



Instituto Politécnico Nacional
Centro de Investigación en Computación



Control de motores de C. D. con aprendizaje por
imitación basado en redes neuronales.

T E S I S

Que para obtener el grado de

Maestra en Ciencias de la Computación

Presenta: **Ing. María Luisa Castillo Vázquez**

Director de tesis: **Dr. Ricardo Barrón Fernández**

México, D. F., febrero de 2010.



INSTITUTO POLITECNICO NACIONAL
SECRETARIA DE INVESTIGACIÓN Y POSGRADO

SIP-14

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 12:00 horas del día 17 del mes de enero de 2010 se reunieron los miembros de la Comisión Revisora de Tesis designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis de grado titulada:

**"CONTROL DE MOTORES DE C.D. CON APRENDIZAJE POR IMITACIÓN
BASADO EN REDES NEURONALES"**

CASTILLO

Apellido paterno

VÁZQUEZ

materno

MARÍA LUISA

nombre(s)

Con registro:

B	0	1	1	3	7	8
---	---	---	---	---	---	---

aspirante al grado de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

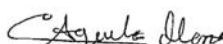
Después de intercambiar opiniones los miembros de la Comisión manifestaron **SU APROBACIÓN DE LA TESIS**, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Presidente


Dr. Sergio Suárez Guerra

Secretario


Dr. Carlos Fernando Aguilar Ibáñez

**Primer vocal
(Director de Tesis)**


Dr. Ricardo Barrón Fernández

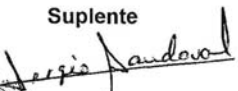
Segundo vocal


Dr. Marco Antonio Moreno Armendáriz

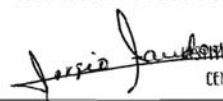
Tercer vocal


Dr. Marco Antonio Ramírez Salinas

Suplente


M. en C. Sergio Sandoval Reyes

El Presidente del Colegio


M. en C. Sergio Sandoval Reyes
CENTRO DE INVESTIGACIÓN
EN COMPUTACIÓN
DIRECCIÓN



INSTITUTO POLITÉCNICO NACIONAL
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México D. F. el día 30 del mes de octubre del año 2009, la que suscribe **María Luisa Castillo Vázquez** alumna del **Programa de Maestría en Ciencias de la Computación** con número de registro **B011378**, adscrito a **Centro de investigación en computación**, manifiesta que es autora intelectual del presente trabajo de Tesis bajo la dirección de **Dr. Ricardo Barrón Fernández** y cede los derechos del trabajo intitulado **Control de motores de C. D. con aprendizaje por imitación basado en redes neuronales**, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección: Tercer callejón de San José No. 3, San José, Iztapalapa, C.P. 09000, e-mail luitarin@yahoo.com.mx y torinja@yahoo.com.mx . Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.



María Luisa Castillo Vázquez
Nombre y firma

A Héctor, Helen y Adri
Mis amados compañeros de travesía

A Carmen y Gabino
Con profundo amor y respeto

Agradecimientos

A Héctor

Porque con su compañía, apoyo y amor incondicional, me ha impulsado para concluir este proyecto.

A mis hijos: Luisa Helena y Héctor Adrián

Por reconfortarme con su ternura y amor en todo momento.

A mis padres: Carmen y Gabino

Por mostrarme con paciencia, amor y comprensión el camino hacia la superación por medio del conocimiento.

A mis hermanos: Jaime, Álvaro, Luz, Jesús, y Lilia.

A mis sobrinos: Antonio, Roberto, Nallely, Luis, Ulises, Alejandro y Leonardo.

Y demás familia: Martha, Fabricio, Jorge, Josefina, Héctor, Patricia, Edith, Pavel y Sofía.

Por apoyarme incondicionalmente en cada proyecto emprendido y por alentarme a ser mejor persona.

A mis compañeros y amigos:

Por llenar de gratos recuerdos mi estancia en este Centro de Investigación.

A mis profesores:

Por aportarme nuevos y valiosos conocimientos. Agradezco especialmente a mi director de tesis: *Dr. Ricardo Barrón Fernández* por impulsarme pacientemente a concluir.

Al Instituto Politécnico Nacional, al Centro de Investigación en Computación y al CONACyT:

Por brindarme los espacios, facilidades y apoyos para poder concluir satisfactoriamente mis estudios de maestría.

Resumen

La imitación es un atributo presente en individuos que viven en sociedad y es un medio muy poderoso para la transferencia de conocimiento y desarrollo de nuevas habilidades. Por medio de la imitación un individuo es capaz de aprender un patrón motor nuevo mediante la observación de la conducta de otro, sin tener conciencia de las relaciones orgánicas internas presentes en el individuo que modela la conducta.

En el desarrollo de sistemas computacionales, en particular en el área de control, actualmente se requiere de un gran número de aplicaciones integradas que mejoren las capacidades de los sistemas sin afectar otros atributos, esto ha llevado a experimentar nuevas alternativas en el campo del control, aplicando técnicas de control inteligente junto con algunos conocimientos sobre habilidades cognitivas, presentes en los humanos y algunos animales con la finalidad de tratar de mejorar el desempeño de los sistemas. En este contexto, la imitación plantea un método en el que un sistema podría aprender un conjunto característico de patrones de control por medio del ejemplo.

Sobre la base de la idea anterior, la presente tesis plantea un esquema básico de aprendizaje por imitación, que se aplica al control de movimiento de motores de corriente directa. El esquema está compuesto por tres elementos: percepción, procesamiento de la información percibida (aprendizaje) y ejecución del acto percibido. El procesamiento de la información se lleva a cabo por una red neuronal artificial que aprende el mapeo entre el movimiento percibido y el que debe ejecutar el motor.

Se plantea para motores de corriente directa sobre todo porque son ampliamente utilizados en aplicaciones de control de posición y velocidad, y a medida que una aplicación crece y requiere de controlar más de un motor, el diseño se complica.

En las pruebas hechas al sistema se utilizan dos motores de corriente directa de diferentes dimensiones, uno de ellos modela los patrones de movimiento que deberá aprender el otro motor. Los resultados obtenidos son aceptables, tanto para perfiles de movimiento fijo, como para perfiles de movimiento variable, ya que aproximadamente en el 80% de los casos, la red responde adecuadamente (en estos casos la diferencia entre los objetivos de salida y la respuesta de la red es mínima, y no es crítica en el momento que el sistema reproduce el movimiento). Se pretende que el sistema sea un punto de partida para la construcción de sistemas con mayores prestaciones, de fácil implementación y programación relativamente simple.

Abstract

Imitation is an attribute present in individuals who live in society, and is an extraordinary form to transfer knowledge and development of new abilities. Through imitation an individual is capable of learning a new behaviorism pattern while observing another's behavior, without being aware of the internal organic relations present in the individual that models the behavior.

In the development of computational systems, particularly in the control area, is required nowadays a great amount of control applications that improve the capabilities of the system without affect other attributes, this has led to experiment new alternatives within the control field, applying intelligent control techniques along with some knowledge about cognitive abilities, such as imitation, present in humans beings and in some animals in order to try to improve the performance of the systems. Within this context, imitation raises a method in which a system might learn a reduced set of control patterns by means of the example.

Considering what is mentioned above, the thesis presents a basic sketch of learning by imitation which applies to the control of moves for direct current motors. The sketch consists of three elements: Perception (observation), processing of the acquired information (learning), and realization of the observed action.

The processing of the information takes place by an artificial neural network that learns the mapping between the movement shown and the movement carried by the motor.

It is focused for direct current motors mainly because they are widely used in control applications for speed and position, and according to the growth of an application and requires to control more than a motor thus the design gets complicated.

In the performed tests to the system, we use two direct current motors of different sizes, one of them models the movement patterns which must be learn by the second motor. The results are acceptable to movement profiles fixed and profiles of variable movements, as approximately 80% of cases, the network responds appropriately (in these cases the difference between output targets and the response of network is minimal and is not critical at the time that the system reproduces movement).

It is projected that this system becomes into the starting point for the building of systems with more benefits, easy implementation and a relatively simple programming.

