transferencia a medida que varía la ganancia del sistema en un determinado intervalo. Con la ayuda de matlab se graficó el lugar geométrico de las raíces de la función de transferencia del motor de cd (figura 3.6).

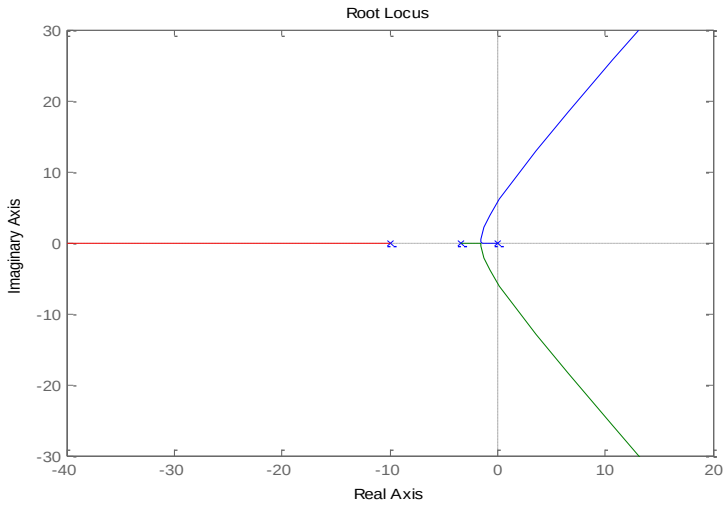


Figura 3.6. Lugar geométrico de las raíces de la función de transferencia del motor

Como se puede observar en la figura 3.6 todos los polos de la función de transferencia se encuentran en el semiplano izquierdo por lo tanto el sistema es estable.

El tipo de sistema se refiere a los polos que se encuentran en el origen del plano s, en la función de transferencia existe un solo polo en el origen, por consiguiente el tipo del sistema es uno. Para determinar la respuesta al escalón unitario es necesario que el sistema sea de segundo orden por esto se utilizó la ecuación 3.12. Como el sistema es de segundo orden se obtienen dos polos -3.33 y -10.

La función de transferencia de un sistema de segundo orden la podemos escribir en términos de la frecuencia natural wn y del factor de amortiguamiento relativo ζ. Esta es la forma general que adopta un sistema de segundo orden en el dominio del tiempo.

$$\mathrm{G2(s)} = \frac{b0\ wn^2}{s^2+2\zeta wns+wn^2} \tag{3.14}$$

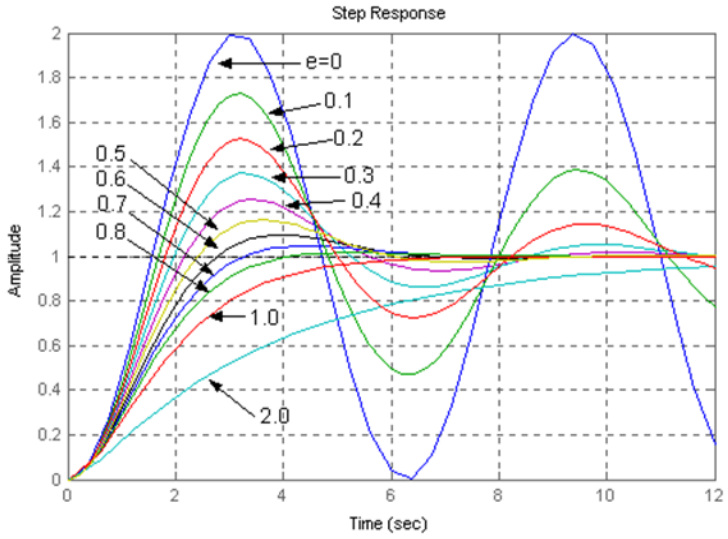
La ecuación característica denotada por Δ(s) está definida por el denominador de la función de transferencia anterior.

$$\Delta(\mathrm{s}) = s^2 + 2\zeta wns + wn^2 \tag{3.15}$$

El comportamiento de la respuesta natural del sistema depende de que valor tenga el factor de amortiguamiento, existen cuatro casos diferentes.

1. Si ζ = 1, los polos son reales e iguales, y la respuesta natural se llama *críticamente amortiguada*.
2. Si ζ > 1, entonces los polos son reales y distintos, y la respuesta natural se llama *sobreamortiguda.*
3. Si ζ < 1, los polos son un par complejo conjugado, y la respuesta natural se llama *subamortiguada.*
4. Si ζ = 0, los polos son complejos, la respuesta natural es senoidal, es decir no hay amortiguamiento.

La figura 3.7 muestra la respuesta natural de un sistema para diferentes valores del factor de amortiguamiento.



Fuente: http://plantscontrol.blogspot.mx/2012/02/8_3049.html

Figura 3.7. Respuesta natural de un sistema para diferentes valores del factor de amortiguamiento.

Para calcular ζ y wn de la ecuación 3.12 se toma la ecuación característica

$$\Delta 1(s) = 0.03s^2 + 0.4s + 1 \qquad (3.16)$$

y se divide entre 0.03 para que tome la forma de la ecuación 3.15

$$\Delta 2(s) = s^2 + 13.3s + 33.3 \qquad (3.17)$$

Igualando la ecuación 3.15 con 3.17

wn^2 = 33.3 $\qquad wn = \sqrt{33.33} = 5.7$

$2\zeta\, wn$ = 13.33 $\qquad \zeta = \frac{13.3}{2wn} = 1.16$

Debido a que los polos son -3.33 y -10, es decir reales y distintos y ζ > 1, la respuesta natural del sistema es sobreamortiguada. Esta respuesta se comprobó mediante la simulación en simulink, mostrada en la figura 3.8.

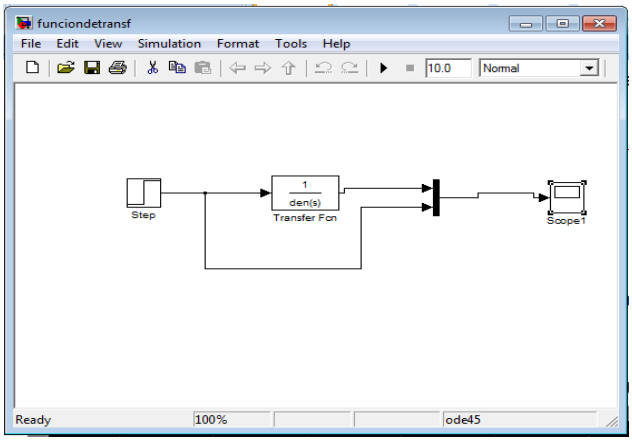


Figura 3.8.Simulación en simulink de la función de transferencia de segundo orden de un motor de cd a una entrada escalón.

Como se puede observar en la figura 3.9 la respuesta al escalón unitario al sistema de segundo orden tiene un comportamiento sobreamortiguado.

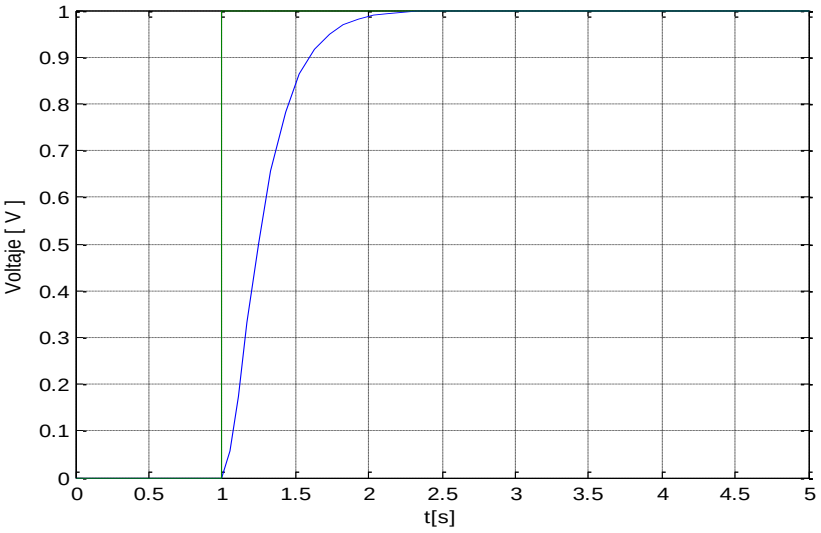


Figura 3.9. Respuesta a una entrada escalón de un sistema de segundo orden de un motor de cd.

Una vez que se verificó que el sistema es estable y se obtuvo la respuesta a una entada escalón se procedió a diseñar y simular el controlador PID.

Para el diseño del controlador PID fue necesario considerar la función de transferencia de tercer grado (ecuación 3.13). Como se mencionó anteriormente esta función se comporta como un sistema de segundo orden más un integrador por lo que no fue necesario sumar la parte integral al controlador reduciéndose a un controlador PD.

La función de transferencia del controlador PD, denotada por Gc(s), es la siguiente.

$$Gc(s) = 0.18s + 3.6 \qquad (3.18).$$

Teniendo la función de transferencia del motor (3.13) y la función de transferencia del controlador (3.18), se multiplican y como resulto se obtiene Gpd (s).

$$Gpd\ (s) = \frac{0.81s+16.2}{0.03s^3+0.4s^2+s} \qquad (3.19)$$

La figura 3.10 muestra la simulación del controlador PID con la función de transferencia 3.15 a una entrada escalón.

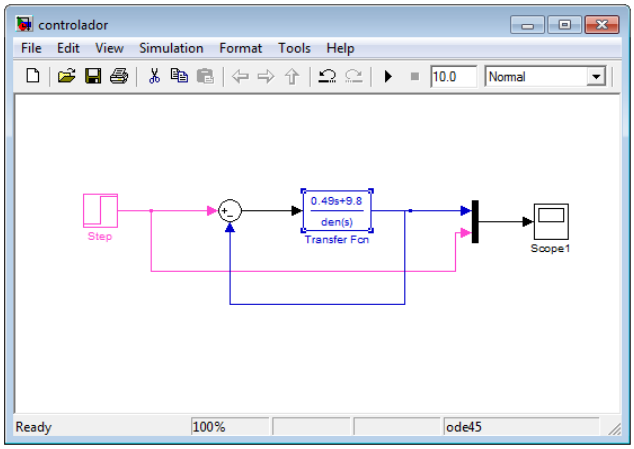


Figura 3.10. Simulación en simulink de un controlador PD a un motor de cd.

La gráfica de la simulación se observa en la figura 3.11.

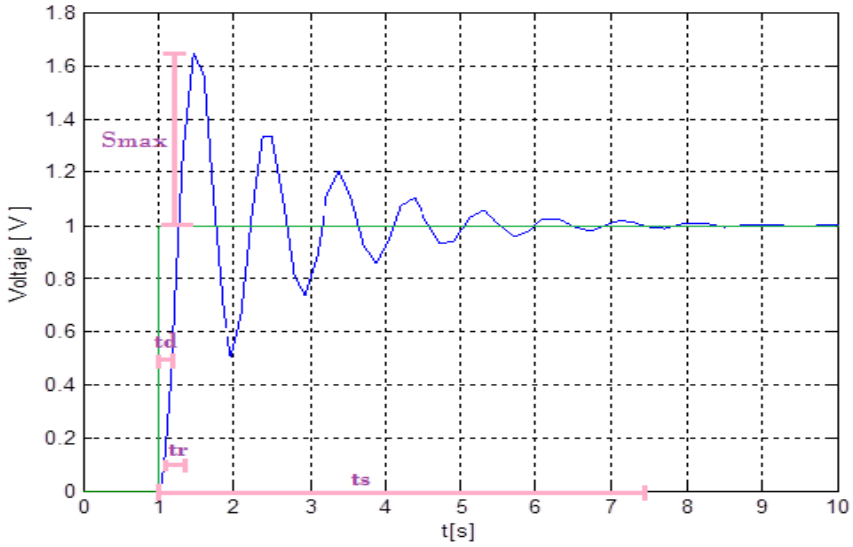


Figura 3.11. Respuesta del controlador PD con parámetros Kp = 3.6 y Kd = 0.18.

Para el análisis de la gráfica anterior es importante señalar que la respuesta en el tiempo de un sistema de control se divide normalmente en dos partes: la respuesta transitoria y la respuesta en estado estable. La respuesta transitoria es la parte de la respuesta en el tiempo que tiende a cero cuando el tiempo se hace muy grande y la respuesta en estado estable es la parte de la respuesta total que permanece después que la transitoria ha desaparecido.

La respuesta transitoria tiene parámetros que definen el comportamiento del sistema de control como: el sobrepaso máximo, el tiempo de retardo, tiempo de levantamiento y el tiempo se asentamiento.

Sobrepaso máximo (Smax). Se representa como un porcentaje del valor final de la respuesta al escalón.

Tiempo de retardo (td). Se define como el tiempo requerido para que la respuesta al escalón alcance el 50% de su valor final.

Tiempo de levantamiento (tr). Se define como el tiempo requerido para que la respuesta al escalón se eleve del 10 al 90% de su valor final.

Tiempo de asentamiento (ts). Se define como el tiempo requerido para que la respuesta al escalón disminuya y permanezca dentro de un porcentaje especifico de su valor final. Una cifra de uso frecuente es 5%.

En la respuesta en estado estable el parámetro importante es el error que esta denotado por ess, donde R(s) es la señal de entrada y G(s) es la función de transferencia.

$$ess = \frac{lim}{s \to 0} = \frac{SR(S)}{1+G(S)} \qquad (3.20)$$

La frecuencia natural Wn y el factor de amortiguamiento ζ están relacionados con tiempo de levantamiento y con el tiempo de asentamiento por las siguientes ecuaciones:

$$wn = \frac{\pi}{2tr} \qquad (3.21)$$

$$\zeta = \frac{4}{tswn} \qquad (3.22)$$

Analizando la gráfica anterior se obtuvieron los siguientes resultados:

Sobrepaso máximo = 66 %

Tiempo de retardo = 0.28s

Tiempo de levantamiento = 0.57s

Tiempo de asentamiento = 6.3s

Error en estado estable = 0

Frecuencia natural wn = 2.7

Factor de amortiguamiento ζ = 0.23

La cantidad del sobrepaso máximo y del tiempo de asentamiento es excesiva esto debido a que el método de sintonización de Ziegler – Nichols para determinar los parámetros de Kp y Kd es solo una aproximación. Para determinar los valores adecuados de Kp y Kd se utilizó el método del lugar geométrico de las raíces.

Como se mencionó anteriormente el lugar geométrico de las raíces permite determinar la posición de los polos y ceros de la función de transferencia en lazo cerrado para un determinado valor de ganancia K a partir de la función de transferencia a lazo abierto. Sin embargo si se requiere de un ajuste diferente al

de la ganancia, se pueden adicionar polos y ceros sobre el lugar geométrico de las raíces.

Efectos de añadir polos. La adición de un polo a la función de transferencia en lazo abierto tiene el efecto de acercar el lugar geométrico de las raíces a la derecha, lo cual disminuye la estabilidad relativa del sistema y a lenta el asentamiento de la respuesta. Es importante explicar que un controlador integral añade un polo en el origen, con lo cual el sistema se puede volver menos estable.
Efectos añadir ceros. La adición de un cero a la función de transferencia en lazo abierto tiene el efecto de acercar el lugar geométrico de las raíces hacia la izquierda, con lo cual el sistema tiende a ser más estable, y se acelera el asentamiento de la respuesta. Un controlador derivativo añade un cero al sistema.

Observando la figura 3.6 lo que conviene es añadir un cero en lugar geométrico de las raíces entre el polo que está en el origen y el polo -3.33, por lo cual el cero se añadió es en -2, como se observa en la figura 3.12.

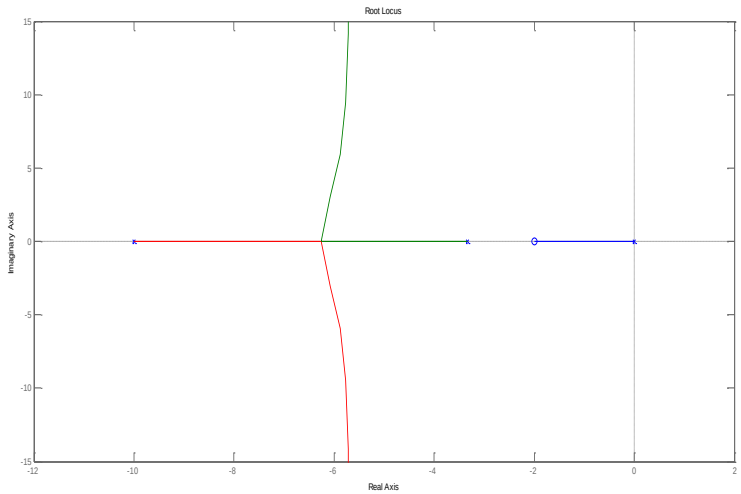


Figura 3.12.Lugar geométrico de las raíces con un cero añadido.

Si multiplicamos la función de transferencia del controlador con la función de transferencia de motor obtenemos Gpd1.

$$\text{Gpd1 (s)} = \frac{4.5\ (Kds+kp)}{0.03s^3+0.4s^2+s} \qquad (3.23)$$

De la ecuación 3.22 el cero de la función de transferencia está determinado por el numerador, este se determina igualando el numerador a cero y despejando a s.

$$4.5\ (\mathrm{Kds+Kp}) = 0 \qquad s = \frac{-Kp}{Kd}$$

Como el cero se desea en -2 se obtiene la siguiente igualdad.

$$s = \frac{-Kp}{Kd} = -2$$

Para que la igualdad se cumpla se propusieron los valores de Kp = 2 y Kd = 1.

Con los nuevos valores de Kp y Kd se obtuvo la respuesta al escalón unitario (figura 3.13)

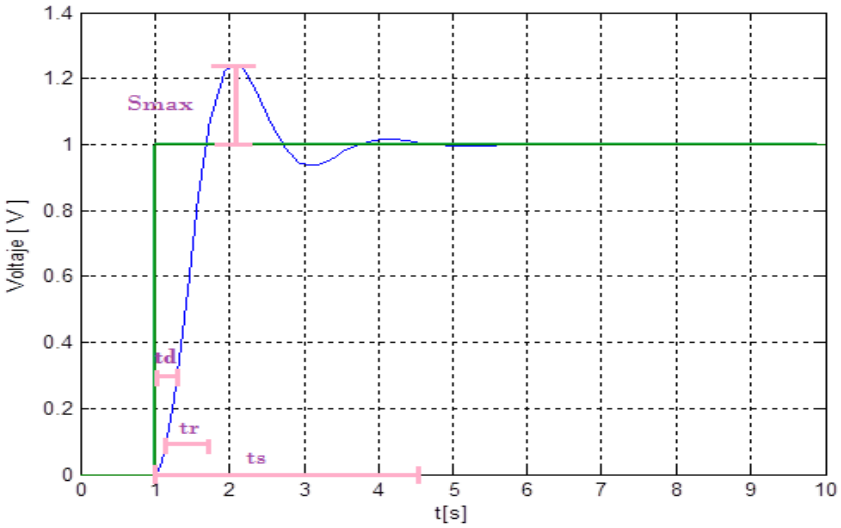


Figura 3.13. Respuesta del controlador PD con parámetros Kp = 2 y Kd = 1.

Teniendo los parámetros de Kp y Kd se calculó el error en estado estable con la ecuación 3.20, para una entrada escalón.

$$\text{ess} = \lim_{s\to 0} \frac{SR(S)}{1+G(S)} = \lim_{s\to 0} \frac{s\frac{R}{s}}{1+G(s)} = \lim_{s\to 0} \frac{s\frac{1}{s}}{1+\frac{s+2}{0.03s^2+0.4s+1}} = \lim_{s\to 0} \frac{1}{\frac{0.03s^2+0.4s+1+s+2}{0.03s^2+0.4s+1}}$$

ess = $\frac{1}{3}$ = 0.3

El resultado anterior se comparó con el error obtenido en las gráficas de respuesta a una entrada escalón.

Analizando la gráfica anterior se obtuvieron los siguientes resultados:
Sobrepaso máximo = 27 %
Tiempo de retardo = 0.28s
Tiempo de levantamiento = 0.57s
Tiempo de asentamiento = 3.5s.
Error en estado estable = 0
Frecuencia natural wn = 2.7
Factor de amortiguamiento ζ = 0.42

Una vez simulado el controlador PD, se procedió a digitalizarlo con la aproximación trapezoidal, que se explicó anteriormente, con base en la ecuación 3.6.
Sustituyendo los parámetros de Kp = 2, Kd = 1 y Ki = 0 en la ecuación 3.6.

$$U(kT) = 2\,e(kT) + \frac{e(kT) - e(kT-T)}{T} \qquad (3.24)$$

El diagrama de flujo del PD que se programó en el microcontrolador es el siguiente:

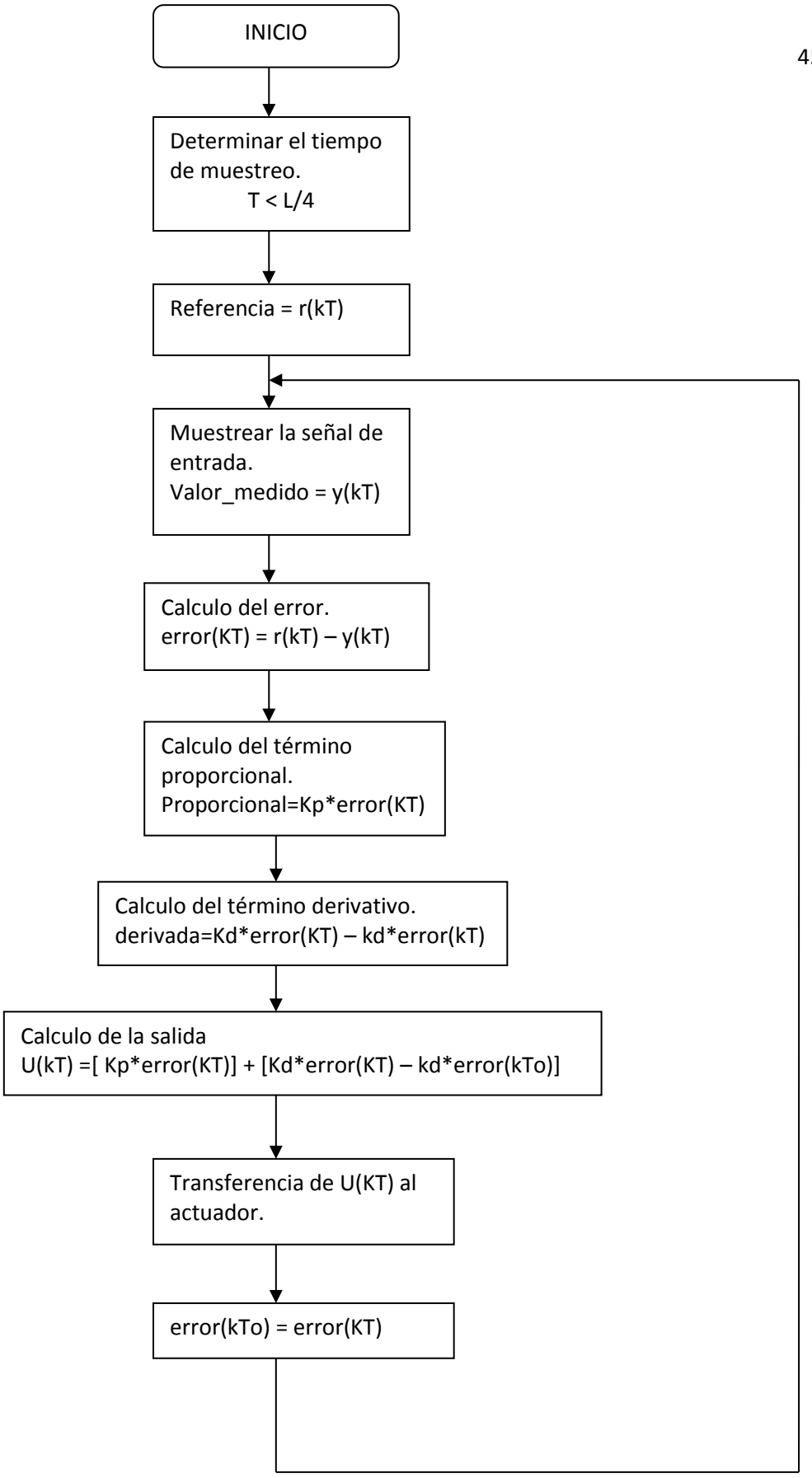


Figura 3.14. Diagrama de flujo del controlador PD.

El algoritmo para programar el microcontrolador es el siguiente:

1. Determinar el tiempo de muestreo.
2. Determinar el punto de ajuste.
3. Muestrear la señal de entrada.
4. Calcular el error.
5. Calcular la parte proporcional.
6. Calcular la parte derivativa.
7. Sumar la parte proporcional y la derivativa
8. Verificar si el error es positivo o negativo.
9. Activar el PWM.

Determinar el tiempo de muestreo. En el modelo de Ziegler – Nichols se toma un valor de T < $\frac{L}{4}$, por lo tanto el tiempo de muestreo tiene que ser menor a 25 ms, para determinar el tiempo de muestreo se utilizara una interrupción con el timer0 cada 16 ms.

Determinar el punto de ajuste. Es el valor de referencia, este valor se determina de acuerdo a la resolución del ADC (10 bits) y se calcula la ecuación de entrada-salida, esto para obtener las mismas unidades de medición del valor medido.

Resolución del ADC.

Resolución = $\frac{Vifs}{2^n-1}$ = $\frac{5}{2^{10}-1}$ = 0.004887.

Ecuación entrada-salida.

D = $\frac{V}{Resolución}$ = $\frac{3}{0.004887}$ = 613.87

Muestrear la señal de entrada. El valor medido es la señal del potenciómetro, esta señal es de voltaje, que varía de 0 a 5 V. La señal de voltaje es analógica por lo que fue necesario utilizar un convertidor de analógico a digital (ADC). Se utilizó

el ADC del PIC16F887, este convertidor es de 10 bits. La relación entre voltaje y posición (ángulo) se muestra en la siguiente tabla.

voltaje	Posición (ángulo)
0	0°
0.5	18°
1	36°
1.5	54°
2	72°
2.5	90°
3	108°
3.5	126°
4	144°
4.5	162°
5	180°

Tabla 3.4. Relación entre voltaje y posición.

Calcular el error. Una vez que el valor medido y el valor deseado tienen las mismas unidades se restan para obtener el error.

Calcular la parte proporcional. La parte proporcional se calcula multiplicando el parámetro Kp por el error.

Proporcional = Kp*error

Calcular la parte derivativa. La parte derivativa se calcula restando el producto del parámetro Kd por el error actual con el producto del parámetro Kd por el error anterior.

Derivada = Kd * error actual – Kd * error anterior.

Sumar la parte proporcional y la derivativa. Se hace la suma de la parte proporcional y la parte derivativa.

Verificar si el error es positivo o negativo. Se verifica si el error es mayor a cero o menor a cero, dependiendo del valor del error se decide la dirección

(izquierda o derecha) del motor de cd. La dirección del motor se basa de acuerdo a la tabla 2.3.

Activar el PWM. El resultado de la suma de la parte proporcional y la derivativa va hacer el porcentaje con el que trabaje el PWM. El PWM del PIC16F887 es de 8 bits, por lo que su límite es de 2^8. La velocidad con que se mueva el motor dependerá del porcentaje del PWM.

3.4. Mediciones de desempeño.

Después de haber realizado la implementación física del controlador PD en el motor de cd se obtuvo la respuesta del desempeño, para esto se utilizó una tarjeta de adquisición de datos de National Instruments y el software de labview para adquirir la señal.

La figura 3.15 muestra la respuesta del controlador PD a una entrada escalón, la señal roja es la entrada escalón y la señal blanca es la respuesta del controlador.

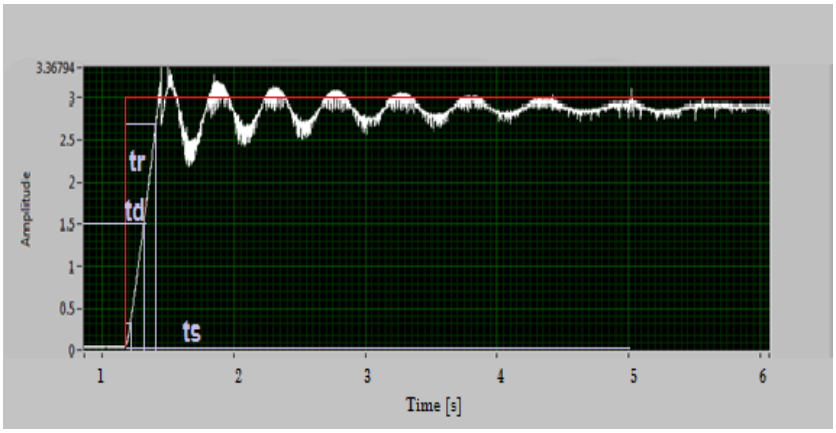


Figura 3.15. Respuesta del controlador PD con parámetros de kp=2 y kd = 1 implementado en el microprocesador PIC16F887.

Analizando la gráfica anterior se obtuvieron los siguientes resultados:

Sobrepaso máximo = 30 %

Tiempo de retardo = 0.2s

Tiempo de levantamiento = 0.5s

Tiempo de asentamiento = 5.5s.

Error en estado estable = 0.1

Frecuencia natural wn = 3.14

Factor de amortiguamiento ζ = 0.23

Comparando la simulación con la implementación física los resultados son similares para el sobrepaso máximo, tiempo de retardo y tiempo de levantamiento, sin embargo el tiempo de asentamiento varia de 3.5s a 5.5s y el error varia de 0 a 0.1 esto se debe a que en la implementación física contribuyen factores físicos que no fueron tomados en el modelo físico del sistema, pues solo se hizo una aproximación.

Otro factor que afecta los resultados de la simulación con la implementación física es el ruido, este ruido es producido por el motor, es por eso que en la simulación no se ve y en la implementación si porque cuando el controlador PD actúa sobre el motor este provoca ruido que se filtra en la adquisición de la señal.

3.4.1 Señales de prueba.

Existen casos en donde las entradas de un sistema de control pueden variar con respecto al tiempo. Esto provoca problemas para el diseñador, ya que es difícil obtener un sistema de control que satisfaga todas las formas posibles de señales de entrada. Para ello es necesario suponer diferentes tipos de señales de entrada de prueba para evaluar el desempeño del sistema de control.

Los diferentes tipos de señales de prueba para un controlador son:

- Función escalón.
- Función rampa.
- Función impulso.
- Función tren de pulsos.
- Función diente de sierra.
- Función seno.

El desempeño del controlador PD se evaluó en la simulación y en la implementación física con las señales de prueba: función tren de pulsos, función seno y función diente de sierra.

La respuesta a la función tren de pulsos en la simulación del controlador PD se muestra en la figura 3.16.

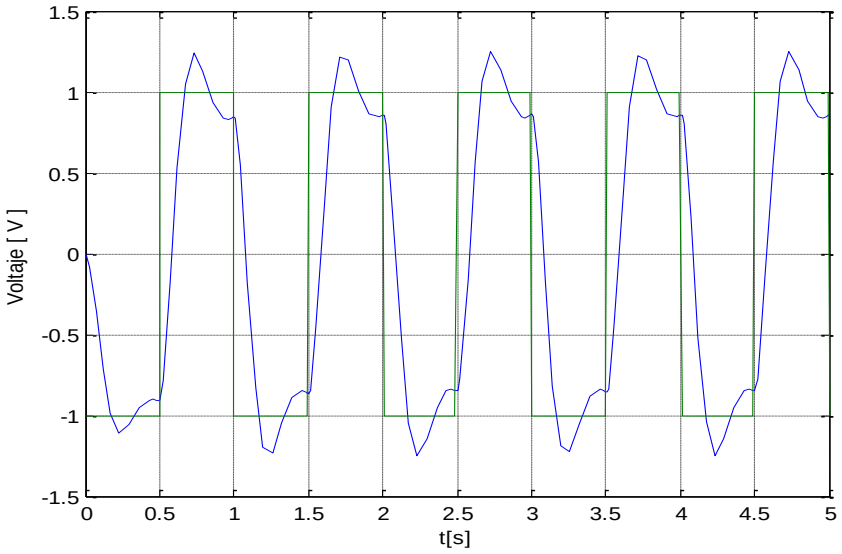


Figura 3.16. Función tren de pulsos como señal de prueba en la simulación del PD

El desempeño del controlador PD a una entrada seno se observa en ala figura 3.17.

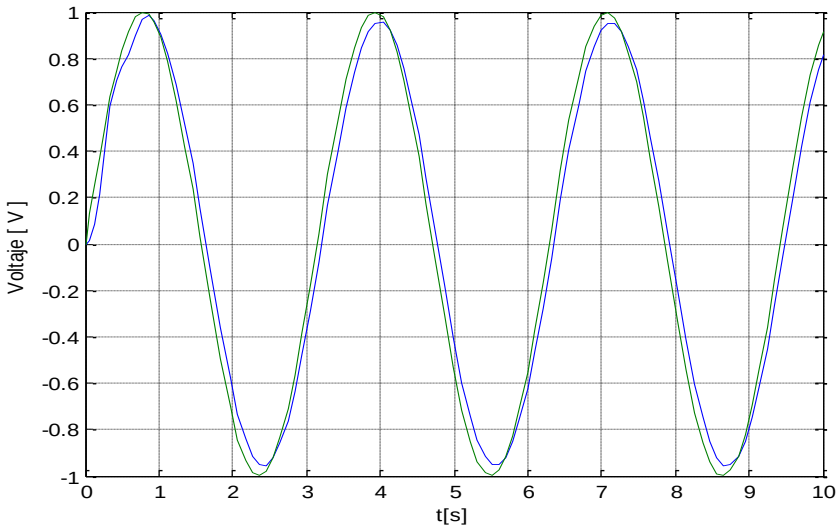


Figura 3.17. Función seno como señal de prueba en la simulación del controlador PD.

La respuesta a la diente de sierra en la simulación del controlador PD se muestra en la figura 3.18.

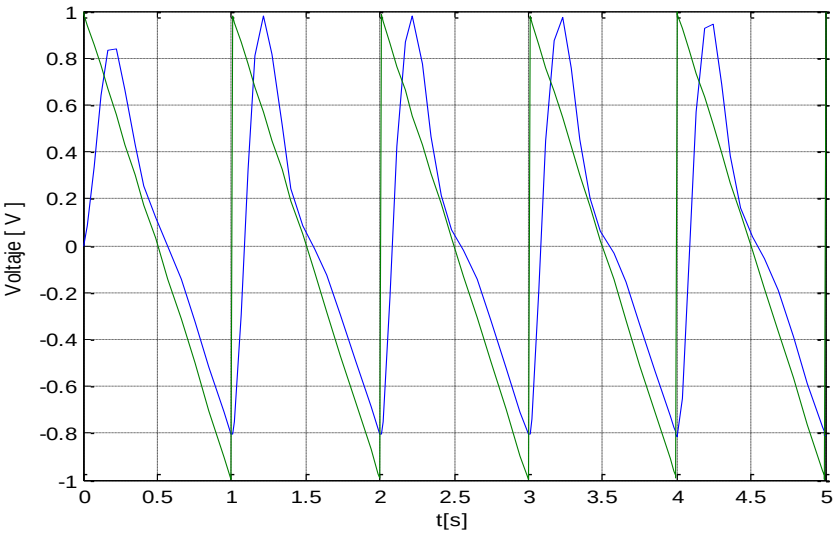


Figura 3.18. Función diente de sierra como señal de prueba en la simulación del controlador PD.

En las gráficas 3.16, 3.17 y 3.18 la señal de color verde es la función de prueba y la señal de color azul es la respuesta del controlador, como se puede observar en las tres gráficas de la simulación en simulink el desempeño del controlador PD es bueno para las tres señales ya que el controlador sigue la trayectoria de las tres señales.

Las siguientes gráficas 3.19, 3.20 y 3.21, muestran del desempeño del controlador PD en la implementación física para las funciones entrada tren de pulsos, seno y diente de sierra respectivamente, las gráficas se adquirieron con la tarjeta de adquisición de datos de National Instruments con el software de Labview.

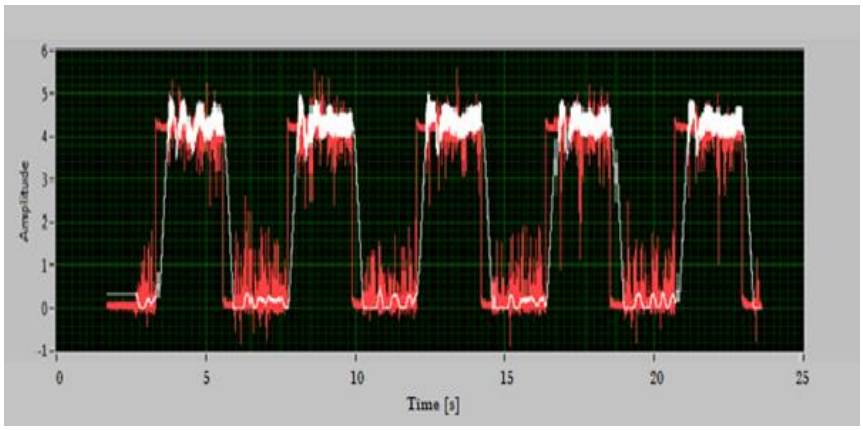


Figura 3.19. Función tren de pulsos como señal de prueba en la implementación física del controlador PD.

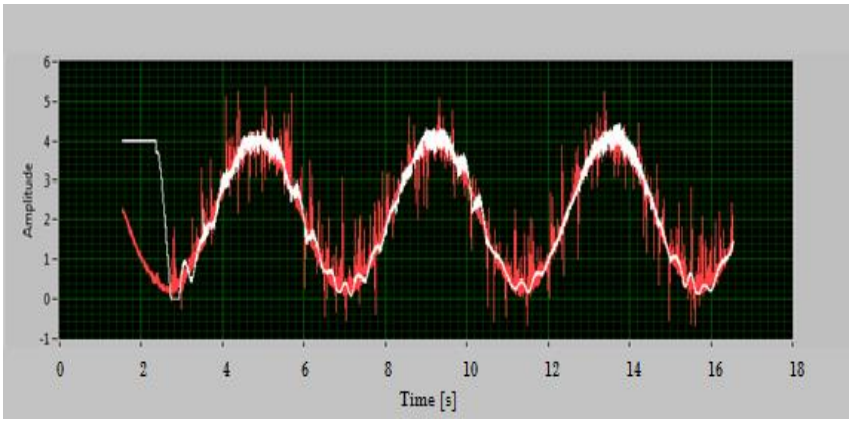


Figura 3.20. Función seno como señal de prueba en la implementación física del controlador PD.

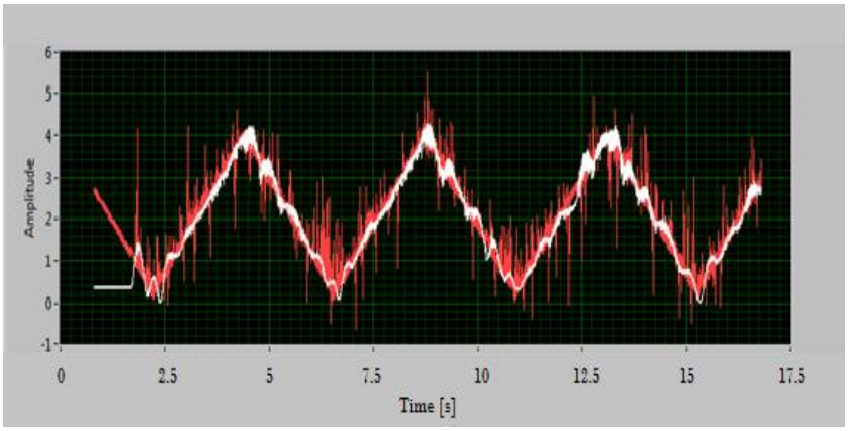


Figura 3.21. Función diente de sierra como señal de prueba en la implementación física del controlador PD.

En las gráficas 3.19, 3.20 y 3.21 la señal de color rojo es la función de prueba y la señal de color blanco es la respuesta del controlador, como se puede observar en las tres gráficas el desempeño del controlador PD es bueno para las tres señales, ya que el controlador sigue bien las tres señales.

CAPITULO 4.
Diseño del controlador PID difuso.

El control difuso se basa en el conocimiento de la lógica difusa, este control necesita del conocimiento del experto, con un buen conocimiento del experto se puede llegar a diseñar controladores difusos con un gran desempeño.

Debido a los buenos resultados de los controladores difusos y la variedad de aplicaciones del controlador PID convencional, existen reglas simples fundadas con la lógica difusa para la sintonización del controlador PID.

4.1 Lógica Difusa.

La creación de la lógica difusa surgió para emular el comportamiento humano, esta lógica se basa en modelar los cuantificadores del lenguaje humano como son: pequeño, grande, mucho, poco, arriba, abajo, etc. La lógica difusa funciona con nuestras expresiones cotidianas como ‘’hace mucho frío’’, ‘’es muy alto’’, ‘’es muy pequeño’’, etc.

La lógica difusa se utiliza en el control cuando el modelo matemático de un sistema no se conoce o es demasiado complejo, sin embargo la lógica difusa requiere del conocimiento profundo del funcionamiento del sistema que se desea controlar, que solo un experto puede tener.

La lógica difusa sirve para manipular información inexacta o demasiado compleja, es por ello que hace uso de la teoría de conjuntos que le facilita este proceso.

4.1.1. Teoría de conjuntos difusos.

Los conjuntos surgieron por la necesidad del hombre de clasificar. Un conjunto es una colección de objetos como: números, letras, colores, figuras, etc. Cada objeto se considera como un elemento del conjunto. Un conjunto está definido únicamente por sus elementos y por nada más, el orden en que se presentan sus elementos no importa.

Un conjunto difuso A en un universo de discurso U está definido por una función de pertenencia o membresía μ que toma valores entre 0 y 1, la transición entre 0 y 1 es gradual.

‘’Un conjunto difuso puede ser visto como una generalización de los conceptos de un conjunto ordinario cuya función de membresía solo toma dos valores [0,1]. De este modo un conjunto difuso A en U puede ser representado como un conjunto de pares ordenados de un elemento genérico u y el grado de su función de membresia’’ [7].

$$A = \{ (u,\mu) \,||u \in U\}$$

Donde: A es el conjunto difuso
U es el universo del discurso
μ es la función de membresía.
u es un elemento.

En la teoría de conjuntos difusos se definen también las operaciones unión e intersección.

- Unión. La unión de dos conjuntos difusos A y B, los cuales están definidos sobre el universo de discurso U, es un conjunto difuso denotado como AUB, con función de membresía definida por:

$$\mu AUB(u) = \max (\mu A(u), \mu B(u))$$

- Intersección. La intersección de dos conjuntos difusos A y B, los cuales están definidos sobre el universo de discurso U, es un conjunto difuso denotado como A∩B, con función de membresía definida por:

$$\mu A \cap B(u) = \min (\mu A(u), \mu B(u) .$$

4.1.2. Funciones de membresía.

Para cada conjunto difuso, existe asociada una función de membresía para sus elementos, la función de membresía indica en qué grado el elemento forma parte de ese conjunto difuso. Las funciones de membresía pueden tener una forma arbitraría; sin embargo solo utilizaremos la triangular y la trapezoidal porque son fáciles de implementar y entregan buenos resultados en el control.

Función triangular. Es una función definida por tres parámetros (α, β, γ) que describen un triángulo (Figura 4.1). El primer y tercer parámetros determinan la base del triángulo, mientras que el segundo parámetro determina la variable con grado de membresía 1.

$$\mu(u) = \begin{cases} 0 & u < \alpha \\ (u - \alpha) / (\beta - \alpha) & \alpha \leq u \leq \beta \\ (\alpha - u) / (\beta - \alpha) & \beta \leq u \leq \gamma \\ 0 & u > \gamma \end{cases}$$

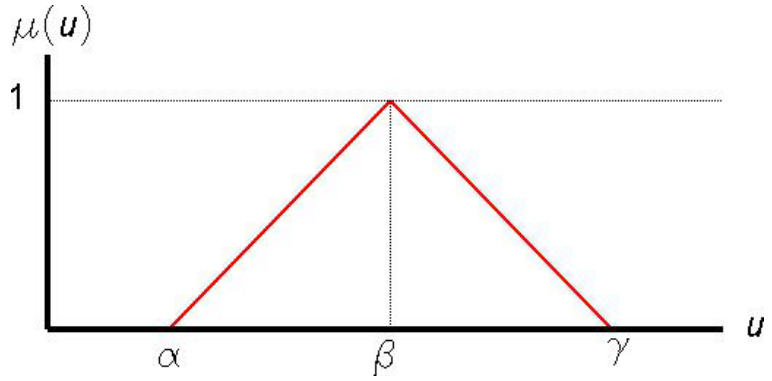


Figura.4.1. Función Triangular.

Función Trapezoidal. Es una función seccionada definida por cuatro parámetros (α, β, γ, δ) que describen un trapecio (figura 4.2). El primer y cuarto parámetros definen la base del trapecio, mientras que el segundo y tercer parámetros definen la región donde la variable posee grado de membresía 1.

$$\mu(u) = \begin{cases} 0 & u < \alpha \\ (u - \alpha) / (\beta - \alpha) & \alpha \leq u < \beta \\ 1 & \beta \leq u \leq \gamma \\ (\delta - u) / (\delta - \gamma) & \gamma < u \leq \delta \\ 0 & u > \delta \end{cases}$$

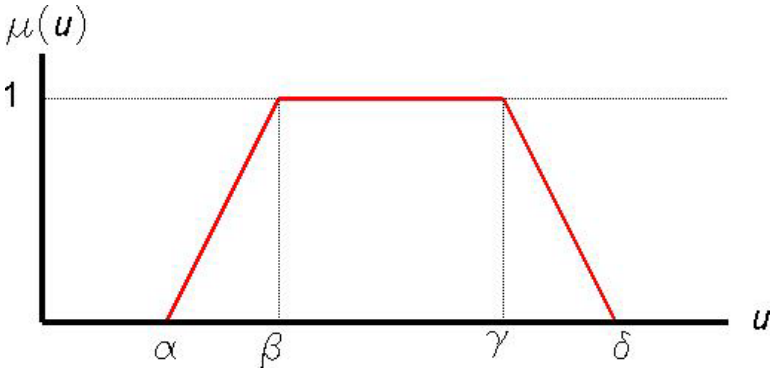


Figura.4.2. Función Trapezoidal.

4.2. Configuración del controlador difuso.

El control difuso utiliza los conocimientos del experto para generar una base de conocimientos que le permitirán al sistema tomar decisiones sobre diferentes acciones que se presenten durante su funcionamiento. El control difuso puede aplicarse a sistemas sencillos así como a sistemas en donde su modelo matemático es muy complejo.

Existen dos tipos de controladores difusos: el tipo Mamdani, en donde las entradas como las salidas son conjuntos difusos y el tipo Sugeno, en el cual las entradas son conjuntos difusos y las salidas son una función relacionada con las variables del proceso.

La estructura de un controlador difuso está dado por: el proceso de fuzzificación, el mecanismo de inferencia y el proceso de defuzzificación.

Proceso de fuzzificación. Este proceso consiste en convertir los valores reales (cuantitativos) a valores difusos (cualitativos). En la fuzzificación se definen los conjuntos difusos y se les asocian funciones de pertenencia a cada conjunto difuso, esto con el objetivo de asignar grados de pertenencia a cada variable de entrada.

Mecanismo de inferencia. El mecanismo de inferencia relaciona los conjuntos de entrada y de salida, para establecer las reglas que definirán al sistema. El mecanismo de inferencia hace uso de los operadores difusos de unión e intersección cuando el sistema tiene más de dos entradas.

Proceso de defuzzificación. El proceso de defuzzificación es lo contrario del proceso de fuzzificación, convierte los valores difusos a valores reales. Para realizar este proceso se utilizan métodos matemáticos simples como son: singletons, centroide y promedio de centro de áreas.

1. Singletons. Consiste en suplir el área que comprende el conjunto difuso y centrarlo en un solo punto, es decir el proceso de defuzzificación se convierte ahora en el cálculo de una media ponderada.

$$u^* = \frac{\sum_{i=1}^{k} Biui}{\sum_{i=1}^{k} Bi} \qquad (4.1)$$

Donde:

Bi es el grado de membresía de las entradas.

ui es el valor de cada función de membresía, en donde el grado de membresía es el máximo (1) de la variable salida.

2. Centroide. ‘’Consiste en determinar el centro de gravedad de todas las áreas generadas por el mecanismo de inferencia difusa’’ [10].

$$u^* = \frac{\int u * \mu(udu)}{\int \mu(u)du} \qquad (4.2)$$

3. Promedio del centro de áreas. Calcula la contribución de cada área en los conjuntos de salida, sin importar los traspales que pudieran existir entre los conjuntos adyacentes.

$$u = \frac{\int u * \sum_{k=1}^{n} \mu(u)du}{\int \sum_{k=1}^{n} \mu(u)du} \qquad (4.3)$$

4.2.1. Proceso de fuzzificación del controlador difuso.

Para el control de posición del motor el tipo de controlador difuso seleccionado fue el Mamdani y el proceso de fuzzificación es el siguiente:

Primeramente se deben establecer los conjuntos difusos. Estos conjuntos se definieron con el conocimiento adquirido en el diseño del controlador PD digital presentado en el capítulo tres.
Para la variable de entrada (error), se establecieron cinco conjuntos difusos para definir el tamaño: error mínimo (emin), error regular (ereg), error mediano (emed), error alto (ealt) y error máximo (emax).
Posteriormente para la variable de salida (PWM), se asignaron cinco conjuntos difusos para definir el porcentaje: salida mínima (smin), salida regular (sreg), salida mediana (smed), salida alta (salta) y salida máxima (smax).

Los conjuntos difusos se definieron tomando el valor absoluto de las dos variables, tanto para la variable de entrada como para la variable de salida, esto con el objetivo de disminuir el número de conjuntos y facilitar el diseño.

En segundo lugar se definieron las funciones de membresía para la variable de entrada (error) y para la variable de salida (PWM), las unidades de medición están en volts y en bits (figura 4.3), esto con el propósito de ver la magnitud de la desviación que existe entre el valor deseado y el de referencia tanto en voltaje como en bits. Sin embargo como las funciones de membresía se programaron en un microcontrolador que tiene un convertidor analógico digital de 10 bits es conveniente basarse en las funciones de membresía con unidades en bits.

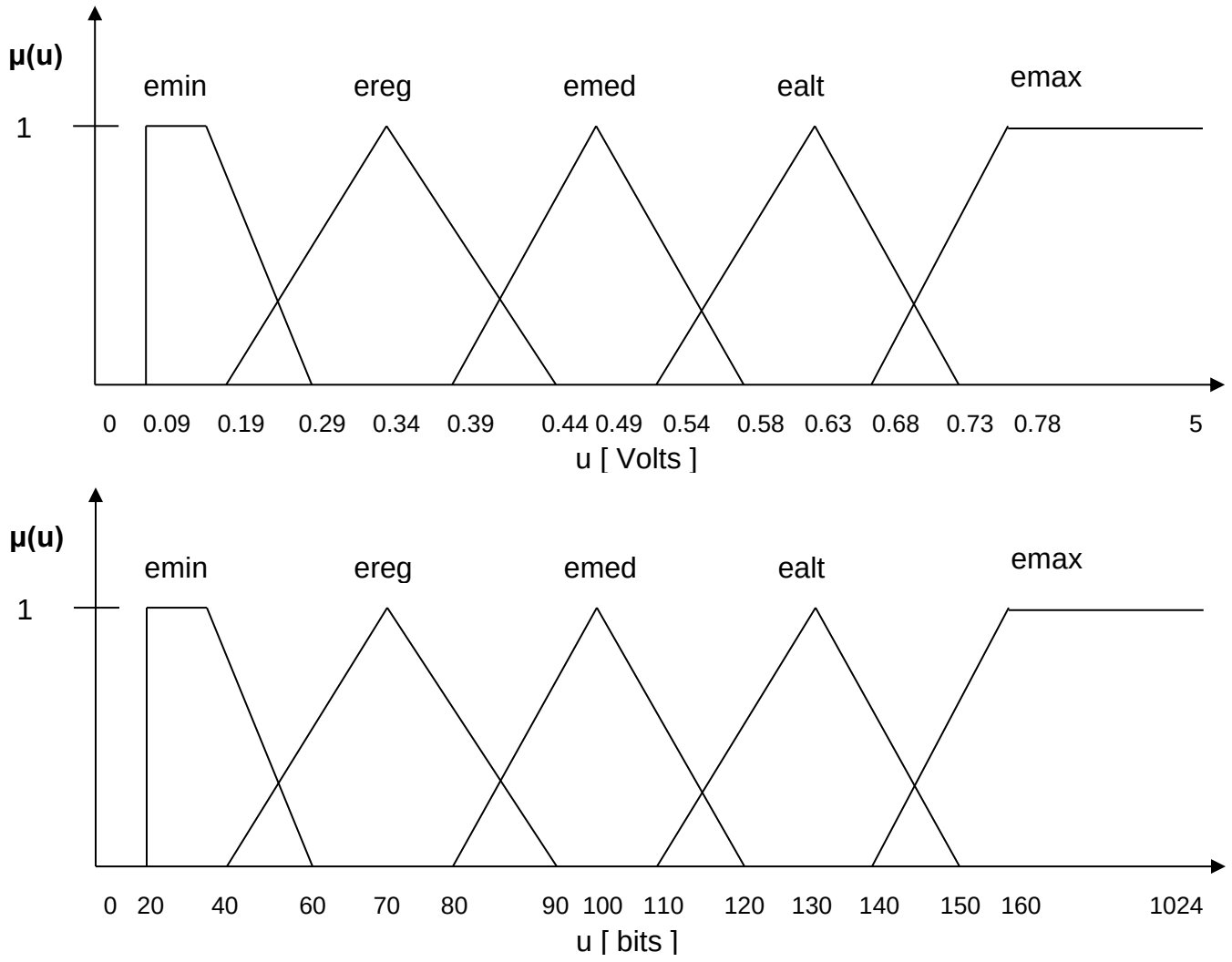


Figura 4.3. Funciones de membresía para la variable de entrada (error).