Índice general

Resumen	
Índice general	
Índice de figuras	
Índice de tablas	
Glosario de términos y abreviaturas	

Capítulo 1

Introducción.....	7
1.1. Motivación	8
1.2. Justificación.....	8
1.3. Planteamiento del problema.....	9
1.4. Hipótesis.....	10
1.5. Objetivos	10
1.6. Límites y alcances.....	10
1.7. Contribuciones	11
1.8. Metodología	11
1.9. Organización de la tesis	12

Capítulo 2

2.1. Aprendizaje e imitación	13
2.1.1. <i>Aprendizaje</i>	13
2.1.2. <i>La imitación</i>	15
2.2. Teoría de control.....	21
2.2.1. <i>Sistema de control en la teoría de control clásico</i>	21
2.2.2. <i>Sistema de control en la teoría de control moderno</i>	22
2.2.3. <i>Control digital</i>	23
2.2.4. <i>Control inteligente</i>	24
2.2.5. <i>Control con redes neuronales</i>	24
2.3. Fundamentos de redes neuronales	25
2.3.1. <i>Arquitectura de redes</i>	26
2.3.2. <i>Redes feedforward</i>	27
2.3.3. <i>Aprendizaje supervisado</i>	27
2.3.4. <i>El algoritmo LMS</i>	28
2.3.5. <i>Backpropagation</i>	29
2.3.6. <i>El tamaño de la red</i>	32
2.4. El sistema de control propuesto.....	33
2.5. Otros trabajos	34
<i>Ventajas del sistema propuesto:</i>	34
2.6. Resumen	35

Capítulo 3

3.1. Generalidades	36
--------------------------	----

3.1.1. Codificador óptico.....	36
3.1.2. PWM Signo-Magnitud.....	37
3.2. Esquema básico para el aprendizaje por imitación	39
3.3. El sistema de control con aprendizaje por imitación.....	40
3.4. La interfaz de comunicación	53
3.5. Resumen.....	55
 Capítulo 4	
4.1. El control de motores con redes neuronales.....	56
4.2. Complejidad computacional.....	58
4.3. El control con aprendizaje por imitación	58
4.3.1. Ajuste de parámetros y entrenamiento de la red neuronal.....	60
4.3.2. Análisis de complejidad.....	61
4.4. Resultados	63
4.5. Resumen.....	68
 Conclusiones	
Trabajos futuros	70
Artículos publicados y trabajos relacionados con la tesis	71
Publicaciones en Congresos.....	71
Proyecto de investigación.....	71
Bibliografía.....	72
Manuales de referencia.....	75
Recursos en línea.....	76
 Apéndice I.....	
Programas del sistema	77
 Apéndice II.....	
Tabla completa de valores obtenidos para el desarrollo del proyecto.....	98
 Apéndice III	
Hojas de especificación.....	102
 Apéndice IV	
Mapa anatómico del cerebro humano.....	104
 Apéndice V	
Fundamentos del hardware del sistema	105
 Apéndice VI	
Teorema de Funahashi.....	125

Índice de figuras

2.1.	Mapa de la composición celular de la corteza cerebral del (a) primate y (b) humano.	19
2.2.	Esquema de un sistema de una entrada y una salida	22
2.3.	Esquema de un sistema multivariable.	23
2.4.	Diagrama a bloques de un sistema de control digital.....	23
2.5.	Neurona artificial con pesos y sesgo.	26
3.1.	Señales proporcionadas por el codificador de cuadratura	37
3.2.	Señal PWM signo-magnitud	38
3.3.	Funcionamiento básico de un puente H	38
3.4.	Ciclo PWM al 10%, 50% y 90%.	39
3.5.	Esquema para el aprendizaje por imitación.....	39
3.6.	Diagrama a bloques del sistema de control con aprendizaje por imitación.....	40
3.7.	Mecanismo de muestreo del sistema sensorial	41
3.8.	Diagrama de flujo del mecanismo de muestreo del sistema sensorial del motor.....	43
3.9.	Esquema gráfico del sentido de giro del eje de un motor	44
3.10.	Diagrama de flujo del mecanismo de decodificación de signo.....	45
3.11.	Graficas: (a) rps vs. periodo para el MM y (b) rps vs. ancho de pulso PWM para el MA	48
3.12.	Configuración de la red neuronal que realiza el mapeo en el aprendizaje por imitación y el control del motor	49
3.13.	Modo de operación del temporizador utilizado en el acto motor.	51
3.14.	Generación de la señal PWM de magnitud	52
3.15.	Generación de la señal PWM de Signo.....	53
3.16.	Interfaz de comunicación del MA	54
4.1.	Fase de entrenamiento de la red por modelo inverso	57
4.2.	Control por modelo inverso	58
4.3.	Diagrama del sistema de control con aprendizaje por imitación	59
4.4.	Perfil de movimiento para probar el sistema el sistema de control con aprendizaje por imitación.....	61
4.5.	(a) Desempeño del error y (b) evolución de los pesos de las capas de la red neuronal durante el entrenamiento	64
4.6.	Comparativo entre los objetivos de entrenamiento, las salidas obtenidas durante las pruebas, y las salidas obtenidas con un control PID, para perfiles de velocidad fija para la variable AP_{PWM}	66
4.7.	Señales de control proporcionadas por el Acto Motor a partir de los datos proporcionados para el Mapeo	67

Índice de tablas

3.1.	Combinación de los bits para cada uno de los cuatro estados de las señales A y B	44
3.2.	Datos para el control obtenidos del MM y del MA.....	46
4.1.	Valores de los datos correspondientes a los vectores de entrada y salida para el entrenamiento de la red neuronal	60
4.2.	Tabla comparativa entre los valores de salida (objetivos) que se utilizaron para el entrenamiento de la red neuronal	65

Glosario de términos y abreviaturas

AIM.- Active Intermodal Mapping – Mapeo Intermodal Activo. Es una de las teorías cognitivas actuales, que tratan de explicar como ocurre el proceso de la imitación.

Apraxia.- Es una enfermedad neurológica caracterizada por la incapacidad para poner una idea en práctica, como consecuencia de alguna interferencia en la transmisión de los impulsos apropiados entre el cerebro y los centros motores.

CD.- Corriente Directa. Se refiere a la corriente eléctrica que mantiene un flujo constante que no varía en magnitud y dirección.

Corteza cerebral.- La corteza cerebral es el manto de tejido nervioso que cubre la superficie de los hemisferios cerebrales.

DB-25.- Es un conector D-sub (D-subminiature) que se utiliza para conectar periféricos a las computadoras. Está formado por dos filas de pines paralelos rodeados por un escudo metálico en forma de D (de ahí su nombre), lo cual garantiza la orientación correcta en la conexión. El puerto paralelo utiliza muy comúnmente un conector de este tipo.

Ideomotor.- Se refiere a un movimiento corporal inconsciente o involuntario hecho en respuesta a un pensamiento o idea antes que a un estímulo sensorial.

Intermodal.- Se refiere al sustrato neurofisiológico común en el cual se relaciona entre sí diversas características sensoriales de un estímulo procedente de los sentidos.

MA.- Siglas utilizada para referirse al término “Motor Aprendiz”.

Microcontrolador.- Es un circuito integrado que está constituido por una unidad central de proceso, memoria, unidades de entrada / salida y una serie de dispositivos auxiliares para su funcionamiento (reloj, contadores, bus, etc.) Se podría considerar al microcontrolador como una computadora completa, pero de prestaciones limitadas.

MM.- Siglas utilizada para referirse al término “Motor Modelo”.

Neuroanatomía.- Anatomía del sistema nervioso. La neuroanatomía es la parte de la anatomía que se ocupa del estudio de las diferentes partes del sistema nervioso y órganos de los sentidos sobre todo en los aspectos descriptivos y topográficos.

Neurociencia.- Ciencia que se ocupa del sistema nervioso o de cada uno de sus diversos aspectos y funciones especializadas. La neurociencia es el estudio de la estructura, función, desarrollo, química, farmacología y patología del sistema nervioso.