- Función de membresía para el conjunto emin.

$$\mu(u) = \begin{cases} 0 & \text{si } u < 20 \\ 1 & \text{si } 20 \leq u \leq 40 \\ -0.05\,u + 3 & \text{si } 40 < u \leq 60 \end{cases}$$

- Función de membresía para el conjunto ereg.

$$\mu(u) = \begin{cases} 0.033\,u - 1.33 & \text{si } 40 < u < 70 \\ -0.05\,u + 4.5 & \text{si } 70 \leq u \leq 90 \end{cases}$$

- Función de membresía para el conjunto emed.

$$\mu(u) = \begin{cases} 0.05\,u - 4 & \text{si } 80 \leq u \leq 100 \\ -0.05\,u + 6 & \text{si } 100 < u < 120 \end{cases}$$

- Función de membresía para el conjunto ealt.

$$\mu(u) = \begin{cases} 0.05\,u - 5.5 & \text{si } 110 \leq u \leq 130 \\ -0.05\,u + 7.5 & \text{si } 130 < u < 150 \end{cases}$$

- Función de membresía para el conjunto emax.

$$\mu(u) = \begin{cases} 0.05\,u - 7 & \text{si } 140 \leq u \leq 160 \\ 1 & \text{si } u > 160 \end{cases}$$

En la figura 4.4 se observan las funciones de membresía para la variable de salida.

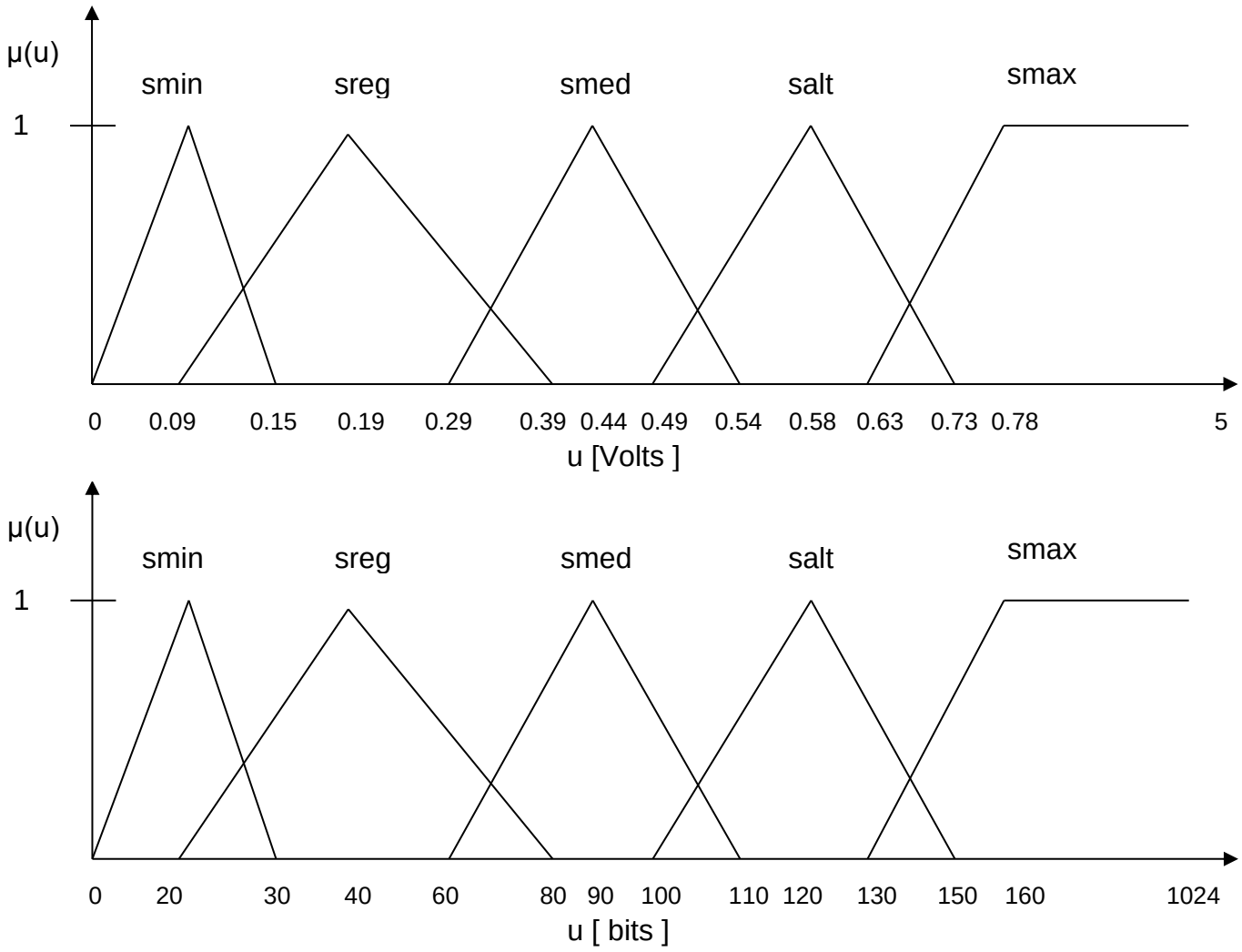


Figura 4.4. Funciones de membresía para la variable de salida (PWM).

- Función de membresía para el conjunto smin.

$$\mu(u) = \begin{cases} 0.05\,u & \text{si } 0 \leq u \leq 20 \\ -0.1\,u + 3 & \text{si } 20 < u < 30 \end{cases}$$

- Función de membresía para el conjunto sreg.

$$\mu(u) = \begin{cases} 0.05\,u - 1 & \text{si } 20 \leq u \leq 40 \\ -0.025\,u + 2 & \text{si } 40 < u < 80 \end{cases}$$

- Función de membresía para el conjunto smed.

$$\mu(u) = \begin{cases} 0.033\,u - 2 & \text{si } 60 < u \leq 90 \\ -0.05\,u + 5.5 & \text{si } 90 < u < 110 \end{cases}$$

- Función de membresía para el conjunto salt.

$$\mu(u) = \begin{cases} 0.05\,u - 5 & \text{si } 100 < u \leq 120 \\ -0.033\,u + 5 & \text{si } 120 < u < 150 \end{cases}$$

- Función de membresía para el conjunto smax.

$$\mu(u) = \begin{cases} 0.033\,u - 4.33 & \text{si } 130 < u \leq 160 \\ 1 & \text{si } u > 160 \end{cases}$$

4.2.2. Mecanismo de inferencia difusa del controlador difuso

Como el sistema de posición solo tiene una entrada, la base de reglas para el control es la siguiente:

1. Si el error es mínimo entonces el porcentaje de salida será mínimo.
2. Si el error es regular entonces el porcentaje de salida será regular.
3. Si el error es mediano entonces el porcentaje de salida será mediano.
4. Si el error es alto entonces el porcentaje de salida será alto.
5. Si el error es máximo entonces el porcentaje de salida será máximo.

4.2.3. Proceso de defuzzificación del controlador difuso.

El proceso de defuzzificación para el control de posición del motor de cd se realizó utilizando el método matemático de Singletons por ser el más sencillo de programar. Este proceso se ejecutó en lenguaje C junto el controlador difuso.

4.3. Realización del controlador difuso.

Una vez configurado el controlador difuso, el siguiente paso es hacer la implementación que comienza realizando el diagrama de flujo y el algoritmo y para programar el controlador difuso.
El diagrama de flujo del controlador difuso es el siguiente:

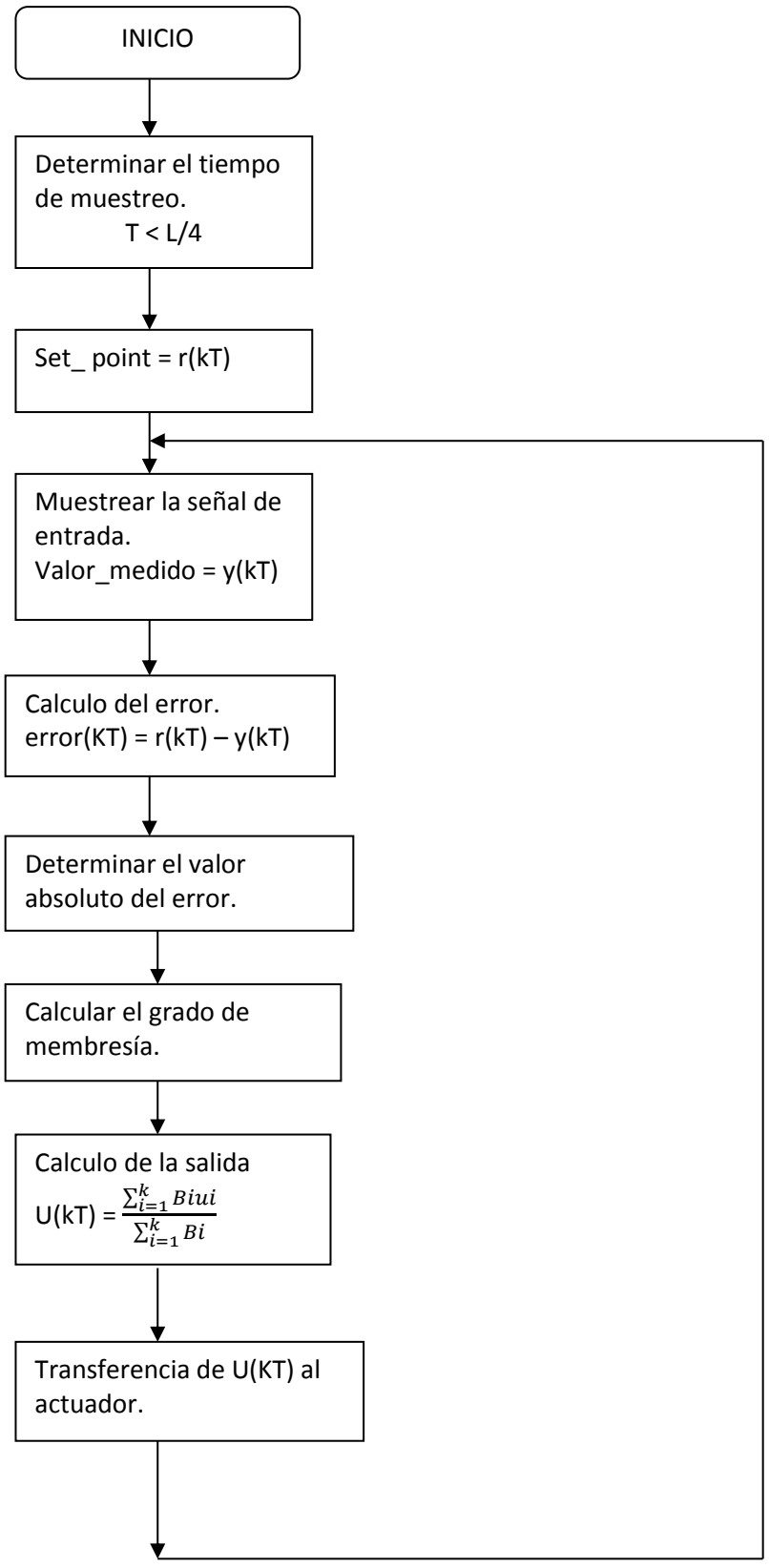


Figura 4.5. Diagrama de flujo del controlador difuso.

El algoritmo del controlador difuso es:

1. Calcular el error.
2. Verificar si el error es positivo o negativo.
3. Determinar el valor absoluto del error.
4. Calcular el grado de membresía dependiendo de cada función de membresía, correspondiente a cada conjunto difuso de la variable error.
5. Calcular la salida con el método matemático de Singletons (ecuación 4.1).

4.4. Configuración del controlador PID difuso.

Debido a que la ecuación del controlador PID es lineal y la ecuación de algunos robots es no lineal, debe tenerse mucho cuidado en compensar las no linealidades del controlador PID, por lo cual se utiliza la lógica difusa para ajustar los parámetros del controlador PID (Kp, Kd, y Ki) para compensar las no linealidades.

4.4.1. Proceso de fuzzificación.

Con la finalidad de simplificar el ajuste de los parámetros del controlador PID se eligió el controlador difuso del tipo Mamdani.

La sintonización de los parámetros del controlador PID necesita dos variables de entrada que son: el error y el cambio de error. Con el objetivo de disminuir el número de conjuntos se tomara el valor absoluto de las dos variables de entrada.

Para las dos variables error y cambio de error, se definirán los mismos cinco conjuntos difusos: mínimo (min), regular (reg), mediano (med), alto (alto) y máximo (max).

Las funciones de membresía se muestran en la figura 4.6.

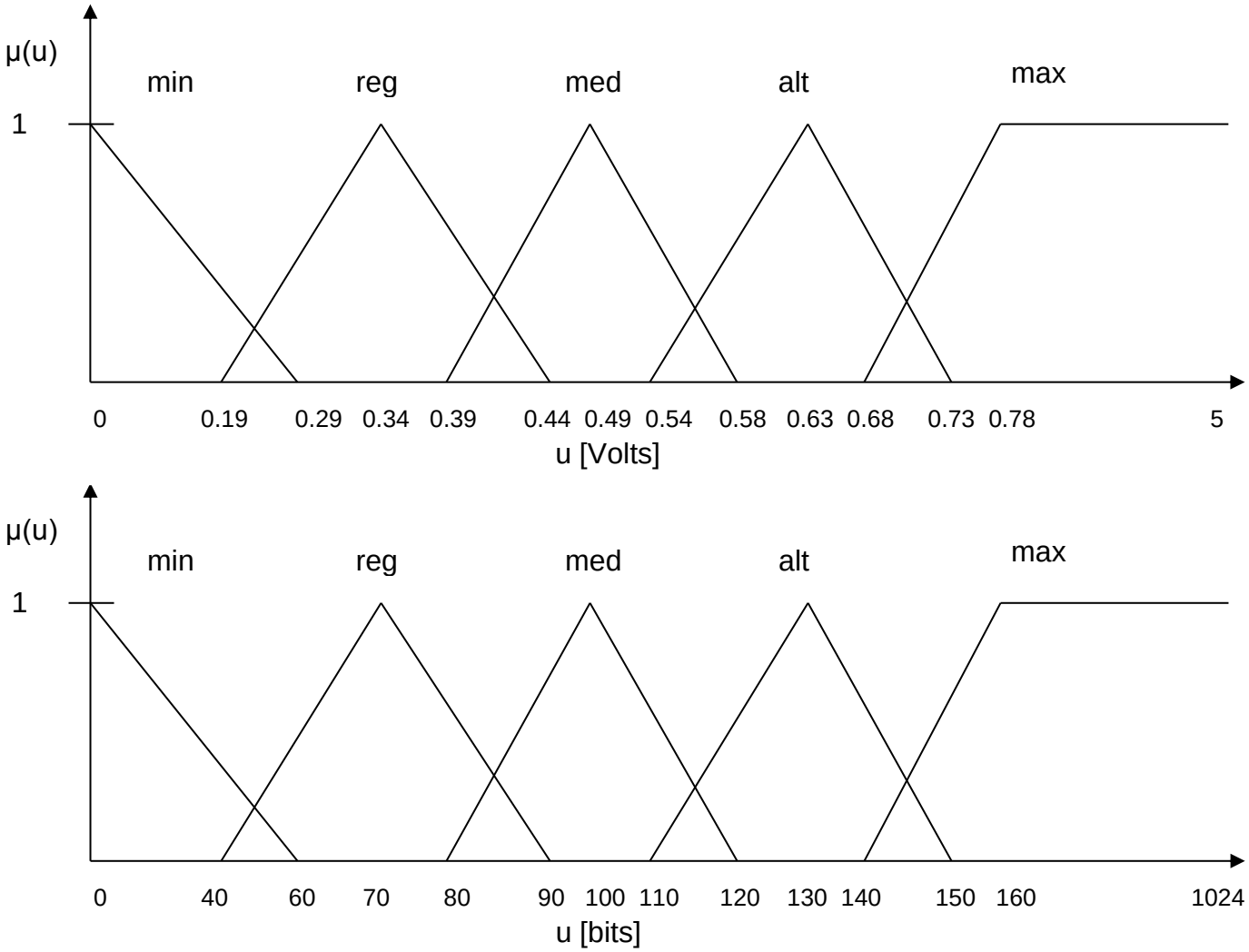


Figura. 4.6. Funciones de membresía para las variables error y cambio de error.

- Función de membresía para el conjunto mínimo.

$$\mu(u) = -0.0166u + 1 \qquad \text{si } 0 \leq u < 60$$

- Función de membresía para el conjunto regular.

$$\mu(u) = \begin{cases} 0.033\,u - 1.33 & \text{si } 40 < u < 70 \\ -0.05\,u + 4.5 & \text{si } 70 \leq u \leq 90 \end{cases}$$

- Función de membresía para el conjunto mediano.

$$\mu(u) = \begin{cases} 0.05\,u - 4 & \text{si } 80 \leq u \leq 100 \\ -0.05\,u + 6 & \text{si } 100 < u < 120 \end{cases}$$

- Función de membresía para el conjunto alto.

$$\mu(u) = \begin{cases} 0.05\,u - 5.5 & \text{si } 110 \leq u \leq 130 \\ -0.05\,u + 7.5 & \text{si } 130 < u < 150 \end{cases}$$

- Función de membresía para el conjunto máximo.

$$\mu(u) = \begin{cases} 0.05\,u - 7 & \text{si } 140 \leq u \leq 160 \\ 1 & \text{si } u > 160 \end{cases}$$

El grado de membresía se calculara más adelante cuando se programe el controlador PID difuso.

4.4.2. Mecanismo de inferencia difusa.

La base de reglas para la sintonización del controlador PID se basa en el artículo de Essam Nats y Khalid A. Buragga [11]. Estos autores determinan la base de reglas de acuerdo a la respuesta del sistema, la cual la dividen en cinco zonas dependiendo del valor del error (e) y del cambio de error (ce). Figura 4.7.

1. Zona1 (z1): e > 0 y ce < 0.
2. Zona2 (z2): e < 0 y ce < 0.
3. Zona3 (z3): e < 0 y ce > 0.
4. Zona4 (z4): e > 0 y ce > 0.
5. Zona5 (z5): e ~ 0 y ce ~ 0.

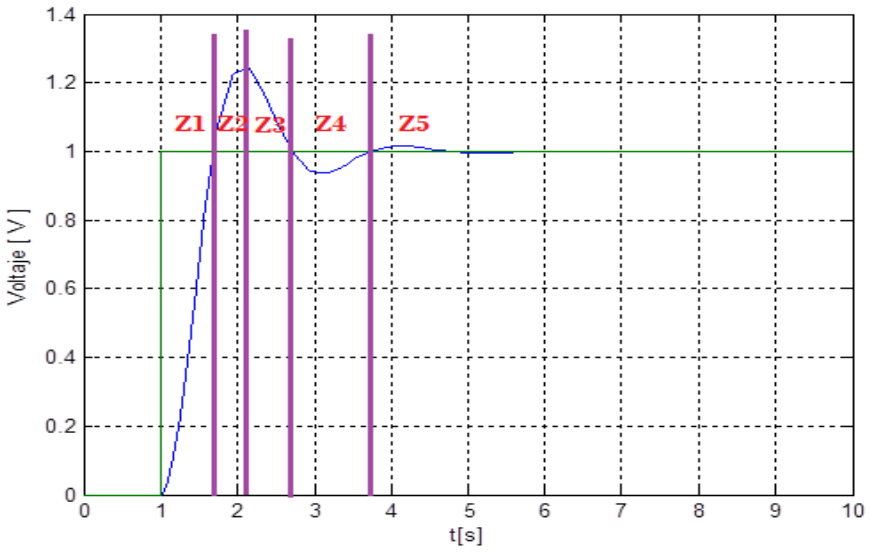


Figura. 4.7. Respuesta del controlador PD con parámetros Kp = 2 y Kd = 1, dividida en cinco zonas de acuerdo a la magnitud del error y del cambio de error.

NOTA: Es importante señalar que el controlador PID difuso para controlar la posición del motor cd, se redujo a un controlador PD difuso el motivo de esta reducción se explicó en el capítulo dos.

En las zonas 1 y 3 el error se autocorrige y la variable de control se mantiene en su configuración actual, en consecuencia en estas zonas, no es necesario modificar los parámetros del controlador PD.
En las zonas 2 y 4 el error no se autocorrige y la variable de salida depende de los cambios en las magnitudes del error y del cambio de error, de modo que en estas zonas es necesario ajustar los parámetros del controlador PD.
Por último en la zona 5 el error se aproxima a cero y la variable de salida también es casi cero, por tanto no es necesario sintonizar los parámetros del controlador PD.

Considerando lo anterior la base de reglas para la sintonización del controlador PD es de la siguiente manera:

- Si el error > 0 y ce < 0 no se ajustaran los parámetros del controlador.
- Si el error < 0 y ce < 0 si se ajustaran los parámetros del controlador.
- Si el error < 0 y ce > 0 no se ajustaran los parámetros del controlador.
- Si el error > 0 y ce > 0 si se ajustaran los parámetros del controlador.
- Si el error ~ 0 y ce ~ 0 no se ajustaran los parámetros del controlador.

Como el sistema de sintonización tiene dos entradas se usó el operador intersección que se mencionó anteriormente.

4.4.3. Proceso de defuzzificación.

De acuerdo con [12], la defuzzificación para la sintonización de los parámetros del controlador PID usando operaciones difusas es la siguiente:

$$K_{Fp} = \frac{\sum_{j=1}^{m} Kp\mu j(e)}{\sum_{j=1}^{m} \min(\mu A(u),\mu B(u))} \quad (4.4.)$$

$$K_{Fi} = \frac{\sum_{j=1}^{m} Ki\mu j(\int e dt)}{\sum_{j=1}^{m} \min(\mu A(u),\mu B(u))} \quad (4.5)$$

$$K_{Fd} = \frac{\sum_{j=1}^{m} Kd\mu j(ce)}{\sum_{j=1}^{m} \min(\mu A(u),\mu B(u))} \quad (4.6)$$

Donde:

Kd, Ki y Kd son los valores iniciales de los parámetros del controlador PID.
K_{Fp}, K_{Fi} y K_{Fd} serán los valores que irán cambiando de acuerdo a cada operación difusa.

Sustituyendo los parámetros K_{Fp}, K_{Fi} y K_{Fd} en la ecuación 3.6 la salida del controlador PID difuso es la siguiente:

$$U(kT) = K_{Fp}\, e(kT) + K_{Fi} \sum_{k=1}^{n} T\, e\,(kT) + K_{Fd}\, \frac{e(kT) - e(kT-T)}{T} \quad (4.7)$$

4.5. Realización del controlador PD difuso.

Considerando la configuración anterior el algoritmo para programar el controlador PD difuso es el siguiente:

1. Calcular el error.
2. Calcular el cambio de error.
3. Verificar si el error es positivo o negativo.
4. Verificar si e > 0 y ce < 0.
5. Si e > 0 y ce < 0 entonces $K_Fp=Kp$ y $K_Fd= Kd$.
6. Verificar si e < 0 y ce < 0.
7. Si e < 0 y ce < 0, determinar el valor absoluto de e y ce, calcular K_Fp y K_Fd de acuerdo a las ecuaciones 4.4 y 4.6 respectivamente.
8. Verificar si e < 0 y ce > 0.
9. Si e < 0 y ce > 0 entonces $K_Fp=Kp$ y $K_Fd= Kd$.
10. Verificar si e > 0 y ce > 0.
11. Si e > 0 y ce > 0, determinar el valor absoluto de e y ce, calcular K_Fp y K_Fd de acuerdo a las ecuaciones 4.4 y 4.6 respectivamente.
12. Verificar si e ~ 0 y ce ~ 0.
13. Si e ~ 0 y ce ~ 0 entonces $K_Fp=Kp$ y $K_Fd= Kd$.
14. Calcular la parte proporcional.
15. Calcular la parte derivativa.
16. Sumar la parte proporcional y derivativa.
17. Activar el PWM.

Siguiendo el algoritmo anterior el diagrama de flujo es el siguiente:

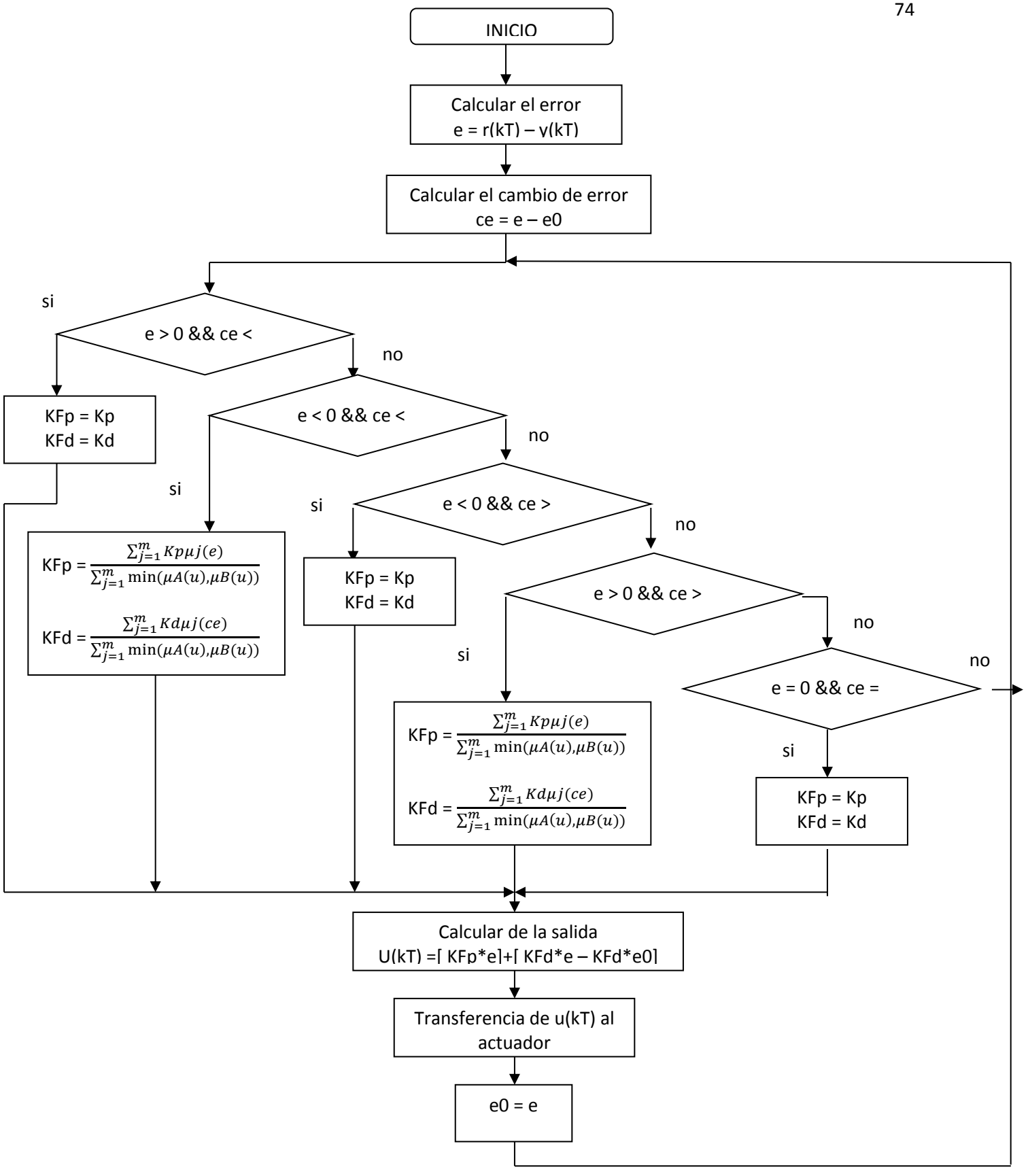


Figura. 4.8. Diagrama de flujo del controlador PD difuso.

4.6. Mediciones de desempeño.

Una vez que se hizo la implementación física del controlador difuso y del controlador PD difuso se evaluará su desempeño con las siguientes señales de prueba: función tren de pulsos, función seno y función diente de sierra.

4.6.1. Desempeño del controlador difuso.

Las siguientes gráficas muestran el desempeño del controlador difuso, las gráficas se adquirieron con la tarjeta de adquisición de datos de National Instruments con el software de Labview.

La señal de color blanco es la función de prueba y la señal de color roja es la respuesta del controlador.

El desempeño del controlador difuso se muestra en las figuras 4.9, 4.10 y 4.11 para las funciones de entrada tren de pulsos, seno y diente de sierra respectivamente.

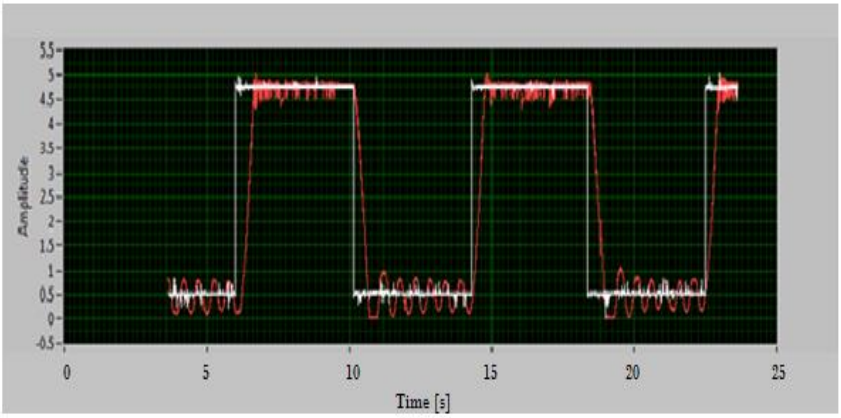


Figura 4.9. Respuesta del controlador difuso a una función tren de pulsos.

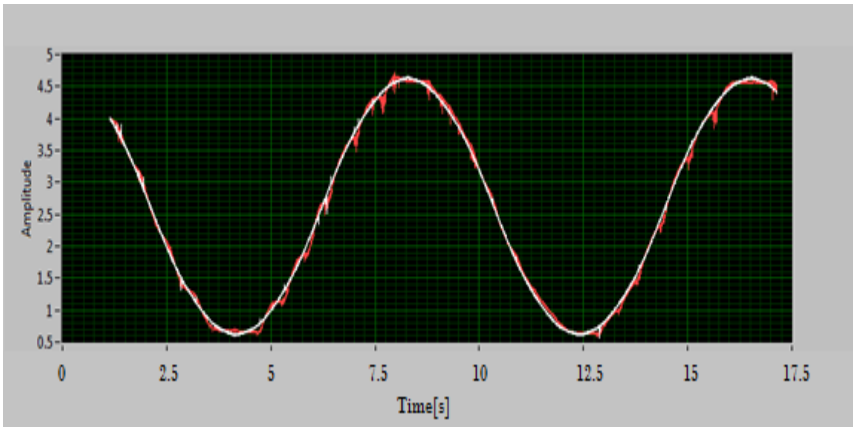


Figura 4.10. Respuesta del controlador difuso a una función seno.

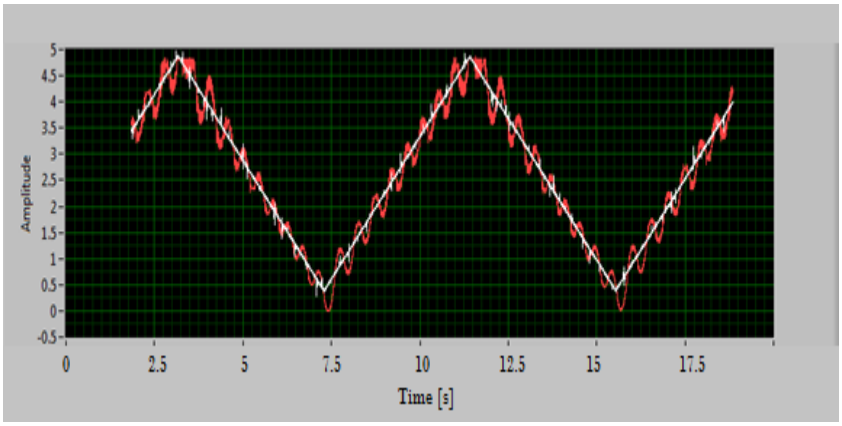


Figura 4.11. Respuesta del controlador difuso a una función diente de sierra.

Como se puede observar en las gráficas anteriores el controlador difuso responde mejor para la señal seno, esto debido a que la señal seno es una función que no tiene picos. Los picos provocan cambios bruscos en la corriente del motor ocasionando que el error sea mayor, por lo tanto la respuesta del controlador se hace más lenta.

4.6.2. Desempeño del controlador PD difuso.

El desempeño del controlador PD difuso para una función de entrada tren de pulsos se muestra en la figura 4.12, para una función seno se observa en la figura 4.13 y por último para una función diente de sierra de muestra en la figura 4.14 .

La señal de color blanco es la función de prueba y la señal de color roja es la respuesta del controlador.

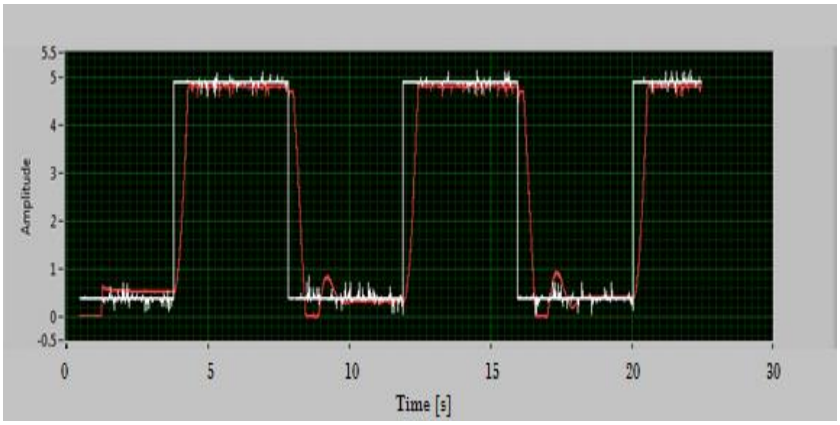


Figura 4.12. Respuesta del controlador PD difuso a una función tren de pulsos.

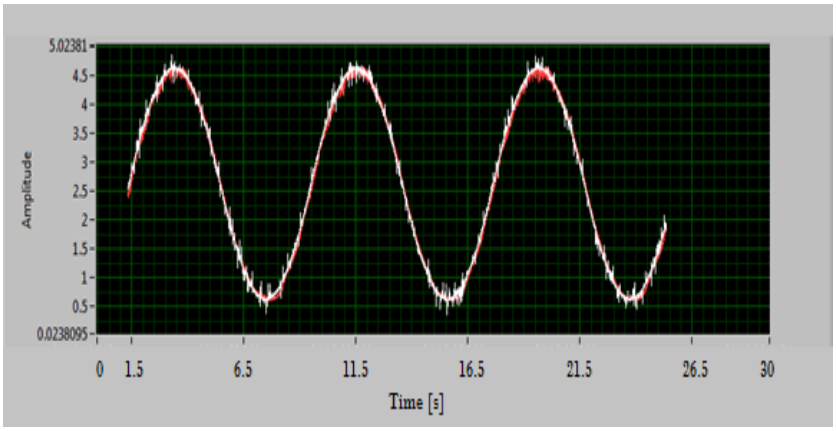


Figura 4.13. Respuesta del controlador PD difuso a una función seno.

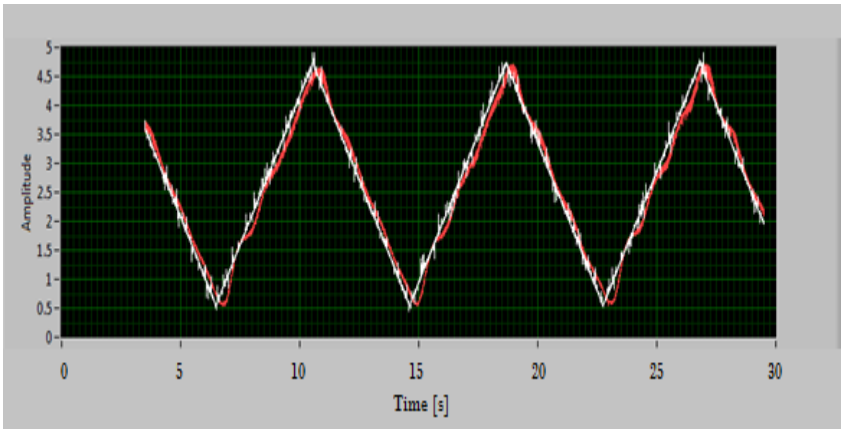


Figura 4.14. Respuesta del controlador PD difuso a una función diente de sierra.

Observando las tres graficas es evidente que el controlador PD difuso responde mejor las tres señales, esto es porque el controlador responde a los cambios no lineales, debido que este controlador tiene propiedades del PID convencional y de la lógica difusa.

CAPITULO 5.
Diseño del controlador PID neurodifuso.

Como ya se mencionó en el capítulo anterior existen reglas para la sintonización del controlador PID y una de estas reglas se basa en la lógica difusa, como se realizó en el capítulo cuatro; sin embargo en este capítulo se utilizó el conocimiento de redes neuronales artificiales para determinar los tres procesos (fuzzificación, mecanismo de inferencia y defuzzificación) del controlador PD difuso.

5.1. Redes de neuronas artificiales.

Uno de los retos de la ciencia y la tecnología es crear sistemas inteligentes y un ejemplo de un sistema inteligente son las redes de neuronas artificiales. Las redes de neuronas artificiales se basan en el funcionamiento de las redes neuronales de los animales y del hombre.

5.1.1. Fundamentos biológicos de las redes neuronales.

El sistema de comunicación neuronal consiste en tres fases:

- Los receptores que se encargan de acumular información del medio ambiente o del interior del organismo en forma de estímulos.
- El sistema nervioso recibe la información acumulada por los receptores, la almacena y la envía a los músculos y glándulas.
- Los músculos y glándulas reciben la información y la interpretan en forma de acciones motoras.

El elemento principal del sistema de comunicación neuronal es la neurona, la neurona es la encargada de transmitir la información, esta información se envía entre las diferentes neuronas, a través de prolongaciones, formando redes. En las redes se crea información y se almacena.

‘’La misión de las neuronas comprende generalmente cinco funciones parciales:

1. Las neuronas recogen la información que llega a ellas en forma de impulsos procedentes de otras neuronas o de receptores.

2. La integran en un código de activación propio de la célula.

3. La transmiten codificada en forma de frecuencia de impulsos a través de su axón.

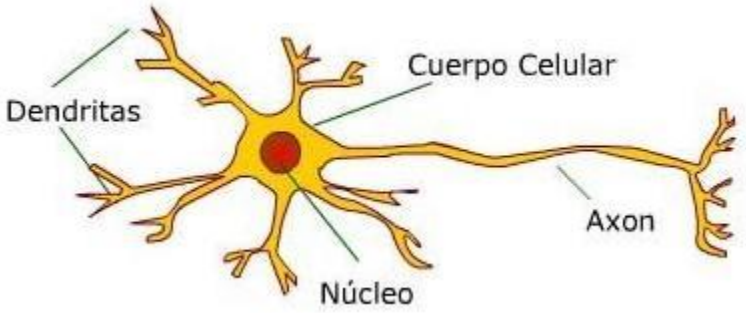
4. A través de sus ramificaciones el axón efectúa la distribución espacial de los mensajes.

5. En sus terminales transmite los impulsos a las neuronas subsiguientes o a las células efectoras” [8].

La neurona o célula nerviosa está compuesta por cinco elementos que son: el cuerpo celular, el núcleo, el axón, las dendritas y la sinapsis. Estos elementos se muestran en la figura 5.1

- Cuerpo celular. El cuerpo celular clasifica y organiza los impulsos que entran y salen de la neurona.
- Núcleo. Es el encargado de recibir la información procesarla y enviarla por las ramificaciones de salida.
- Axón. Es una ramificación de salida de la neurona, en donde se propagan un conjunto de impulsos electro-químicos.

- Dendritas. Son ramificaciones de entrada, donde se propaga la señal al interior de la neurona.



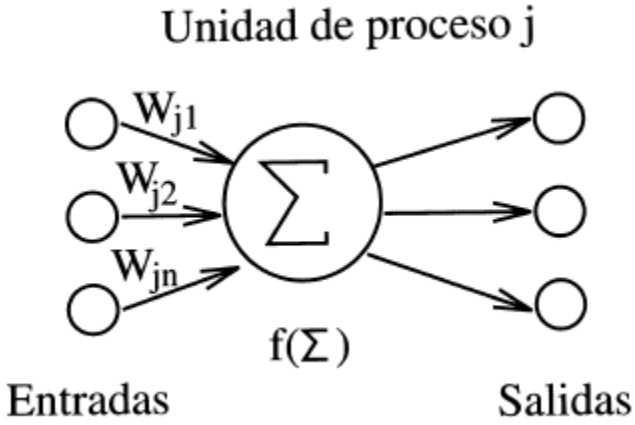
Fuente: http://www.educarchile.cl/ech/pro/app/detalle?ID=137486

Figura 5.1. Descripción de una célula nerviosa.

5.1.2 La neurona artificial.

‘’La neurona artificial, célula o autómata, es un elemento que posee un estado interno, llamado nivel de activación, y recibe señales que le permiten, en su caso, cambiar de estado’’ [8].

La neurona artificial cuenta con entradas, salidas y una unidad de proceso como se muestra en la figura 5.2.
Dependiendo de las entradas que reciba la neurona artificial es como su nivel de activación cambia, la neurona posee una función de transición de estado o función de activación que determina el nivel de activación.



Fuente: http://futurointeligente.wordpress.com/2011/01/04/redes-neuronales-artificiales-i-introduccion/

Figura.5.2. Esquema de una neurona artificial.

Para calcular el estado de activación de la neurona primero se calcula la entrada total, la entrada total se calcula como la suma de todas las entradas por unos ciertos valores llamados pesos, como se muestra en la ecuación 5.1.

$$E = x1*w1 + x2*w2 + \ldots\ldots\ldots + xn*wn. \qquad (5.1)$$

Donde: E es la entrada total.
x1…xn son las diferentes entradas.
w1...wn son los distintos pesos.

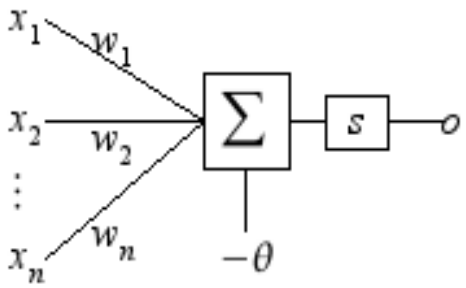
Después de calcular la entrada total (E), está es procesada por la función de activación, con el fin de determinar la salida de la neurona. La función de activación dependerá del tipo de modelo neuronal que se desea utilizar.

5.1.3. Primeros modelos de neuronas artificiales.

Los primeros modelos de neuronas artificiales fueron: la neurona McCulloch-Pitts, la neurona Perceptron y la neurona Adaline.

5.1.3.1. Neuronas McCulloch-Pitts.

‘’Este primer modelo de neurona fue propuesto por Warren McCulloch y Walter Pitts en 1943 en el artículo *A Logical Calculus of the Ideas Immanent in Nervous Activity* ‘’ [8]. La estructura de este modelo se muestra en la figura 5.3.



Fuente: http://es.wikipedia.org/wiki/Neurona_de_McCulloch-Pitts

Figura.5.3. Estructura de una neurona McCulloch-Pitts.

Este modelo es una estructura simple de una neurona cerebral, solo considera dos estados: apagado (0) y encendido (1). Este modelo puede tener un número de entradas n binarias y solo una salida binaria.

La función de activación de este modelo es la siguiente:

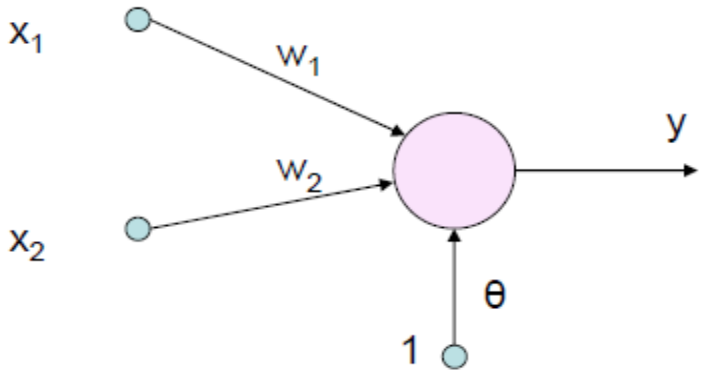
$$S = \begin{cases} 1 \text{ si } \sum_{i=1}^{n} xi * wi > \theta \\ 0 \text{ en caso contrario} \end{cases}$$

Donde: S es la salida binaria
xi son las posibles entradas.
wi son los posibles pesos.
Θ es el umbral.

El umbral es un parámetro que se utiliza para comparar y así determinar la salida.

5.1.3.2. Neuronas Perceptrón.

Consiste en una estructura monocapa, donde hay un conjunto de células de entrada, tantas como sea necesario, dependiendo del problema; y una o varias células de salida. Cada una de las células de entrada tiene conexiones con todas las células de salida, estas conexiones son las que determinan las superficies de discriminación del sistema. La estructura de este modelo se muestra en la figura 5.4.



Fuente: http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/perceptron-simple.html

Figura.5.4. Estructura de una neurona Perceptron.

La función de activación de este modelo es la siguiente:

$$y = F\left(\sum_{i=1}^{n} wi * xi + \theta\right)$$

Donde: y es la salida.

wi son los distintos pesos.

xi son las diferentes entradas.

Θ es el umbral.

F es la función de sigma.

La función F es la siguiente:

$$F(s) = \begin{cases} 1 \text{ si } s > 0 \\ -1 \text{ si } s < 0 \end{cases}$$

Este modelo surgió para realizar tareas de clasificación, por ejemplo si la salida es igual a 1, la entrada pertenece al grupo A y si la salida es igual a -1, la entrada pertenece al grupo B.

5.1.3.3. Neuronas Adaline.

Este modelo es similar al modelo Perceptrón, sin embargo la neurona Adaline es capaz de realizar aprendizaje. Consiste en recibir un conjunto de entradas y las combina para determinar una salida, es un modelo combinador adaptivo.

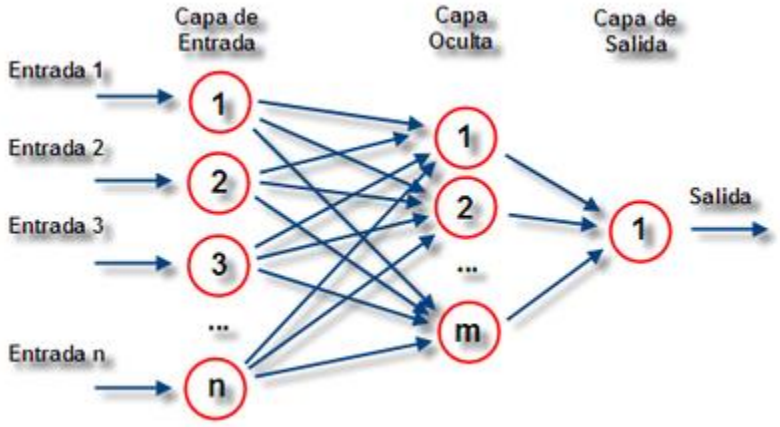
El aprendizaje de este modelo consiste en determinar el error, que se obtiene con la diferencia entre la salida deseada (d) y el valor real (y) producido en la capa de salida para un patrón de entrada x. A este tipo de aprendizaje se le llama regla Delta (| d-y |). Otro aprendizaje que debe realizar el modelo es establecer los valores de los pesos adecuados para que el error sea igual a cero.
La función de activación de este modelo es la siguiente:

$$\Delta wi = \begin{cases} \alpha\, xi & \text{si } d > y \\ -\alpha\, xi & \text{si } d < y \\ 0 & \text{en caso contrario} \end{cases}$$

Donde: Δwi son los diferentes pesos.
xi son la distintas entradas
d es la salida deseada
y es el valor real producido en la capa de salida.
α es la taza de aprendizaje.

5.1.4. Estructura básica de una red neuronal artificial.

Una red neuronal artificial es un conjunto de neuronas artificiales, donde sus salidas están conectadas a las entradas de otras neuronas. Una vez que se calcula la salida de una neurona como se explicó anteriormente esta se conecta como entrada de otra neurona. La estructura básica de una red neuronal artificial se muestra en la figura 5.5.



Fuente: http://es.wikipedia.org/wiki/Red_neuronal_artificial

Figura.5.5. Estructura básica de una red neuronal artificial.

La forma en cómo se conectan las neuronas entre sí se le llama patrón de conectividad o arquitectura de la red. La figura 5.5 muestra una red multicapa, en donde primero están las células de entrada que son las que reciben los valores de entrada, después se encuentran las células ocultas o capa oculta, puede haber una o varias capas ocultas, y por último están las células de salida o capa de salida, esta capa determina la salida de toda la red.

5.1.5 Aprendizaje de la red neuronal.

El aprendizaje es la parte más importante de la red neuronal artificial. El proceso de aprendizaje determina la solución de problemas que la red será capaz de resolver. Este proceso se basa en ejemplos.
El aprendizaje de una red neuronal artificial consiste en determinar los pesos adecuados de todas sus conexiones que la habiliten para resolver el problema deseado. Generalmente el proceso de aprendizaje consiste en ir introduciendo todos los ejemplos de aprendizaje y modificar los pesos de las conexiones dependiendo del tipo de aprendizaje. El tipo de aprendizaje varía de acuerdo al modelo de neuronas que se esté utilizando.

‘’Desde el punto de vista de los ejemplos, el conjunto de aprendizaje debe poseer las siguientes características:

- **Ser significativo**. Debe haber un número suficiente de ejemplos. Si el conjunto de aprendizaje es reducido, la red no será capaz de adaptar sus pesos de forma eficaz.
- **Ser representativo**. Los componentes del conjunto de aprendizaje deberán ser diversos. Si el conjunto de aprendizaje tiene muchos más ejemplos de un tipo que el resto, la red se especializará en dicho subconjunto de datos y no será de aplicación general. Es importante que todas las regiones significativas del espacio de estados estén suficientemente representadas en el conjunto de aprendizaje’’ [8].

5.2. Configuración del controlador neurodifuso.

La configuración del controlador neurodifuso consiste en sustituir el proceso de fuzzificación, el mecanismo de inferencia y el proceso defuzzificación del controlador difuso por redes neuronales. Esta configuración se realizara con el modelo neuronal artificial Adaline por ser un modelo capaz de utilizar aprendizaje como se explicó anteriormente.

En primer lugar se produjeron los ejemplos para el aprendizaje, de acuerdo al controlador difuso que se realizó en el capítulo cuatro. Toman en cuenta el proceso de fuzzificación, el mecanismo de inferencia y el proceso de defuzzificación, la tabla 5.1 relaciona la entrada (error) con la salida (PWM) del controlador difuso es la siguiente:

Error (x)	PWM (d)
20	10
40	10
60	40
80	40
100	90
120	120
140	120
160	160
180	180
200	180
400	200
600	220
800	220
1024	240

Tabla. 5.1. Conjunto de entrenamiento del controlador difuso.

5.3. Proceso de aprendizaje del controlador neurodifuso.

El proceso de aprendizaje definido por la regla Delta de acuerdo al autor Pedro Isasi Viñuela [8] está dado por seis pasos.

1. Iniciar los pesos de forma aleatoria.
2. Introducir un patrón de entrada.
3. Calcular la salida de la red, compararla con la deseada y obtener la diferencia: (d - y).
4. Para todos los pesos, multiplicar dicha diferencia por la entrada correspondiente, y ponderarla por una tasa de aprendizaje α.
5. Modificar el peso restando del valor antiguo la cantidad obtenida en 4.
6. Si no se ha cumplido el criterio de convergencia, regresar a 2; si se han acabado todos los patrones, empezar de nuevo a introducir patrones.

Debido a que este entrenamiento es repetitivo se debe de realizar con la ayuda de algún lenguaje de programación.
El siguiente ejemplo ilustra el algoritmo de aprendizaje, con el fin de explicar cómo se determinan los pesos adecuados que resuelvan el problema.

Se elegirá el conjunto de entrenamiento del controlador difuso mostrado en la tabla 5.1, solo para las dos primeras entradas de la tabla.

Primera entrada:

Como solo hay una entrada en el controlador, se inicializa un solo peso w = 0.1.

De acuerdo a la tabla 5.1 el patrón de entrada uno es: x1 = 20.

La salida de la red es: y = w*x1 = 2. La salida deseada de acuerdo a la tabla 5.1 es: d = 10. La diferencia es: d - y = 8.

La taza de aprendizaje se inicializa de forma aleatoria α= 0.000001.Multiplicando la diferencia (d-y), por la entrada (x1) y por la taza de aprendizaje (α), el resultado es: Δw = (d-y)*x1*α = 0.00016.

Restando w con Δw se modifica el peso: w = w – Δw = 0.09984.

Segunda entrada:

El peso ahora es: w = 0.09984

De acuerdo a la tabla 5.1 el patrón de entrada dos es x2= 40.

La salida de la red es: y = w*x2 = 3.9936. La salida deseada de acuerdo a la tabla 5.1 es d = 10. La diferencia es: d - y = 6.0064.

La taza de aprendizaje se inicializa de forma aleatoria α= 0.000001.Multiplicando la diferencia por la entrada y por α, el resultado es: Δw = (d-y)*x2*α = 0.00024.

Restando w con Δw se modifica el peso: w = w – Δw = 0.09976.

Siguiendo este proceso es como se entrena a la neurona artificial, para obtener el peso adecuado que resuelva el problema deseado.

5.4. Realización del controlador neurodifuso.

Es importante aclarar que como el controlador tiene una entrada (error) y una salida (PWM) solo se necesita una neurona artificial, no es necesaria una red neuronal artificial.

El esquema de la neurona artificial para el controlador neurodifuso se muestra en la figura 5.6.

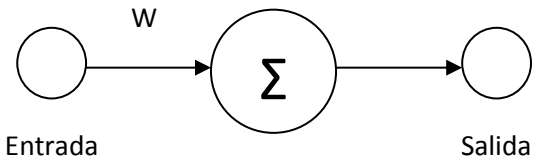


Figura. 5.6. Neurona artificial para el controlador neurodifuso.

El algoritmo del controlador neurodifuso para la implementación es:

6. Calcular el error.
7. Verificar si el error es positivo o negativo.
8. Determinar el valor absoluto del error.
9. Calcular el la salida multiplicando el peso por el error.