Percepción.- Sensación interior que resulta de una impresión material hecha en nuestros sentidos.

PI.- Proportional Integral – Integral Proporcional. Es una metodología de control, que realiza las acciones correctivas para los actuadores por medio del cálculo basado en el error (proporcional) y la suma de todos los errores (integral).

PID.- Proportional Integral Derivative – Proporcional Integral Derivativo. Es una metodología de control muy común en procesos de control. Es un lazo de retroalimentación continua que mantiene a un proceso fluyendo normalmente realizando una acción correctiva cuando existe una desviación del valor objetivo. El control PID recibe señales de sensores y calcula las acciones correctivas para los actuadores por medio de un cálculo basado en el error (proporcional), la suma de todos los errores previos (integral) y la tasa de cambio del error (derivativa).

Proprioceptiva. Función sensorial que está relacionada con la percepción relativa las partes del cuerpo. La propiocepción informa de los diferentes estados de los músculos y articulaciones.

PWM. Pulse Width Modulation - Modulación por Ancho de Pulso. Es una técnica de modulación en la que se varía el ciclo de trabajo de una señal periódica.

Sistema de control. Es un sistema cuyo propósito es regular o ajustar el flujo de energía con un propósito específico. Existen sistemas de control de lazo cerrado y sistemas de control de lazo abierto. Un sistema de control de lazo cerrado utiliza la salida del sistema como retroalimentación, con el propósito de modificar las acciones encaminadas a llevar a cabo un objetivo. Un sistema de lazo abierto no tiene retroalimentación, utiliza otro esquema para ajustar el flujo de energía.

SPP. Standard Parallel Port – En computación, un puerto es un conjunto de líneas de señales que el microprocesador, o el CPU, usa para intercambiar datos con otros componentes.

Supramodal. Se refiere al sustrato neurofisiológico común donde se genera una representación mental del mundo por parte del cerebro, debido a la información percibida por las distintas vías sensoriales (auditiva, visual, olfativa, gustativa y táctil) y que es independiente de la vía sensorial por la que llega la información.

Capítulo 1

Introducción

La imitación es una habilidad innata que poseen los humanos y algunos animales para reproducir las acciones de otros. Es un medio muy poderoso para la transmisión de conocimiento que acelera el aprendizaje y el desarrollo de nuevas habilidades.

La imitación de habilidades motoras involucra un conjunto muy complejo de mecanismos que permiten mapear el movimiento observado de un modelo, al movimiento que debe ser ejecutado por el observador. En su forma más básica, este tipo de imitación comprende tres aspectos: la percepción de una acción, el procesamiento de la información adquirida por medio de la percepción y la reproducción de la acción.

La importancia que tiene la imitación en el aprendizaje ha sido motivo de estudio a través del tiempo. En la psicología, por ejemplo, se comienza a estudiar formalmente desde el siglo XIX, y en la actualidad, en la psicología cognitiva, se siguen desarrollando algunas teorías que tratan de precisar cómo se da el proceso de la imitación. También, estudios recientes en neurociencia del comportamiento han identificado un conjunto de neuronas en la corteza frontal de primates y humanos que está relacionada con el comportamiento imitativo; este conjunto de neuronas, llamado neuronas espejo, es un elemento clave en la imitación que ayuda a igualar el movimiento observado con el movimiento que ejecuta el observador.

Hoy en día, el conocimiento que se tiene sobre la imitación y las ventajas que proporciona para la adquisición de conocimiento y habilidades, han motivado su aplicación en diversas áreas de la ciencia, incluyendo a las ciencias computacionales.

Desde el punto de vista computacional el aprendizaje por imitación es un problema altamente complejo que requiere del mapeo de una acción percibida (ejecutada por un modelo), en un marco de referencia externo, a un marco de referencia interno en el observador. Por lo que la imitación plantea la posibilidad de transferir habilidades entre sistemas por medio de ejemplos de acciones.

En el caso concreto de un sistema de control la imitación permitiría transferir las habilidades de control entre sistemas por medio de ejemplos.

En este sentido, la tesis propone un método básico de control para motores de corriente directa, en el que el sistema aprende por imitación la regla de control. El aprendizaje se lleva a cabo por medio de una red neuronal artificial, que junto con un módulo de percepción, y un módulo que reproduce la acción, forman parte del sistema de aprendizaje por imitación, el cual a su vez forma parte del sistema de control para el motor. Se espera que el sistema planteado sirva como un esquema básico que pueda ser empleado para construir sistemas de mayor potencialidad, disminuyendo la dificultad que implica la programación en estos sistemas.

1.1. Motivación

Desde que el humano fue capaz de crear sistemas que le facilitaran el desarrollo de sus tareas diarias, comenzó también la inquietud de dotar a dichos sistemas con las capacidades necesarias para brindarles mayor eficiencia. En los sistemas de control por ejemplo, desde que inicia la era de las computadoras digitales, se han desarrollado algoritmos de control, que en su concepción son cada vez más abstractos y que pretenden brindar más eficiencia a las tareas de control.

En el caso concreto de un sistema que tiene como elemento final a un motor de corriente directa, cuando se toman en cuenta las características no lineales del motor, y el número de motores a controlar aumenta, el costo y esfuerzo computacional también aumenta. Una opción con miras a disminuir esta complicación es utilizar técnicas de Inteligencia Artificial, como las redes neuronales artificiales; sin embargo, algunas veces esto no es suficiente y la inquietud de brindar más eficiencia a los sistemas más complejos motiva a explorar nuevas alternativas en diferentes áreas de la ciencia. Así, en años recientes se ha comenzado a explorar en las áreas que estudian cómo los humanos y algunas otras especies animales acceden al conocimiento y de esta manera trasladar estas habilidades a los sistemas artificiales. Un ejemplo de esto son las investigaciones que se han desarrollado para implementar el aprendizaje por imitación en diversas áreas de las ciencias computacionales, con el fin de trasladar los beneficios que proporciona la imitación en los humanos y animales, a los sistemas artificiales, y de esta manera tratar de resolver los problemas más complejos de una manera más sencilla.

1.2. Justificación

Aunque actualmente existen métodos modernos bien definidos para el control de motores de corriente directa, el aprendizaje por imitación ofrece un método con gran potencial, sobre todo cuando la información sobre el comportamiento del sistema está sólo en la forma de pares de

datos de entrada-salida, ya que permite transferir una regla de control al sistema por medio de ejemplos de comportamiento, aún sin tener un modelo matemático exacto.

Tomando como base la idea que una regla de control es una función tal que, para una entrada específica se produce una salida específica, cuando se requiere establecer una regla de control cuando la información sobre el comportamiento del sistema está sólo en la forma de pares de datos de entrada-salida, será necesario encontrar la función que se adapte a esos datos con un buen grado de exactitud. Para esto, será necesario establecer un criterio de desempeño y la función que cumple con ese criterio, en este caso una red neuronal es una herramienta que cumple con ambos requerimientos.

En el sistema propuesto se justifica el uso de una red neuronal, sobre todo porque la información sobre el comportamiento del sistema está constituida por pares de datos de entrada-salida y dada la naturaleza no lineal del motor, es muy complicado obtener un modelo no lineal. Además gracias a la capacidad de generalización que tienen las redes neuronales se pueden usar como dispositivos para aproximar funciones, lo cual permite que se puedan aplicar en control. Por otra parte, cuando una aplicación implica relaciones no lineales, como es el caso del motor, las redes neuronales con múltiples capas son las más adecuadas.

Teóricamente, el sistema aprende mediante la observación y es capaz de reconocer y reproducir el movimiento percibido. En la práctica se obtiene un sistema que aprende un conjunto reducidos de patrones de control y a partir de esto puede reproducir perfiles de movimientos variables.

La aportación principal es que a diferencia de un sistema de control convencional la regla de control se establece por medio de ejemplos de movimiento, ya que el sistema tiene la capacidad de percibir una acción, procesar la información percibida y reproducir la acción, tal y como sucede cuando un individuo aprende por imitación.

1.3. Planteamiento del problema

El control de motores de corriente directa requiere de obtener un modelo matemático exacto del sistema, pero cuando la información sobre el comportamiento del sistema está sólo en la forma de datos de entrada y salida, y dadas las características no lineales del motor, es muy difícil obtener el modelo exacto, se puede entonces aplicar un esquema básico de aprendizaje por imitación y la teoría de redes neuronales artificiales para implementar un sistema de control que sea capaz de aprender un conjunto reducido de patrones de movimiento, y a partir de esto pueda reproducir una gran variedad de perfiles de movimientos variables.

Se parte de analizar el funcionamiento de un sistema de control implementado con un microcontrolador de propósito específico, para determinar los parámetros característicos del control.

1.4. Hipótesis

Considerando las características no lineales de un motor de corriente directa y la dificultad que representa obtener un modelo matemático no lineal, se puede aplicar un esquema básico de aprendizaje por imitación y la teoría sobre redes neuronales para implementar un sistema de control no lineal, el cual será capaz de percibir, procesar la información percibida y reproducir cualquier perfil de movimiento.

1.5. Objetivos

Objetivo general

Aplicar un esquema básico de aprendizaje por imitación y la teoría de redes neuronales para implementar un sistema básico de control para motores de corriente directa.

Objetivos específicos

Para alcanzar el objetivo principal, fue necesario:

- Plantear un esquema básico de aprendizaje por imitación.
- Proponer un sistema básico de control para motores de corriente directa, basado en el esquema de aprendizaje por imitación.
- Implementar el esquema de control propuesto con herramientas de hardware y software de propósito general.
- Diseñar e implementar una interfaz de entrada / salida para la comunicación de datos para el sistema propuesto.
- Diseñar e implementar los programas para la adquisición de datos y para generar las señales de control necesarias para el control del Motor.
- Realizar las pruebas y ajustes necesarios para el funcionamiento del sistema.

1.6. Limites y alcances

Limites

El sistema de control propuesto, es un sistema de control no lineal, que está limitado al control de movimiento de motores de corriente directa.

Alcances

Teóricamente se plantea un método de control para motores de corriente directa, que utiliza un esquema básico de aprendizaje por imitación, que utiliza una red neuronal para el proceso de aprendizaje del sistema. En la práctica, el sistema desarrollado será capaz de aprender, reconocer y reproducir el movimiento percibido.

1.7. Contribuciones

A diferencia de un sistema de control convencional, la aportación principal al sistema es la implementación de un método que permite transferir la ley de control por medio de un conjunto reducido de ejemplos (parámetros del comportamiento del sistema), a partir de lo cual el sistema será capaz de proporcionar las señales de control necesarias para que el motor desarrolle un número muy variado de perfiles de movimiento.

1.8. Metodología

- Se planteó un esquema básico de aprendizaje por imitación que está documentado en artículos y libros de divulgación científica.
- Como primera aproximación al diseño del control de **velocidad**, se analizó el comportamiento del motor, y se identificaron las señales necesarias para su control.
- Basándose en lo anterior, se establecieron las entradas y salidas del sistema de control.
- Se caracterizó el comportamiento del sistema por medio de un conjunto de datos de entrada-salida.
- Sobre la base de lo anterior y dada la dificultad que representa obtener un modelo no lineal del motor, y dada la naturaleza de la imitación, se planteó que el aprendizaje en el sistema se haría por medio de una red neuronal artificial. Por lo que se estudió la teoría de redes neuronales para determinar el tipo, arquitectura y método de entrenamiento más adecuados para el sistema.
- Se hizo una simulación para verificar la viabilidad del uso de la red neuronal.
- Para la implementación se eligieron herramientas de hardware y software de propósito general.
- Se diseñaron e implementaron cada una de las etapas del sistema (interfaces de entrada y salida de datos, procesamiento y ejecución del movimiento) y todo el sistema de control.
- Se diseñaron e implementaron los programas para la adquisición y procesamiento de datos y el control.

- Se probó el funcionamiento del sistema etapa por etapa, y se hicieron los ajustes necesarios para que el funcionamiento fuera el adecuado.
- Se probó experimentalmente el funcionamiento del sistema de control del motor.
- Se realizaron las pruebas para el funcionamiento del motor para perfiles de movimientos fijos y variables.

1.9. Organización de la tesis

La tesis está dividida principalmente en cuatro capítulos. Para mejorar su comprensión, se proporciona una descripción del contenido de cada capítulo.

El Capítulo 1, el presente, describe el objetivo general, los objetivos específicos, así como la justificación, límites, alcances, metodología y organización de este trabajo de tesis.

El Capítulo 2, contiene una descripción de los trabajos más recientes realizados en el campo del control de motores de CD y en el aprendizaje por imitación. Proporciona un marco teórico sobre las teorías del aprendizaje e imitación, que dan un antecedente a este trabajo de tesis.

El Capítulo 3, describe cómo está constituido el Sistema de Control de Motores de CD con Aprendizaje por Imitación.

El Capítulo 4, proporciona los resultados de las pruebas hechas al sistema, así como las conclusiones y trabajos futuros.

La bibliografía contiene los artículos, libros y recursos en línea que se consultaron para el desarrollo de la presente tesis.

En el Apéndice I, se puede encontrar el código completo del software desarrollado para el sistema. El Apéndice II contiene la lista completa de los datos que se obtuvieron experimentalmente para el control de los motores. El Apéndice III contiene hojas de especificación e información adicional sobre los dispositivos de hardware utilizados para el desarrollo del sistema. El apéndice IV contiene el diagrama anatómico del cerebro humano. El Apéndice V muestra los fundamentos del hardware. El apéndice VI contiene algunos teoremas mencionados en la tesis.

Capítulo 2

Estado del arte

Este capítulo proporciona el marco teórico sobre las teorías del aprendizaje e imitación, y los mecanismos neuronales que se cree están fuertemente relacionados con la habilidad de imitar. También se da la descripción de un sistema de control visto desde la teoría de control clásica, la teoría de control moderna y desde el punto de vista de la inteligencia artificial, en particular el uso de redes neuronales artificiales en los sistemas de control.

2.1. Aprendizaje e imitación

2.1.1. *Aprendizaje*

En psicología el aprendizaje está definido como la adquisición de una conducta duradera por medio de la práctica [77]. La importancia que tiene el aprendizaje en muchos aspectos de la vida humana ha sido motivo de debates desde los orígenes de la filosofía. Platón, por ejemplo, sugiere que el aprendizaje es una manera de relacionar estímulos actuales con experiencias del pasado, por lo que “aprender es recordar”. Aristóteles por su parte explica que los conceptos se acumulan en la mente por medio de asociación de ideas, lo cual se da por semejanza, contraste y contigüidad. Varios siglos después, en 1670 el inglés John Locke retoma lo dicho por Aristóteles, y lo aplica en su obra “Ensayo Sobre el Conocimiento Humano” en donde trata de explicar el origen, la certeza y los alcances del conocimiento. Los estudios del aprendizaje siguieron dentro del marco de la filosofía hasta el surgimiento de la Psicología Científica¹ en el siglo XVIII, cuando el alemán Wilhelm Wundt crea el primer laboratorio científico de psicología. Desde entonces y debido a la gran complejidad que representa el proceso de aprendizaje, han surgido varias corrientes psicológicas que han estudiado cómo es el proceso de aprendizaje en los humanos y algunos animales, dentro de estas corrientes se considera principalmente a cinco: *La*

¹ Antes la psicología era considerada una rama de la filosofía.

escuela tradicional de la psicología científico filosófica, las doctrinas del condicionamiento clásico (I. P. Pavlov y J. B. Watson), la Gestalt (Wertheimer, Koffka y Köhler), el Conexionismo, y la corriente Psicoanalítica (S. Freud). Estas corrientes estudiaron los procesos mentales del aprendizaje de manera sistemática y dieron origen a lo que se conoce como “Teorías del Aprendizaje”.