El diagrama de flujo del controlador neurodifuso es el siguiente:

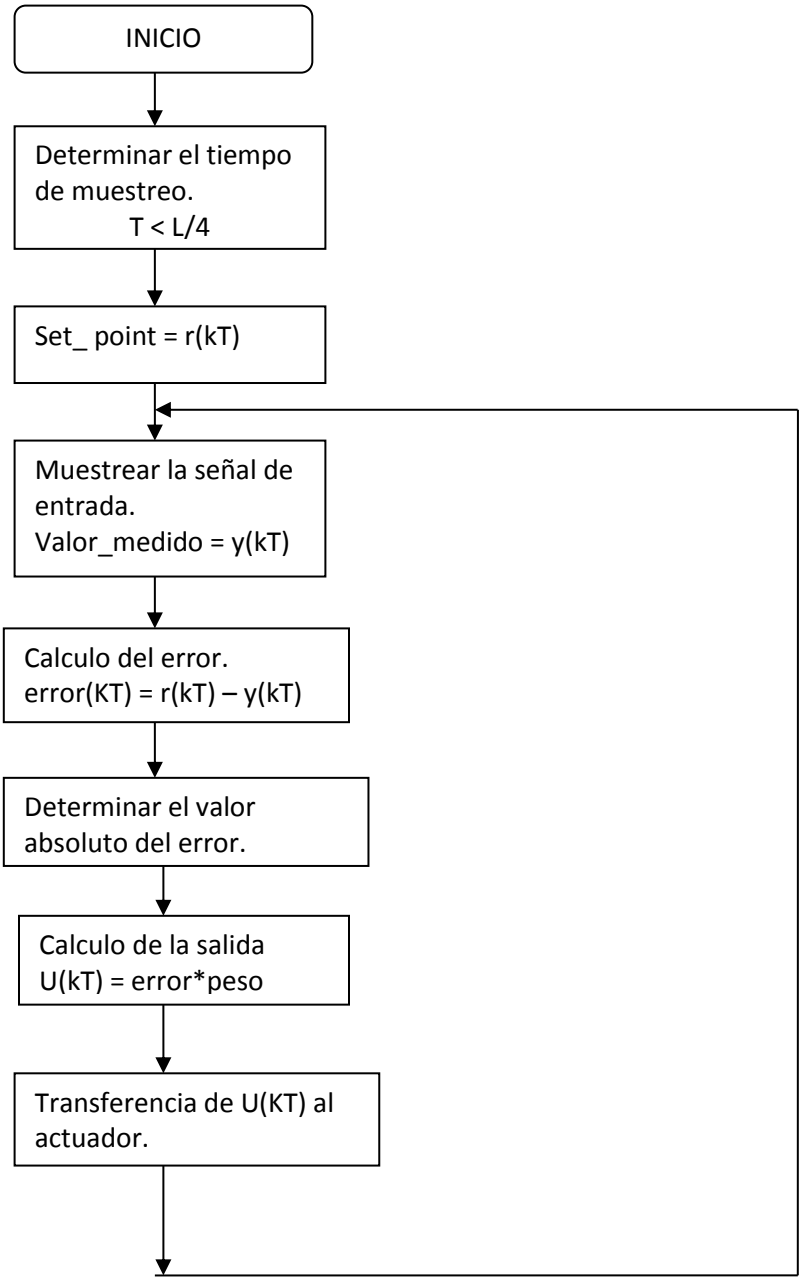


Figura. 5.7. Diagrama de flujo del controlador neurodifuso.

5.5. Configuración del controlador PID neurodifuso.

Se utilizó el modelo neuronal artificial Adaline para sustituir la estructura del controlador difuso para la sintonización del controlador PD.

Los ejemplos se produjeron mediante el controlador PD difuso que se realizó en el capítulo cuarto. Para cada patrón de entrada se obtendrá su salida correspondiente siguiendo los tres procesos del controlador (fuzzificación, mecanismo de inferencia y defuzzificación).

Debido a que la sintonización del controlador PD consiste en determinar los parámetros de K_Fp y K_Fd, es necesario producir dos conjuntos de entrenamiento uno para el parámetro K_Fp y otro para el parámetro K_Fd

La tabla 5.2 muestra el conjunto de entrenamiento para determinar el parámetro K_Fp y la tabla 5.3 muestra el conjunto de entrenamiento para determinar el parámetro K_Fd del controlador PD.

Cabe aclarar que los conjuntos de entrenamiento se obtuvieron mediante un programa en lenguaje se C, siguiendo el algoritmo de entrenamiento de la regla Delta. El programa se muestra en apéndice.

Error	K_Fp
40.98	2
81.96	1.95
122.95	1.97
163.93	2.057
204.9	2.13

Tabla. 5.2. Conjunto de entrenamiento para determinar el parámetro K_Fp del controlador PD.

Error	K_{Fd}
40.98	0.00910
81.96	0.00913
122.95	0.00918
163.93	0.00923
204.9	0.00927

Tabla. 5.3. Conjunto de entrenamiento para determinar el parámetro K_{Fd} del controlador PD.

5.6. Proceso de aprendizaje del controlador PD neurodifuso.

El entrenamiento de la neurona artificial para obtener los pesos adecuados que determinaran los parámetros K_{Fp} y K_{Fd} del controlador PD se hará de acuerdo al algoritmo Delta. Este programa se muestra en el apéndice.

Después de realizar los programas los pesos adecuados son: wp= 0.008520 y wd= 0.000042.

5.7. Realización del controlador PD neurodifuso.

La sintonización del controlador PD neurodifuso necesita dos neuronas artificiales porque tiene una entrada (error) y dos salidas que son los parámetros $K_F p$ y $K_F d$. La figura 5.8 muestra el esquema de la red neuronal artificial para la sintonización del controlador PID neurodifuso.

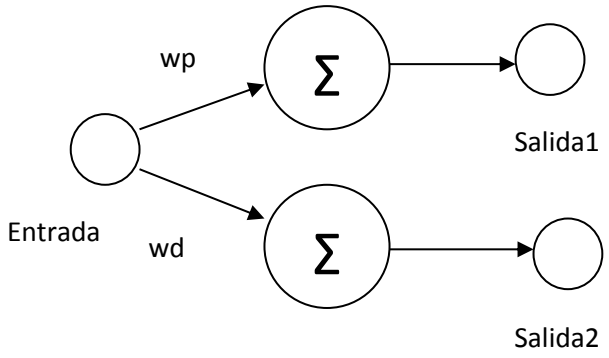


Figura.5.8. Red neuronal artificial para la sintonización del controlador PD neurodifuso.

A continuación se muestra el algoritmo para la sintonización del controlador PD neurodifuso:

18. Calcular el error.
19. Calcular el cambio de error.
20. Verificar si el error es positivo o negativo.
21. Verificar si e > 0 y ce < 0.
22. Si e > 0 y ce < 0 entonces $K_F p = Kp$ y $K_F d = Kd$.
23. Verificar si e < 0 y ce < 0.

24. Si e < 0 y ce < 0, determinar el valor absoluto de e y calcular K_Fp=wp*e y K_Fd= wd*e.
25. Verificar si e < 0 y ce > 0.
26. Si e < 0 y ce > 0 entonces K_Fp=Kp y K_Fd= Kd.
27. Verificar si e > 0 y ce > 0.
28. Si e > 0 y ce > 0, determinar el valor absoluto de e y calcular K_Fp=wp*e y K_Fd= wd*e.
29. Verificar si e ~ 0 y ce ~ 0.
30. Si e ~ 0 y ce ~ 0 entonces K_Fp=Kp y K_Fd= Kd.
31. Calcular la parte proporcional.
32. Calcular la parte derivativa.
33. Sumar la parte proporcional y derivativa.
34. Activar el PWM.

El diagrama de flujo del controlador PD neurodifuso se muestra en la figura 5.9.

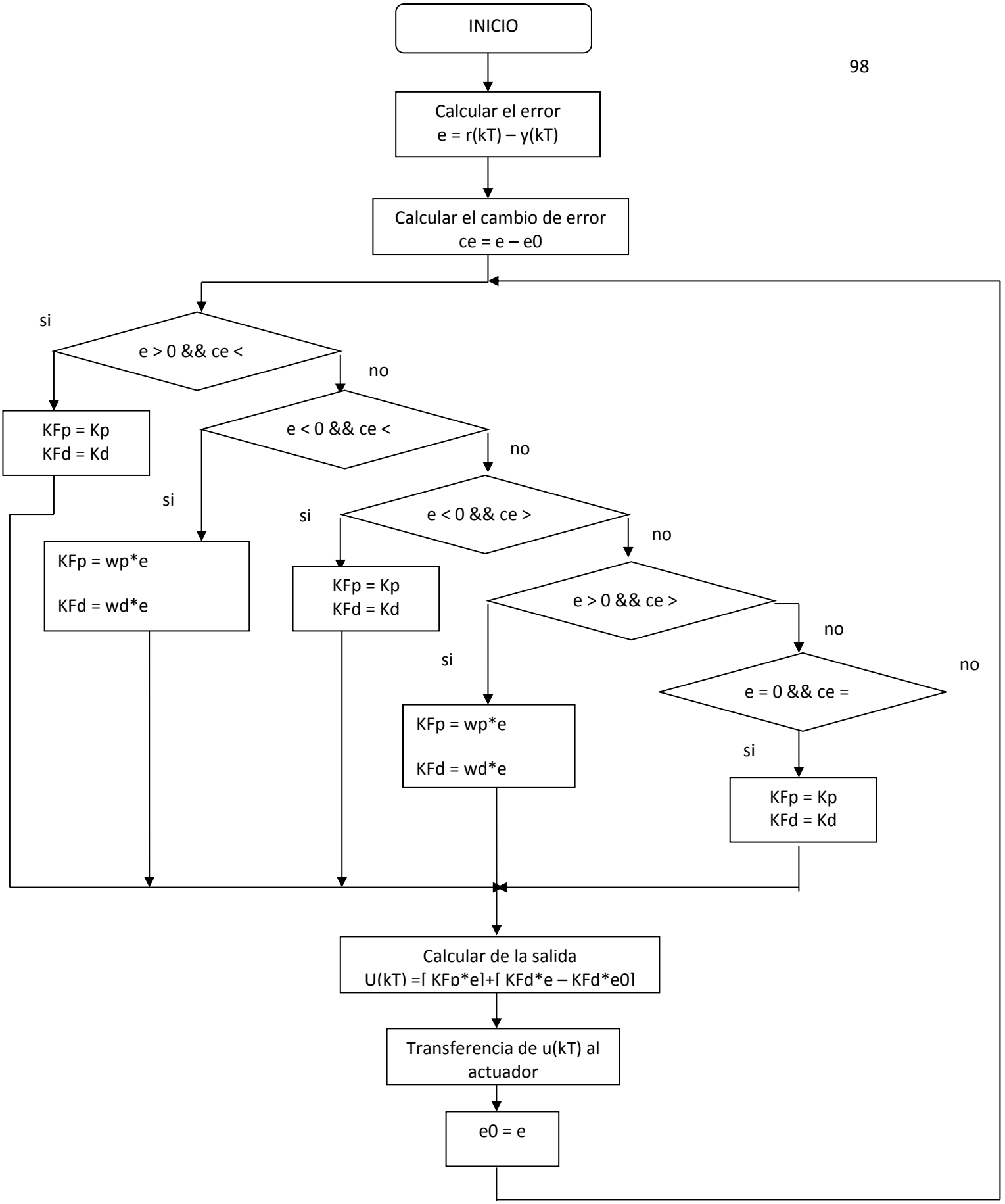


Figura. 5.9. Diagrama de flujo del controlador PD neurodifuso.

5.8. Mediciones de desempeño.

Después de la implementación física de los controladores Neurodifuso y PD neurodifuso es importante evaluar su desempeño ingresándoles las señales de pruebas tren de pulsos, seno y diente de sierra.

5.8.1. Desempeño del controlador neurodifuso.

A continuación se exponen las gráficas 5.10, 5.11 y 5.12 del desempeño del controlador neurodifuso para una función de entrada tren de pulsos, seno y diente de sierra respectivamente, las gráficas se obtuvieron con la tarjeta de adquisición de datos de National Instruments con el software de Labview.

La señal de color blanco es la función de prueba y la señal de color roja es la respuesta del controlador.

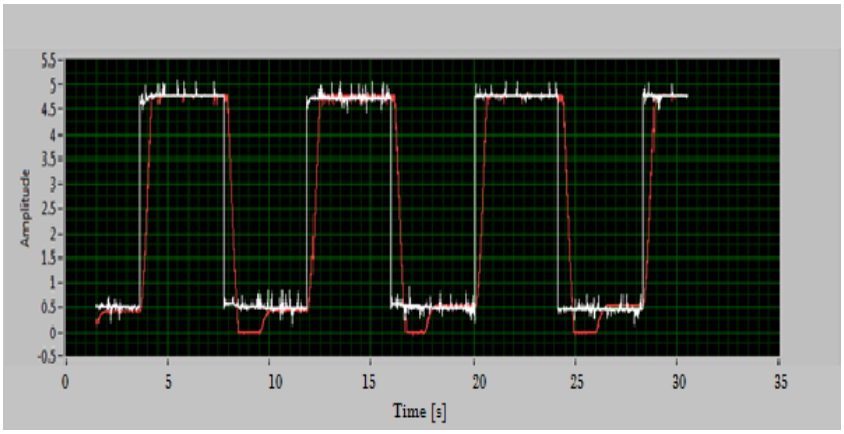


Figura 5.10. Respuesta del controlador neurodifuso a una función tren de pulsos.

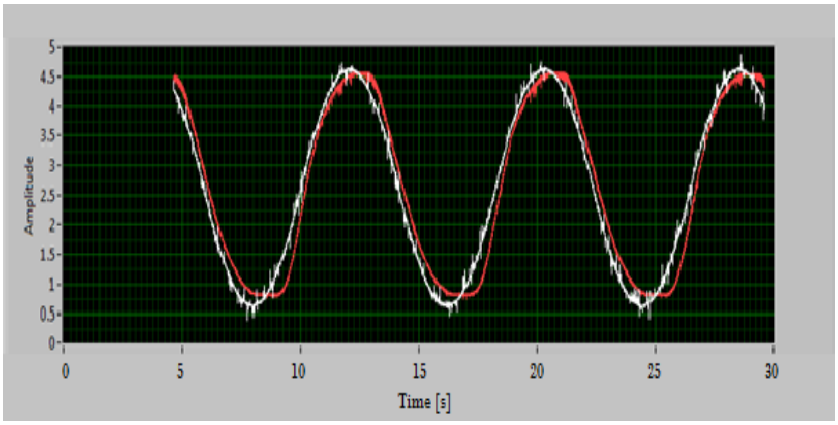


Figura 5.11. Respuesta del controlador neurodifuso a una función seno.

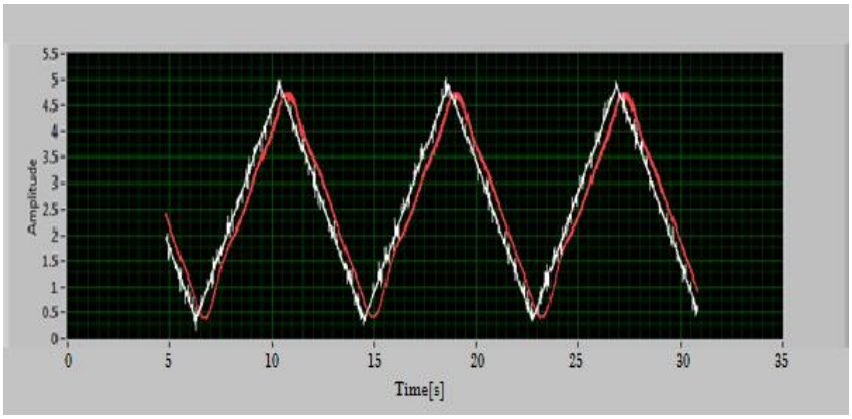


Figura 5.12. Respuesta del controlador neurodifuso a una función diente de sierra.

Como se puede observar en las gráficas anteriores el controlador neurodifuso siguió bien a las tres señales a diferencia del controlador difuso que seguía bien solo a la señal seno, esto debido a que gracias a la neurona artificial el controlador neurodifuso es más rápido porque el aprendizaje neuronal realizó los tres procesos (fuzzificación, mecanismo de inferencia y defuzzificación) del controlador difuso reduciendo tiempo en el cálculo matemático que conllevan los tres procesos.

5.8.2. Desempeño del controlador PD neurodifuso.

La evaluación del desempeño del controlador PD neurodifuso se muestran en las gráficas 5.13, 5.14 y 5.15 para la entrada con funciones tren de pulso, seno y diente de sierra respectivamente. La señal de color blanco es la función de prueba y la señal de color rojo es la respuesta del controlador. Las tres gráficas se obtuvieron también con la tarjeta de adquisición de datos de National Instruments y con el software de Labview.

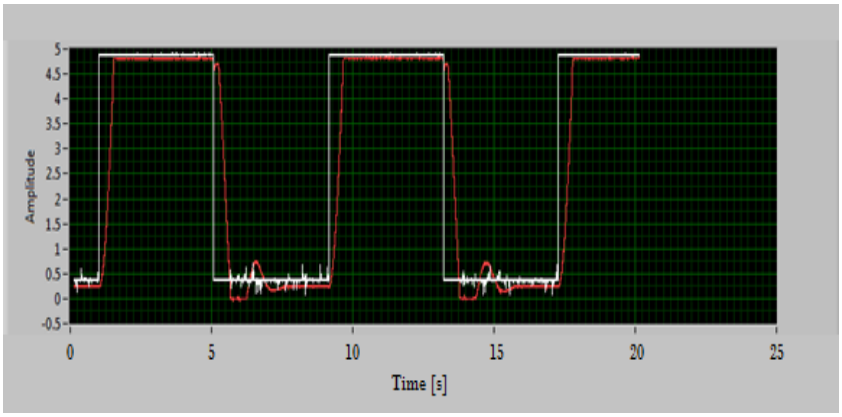


Figura 5.13. Respuesta del controlador PD neurodifuso a una función tren de pulsos.

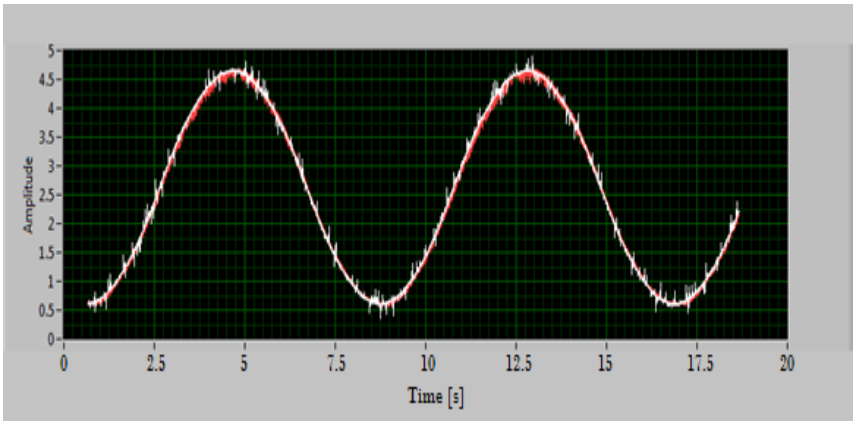


Figura 5.14. Respuesta del controlador PD neurodifuso a una función seno.

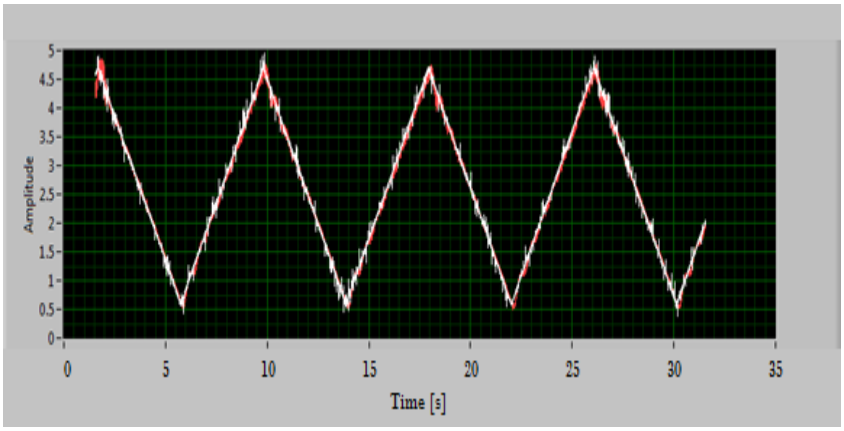


Figura 5.15. Respuesta del controlador PD neurodifuso a una función diente de sierra.

En las tres gráficas se puede observar que el controlador responde más rápido a los cambios de las tres señales, debido que este controlador tiene propiedades del PID convencional, de la lógica difusa y de redes neuronales artificiales.

CAPITULO 6.
Análisis de resultados y conclusiones.

En este capítulo se comparara el desempeño de los tres controladores el controlador PD, el controlador PD difuso y el controlador PD neurodifuso, aplicados a el motor ya montado a el robot móvil (figura 6.1), para controlar la posición de la llanta de dirección, con el propósito de ver las ventajas y desventajas de cada controlador.

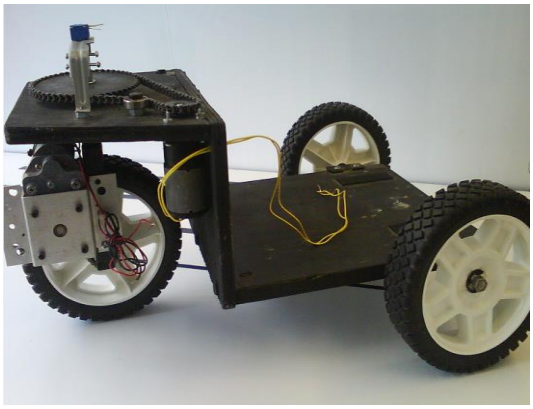


Figura 6.1. Robot móvil a controlar.

6.1. Resultados.

Las pruebas de los tres controladores se llevaron a cabo colocando al robot móvil en tres diferentes superficies, la primera superficie de prueba fue en una mesa de formica, la segunda en una tabla de madera barnizada, y la tercera en un piso de concreto, cada una de estas superficies cuenta con una rugosidad diferente, el objetivo de la prueba es verificar el desempeño de cada uno de los controladores en superficies distintas.

- **Superficie de formica**.

En las figuras 6.2, 6.3 y 6.4 se muestran las respuestas de los tres controladores PD, PD disuso y PD neurodifuso a una entrada escalón respectivamente, aplicados al robot móvil en una mesa de formica.

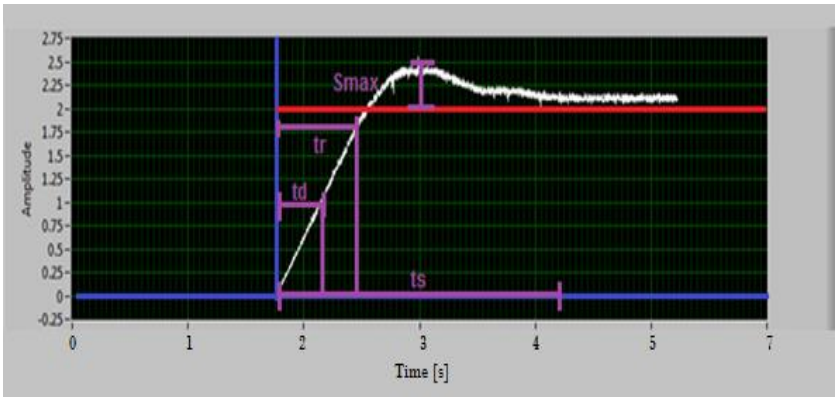


Figura 6.2. Respuesta del controlador PD a una entrada escalón, aplicado al motor de un robot móvil en una superficie de formica.

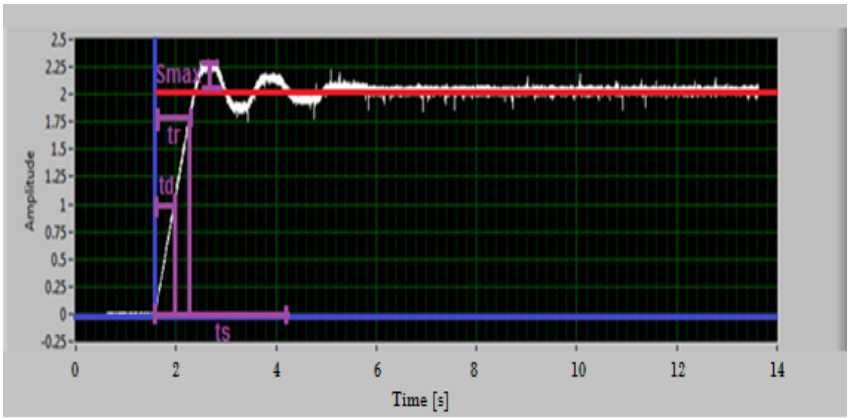


Figura 6.3. Respuesta del controlador PD difuso a una entrada escalón, aplicado al motor de un robot móvil en una superficie de formica.

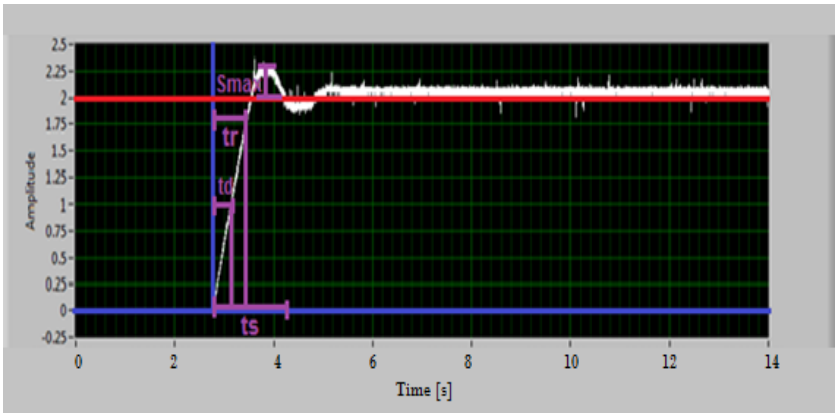


Figura 6.4. Respuesta del controlador PD neurodifuso a una entrada escalón, aplicado al motor de un robot móvil en una superficie de formica.

Analizando las tres graficas anteriores se obtuvieron los siguientes resultados:

	PD	PD difuso	PD neurodifuso
Sobrepaso máximo (Smax)	25%	12.5%	12.5%
Tiempo de retardo (td)	0.35s	0.2s	0.2s
Tiempo de levantamiento (tr)	0.65s	0.3s	0.3s
Tiempo de asentamiento (ts)	2.4s	1.3s	0.8s
Error en estado estable	0.12	0.06	0.06
Frecuencia natural (Wn)	2.4	5.2	5.2
Factor de amortiguamiento (ζ)	0.69	0.59	0.96

Tabla 6.1. Características principales de la respuesta de los tres controladores a una entrada escalón, aplicados al motor de un robot móvil en una superficie de formica.

Después de haber obtenido las características principales de la respuesta de los tres controladores a una entrada escalón, es importante evaluar el desempeño de cada controlador a diferentes entradas, y para ello utilizaremos las señales de prueba tren de pulsos, seno y diente de sierra.

En primer lugar empezaremos a evaluar el desempeño del controlador PD.

En las siguientes figuras la señal de prueba es la de color blanco y la respuesta del controlador es la de color rojo.

La figura 6.5 muestra la respuesta a una entrada tren de pulsos del controlador PD, aplicado al motor de un robot móvil en una superficie de formica.

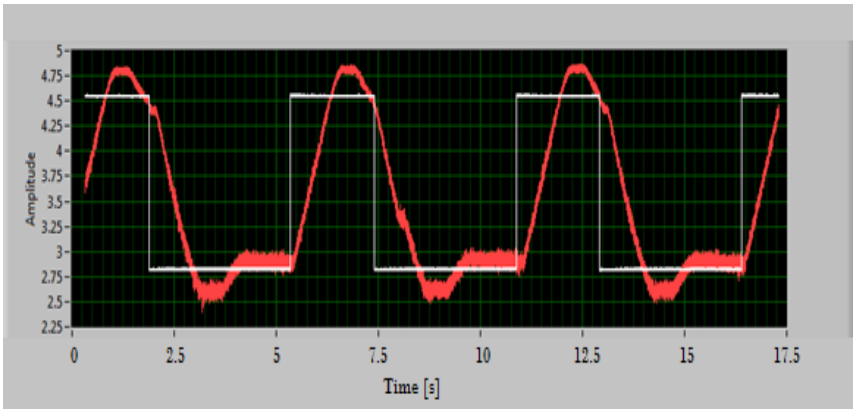


Figura 6.5. Respuesta a una entrada tren de pulsos del controlador PD, aplicado al motor de un robot móvil en una superficie de formica.

La respuesta a una entrada seno del controlador PD, aplicado al motor de un robot móvil en una superficie de formica se muestra en la figura 6.6

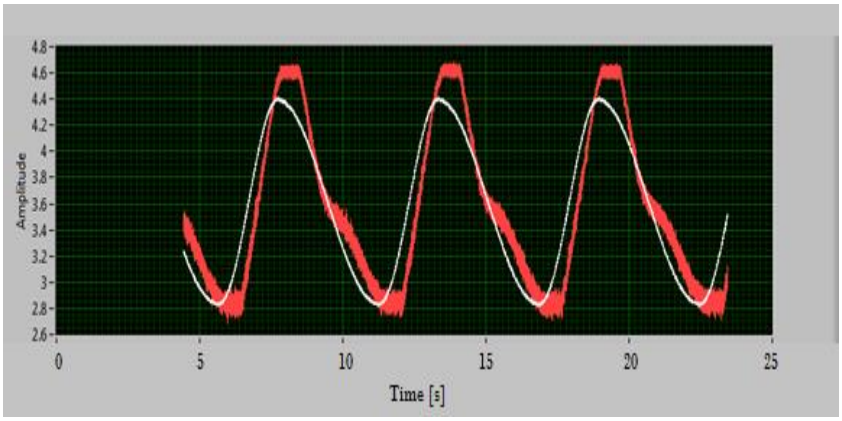


Figura 6.6. Respuesta a una entrada seno del controlador PD, aplicado al motor de un robot móvil en una superficie de formica.

La figura 6.7 muestra la respuesta a una entrada diente de sierra del controlador PD, aplicado al motor de un robot móvil en una superficie de formica.

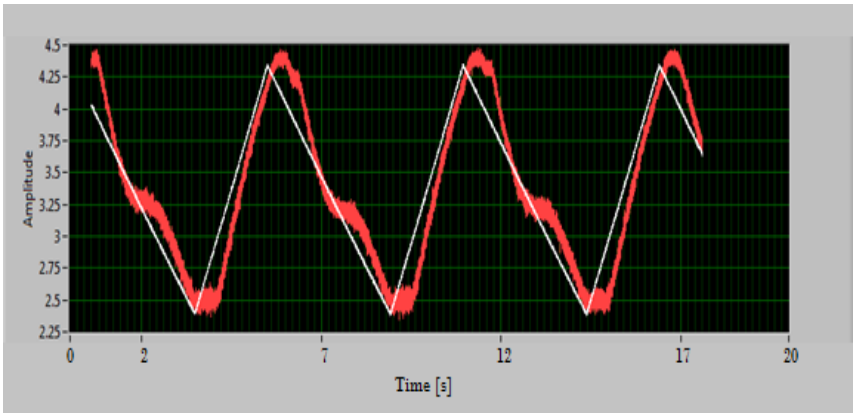


Figura 6.7. Respuesta a una entrada diente de sierra del controlador PD, aplicado al motor de un robot móvil en una superficie de formica.

En segundo lugar se evaluara el desempeño del controlador PD difuso para las señales de entrada tren de pulsos mostrada en la figura 6.8, seno mostrada en la figura 6.9 y diente de sierra mostrada en la figura 6.10.

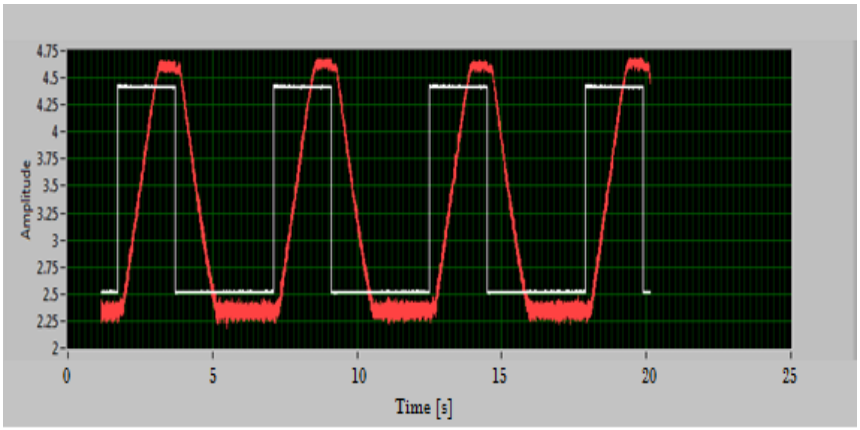


Figura 6.8. Respuesta a una entrada tren de pulsos del controlador PD difuso, aplicado al motor de un robot móvil en una superficie de formica.

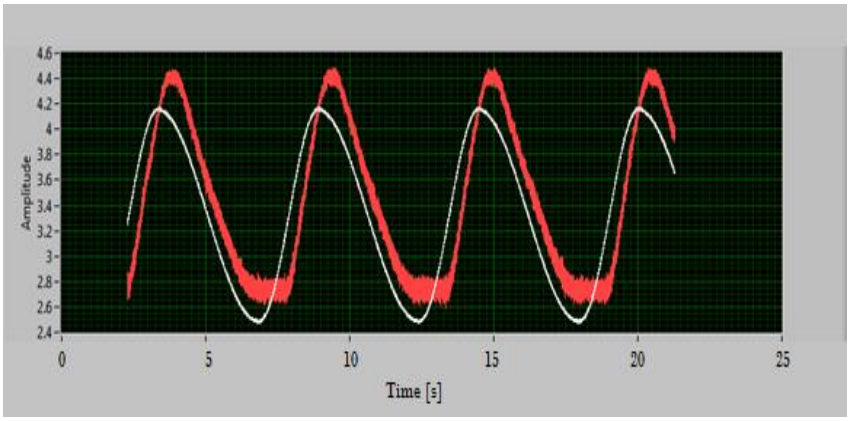


Figura 6.9. Respuesta a una entrada seno del controlador PD difuso, aplicado al motor de un robot móvil en una superficie de formica.

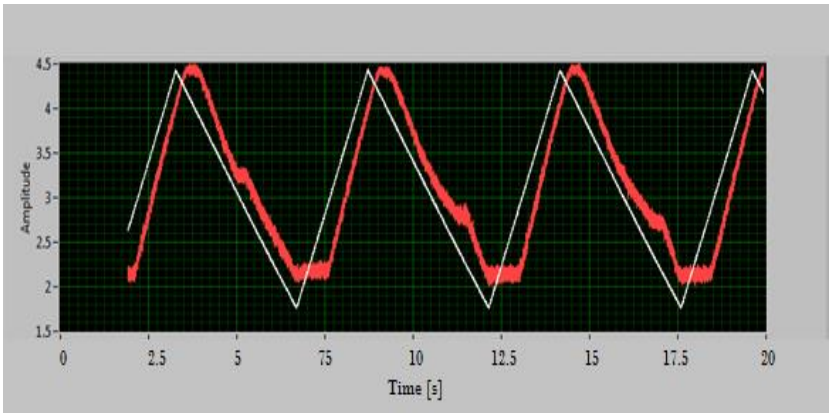


Figura 6.10. Respuesta a una entrada diente de sierra del controlador PD difuso, aplicado al motor de un robot móvil en una superficie de formica.

Por último se evaluara el comportamiento del controlador PD neurodifuso para las señales de entrada tren de pulsos mostrada en la figura 6.11, seno mostrada en la figura 6.12 y diente de sierra mostrada en la figura 6.13.

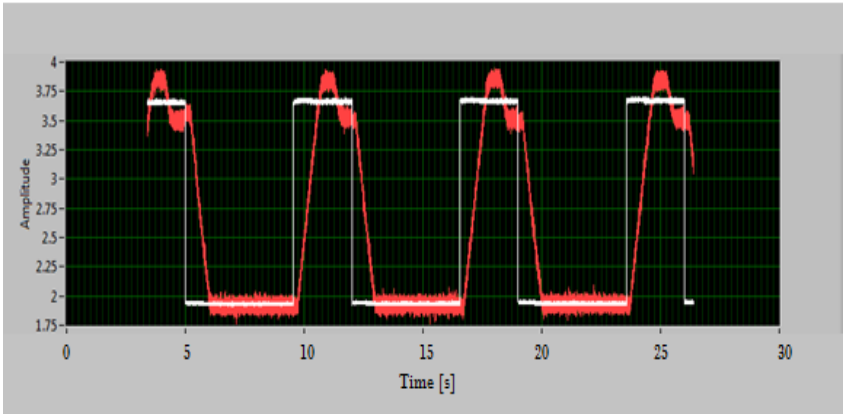


Figura 6.11. Respuesta a una entrada tren de pulsos del controlador PD neurodifuso, aplicado al motor de un robot móvil en una superficie de formica.

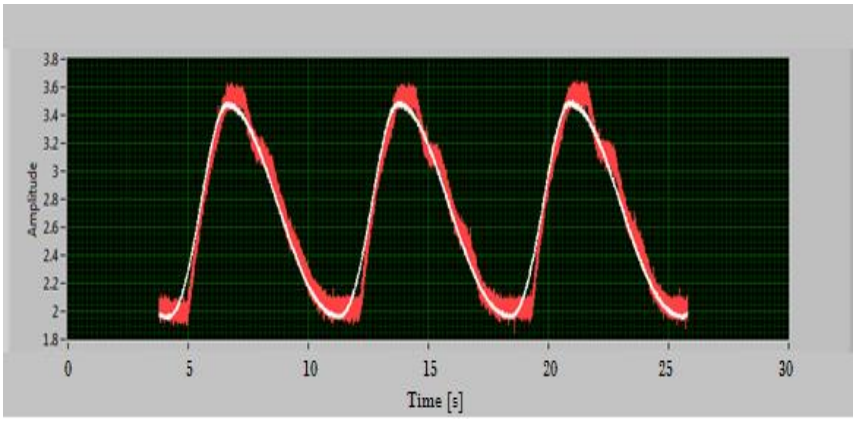


Figura 6.12. Respuesta a una entrada seno del controlador PD neurodifuso, aplicado al motor de un robot móvil en una superficie de formica.

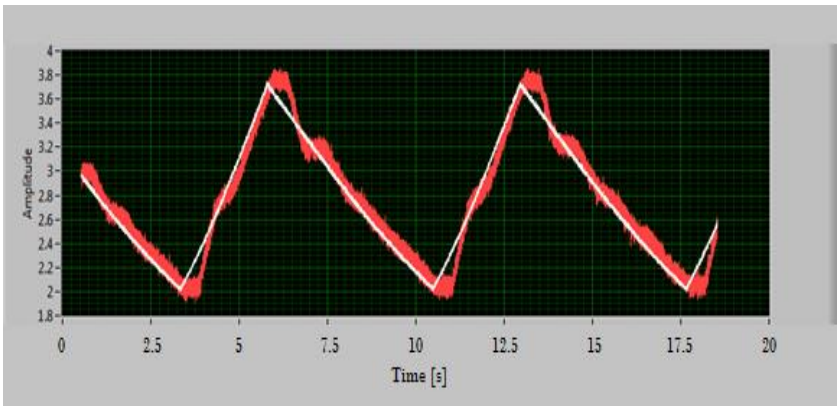


Figura 6.13. Respuesta a una entrada diente de sierra del controlador PD neurodifuso, aplicado al motor de un robot móvil en una superficie de formica.

Comparando el desempeño de los tres controladores se puede observar en las gráficas que el controlador PD neurodifuso, responde mejor a las tres señales, e incluso para las señales seno y diente de sierra mostradas en las figuras 6.12 y 6.13 respectivamente, esto indica que el controlador responde más rápido, en comparación de los otros dos controladores.

- **Superficie de madera barnizada.**

Las figuras 6.14, 6.15 y 6.16 muestran las respuestas de los tres controladores PD, PD difuso y PD neurodifuso a una entrada escalón respectivamente, aplicados al robot móvil en una superficie de madera barnizada.

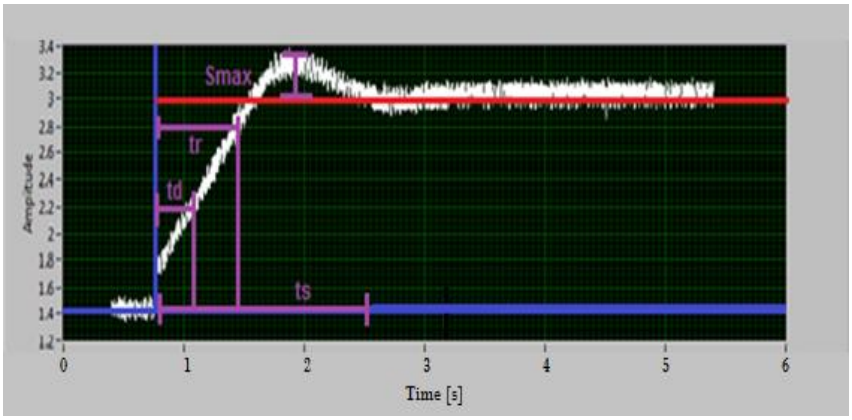


Figura 6.14. Respuesta del controlador PD a una entrada escalón, aplicado al motor de un robot móvil en una superficie de madera barnizada.

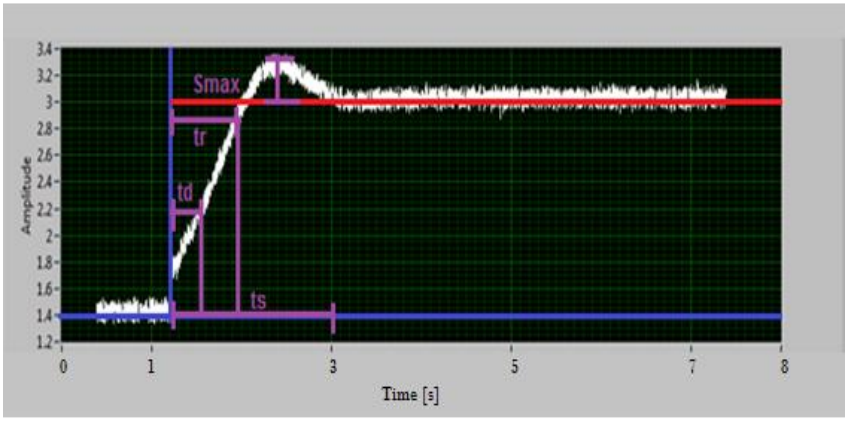


Figura 6.15. Respuesta del controlador PD difuso a una entrada escalón, aplicado al motor de un robot móvil en una superficie de madera barnizada.

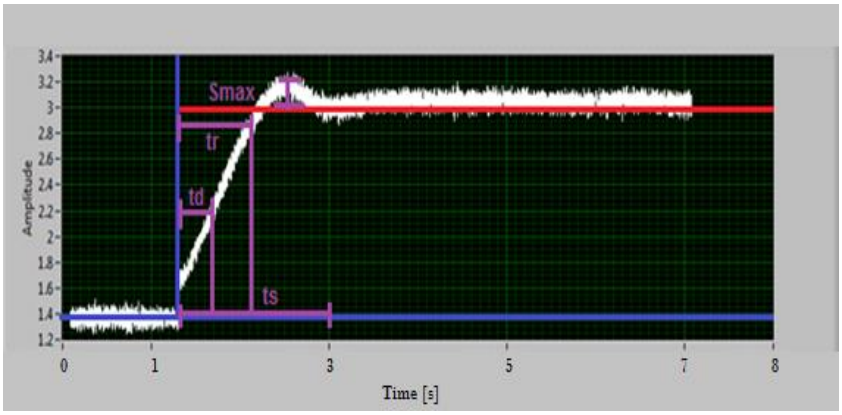


Figura 6.16. Respuesta del controlador PD difuso a una entrada escalón, aplicado al motor de un robot móvil en una superficie de madera barnizada.

Los resultados del análisis de las tres graficas anteriores son los siguientes:

	PD	PD difuso	PD neurodifuso
Sobrepaso máximo (Smax)	25%	18.75%	12.5%
Tiempo de retardo (td)	0.35s	0.2s	0.2s
Tiempo de levantamiento (tr)	0.65s	0.3s	0.3s
Tiempo de asentamiento (ts)	1.75s	1s	0.8s
Error en estado estable	0.12	0.10	0.10
Frecuencia natural wn	2.4	5.2	5.2
Factor de amortiguamiento	0.95	0.77	0.96

Tabla 6.2. Características principales de la respuesta de los tres controladores a una entrada escalón, aplicados al motor de un robot móvil en una superficie de madera barnizada.

Ahora se evaluara el desempeño de los tres controladores para las diferentes señales de prueba. Donde la señal de prueba es la de color blanco y la respuesta del controlador es la de color rojo.

A continuación se muestran las gráficas del desempeño del controlador PD, empezando por la señal tren de pulsos mostrada en la figura 6.17, la señal seno mostrada en la figura 6.18 y la señal diente de sierra mostrada en la figura 6.19.

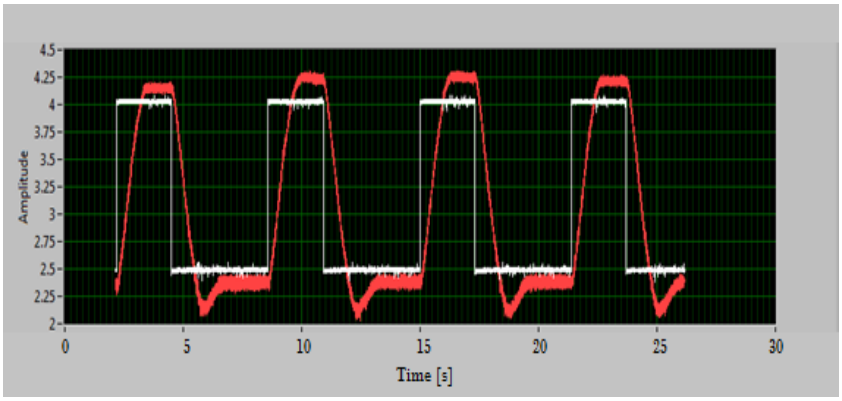


Figura 6.17. Respuesta a una entrada tren de pulsos del controlador PD, aplicado al motor de un robot móvil en una superficie de madera barnizada.

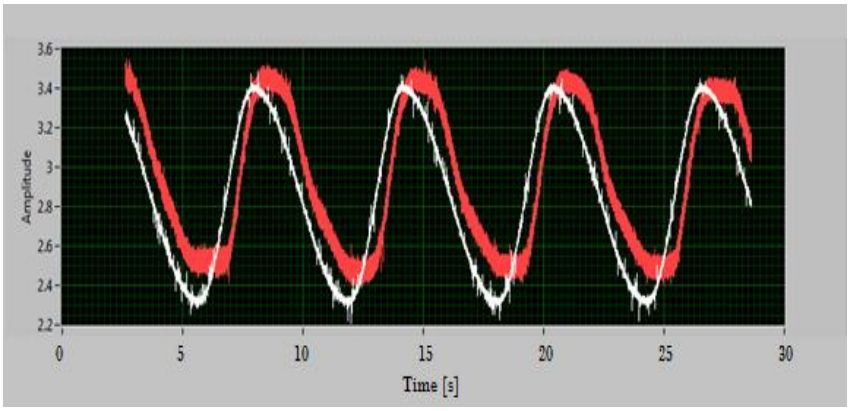


Figura 6.18. Respuesta a una entrada seno del controlador PD, aplicado al motor de un robot móvil en una superficie de madera barnizada.

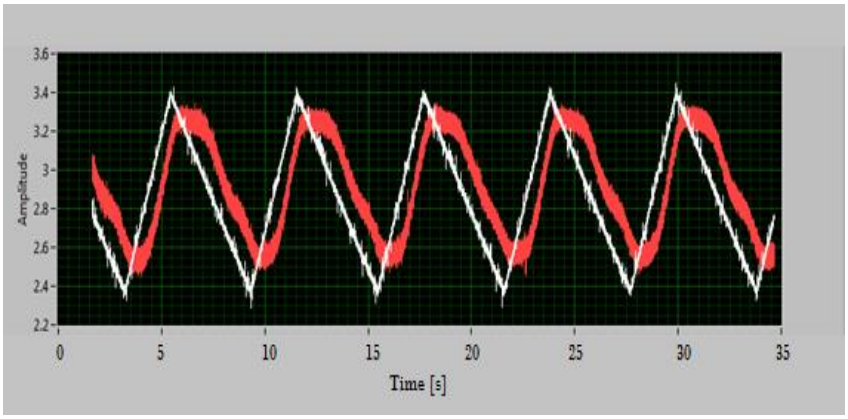


Figura 6.19. Respuesta a una entrada diente de sierra del controlador PD, aplicado al motor de un robot móvil en una superficie de madera barnizada.

Ahora se muestran las gráficas del desempeño del controlador PD difuso, empezando por la señal tren de pulsos mostrada en la figura 6.20, la señal seno mostrada en la figura 6.21 y la señal diente de sierra mostrada en la figura 6.22.

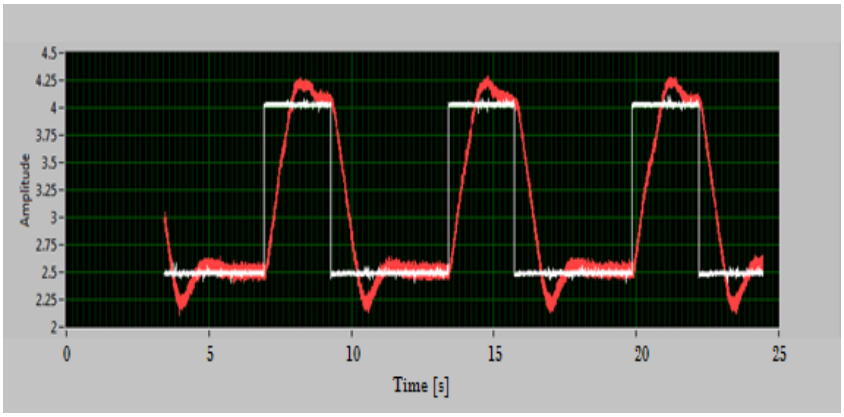


Figura 6.20. Respuesta a una entrada tren de pulsos del controlador PD difuso, aplicado al motor de un robot móvil en una superficie de madera barnizada.

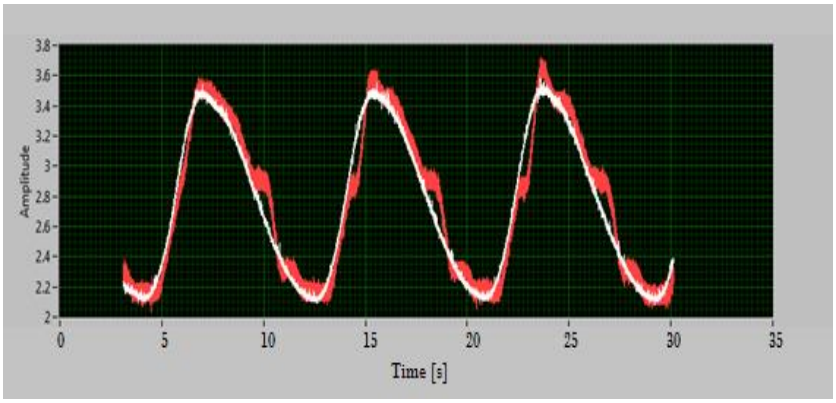


Figura 6.21. Respuesta a una entrada seno del controlador PD difuso, aplicado al motor de un robot móvil en una superficie de madera barnizada.

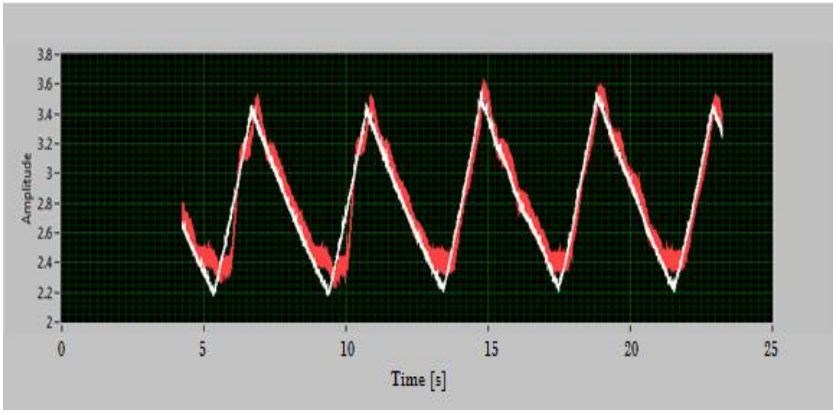


Figura 6.22. Respuesta a una entrada diente de sierra del controlador PD difuso, aplicado al motor de un robot móvil en una superficie de madera barnizada.

Por último el desempeño del controlador PD neurodifuso para las tres señales de prueba tren de pulsos, seno y diente de sierra se muestran en las figuras 6.23, 6.24 y 6.25 respectivamente.

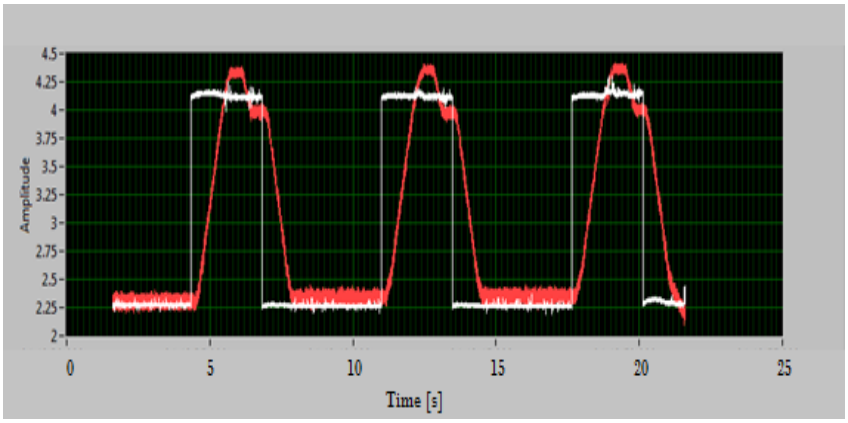


Figura 6.23. Respuesta a una entrada tren de pulsos del controlador PD neurodifuso, aplicado al motor de un robot móvil en una superficie de madera barnizada.