Teorías del aprendizaje

Las Teorías del Aprendizaje tratan de explicar cómo los sujetos acceden al conocimiento, centrándose en el estudio de la adquisición de destrezas y habilidades, y en el razonamiento y adquisición de conceptos. Los paradigmas más importantes dentro de los cuales tienden a caer las teorías del aprendizaje son: *el conductismo, el cognitivismo, el constructivismo y la teoría del aprendizaje social*. Estas teorías se mencionan a continuación.

El conductismo.

Fue propuesto por J. B. Watson, B. F. Skinner, I. Pavlov y otros a principios del siglo XX. El conductismo estudia el comportamiento de la relación estímulo respuesta sin tener en cuenta los procesos mentales que éste implica. Asume que el aprendiz es un sujeto pasivo respondiendo a estímulos del medio ambiente. Considera que en un principio el sujeto está como una hoja en blanco y el comportamiento se forma a través de refuerzos positivos o negativos que aumentan la posibilidad de aprendizaje. Así el conocimiento es un producto que puede medirse, cuantificarse y evaluarse objetivamente. Las teorías conductistas más prominentes son: *El conductismo (Watson, Pavlov, Skinner, Thorndike, Bandura y Tolman), el condicionamiento clásico (Pavlov), la teoría del aprendizaje social (Bandura) y el modelo GOMS (Card, Moran y Newell).*

El cognitivismo

En 1960 el cognitivismo reemplazó al conductismo como paradigma dominante. Esta teoría considera que el individuo es un ser racional que requiere de la participación activa en el proceso de aprendizaje y sus acciones son el resultado de pensar, por lo que el individuo es visto como un procesador de información. En este caso la teoría se enfoca en las actividades mentales implícitas en el aprendizaje. Los procesos mentales como el pensamiento, la memoria y la resolución de problemas, son analizados para comprender como el individuo aprende. Las teorías más prominentes son: *El cognitivismo (Merrill, Reigeluth, Gagne, Briggs, Wager, Bruner, Schank, Scandura), la teoría del aprendizaje social (Bandura) y la teoría de la atribución (Weiner).*

El constructivismo

Fue propuesta por J. Dewey, J. Piaget, G. Bruner, L. Vygotsky y otros. El constructivismo puntualiza que el aprendizaje es un proceso constructivo activo, en donde el aprendiz es un constructor de información, que crea sus representaciones subjetivas de la realidad objetiva, por lo que el conocimiento se construye sobre la base de experiencias personales y conocimientos

previos. Las teorías más prominentes son: *el constructivismo* (Vigotsky, Piaget, Dewey, Vico, Rorty, Bruner) *la teoría del desarrollo social* (Vygotsky), *comunidades de práctica* (Lave y Wenger), *descubrimiento del aprendizaje* (Bruner) y *la teoría de la etapa del desarrollo cognitivo* (Piaget).

La teoría del aprendizaje social

Fue propuesta por A. Bandura en 1977. La teoría del aprendizaje social, puntualiza que los individuos aprenden observando el comportamiento, las actitudes y el resultado del comportamiento de otros. A partir de la observación el individuo forma una idea de cómo se deben llevar a cabo nuevos comportamientos, y posteriormente la información codificada sirve como una guía de acción. La teoría del aprendizaje social, explica el comportamiento humano en términos de una interacción continua y recíproca entre la influencia cognitiva, conductual y del medio ambiente. Un antecedente fue publicado en 1941 con el trabajo de Neal Miller y John Dollard llamado *El Aprendizaje Social e Imitación*. Este trabajo explica cómo a través del comportamiento observado de modelos, los humanos y algunas especies de animales llegarán a aprender dicho comportamiento con la ayuda del reforzamiento del ambiente.

2.1.2. La imitación

De acuerdo con Billard en [12] la imitación es la habilidad para reconocer y reproducir las acciones de otros, y es un poderoso medio de aprendizaje y desarrollo de nuevas habilidades. Por medio de la imitación se pueden aprender patrones de conducta nuevos o cambiar los patrones ya existentes, observando el comportamiento de otros [8], y también actúa como un mecanismo de adaptación y sobrevivencia. Meltzoff y Moore en [46] hacen distinción entre la imitación manual, vocal y facial. Heyes en [31] afirma que la imitación consiste del aprendizaje de un patrón motor nuevo (por ejemplo, gestos faciales o movimiento del cuerpo), por medio de la observación de otro individuo, llamado modelo, realizando ese patrón motor.

Desde el punto de vista conductista, la imitación es el desempeño motor o verbal de actos específicos o sonidos que son previamente realizados por un modelo. En todas las culturas el aprendizaje por imitación a través de modelos, ha sido utilizado como una manera de transmitir patrones de comportamiento de importancia social y cultural, además de que alienta la interacción social [2].

En la literatura sobre imitación, se hace una distinción entre la “verdadera imitación²” y la mímica³ o copia. La verdadera imitación es la habilidad para aprender y reproducir habilidades que no son parte del repertorio previo del observador. Thorndike define a la verdadera imitación como un comportamiento que es observado, entendido y reproducido. Por otra parte, la mímica es la habilidad para reproducir un comportamiento que ya forma parte del repertorio del

² También es llamada “imitación de alto nivel”.

³ También es llamada “imitación de bajo nivel”.

observador [7, 11, 12]. También la imitación puede ser inmediata cuando la reproducción ocurre dentro de un período de tiempo corto; o diferenciada cuando la reproducción ocurre dentro de un período de tiempo largo. Cuando sólo una parte del comportamiento imitativo es reproducido la imitación se considera parcial o selectiva; también puede ser exacta. Considerando el proceso cognitivo detrás de la imitación, cuando el sistema motor está activo, se lleva a cabo una imitación activa, cuando el sistema motor está pasivo durante la percepción, la imitación es pasiva [12]. Durante mucho tiempo la imitación ha sido considerada como un tipo especial de aprendizaje, debido a su complejidad psicológica y su potencial para soportar la transmisión cultural [29], por lo que a través del tiempo ha surgido una serie de teorías de la imitación que tratan de explicar los mecanismos que implica.

Teorías de la imitación

El concepto de imitación en la teoría psicológica tiene una larga historia que probablemente data desde antes de William James (1890)⁴, Lloyd Morgan (1896), Tarde (1903) y McDougall (1908), quienes consideraban a la imitación como una propensión innata e instintiva. Humphrey (1921) y Allport (1924) intentan explicar el comportamiento imitativo por medio de los principios del condicionamiento Pavloviano. Holt (1931) ofreció un análisis del aprendizaje observacional que es de alguna manera similar al propuesto por Allport, pero revela una de las principales debilidades de las teorías del condicionamiento clásico de la imitación: La falla para representar satisfactoriamente la emergencia de respuestas nuevas durante la secuencia modelo-observador [8]. A principios del siglo XX, el problema de la imitación recibe una creciente atención, pero fue hasta la publicación de “El Aprendizaje Social e Imitación” de Miller y Dollard en 1941, que el concepto de imitación fue completamente integrado al marco de trabajo del conductismo. De acuerdo con esta publicación, para que un sujeto pueda aprender por medio de la imitación debe ser motivado y reforzado positivamente para que su comportamiento sea igual al del modelo. Para alcanzar este objetivo, el sujeto comienza respondiendo por “ensayo y error” hasta que alcanza la respuesta correcta. Para Skinner (1953), la imitación es condicionada y aparece cuando el sujeto observador reproduce el mismo comportamiento que el modelo, y recibe un estímulo reforzador. Mowrer (1960) se enfoca casi exclusivamente en una retroalimentación propioceptiva evaluada positivamente como el proceso crucial en el aprendizaje imitativo. Según Piaget (1962), el desarrollo limita la imitación. Por lo que es necesaria la consolidación de las estructuras mentales en cada etapa de desarrollo para tener las habilidades necesarias para imitar [61]. Después de los trabajos de Piaget la imitación no recibió más atención, principalmente por que se creía que ésta no constituía una expresión de alta inteligencia [55]. Esta actitud cambió cuando en la década de 1970 Meltzoff y Moore publicaron una serie de estudios donde sugerían que los niños recién nacidos pueden imitar expresiones faciales. En

⁴ Con la publicación de “The principles of psychology”.

trabajos posteriores ellos proponen un modelo del mecanismo que se encuentra detrás de la imitación facial en recién nacidos, este mecanismo llamado “Mapeo Intermodal Activo” (AIM⁵) [46], se convertiría en una de las teorías de la imitación que sigue en dinámico desarrollo.