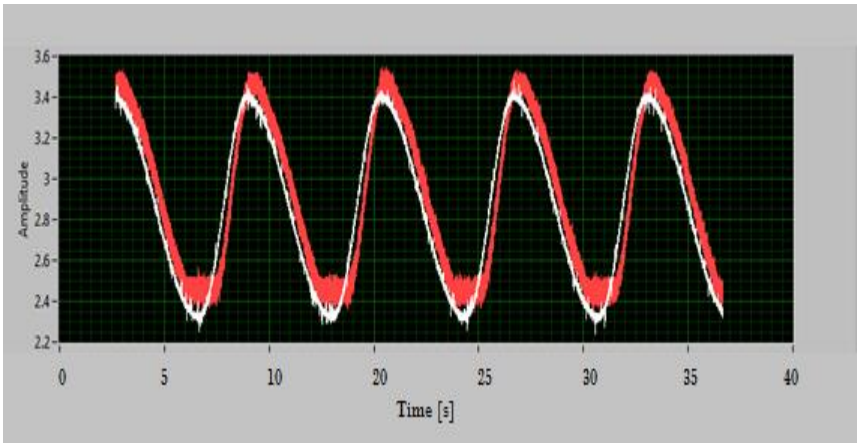


Figura 6.24. Respuesta a una entrada seno del controlador PD neurodifuso, aplicado al motor de un robot móvil en una superficie de madera barnizada.

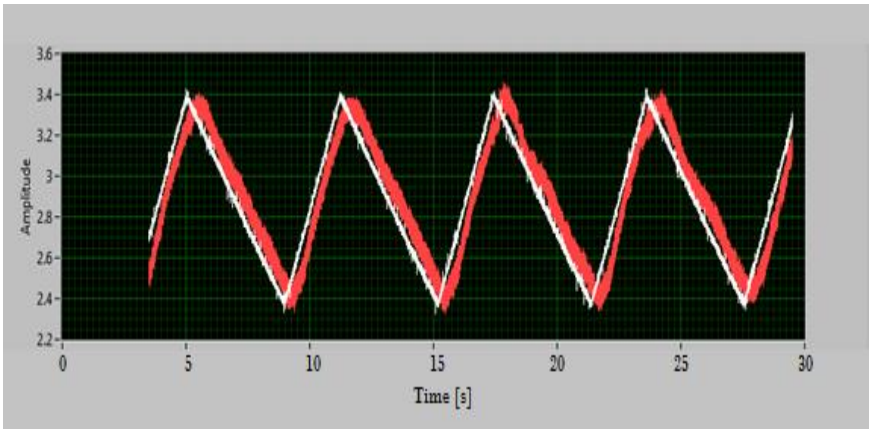


Figura 6.25. Respuesta a una entrada diente de sierra del controlador PD neurodifuso, aplicado al motor de un robot móvil en una superficie de madera barnizada.

Después de evaluar el desempeño de los tres controladores es evidente que el controlador que responde más rápido y sigue mejor la trayectoria de las señales de prueba es el controlador PD neurodifuso.

- **Superficie de concreto.**

Las figuras 6.26, 6.27 y 6.28 muestran las respuestas de los tres controladores PD, PD difuso y PD neurodifuso a una entrada escalón respectivamente, aplicados al robot móvil en una superficie de concreto.

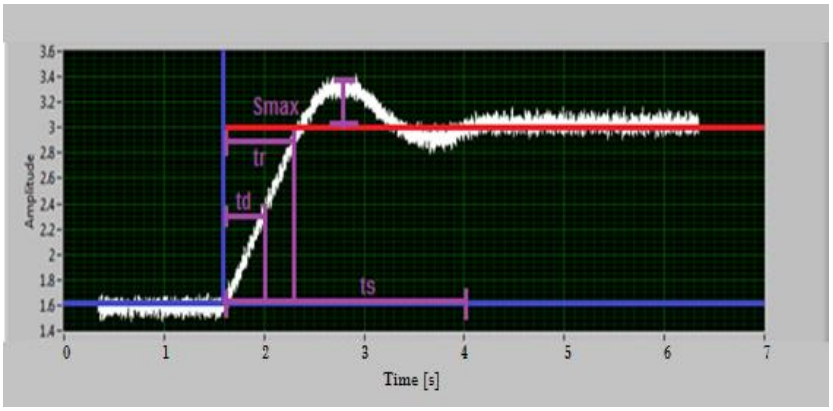


Figura 6.26. Respuesta del controlador PD a una entrada escalón, aplicado al motor de un robot móvil en una superficie de concreto.

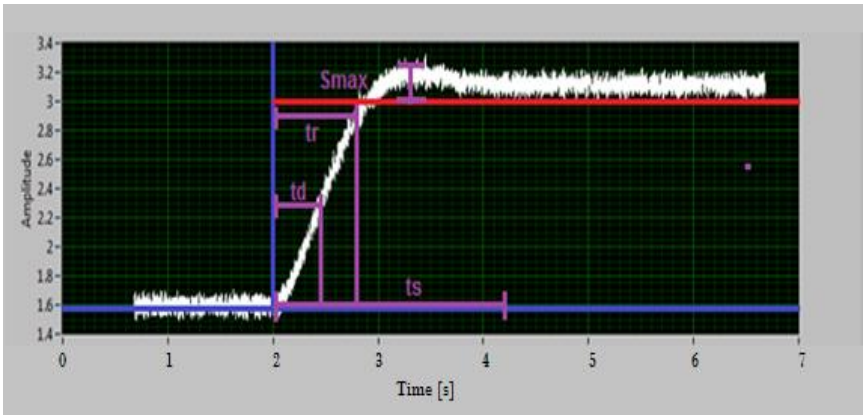


Figura 6.27. Respuesta del controlador PD difuso a una entrada escalón, aplicado al motor de un robot móvil en una superficie de concreto.

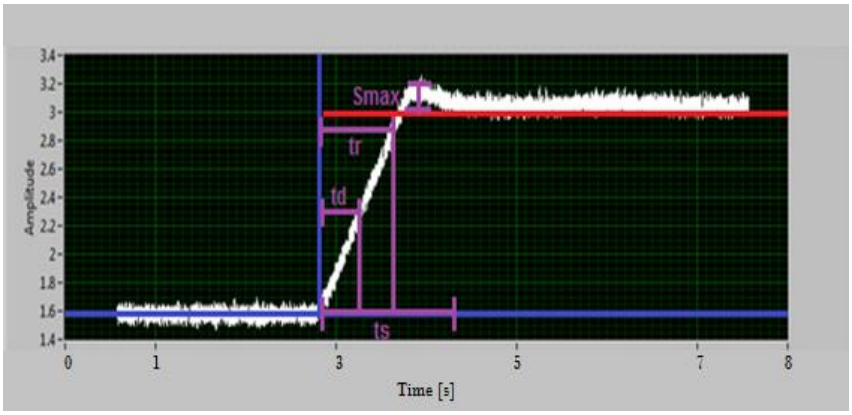


Figura 6.28. Respuesta del controlador PD neurodifuso a una entrada escalón, aplicado al motor de un robot móvil en una superficie de concreto.

Con el análisis de las gráficas anteriores se obtuvieron los siguientes resultados:

	PD	PD difuso	PD neurodifuso
Sobrepaso máximo (Smax)	28.57%	14.28%	14.28%
Tiempo de retardo (td)	0.4s	0.4s	0.4s
Tiempo de levantamiento (tr)	0.8s	0.8s	0.8s
Tiempo de asentamiento (ts)	2.4s	2.2s	1.5s
Error en estado estable	0.12	0.20	0.10
Frecuencia natural wn	1.96	1.96	1.96
Factor de amortiguamiento ζ	0.85	0.92	1.36

Tabla 6.3. Características principales de la respuesta de los tres controladores a una entrada escalón, aplicados al motor de un robot móvil en una superficie de concreto.

A continuación se evaluara el desempeño de los tres controladores para las diferentes señales de prueba. Donde la señal de prueba es la de color blanco y la respuesta del controlador es la de color rojo.

En primer lugar se muestran las gráficas del desempeño del controlador PD, empezando por la señal tren de pulsos mostrada en la figura 6.29, la señal seno mostrada en la figura 6.30 y la señal diente de sierra mostrada en la figura 6.31.

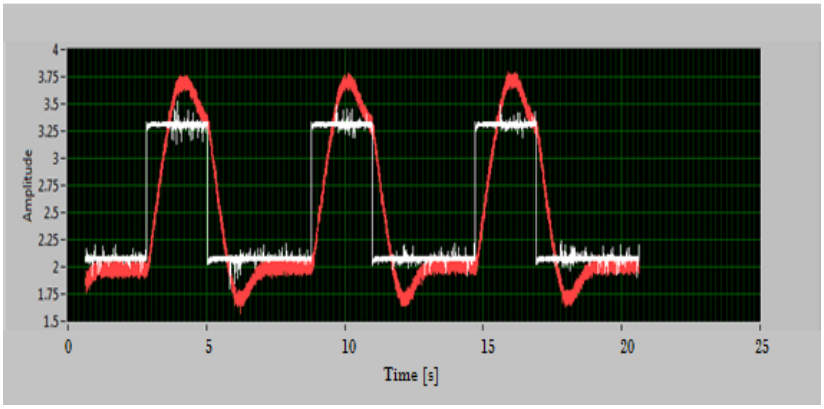


Figura 6.29. Respuesta a una entrada tren de pulsos del controlador PD, aplicado al motor de un robot móvil en una superficie de concreto.

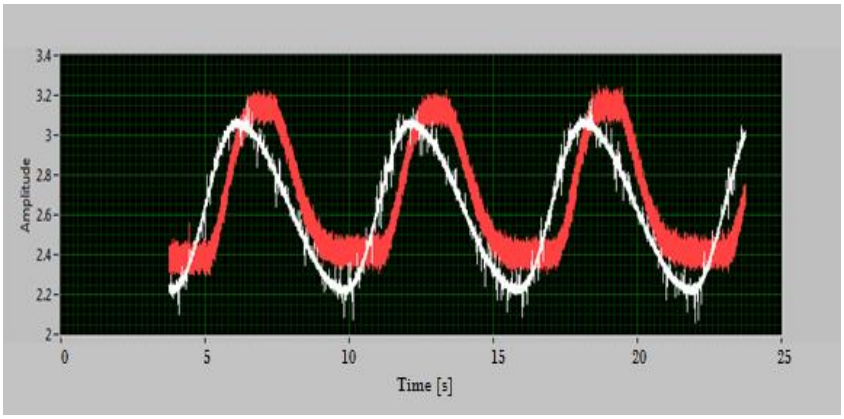


Figura 6.30. Respuesta a una entrada seno del controlador PD, aplicado al motor de un robot móvil en una superficie de concreto.

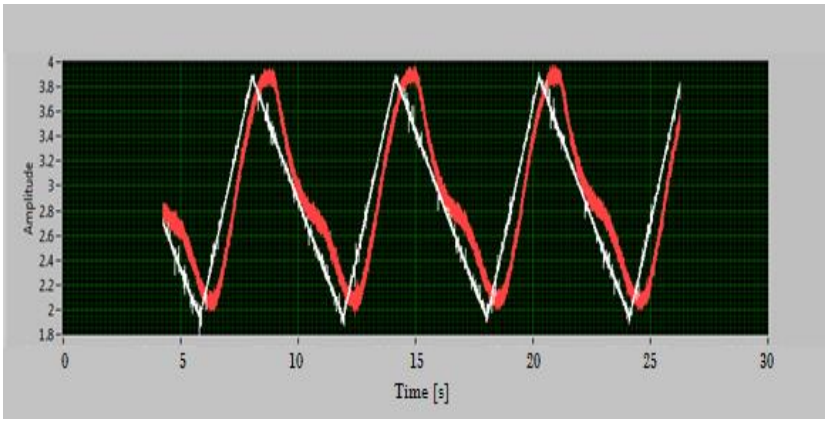


Figura 6.31. Respuesta a una entrada diente de sierra del controlador PD, aplicado al motor de un robot móvil en una superficie de concreto.

En segundo lugar se muestran las gráficas del desempeño del controlador PD difuso, empezando por la señal tren de pulsos mostrada en la figura 6.32, la señal seno mostrada en la figura 6.33 y la señal diente de sierra mostrada en la figura 6.34.

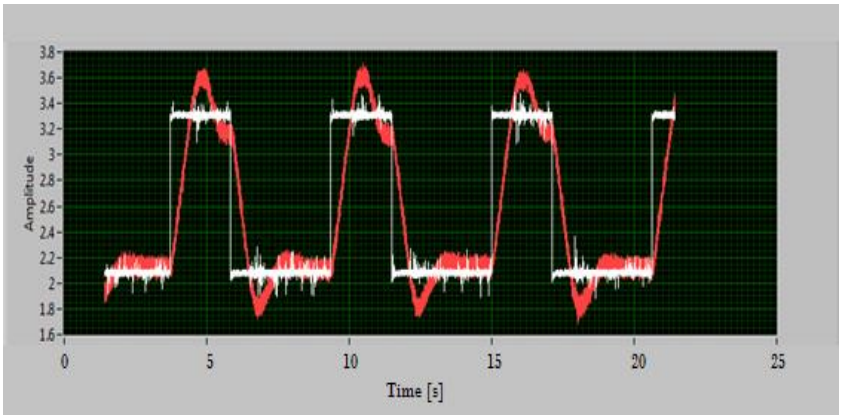


Figura 6.32. Respuesta a una entrada tren de pulsos del controlador PD difuso, aplicado al motor de un robot móvil en una superficie de concreto.

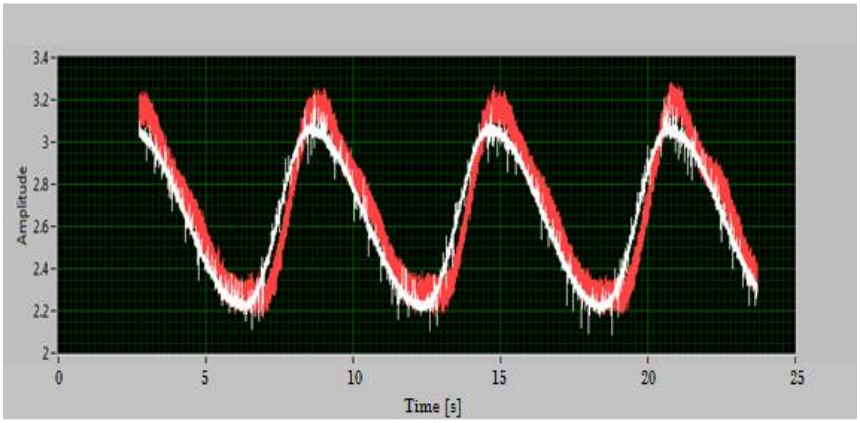


Figura 6.33. Respuesta a una entrada seno del controlador PD difuso, aplicado al motor de un robot móvil en una superficie de concreto.

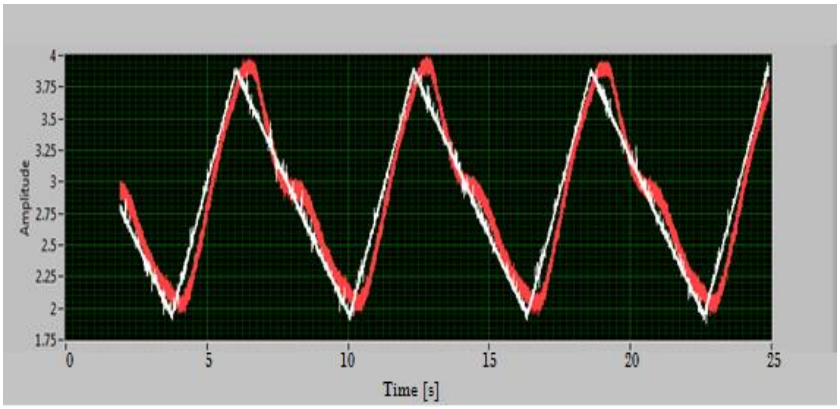


Figura 6.34. Respuesta a una entrada diente de sierra del controlador PD difuso, aplicado al motor de un robot móvil en una superficie de concreto.

Por último el desempeño del controlador PD neurodifuso para las tres señales de prueba tren de pulsos, seno y diente de sierra se muestran en las figuras 6.35, 6.36 y 6.37 respectivamente.

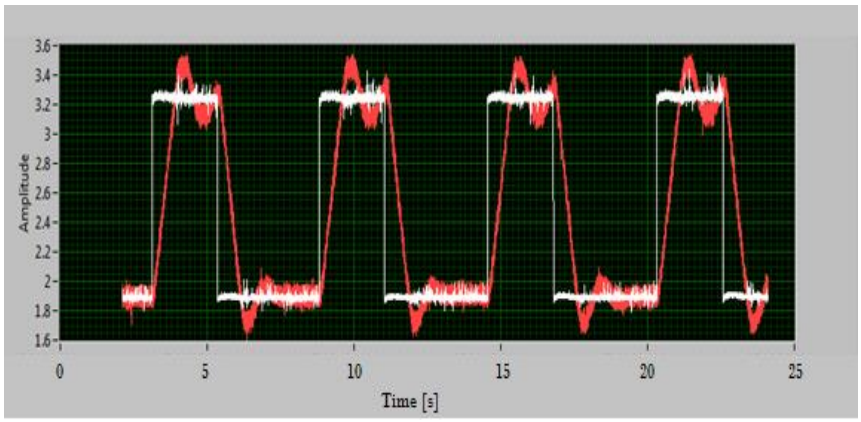


Figura 6.35. Respuesta a una entrada tren de pulsos del controlador PD neurodifuso, aplicado al motor de un robot móvil en una superficie de concreto.

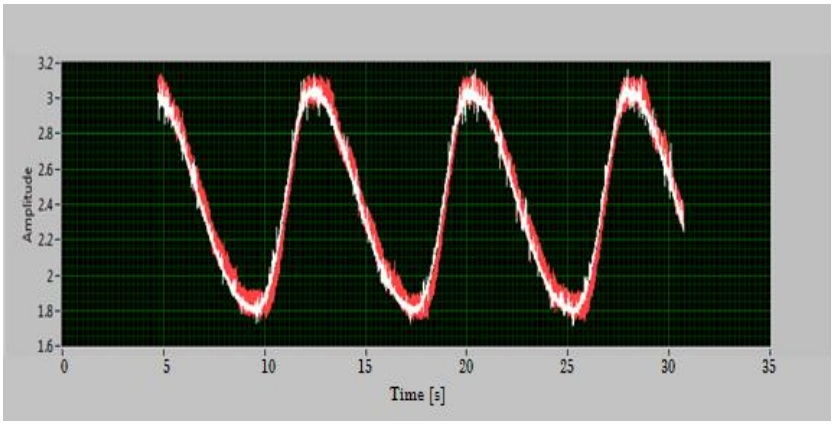


Figura 6.36. Respuesta a una entrada seno del controlador PD neurodifuso, aplicado al motor de un robot móvil en una superficie de concreto.

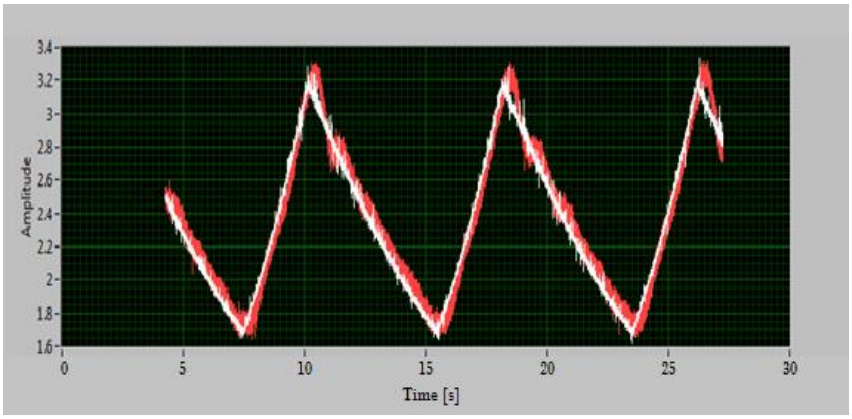


Figura 6.37. Respuesta a una entrada diente de sierra del controlador PD neurodifuso, aplicado al motor de un robot móvil en una superficie de concreto.

Como se puede observar en las gráficas anteriores el desempeño del controlador PD neurodifuso es el mejor ya que responde más rápido a las señales de prueba.

Comparando el desempeño de los tres controladores en las diferentes superficies se puede concluir que el desempeño de los controladores varía dependiendo de la superficie, esto debido a que cada superficie tiene una fricción diferente, haciendo que el controlador sea más lento o más rápido. Entre más fricción el controlador es más lento y entre menos fricción el controlador es más rápido.

Los resultados de los tres controladores en las diferentes superficies, se muestran en la tabla 6.4, se puede observar que los tres controladores respondieron más

rápido en la superficie de formica, esto debido a que esta superficie tiene una menor fricción, y por el contrario los tres controladores tuvieron una respuesta más lenta en la superficie de concreto, debido a su mayor fricción.

Superficie de formica	PD	PD difuso	PD neurodifuso
Sobrepaso máximo (Smax)	25%	12.5%	12.5%
Tiempo de retardo (td)	0.35s	0.2s	0.2s
Tiempo de levantamiento (tr)	0.65s	0.3s	0.3s
Tiempo de asentamiento (ts)	2.4s	1.3s	0.8s
Error en estado estable	0.12	0.06	0.06
Frecuencia natural (Wn)	2.4	5.2	5.2
Factor de amortiguamiento (ζ)	0.69	0.59	0.96
Superficie de madera barnizada			
Sobrepaso máximo (Smax)	25%	18.75%	12.5%
Tiempo de retardo (td)	0.35s	0.2s	0.2s
Tiempo de levantamiento (tr)	0.65s	0.3s	0.3s
Tiempo de asentamiento (ts)	1.75s	1s	0.8s
Error en estado estable	0.12	0.10	0.10
Frecuencia natural wn	2.4	5.2	5.2
Factor de amortiguamiento	0.95	0.77	0.96
Superficie de concreto			
Sobrepaso máximo (Smax)	28.57%	14.28%	14.28%
Tiempo de retardo (td)	0.4s	0.4s	0.4s

Tiempo de levantamiento (tr)	**0.8s**	**0.8s**	**0.8s**
Tiempo de asentamiento (ts)	**2.4s**	**2.2s**	**1.5s**
Error en estado estable	**0.12**	**0.20**	**0.10**
Frecuencia natural wn	**1.96**	**1.96**	**1.96**
Factor de amortiguamiento ζ	**0.85**	**0.92**	**1.36**

Tabla 6.4. Características principales de la respuesta de los tres controladores a una entrada escalón, aplicados al motor de un robot móvil en tres superficie diferentes.

6.2. Conclusiones.

Este trabajo se hizo con el propósito de diseñar y comparar el desempeño de tres controladores un controlador PD digital, un controlador PD difuso y un controlador PD neurodifuso aplicados al control de la dirección de un robot móvil en configuración triciclo. Se probó el desempeño sobre tres diferentes superficies formica, madera barnizada y concreto.

El análisis del desempeño del controlador se observa en la tabla 6.4, en donde se muestran los parámetros obtenidos para cada controlador.

Sobrepaso máximo. Entre más grande sea el sobrepaso máximo menor es el desempeño del controlador.

Tiempo de retardo, levantamiento y asentamiento. Entre más grandes son estos parámetros más lento el controlador y entre más pequeños es más rápido el controlador.

Error. Idealmente el error en un controlador debe de ser cero, sin embargo en la realidad el error en un controlador siempre va existir aunque sea muy pequeño, lo que se busca cuando se diseña un controlador es que el error sea lo más pequeño posible.

Frecuencia natural. La frecuencia natural tiene un efecto directo con los tiempos de retardo, levantamiento y asentamiento, es decir si la frecuencia es mayor los tiempos se reducen, también si la frecuencia es mayor indica que los polos del sistema están más lejos del origen del plano s, es decir el sistema es más estable.

Factor de amortiguamiento relativo. Si el factor de amortiguamiento se aproxima a cero indica que el sistema es menos estable, es decir que puede llegar a oscilar.

PID digital. De acuerdo a los resultados de la tabla 6.4 el desempeño del controlador en la superficie formica y de madera barnizada fue parecido ya que el sobrepaso máximo, el tiempo de retardo, el tiempo de levantamiento y el error

tuvieron los mismos valores, sin embargo el tiempo se asentamiento fue más pequeño en la superficie de madera barnizada, esto se debe a que el factor de amortiguamiento en esta superficie es mayor lo que indica que el sistema es más estable en esta superficie. En la superficie de concreto los parámetros (smax, td, tr y ts) del controlador son mayores, esto debido a que la frecuencia natural y el factor de amortiguamiento disminuyeron, cuando estos factores disminuyen el sistema se hace más inestable.

PD difuso. De acuerdo a la tabla 6.4 el controlador tuvo un mejor desempeño en la superficie formica teniendo menor el sobrepaso máximo y el error, debido a que la frecuencia natural fue mayor en esta superficie los tiempos de retardo, levantamiento y asentamiento fueron menores en esta superficie, es decir el controlador PD difuso tuvo un mejor desempeño en la superficie de formica.

PD neurodifuso. Los resultados de la tabla 6.4 indican que este controlador en la superficie formica tuvo un mejor desempeño, con error de 0.06 menor que en las otras superficies y una frecuencia natural mayor de 5.2 que ocasiono que los tiempos de retardo levantamiento y asentamiento fueran menores.

En general los tres controladores tuvieron un mejor desempeño en la superficie formica, sin embargo el controlador PD neurodifuso tuvo el mejor desempeño de los tres teniendo un error de 0.06, una frecuencia natural de 5.2 que ocasionó que los tiempos de retardo, levantamiento y asentamiento fueran menores, también el sobrepaso máximo fue de 12.5% menor que los demás y un factor de amortiguamiento mayor de 0.96.

En conclusión el controlador PD neurodifuso es el controlador que tiene un mejor desempeño en las tres superficies y mantiene al sistema más estable en la superficie formica, en consecuencia es el controlador que responde mejor a las tres señales de prueba (tren de pulsos, seno y diente de sierra).

Este trabajo resulta relevante ya que posicionar con exactitud la llanta de dirección de un robot móvil tipo triciclo es un aspecto fundamental para establecer la trayectoria a seguir por el robot, de la misma forma esto permitirá estimar la posición y orientación del mismo, estos dos aspectos son fundamentales para el control y la navegación autónoma.

El posicionar con exactitud la dirección del robot sin importar la superficie en la que se encuentra, permitirá como trabajo futuro el desarrollo de tareas como la planificación de caminos y trayectorias, evitar obstáculos, seguimiento de patrones, etc.

Como trabajo posterior se deberán implementar los algoritmos de navegación al robot los cuales deberán tener un buen desempeño considerando que el posicionamiento de la llanta de dirección se realizó con una buena exactitud y precisión, permitiendo una navegación confiable.

Bibliografía.