Hoy en día las investigaciones sobre la imitación se han enfocado a resolver lo que se conoce como “*el problema de correspondencia*”, el cual cuestiona: ¿Cómo es que el imitador conoce qué patrones de activación motora harán a su acción parecida a la del modelo? La respuesta a este cuestionamiento ha dado lugar a lo que hoy se conoce como las teorías actuales de la imitación.

Teorías actuales de la imitación

Actualmente las teorías de la imitación ofrecen soluciones especialistas y soluciones generalistas al *problema de correspondencia*. Las teorías especialistas sugieren que la imitación se da a través de un mecanismo de propósito especial compuesto por mecanismos funcionales y neurológicos dedicados a controlar la imitación. En este mecanismo de propósito especial la percepción y el proceso motor comparten un lenguaje en común. La teoría especialista más prominente es el *Mapeo Intermodal Activo*. Por otra parte, las teorías generalistas asumen que la imitación se lleva a cabo debido a la activación de representaciones motoras a través de la observación de una acción. También proponen que la imitación se da por medio del aprendizaje general y mecanismos de control motor. Actualmente existen dos teorías generalistas complementarias: La teoría *Ideomotora* (IM⁶), y el modelo *Aprendizaje Secuencia Asociativa* (ASL⁷) [17].

El mapeo intermodal activo

El AIM fue formulado por Meltzoff y Moore en 1997, para explicar como es que los infantes copian los gestos faciales de los adultos, actualmente se aplica a la imitación en general. El AIM propone que cuando se observa un cuerpo en movimiento con la intención de imitarlo, la representación visual del movimiento es convertida en una representación supramodal, la cual contiene información sobre las ‘relaciones orgánicas’ que implica el movimiento [1, 17, 30]. De esta manera, la imitación es un proceso de correspondencia-a-objetivo⁸. De acuerdo con este punto de vista, la percepción y los actos humanos son codificados dentro de una estructura común, que capacita al humano para detectar equivalencias entre sus propios actos y los que observa [46]. Hasta ahora, el AIM no aborda el *problema de correspondencia*.

La teoría ideomotora

Asume que la información sensorial que producen las acciones es representada en la forma de imágenes, y tales representaciones son utilizadas para iniciar y controlar el movimiento del

⁵ De las siglas en inglés de “Active Intermodal Mapping”. Algunos autores lo escriben como “Active Intermodal Matching”.

⁶ Del inglés “Ideomotor”

⁷ De las siglas en inglés de “Associative Sequence Learning”.

⁸ En inglés “matching-to-target”.

cuerpo. Desde este punto de vista, observar la ejecución de una acción conduce al observador a la activación de una representación motora interna, y es precisamente esta representación motora la que es usada para imitar el comportamiento observado y por lo tanto, la facilidad con que los humanos imitan es debido a la organización general del control motor, y no a un mecanismo de propósito especial dedicado a la imitación.

El modelo aprendizaje secuencia asociativa

El modelo ASL afirma que la capacidad para imitar es producto de procesos generales de aprendizaje asociativo [17]. También sugiere que el desarrollo de los mecanismos de la imitación es altamente dependiente de la experiencia, y que estos consisten de un conjunto de enlaces bidireccionales excitatorios entre representaciones sensoriales y motoras de unidades de movimiento, en vez de un dispositivo innato que activa internamente transformaciones de entradas visuales a salidas motoras vía representaciones supramodales [30]. Fue delineada por Heyes y Ray en el año 2000 [32].

Imitación y neurociencia

Un factor importante dentro de la imitación es la conexión entre las acciones percibidas y las acciones generadas. Para la percepción de acciones es necesario un sistema sensorial y para la generación de acciones es necesario un sistema motor. Ambos sistemas deberán interactuar de manera que las acciones percibidas puedan ser mapeadas a las acciones apropiadas.

En estudios realizados sobre la neuroanatomía de primates en la década de 1980, se encontró que las neuronas dentro del Surco Temporal Superior (STS) del cerebro de los macacos están activas al observar la forma y movimiento de objetos (figura 2.1(a)). Posteriormente en 1996 se descubre que un circuito neuronal en el área F5 de la corteza premotora del cerebro de los primates, se encuentra activo no solamente cuando éste observa a otros primates o humanos manipular objetos, sino también cuando éste realiza la misma manipulación [11, 44, 56]. El hallazgo de este conjunto de neuronas, llamadas neuronas espejo, es posiblemente una explicación de cómo se realiza el mapeo entre percepción y acción en el aprendizaje por imitación. Se cree que las neuronas espejo son responsables de igualar el comando neuronal de una acción percibida, con el código para el reconocimiento de la misma acción, llevada a cabo por el observador, por lo que el sistema de neuronas espejo ha sido propuesto como el enlace entre la representación sensorial y la motora para aprender por medio de la observación⁹ y la imitación de las acciones de otros [11, 12]. A diferencia de las neuronas espejo, las neuronas en el STS no están activas durante la ejecución de actos motores.

El descubrimiento de las neuronas espejo en el cerebro de los primates, despertó el interés de encontrar un mecanismo similar en el cerebro humano (figura 2.1 (b)). Estudios recientes en

⁹ Aunque se maneja el término observar, en realidad se refiere a cualquier tipo de percepción sensorial.

neurología del comportamiento [15] y de neuroimágenes sugieren que existe un área similar en el cerebro humano, comprendida predominantemente en el área de Broca (regiones 44 y 45), en el lóbulo parietal (área 40), y el surco temporal superior (área 21), que registran gran actividad durante la observación y ejecución de acciones de la mano y están involucradas en la imitación¹⁰ [11, 12].

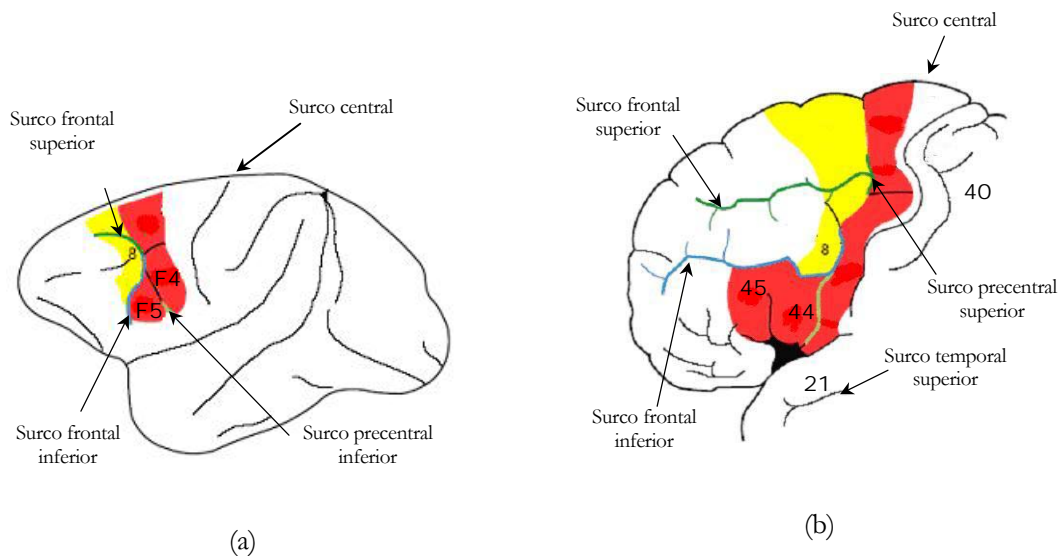


Figura 2.1. Mapa de la composición celular de la corteza cerebral del (a) primate y (b) humano. Las coloraciones del mismo tono indican áreas con homologías anatómicas y funcionales. En (a) se muestra la región F5 de la corteza premotora del cerebro de los primates donde se concentra el conjunto de neuronas espejo. En (b) se muestra una región similar en el cerebro humano, comprendida predominantemente en las regiones 44 y 45 del área de Broca.