[1] Ogata, Katsuhiko. *Ingeniería de control moderna*, Prentice Hall, Estado de México, pag 997.

[2] Josep M. Nogués. Robots de servicio como sistemas de control y vigilancia móviles. [en línea].

[3] <http://www.borrmart.es/articulo_seguritecnia.php?id=1033&numero=319> [citado en 19 de Mayo de 2012].

[4] José Andrés Vicente Lober. Técnicas de inteligencia artificial en la construcción de robots móviles autónomos. Universidad de Salamanca. Julio 2003.

[5] Juan González Gómez. Robótica modular y locomoción. Universidad Autónoma de Madrid. Noviembre 2006.

[6] J. Ruíz del Solar. R. Salazar. Robots Móviles. Universidad de Chile.

[7] P. Reynoso Mora y D. Mocencahua Mora. Sintonización Difusa de un PID para Robots Manipuladores.

[8]. Isasi, Viñuela, Pedro, et.al. *Redes de neuronas artificiales. Un enfoque práctico*. Prentice Hall. Madrid, 2004, 248 p.p.

[9] Benjamín C. Kuo, Sistemas de Control Automático, Prentice Hall, México D.F, pag.897.

[10]. Gilberto Reynoso Meza. Notas en Control Difuso. Monterrey, Junio 2005, pag 23. ITESM Campus Monterrey. Departamento de Mecatrónica y Automatización.

[11]. Essam Natsheh.et.al. Comparison Between Conventional and Fuzzy Logic PID Controllers for Controlling DC Motors. Rule-base Matrix of the PID-like FLC. International Journal of Computer Science Issues. Vol 7, Issue 5, Septiembre 2010.

[12]. P. Reynoso Mora. et.al. Sintonización Difusa de un PID para Robots Manipuladores. Sintonización difusa del PID. Seminario de control difuso, Marzo 2004.

APÉNDICE. Códigos de los programas en lenguaje C de los controladores en el compilador mikroC.

1. Controlador PD.

```
float lsb = 0.00488;   //Resolución del ADC.
float Kp=2;          //Parámetro de proporcionalidad.
float Kd=1;          //Parámetro de la derivada.
int v = 3;           //Voltaje deseado.
float ref;           //variable para el valor deseado.
float sensor;          //variable para el valor medido.
float u;             //variable para la suma.
float error;           // variable para el error.
float proporcional=0;    //variable para la parte proporcional.
float derivada=0;        //variable para la parte derivativa.
float T0=0;      //variable para la multiplicación de kp*erroranterior.
void interrupt()           //Función de la interrupción.
{
  if (INTCON.T0IF==1)  //Verifica que la bandera TIOF se active.
  {
  ref = v/lsb;       //Calculo del valor deseado.
 sensor = ADC_Read(0);  //Lee la entrada del ADC.
  error = sensor - ref;    // Calculo del error.
  proporcional = Kp*error;   //Calculo de la parte proporcional
  derivada = (Kd*error)-T0;     //Calculo de la parte derivativa.
  //Suma de la parte proporcional, integral y derivativa.
  u = proporcional+integral+derivada;
  u = floor(u); // Redondea el valor de u.
  u = (int)u;   //Convierte el valor de u a un entero.
  u = abs(u);  //Determina el valor absoluto de u.
  if (error<0)    //Verifica si el error es menor a cero
  {
     PORTA = 0x02;
  }
  if (error>0)  //Verifica si el error es mayor a cero
  {
     PORTA = 0x04;
  }

  if (u>255)   //Verifica que u no sea mayor a 2^8.
```

```
    {
      PWM1_Set_Duty(255);  //Asignación de trabajo del PWM
    if (u<255)  //Verifica que u no sea menor a 2^8.
    {
     PWM1_Set_Duty(u);    //Asignación de trabajo del PWM
    }
    T0=Kd*error;     //Multiplica Kd*erroranterior.
    }
  }
void main()          //Función principal
{
  TRISA = 0x01;   //Configura el pin A0 como entrada y los demás como salida
  PORTA = 0x00;    //Inicializa el puerto A en cero.
  ANSEL = 0X01;    //Configura el pin A0 como entrada digital.
  OPTION_REG = 0x05;   //Configura la preescala del timer0.
  INTCON = 0xA0;       //Configura el timer0.
  PWM1_Start();        //Inicializa el PWM.
  for(;;);             //Ciclo infinito.
 }
```

2. Controlador Difuso.

```
   float lsb = 0.00488;   //Resolucion del ADC.
  int v = 2;              //Voltaje deseado.
  int ref;               //variable para el valor deseado.
  int sensor;            //variable para el valor medido.
  float u;               //variable para la suma.
  float error;           // variable para el error.
  int uval[5]={20,40,90,120,160}; //variable ui
  float peso[5];                 //variable para grados de membresía
  int i;
  float suma=0;
  float suma1=0;
  void interrupt()           //Función de la interrupción.
  {
    if (INTCON.T0IF==1)  //Verifica que la bandera TIOF se active.
    {
        for(i=0;i<5;i++)
       {
         peso[i]=0;       //Inicializa los grados de membresía en cero
       }
        suma=0;
        suma1=0;
       sensor = ADC_Read(0);  //Lee la entrada del ADC.
       //ref = ADC_Read(3);  //Lee la entrada del ADC.
```

```
ref = v/lsb;        //Calculo del valor deseado.
error = ref - sensor;    // Calculo del error.
if (error<0)    //Verifica si el error es menor a cero
{
   PORTA = 0x02;
}
if (error>0)  //Verifica si el error es mayor a cero
{
   PORTA = 0x04;
}
error = abs(error);   //Determina el valor absoluto del error.
/*Calculo de los grados de membresía*/
if (error<=20)
{
  peso[0]=0;
}
else if (error<=40)
{
  peso[0]=1;
} else  if (error<=60)
{
   peso[0]= (-0.05*error)+3;
   peso[1]= (0.033*error)-1.3;
} else if (error<=70)
{
   peso[1] = (0.033*error)-1.3;
}  else if (error<=80)
{
   peso[1] = (-0.05*error)+4.5;
} else if (error<90)
{
   peso[1] = (-0.05*error)+4.5;
   peso[2] = (0.05*error)-4;
} else if (error<=100)
{
   peso[2] = (0.05*error)-4;
} else if (error<=110)
{
   peso[2] = (-0.05*error)+6;
} else if (error<120)
{
   peso[2] = (-0.05*error)+6;
   peso[3] = (0.05*error)-5.5;
}  else if (error<=130)
{
   peso[3] = (0.05*error)-5.5;
}  else if (error<=140)
```

```
        {
           peso[3] = (-0.05*error)+7.5;
        }  else if (error<150)
        {
           peso[3] = (-0.05*error)+7.5;
           peso[4] = (0.05*error)-7;
        } else if (error>=160)
        {
           peso[4] = (0.05*error)-7;
        } else if (error>160)
        {
           peso[4] = 1;
        }
        /* Caculo de Singletons*/
        for(i=0;i<5;i++)
        {
          suma = (peso[i]*uval[i])+suma;
          suma1 = peso[i]+suma1;
        }
        u = suma/suma1;
        u = floor(u); // Redondea el valor de u.
        u = (int)u;   //Convierte el valor de u a un entero.
        u = abs(u);  //Determina el valor absoluto de u.   */
        if (u>255)   //Verifica que u no sea mayor a 2^8.
        {
          PWM1_Change_Duty(255);  //Asignacion de trabajo del PWM
        }
        if (u<255)  //Verifica que u no sea menor a 2^8.
        {
          PWM1_Change_Duty(u);    //Asignación de trabajo del PWM
        }
         INTCON.T0IF=0;
      }
   }
void main()          //Funcion principal
{
   TRISA = 0x01;   //Configura el pin A0 como entrada y los demas como salida
   PORTA = 0x00;    //Inicializa el puerto A en cero.
   ANSEL = 0X01;    //Configura el pin A0 como entrada digital.
   OPTION_REG = 0x03;   //Configura la preescala del timer0.
   INTCON = 0xA0;       //Configura el timer0.
   PWM1_Start();        //Inicializa el PWM.
   for(;;);              //Ciclo infinito.
 }
```

3. Controlador PID difuso.

```
float lsb = 0.00488;   //Resolución del ADC.
float Kp=0.7;          //Parámetro de proporcionalidad.
float Kd=0.01;       //Parámetro de derivación
float Kfp;           // Parámetro variable de proporcionalidad.
float Kfd;           // Parámetro variable de derivación.
int v = 2;             //Voltaje deseado.
int ref;             //variable para el valor deseado.
int sensor;          //variable para el valor medido.
float u;             //variable para la suma.
float error=0;           // variable para el error.
float camerror=0;   //variable para el cambio de error
float pesomin;       //variable para el mínimo grado de membresía.
float peso1;        //variable para el grado de membresía.
float peso2;        //variable para el grado de membresía.
float pesocammin;    //variable para el mínimo grado de membresía.
float pesocam1;        //variable para el grado de membresía.
float pesocam2;      //variable para el grado de membresía.
int i;                 //variable para una iteración
float funmin1=0;
float funmin2=0;
float funmin3=0;
float sumapro=0;
float sumapro1=0;
float sumaderi=0;
float sumaderi1=0;
float proporcional=0;    //variable para la parte propocional.
float derivada=0;        //variable para la parte derivativa.
float T0=0;      //variable para la multiplicacion de kp*erroranterior.
float error0=0;
float proporcionalerror();
float diferenciaerror();
float calculo();
void interrupt()          //Función de la interrupción.
{
  if (INTCON.T0IF==1)  //Verifica que la bandera TIOF se active.
  {
      peso1=0;
      peso2=0;
      pesomin=0;
      pesocam1=0;
      pesocam2=0;
      pesocammin=0;
      sumapro=0;
      sumapro1=0;
```

```
 sumaderi=0;
 sumaderi1=0;
sensor = ADC_Read(0);  //Lee la entrada del ADC.
ref = v/lsb;        //Calculo del valor deseado.
error = ref - sensor;    // Calculo del error.
camerror = error - error0;
if (error<0)    //Verifica si el error es menor a cero
{
   PORTA = 0x02;
}
if (error>0)  //Verifica si el error es mayor a cero
{
   PORTA = 0x04;
}
if (error>0 && camerror<0)
{
    Kfp=Kp;
    Kfi=Ki;
    Kfd=Kd;
}
else if(error<0 && camerror<0)
{
  calculo();
}
else if (error<0 && camerror>0)
{
   Kfp=Kp;
   Kfi=Ki;
   Kfd=Kd;
}
else if (error>0 && camerror>0)
{
   calculo();
}
else if(error==0 && camerror==0)
{
   Kfp=0;
   Kfd=0;
   Kfi=0;
}
error = ref - sensor;    // Calculo del error.
proporcional = Kfp*error;  //Calculo de la parte proporcional
derivada = (Kfd*error)-T0;   //Calculo de la parte derivada
//Suma de la parte proporcional y derivativa.
u = proporcional+derivada;
u = floor(u); // Redondea el valor de u.
u = (int)u;   //Convierte el valor de u a un entero.
```

```
        u = abs(u);  //Determina el valor absoluto de u.   */
        if (u>255)   //Verifica que u no sea mayor a 2^8.
        {
          PWM1_Change_Duty(255);  //Asignacion de trabajo del PWM
        }
        if (u<255)  //Verifica que u no sea menor a 2^8.
        {
         PWM1_Change_Duty(u);    //Asignación de trabajo del PWM
        }
        T0=Kfd*error;     //Multiplica Kd*erroranterior.
        error0 = error;
        INTCON.T0IF=0;
    }
  }
void main()          //Funcion principal
{
  TRISA = 0x09;   //Configura el pin A0 como entrada y los demas como salida
  PORTA = 0x00;    //Inicializa el puerto A en cero.
  ANSEL = 0X09;    //Configura el pin A0 como entrada digital.
  OPTION_REG = 0x03;   //Configura la preescala del timer0.
  INTCON = 0xA0;       //Configura el timer0.
  PWM1_Start();        //Inicializa el PWM.
  for(;;);
 }
float proporcionalerror()
{
        if (error<=60)
        {
           peso1= (-0.0166*error)+1;
           peso2= (0.033*error)-1.3;
           if (peso1<=peso2)
           {
             pesomin=peso1;
           }
           else if (peso2<=peso1)
           {
              pesomin=peso2;
           }
        } else if (error<=70)
        {
           pesomin = (0.033*error)-1.3;
        }  else if (error<=80)
        {
           pesomin = (-0.05*error)+4.5;
        } else if (error<90)
        {
           peso1 = (-0.05*error)+4.5;
```

```
    peso2 = (0.05*error)-4;
    if (peso1<=peso2)
    {
      pesomin=peso1;
    }
    else if (peso2<=peso1)
    {
       pesomin=peso2;
    }
} else if (error<=100)
{
    pesomin = (0.05*error)-4;
} else if (error<=110)
{
    pesomin = (-0.05*error)+6;
} else if (error<120)
{
    peso1 = (-0.05*error)+6;
    peso2 = (0.05*error)-5.5;
    if (peso1<=peso2)
    {
      pesomin=peso1;
    }
    else if (peso2<=peso1)
    {
       pesomin=peso2;
    }
}  else if (error<=130)
{
    pesomin = (0.05*error)-5.5;
}  else if (error<=140)
{
    pesomin = (-0.05*error)+7.5;
}  else if (error<150)
{
    peso1 = (-0.05*error)+7.5;
    peso2 = (0.05*error)-7;
    if (peso1<=peso2)
    {
      pesomin=peso1;
    }
    else if (peso2<=peso1)
    {
       pesomin=peso2;
    }
} else if (error>=150)
{
```

```
            pesomin = (0.05*error)-7;
        } else if (error>160)
        {
            pesomin = 1;
        }
        return (pesomin);
}
 float diferenciaerror()
 {
        if (camerror<=60)
        {
            pesocam1= (-0.0166*camerror)+1;
            pesocam2= (0.033*camerror)-1.3;
            if (pesocam1<=pesocam2)
            {
                pesocammin=pesocam1;
            }
            else if (pesocam2<=pesocam1)
            {
                pesocammin=pesocam2;
            }
        } else if (camerror<=70)
        {
            pesocammin = (0.033*camerror)-1.3;
        }  else if (camerror<=80)
        {
            pesocammin = (-0.05*camerror)+4.5;
        } else if (camerror<90)
        {
            pesocam1 = (-0.05*camerror)+4.5;
            pesocam2 = (0.05*camerror)-4;
             if (pesocam1<=pesocam2)
            {
                pesocammin=pesocam1;
            }
            else if (pesocam2<=pesocam1)
            {
                pesocammin=pesocam2;
            }
        } else if (camerror<=100)
        {
            pesocammin = (0.05*camerror)-4;
        } else if (camerror<=110)
        {
            pesocammin = (-0.05*camerror)+6;
        } else if (camerror<120)
        {
```

```
            pesocam1 = (-0.05*camerror)+6;
            pesocam2 = (0.05*camerror)-5.5;
             if (pesocam1<=pesocam2)
            {
               pesocammin=pesocam1;
            }
            else if (pesocam2<=pesocam1)
            {
               pesocammin=pesocam2;
            }
         }  else if (camerror<=130)
         {
            pesocammin = (0.05*camerror)-5.5;
         }  else if (camerror<=140)
         {
            pesocammin = (-0.05*camerror)+7.5;
         }  else if (camerror<150)
         {
            pesocam1 = (-0.05*camerror)+7.5;
            pesocam2 = (0.05*camerror)-7;
             if (pesocam1<=pesocam2)
            {
               pesocammin=pesocam1;
            }
            else if (pesocam2<=pesocam1)
            {
               pesocammin=pesocam2;
            }
         } else if (camerror>=160)
         {
            pesocammin = (0.05*camerror)-7;
         } else if (camerror>160)
         {
            pesocammin= 1;
         }
         return (pesocammin);
}
float calculo()
{
            error = abs(error);
            camerror = abs(camerror);
            sumaerror = abs(sumaerror);
            proporcionalerror();
            diferenciaerror();
            //Calculo de Kfp.
            sumapro=pesomin*kp+sumapro;
```

```
        if (pesomin<=pesocammin)
        {
           funmin1 = pesomin;
        }
        else if (pesocammin<=pesomin)
        {
           funmin1 = pesocammin;
        }
        sumapro1= funmin1+sumapro1;
        Kfp = sumapro/sumapro1;
        //Calculo de Kfd.
        sumaderi=pesocammin*kd+sumaderi;
        sumaderi1= funmin1+sumaderi1;
        Kfd = sumaderi/sumaderi1;
        return (Kfp,Kfd);
      }
```

4. Aprendizaje de la neurona artificial.

A continuación se muestra el programa del algoritmo Delta para el aprendizaje de la neurona artificial.

```
#include<stdio.h>
#include<stdlib.h>

main ()
{
   int X[14]={20,40,60,80,100,120,140,160,180,200,400,600,800,1024};
   float d[14]={10,10,40,40,90,120,120,160,180,180,200,220,220,240};
   float W=0.1;
   float CW=0;
   float alfa = 0.000001;
   float y=0;
   int i;
   int k;
   for (k=0;k<10;k++)
   {
         for (i=0;i<14;i++)
         {
            y= X[i]*W;
            CW= alfa*(d[i]-y)*X[i];
            W=W+CW;
            CW=0;
            y=0;
         }
   }
   printf ("%.6f\n",W);
   system("pause");
}
```

Después de 5 iteraciones el programa convergió y el peso adecuado para la neuronal artificial es de w = 0.230908.

5. Control neurodifuso.

```
float lsb = 0.00488;   //Resolucion del ADC.
int v = 2;             //Voltaje deseado.
int ref;              //variable para el valor deseado.
int sensor;           //variable para el valor medido.
float u;              //variable para la suma.
float error;           // variable para el error
void interrupt()           //Función de la interrupción.
{
  if (INTCON.T0IF==1)  //Verifica que la bandera TIOF se active.
  {
     sensor = ADC_Read(0);  //Lee la entrada del ADC.
     ref = v/lsb;        //Calculo del valor deseado.
     error = ref - sensor;    // Calculo del error.
     if (error<0)    //Verifica si el error es menor a cero
     {
        PORTA = 0x02;
     }
     if (error>0)  //Verifica si el error es mayor a cero
     {
        PORTA = 0x04;
     }
     error = abs(error);
     u = error*0.230908;
     u = floor(u); // Redondea el valor de u.
     u = (int)u;   //Convierte el valor de u a un entero.
     u = abs(u);  //Determina el valor absoluto de u.   */
     if (u>255)   //Verifica que u no sea mayor a 2^8.
     {
       PWM1_Change_Duty(255);  //Asignacion de trabajo del PWM
     }
     if (u<255)  //Verifica que u no sea menor a 2^8.
     {
      PWM1_Change_Duty(u);    //Asignación de trabajo del PWM
     }
     INTCON.T0IF=0;
  }
}
```

```
void main()          //Funcion principal
{
  TRISA = 0x01;   //Configura el pin A0 como entrada y los demas como salida
  PORTA = 0x00;    //Inicializa el puerto A en cero.
  ANSEL = 0X01;    //Configura el pin A0 como entrada digital.
  OPTION_REG = 0x03;   //Configura la preescala del timer0.
  INTCON = 0xA0;       //Configura el timer0.
  PWM1_Start();        //Inicializa el PWM.
  for(;;);
 }
```

6. Aprendizaje del controlador PID neurodifuso.

El programa en el software Dev C++ para encontrar el peso adecuado que sintonice el parámetro K_{Fp} se muestra a continuación:

```
#include<stdio.h>
#include<stdlib.h>
main ()
{
    float X[5]={41.66,81.96,122.95,163.93,204.91};
    float d[5]={2,1.95,1.97,2.057,2.13};
    float W=0.001;
    float CW=0;
    float alfa = 0.0001;
    float y=0;
    int i;
    int k;
    for (k=0;k<10;k++)
    {
          for (i=0;i<5;i++)
          {
             y= X[i]*W;
             CW= alfa*(d[i]-y)*X[i];
             W=W+CW;
             CW=0;
             y=0;
          }
    }
    printf ("%.6f\n",W);
```

```
    system("pause");
}
```

El siguiente programa es para obtener el peso adecuado que determine el parámetro KFd del controlador PID.

```
#include<stdio.h>
#include<stdlib.h>
main ()
{
    float X[5]={41.66,81.96,122.95,163.93,204.91};
    float d[5]={0.00910,0.00913,0.00918,0.00923,0.00927};
    float W=0.001;
    float CW=0;
    float alfa = 0.0001;
    float y=0;
    int i;
    int k;
    for (k=0;k<100;k++)
    {
          for (i=0;i<5;i++)
          {
             y= X[i]*W;
             CW= alfa*(d[i]-y)*X[i];
             W=W+CW;
             CW=0;
             y=0;
          }
    }
    printf ("%.6f\n",W);
    system("pause");
}
```

7. Control PID neurodifuso.

El programa en el software mikroC para la implementación en el microprocesador es el siguiente:

```
  float lsb = 0.00488;   //Resolucion del ADC.
  float Kp=2;             //Parámetro de proporcionalidad.
  float Kd=1;            //Parámetro derivativo.
  float wp=0.008520;    //Peso uno
```

```
float wd=0.000043;    //Peso dos
int v = 2;              //Voltaje deseado.
int ref;              //variable para el valor deseado.
int sensor;           //variable para el valor medido.
float u;              //variable para la suma.
int i;
float proporcional=0;    //variable para la parte propocional.
float derivada=0;         //variable para la parte derivativa.
float T0=0;      //variable para la multiplicación de kp*erroranterior.
float error=0;
float Kfp=0;
float Kfd=0;
float camerror=0;
float error0=0;
void interrupt()           //Función de la interrupción.
{
  if (INTCON.T0IF==1)  //Verifica que la bandera TIOF se active.
  {
     sensor = ADC_Read(0);  //Lee la entrada del ADC.
     ref = v/lsb;        //Calculo del valor deseado.
     error = ref - sensor;    // Calculo del error.
     camerror = error - error0;  //Calculo del cambio del error
     if (error<0)    //Verifica si el error es menor a cero
     {
        PORTA = 0x02;
     }
     if (error>0)  //Verifica si el error es mayor a cero
     {
        PORTA = 0x04;
     }
     if (error>0 && camerror<0)
     {
         Kfp=Kp;
         Kfd=Kd;
     }
     else if(error<0 && camerror<0)
     {
         error=abs(error);
         Kfp=0.008520*error;
         Kfd=0.000043*error;
     }
     else if (error<0 && camerror>0)
     {
        Kfp=Kp;
        Kfd=Kd;
     }
```

```
        else if (error>0 && camerror>0)
        {
            error=abs(error);
            Kfp=0.008520*error;
            Kfd=0.000043*error;
        }
        else if(error==0 && camerror==0)
        {
           Kfp=0;
           Kfd=0;
        }
        error = ref - sensor;    // Calculo del error.
        proporcional = Kfp*error;
        derivada = (Kfd*error)-T0;
    //Suma de la parte proporcional, integral y derivativa.
        u = proporcional+derivada;
        u = floor(u); // Redondea el valor de u.
        u = (int)u;   //Convierte el valor de u a un entero.
        u = abs(u);  //Determina el valor absoluto de u.   */
        if (u>255)   //Verifica que u no sea mayor a 2^8.
        {
          PWM1_Change_Duty(255);  //Asignacion de trabajo del PWM
        }
        if (u<255)  //Verifica que u no sea menor a 2^8.
        {
         PWM1_Change_Duty(u);    //Asignación de trabajo del PWM
        }
        T0=Kd*error;     //Multiplica Kd*erroranterior.
        INTCON.T0IF=0;
    }
  }

void main()          //Funcion principal
{
  TRISA = 0x09;   //Configura el pin A0 como entrada y los demas como salida
  PORTA = 0x00;    //Inicializa el puerto A en cero.
  ANSEL = 0X09;    //Configura el pin A0 como entrada digital.
  OPTION_REG = 0x03;   //Configura la preescala del timer0.
  INTCON = 0xA0;       //Configura el timer0.
  PWM1_Start();        //Inicializa el PWM.
  for(;;);
 }
```

Glosario

Sistema. Es el resultado de la interacción de partes o componentes de una manera única y tiene asociadas una o más señales denominadas entradas y una o más señales denominadas salidas.

Señal eléctrica. Es generada por algún fenómeno electromagnético, esta señal puede ser analógica, si varia de forma continua en el tiempo, o digital si varia de forma discreta (con valores dados como 0 y 1).

Mecanismo. Es el conjunto de sólidos resistentes, elementos elásticos unidos entre sí, mediante diferentes tipos de uniones.

Motor. Es el encargado de transformar algún tipo de energía (eléctrica, de combustibles fósiles, etc), en energía mecánica capaz de realizar un trabajo.

Motor cd. Un motor de corriente directa es una máquina que convierte energía eléctrica en movimiento o trabajo mecánico a través de medios electromagnéticos.

Servomecanismo. Es un sistema formado de partes mecánicas y electrónicas que en ocasiones son usadas en robots con parte móvil o fija.

Automatización. Es el uso de sistemas o elementos computarizados y electromecánicos para controlar maquinarias y/o procesos industriales.

Algoritmo. Es un conjunto de instrucciones o reglas bien definidas, ordenadas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad.

Diagrama de flujo. Es la representación gráfica del algoritmo o proceso.

Microcontrolador. Es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Sus tres principales unidades de funcionamiento son: la unidad central de procesamiento, la memoria y los periféricos de entrada y salida.

Oscilador. Es un circuito capaz de convertir la corriente continua en una corriente que varía de forma periódica en el tiempo; estas oscilaciones pueden ser senoidales, cuadradas, triangulares, etc.,

Temporizador. Es un dispositivo, con frecuencia programable, que permite medir el tiempo.

Convertidor analógico digital. Es un dispositivo electrónico capaz de convertir una señal analógica de voltaje en una señal digital con un valor binario.

Memoria EEPROM. Es un tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente.

Memoria Flash. Permite la lectura y escritura de múltiples posiciones de memoria en la misma operación.

PWM. La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de *pulse-width modulation*) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica.

Puente h. Es un circuito electrónico que permite a un motor de cd girar en ambos sentidos, avance y retroceso.

Función de transferencia. Es un modelo matemático que a través de un cociente relaciona la respuesta de un sistema a una señal de entrada o excitación.

Respuesta al escalón unitario. Es la respuesta de un sistema de control cuando la entrada es una función al escalón unitario.

Función al escalón unitario. Es una función cuyo valor es 0 para cualquier argumento negativo, y 1 para cualquier argumento positivo:

$$\forall x \in \mathbb{R}: \quad u(x) = H(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ 1 & \text{si} \quad x \geq 0 \end{cases}$$

Sobrepaso máximo. Es el valor pico máximo de la curva de respuesta, medido a partir de la unidad.

Tiempo de retardo (td). Se define como el tiempo requerido para que la respuesta al escalón alcance el 50% de su valor final.

Tiempo de levantamiento (tr). Se define como el tiempo requerido para que la respuesta al escalón se eleve del 10 al 90% de su valor final.

Tiempo de asentamiento (ts). Se define como el tiempo requerido para que la respuesta al escalón disminuya y permanezca dentro de un porcentaje especifico de su valor final. Una cifra de uso frecuente es el 5%.

Formica. Laminado plástico y brillante con que se forran algunas maderas, especialmente el conglomerado de madera.