En estudios sobre el comportamiento imitativo anormal en humanos, se ha mostrado que cuando existen lesiones en ciertas regiones del cerebro los pacientes son incapaces de llevar a cabo un comportamiento imitativo (apraxia ideomotora), o muestran un comportamiento imitativo obsesivo (imitación compulsiva), lo cual muestra una clara evidencia de que ciertas regiones del cerebro están involucradas en la imitación [12].

Actualmente un gran número de estudios neuropsicológicos han demostrado que existe un mecanismo de resonancia motora en la corteza premotora y la corteza parietal posterior, que se activa cuando los sujetos observan acciones de objetivo dirigido que son ejecutados por otra persona, o cuando sólo el objetivo de esa acción es visible. Varios estudios de imágenes han reportado que este mecanismo de resonancia motora también está involucrado en la imitación de una acción [35].

¹⁰ En el Apéndice IV se presenta un mapa anatómico del cerebro humano, en donde se ubican sus principales partes.

Aprendizaje por imitación

El aprendizaje por imitación es la habilidad para aprender por medio de la observación de las acciones que realizan otros, y es uno de los mecanismos más poderosos utilizado para transmitir el conocimiento y las habilidades, por lo que ha tenido una gran variedad de aplicaciones en computación, robótica e inteligencia artificial. En el campo de la computación se han desarrollado sistemas que utilizan metodologías computacionales inspiradas en organismos biológicos [11, 13, 57, 63]. En el campo de la robótica, el aprendizaje por imitación ha sido aplicado principalmente en el aprendizaje y entrenamiento de robots humanoides [7, 20, 27]. En el campo de la inteligencia artificial, se empieza a utilizar el concepto de aprendizaje por imitación en realidad virtual [19, 37], y en juegos computacionales [23].

Sistemas inspirados en los principios biológicos de la imitación

Son sistemas que toman como inspiración una parte o todo un organismo biológico para modelar sistemas cuya funcionalidad trata de asemejarse a la funcionalidad del organismo que sirve como inspiración. De esta manera, tomando como punto de partida cómo ocurre el aprendizaje por imitación en humanos, han surgido aplicaciones en donde se proponen modelos para la imitación de habilidades motoras inspiradas biológicamente. Un ejemplo de ellos es un modelo que está compuesto de módulos que son abstracciones de alto nivel de la médula espinal, la corteza motora primaria y premotora, el cerebelo y la corteza temporal, en donde cada módulo es modelado como un nivel conexionista. Los comportamientos motores primarios, como el movimiento rítmico de manos y piernas al caminar, son predefinidos en la médula espinal, y el aprendizaje de nuevas combinaciones de movimientos es hecho en la corteza premotora y el cerebelo [11, 13]. Otra aplicación describe un sistema motor computacional para producir movimiento en un brazo robot tal y como lo haría un humano. Dentro de este sistema motor son presentados diferentes movimientos en una estructura modular. Cuando el sistema observa un movimiento demostrado, el sistema motor utiliza esos módulos para producir comandos motores los cuales son usados para actualizar una representación del estado interno. Esto es usado para que el sistema pueda reconocer movimientos conocidos y mover el brazo robot de acuerdo con las características exactas del movimiento demostrado y usarlos para aprender nuevos módulos [58]. En [63] se presenta una combinación de los elementos del mecanismo perceptual y del mecanismo motor involucrados en el aprendizaje por imitación con la función que las neuronas espejo tienen en el cerebro, para generar las primitivas que son usadas como un conjunto base de movimientos, sirviendo como un vocabulario para clasificar e imitar los movimientos observados.

2.2. Teoría de control

La teoría de control es un campo interdisciplinario de la ingeniería y las matemáticas, que estudia el comportamiento de los sistemas. Fue desarrollada principalmente para soportar el creciente avance del control automático. Históricamente, el control automático ha estado relacionado con la idea controlar y hacer de manera automática las tareas más complicadas, reemplazando el trabajo humano por un control automático. Los registros más antiguos sobre sistemas de control datan del siglo III a. C. en Grecia, pero fue hasta la segunda mitad del siglo XIX, que las bases matemáticas (la transformada de Laplace, la transformada de Fourier y la teoría de la variable compleja de Cauchy), y analíticas de la teoría de control comenzaron a establecerse. Para la década de 1930 las aplicaciones eran numerosas: amplificadores en sistemas telefónicos, plantas eléctricas, estabilización de aviones, mecanismos eléctricos para la industria química, del acero, del petróleo, etc. Pero frecuentemente el análisis de los sistemas se complicaba debido a la gran cantidad de elementos independientes que lo constituían y a la dificultad de representarlos por medio de un conjunto de ecuaciones diferenciales. Esta situación cambió con la introducción del análisis en frecuencia, el cual permite conocer y analizar la relación que existe entre la amplitud y la fase de entrada con respecto a la amplitud y fase de salida, con relación a la frecuencia. Entre los trabajos más destacados sobre el tema están los de Nyquist (1932), Black (1934) y Bode (1940). A finales de la década de 1940, Evans presenta un método sencillo para encontrar las raíces de la ecuación característica de un sistema, llamado método de lugar de las raíces. Este método indica cómo deben modificarse los polos y los ceros en lazo abierto para que la respuesta cumpla con las especificaciones de comportamiento del sistema, lo cual simplifica significativamente el análisis de los sistemas. A finales de la década de 1950, se desarrollan métodos matemáticos temporales para aplicaciones de control, para resolver principalmente problemas aeroespaciales, en donde las computadoras digitales cobraron gran importancia como herramienta de desarrollo. La manera de representar a los sistemas y las herramientas utilizadas para analizarlos ha dado origen a diferentes etapas en la teoría de control. De esta manera, los métodos de respuesta en frecuencia y del lugar de las raíces son el eje de la teoría de control clásica (1900-1960). El desarrollo de métodos matemáticos temporales marca los inicios de la teoría de control moderna (a partir de 1960). El uso de computadoras en aplicaciones de control ha dado origen al control digital. Y la interacción de disciplinas como la inteligencia artificial y la teoría de sistemas ha dado origen a lo que se conoce como control inteligente. A continuación se presenta una breve descripción de las metodologías utilizadas en control, para el análisis y desarrollo de sistemas.

2.2.1. *Sistema de control en la teoría de control clásico*

Las bases matemáticas utilizadas en la teoría de control clásico incluyen temas como la teoría de la variable compleja, ecuaciones diferenciales y de diferencias, y las transformadas de Laplace y z .

El control clásico utiliza el análisis en frecuencia y el método del lugar de las raíces para plantear modelos matemáticos cuantitativos de sistemas físicos, que pueden ser descritos por ecuaciones diferenciales ordinarias. En el análisis en frecuencia se aplica el método de *respuesta en frecuencia*, para lo cual será necesario obtener la *función de transferencia*¹¹ del sistema y evaluarla para $s = j\omega$. La respuesta en frecuencia es la respuesta en *estado-estacionario* de un sistema a una señal de entrada sinusoidal. Para un sistema lineal e invariante en el tiempo de una entrada y una salida (SISO), como el que se muestra en la figura 2.2, la función de transferencia estará dada por la ecuación 2.1.a. La función de transferencia sinusoidal en estado estacionario puede ser expresada en términos de sus partes real e imaginaria (ecuación 2.2.b).

El lugar de las raíces es un método muy poderoso de análisis y diseño para la estabilidad y respuesta transitoria de un sistema. Este método se basa en representar gráficamente la ubicación de los polos de la función de transferencia característica en lazo cerrado.

$$G(s) = \frac{Y(s)}{U(s)} \quad 2.1.a$$

$$G(s)|_{s=j\omega} = G(j\omega) = R(\omega) + jX(\omega) \quad 2.2.b$$

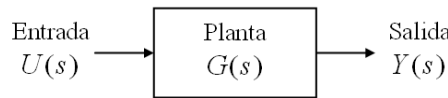


Figura 2.2 Esquema de un sistema de una entrada y una salida.

Algunas veces el control clásico puede ser aplicado con cierto éxito en sistemas no lineales, pero solamente en la región donde los sistemas tienen un comportamiento lineal.

2.2.2. Sistema de control en la teoría de control moderno

Además de las herramientas matemáticas utilizadas por la teoría de control clásico, el control moderno utiliza la teoría de conjuntos y el álgebra lineal para plantear, en el dominio del tiempo, los modelos matemáticos de sistemas físicos. En este caso, el comportamiento de los sistemas se representa a través de un conjunto de *variables de estado* (x_1, x_2, \dots, x_n) , que muestran cómo evoluciona el *estado* del sistema. Para un sistema con múltiples entradas y múltiples salidas, como el que se muestra en la figura 2.3, u es un vector de entradas, x es un vector de variables de estado y y es el vector de salidas. El conocimiento de los valores iniciales de las variables de

¹¹ La función de transferencia es la relación que existe entre la respuesta de un sistema y una señal de entrada o excitación.

estado en el tiempo t_0 , $[x_1(t_0), x_2(t_0), \dots, x_n(t_0)]$ y de las entradas $u_1(t), u_2(t), \dots, u_n(t)$, para $t \geq t_0$, es suficiente para determinar los valores futuros de las salidas y las variables de estado. Para un sistema lineal, la notación compacta para representar al sistema está formada por una *ecuación diferencial de estado o ecuación de estado* (ecuación 2.2.a) y una *ecuación de salida* (ecuación 2.2.b). La *ecuación de estado* describe la tasa de cambio del estado del sistema y las señales de entrada dadas en un tiempo t .

$$\frac{dx}{dt} = Ax + Bu \quad 2.2.a$$

$$y = Cx + Du \quad 2.2.b$$

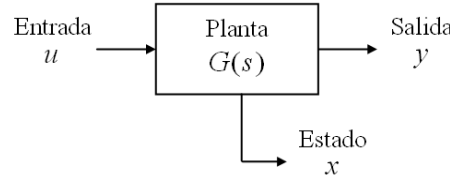


Figura 2.3. Esquema de un sistema multivariable.

2.2.3. Control digital

El control digital es una rama de la teoría de control que utiliza computadoras digitales para operar un sistema, en este caso la computadora puede ser utilizada como controlador y la información que ingresa o sale de ella debe estar en la forma de datos discretos (figura 2.4).

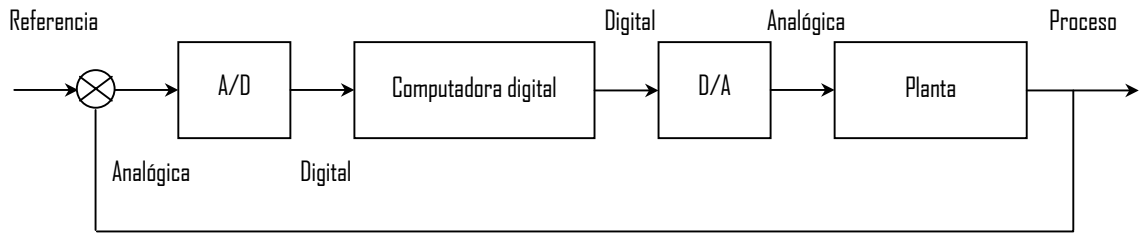


Figura 2.4. Diagrama a bloques de un sistema de control digital.

Cuando el sistema procesa tanto señales digitales como analógicas, será necesario proveer un medio que permita hacer la conversión de una forma a otra, según lo requiera el sistema. Un dispositivo que convierte señales analógicas a digitales se llama *convertidor analógico digital* (A/D). Un dispositivo que convierte señales digitales a analógicas se llama *convertidor digital analógico*.

(D/A). En la conversión analógica-digital, la señal analógica primero es convertida a una señal muestreada y después es transformada en una secuencia de dígitos binarios. En este caso, utilizar la transformada de Laplace para analizar al sistema se torna un poco difícil, por lo que es necesario utilizar un método como la *transformada z*, para describir y analizar el desempeño del sistema. La transformada z es un método matemático que permite transformar ecuaciones de diferencias a ecuaciones algebraicas lineales, que se pueden resolver fácilmente.

2.2.4. *Control inteligente*

El control inteligente es una metodología de control, que está fundamentada en la interacción de disciplinas como la inteligencia artificial, la investigación de operaciones y la teoría de sistemas. Es una aproximación del concepto de control que se basa en el estudio y entendimiento de cómo los seres humanos y algunos animales logran realizar ciertas tareas, lo cual proporciona un medio para obtener ideas de cómo resolver problemas de control con alto grado de dificultad. De esta manera, para resolver un problema de control dentro del campo del control inteligente, se trabaja considerando el uso de una aproximación de control motivada por la forma de representación y decisión hecha por humanos ó animales. El control inteligente utiliza diferentes metodologías para diseñar y construir un controlador. Una metodología de control es inteligente si usa técnicas y procedimientos motivados en humanos, animales o sistemas biológicos. Aunque existe una variedad de metodologías de control inteligente (control con redes neuronales, control difuso, control experto, algoritmos genéticos, etc.), en este trabajo se considera exclusivamente al control con redes neuronales.

2.2.5. *Control con redes neuronales*

El uso de las redes neuronales en aplicaciones de control ha crecido muy rápidamente, principalmente porque proporcionan una alternativa en el diseño de sistemas de control cuya complejidad es difícil de representar por medio de un modelo matemático o computacional [37]. Por otra parte, cuando la información sobre el comportamiento de un sistema consiste primordialmente de datos numéricos, el uso de redes neuronales artificiales para implementar un sistema de control es una buena opción. A la metodología de control que utiliza redes neuronales para el diseño de controladores, se le llama “neurocontrol”.

Neurocontrol

El *neurocontrol*¹², se refiere al uso de redes neuronales artificiales o naturales para controlar directamente ciertas acciones encaminadas a producir un resultado específico. Es una metodología de control en la cual el controlador en sí mismo es una red neuronal, o los

¹² También llamado Control Neural.

controladores son diseñados basados en una red neuronal que modela a la planta [37]. Está definido como el uso o estudio de redes neuronales bien específicas empleadas como controladores, en donde la red proporciona una señal de control vectorial $u(t)$ como función del tiempo. Así una red neuronal puede ser usada como un sistema de control “per se” donde la salida de la red neuronal puede ser la acción de control. Se tienen tres niveles de análisis en *neurocontrol*, en el primero de ellos se construyen sistemas con aprendizaje supervisado, en donde el sistema aprende una función no lineal o un mapeo estático de un vector X a un vector Y . En el segundo nivel se construyen sistemas complejos como son la reproducción y optimización dinámica de sistemas. En el tercer nivel, en investigación aplicada, se usan sistemas de *neurocontrol* junto con otros sistemas para construir sistemas complejos para aplicaciones específicas [26].

La capacidad de generalización que tienen las redes neuronales hace posible que puedan usarse como dispositivos de aproximación de funciones, lo que permite su aplicación en control, sobre todo cuando la información disponible sobre el comportamiento del sistema, está sólo en la forma de datos numéricos de entrada y salida, observados de la planta real o de un modelo simulado. Específicamente cuando los modelos matemáticos de la dinámica de la planta no están disponibles, las redes neuronales ofrecen un método para diseñar controladores, proporcionando información numérica sobre el comportamiento del sistema en la forma de datos de entrada-salida. En otras palabras una red neuronal puede ser usada como un “caja negra”, que representa un modelo de la planta.

Para construir un controlador basado en una red neuronal que pueda forzar a una planta a comportarse de una manera deseada, es necesario ajustar los parámetros de la red por medio del error observado, que es la diferencia entre la salida de la planta y la salida deseada. El ajuste de los parámetros de controlador será hecho por la propagación de los errores a través de la estructura de la red neuronal. Cuando el modelo matemático de la planta es desconocido necesitamos conocer al menos un modelo aproximado de la planta. Un modelo conocido de la planta se llama modelo identificado. Cuando usamos datos de entrada-salida de la planta para entrenar una red neuronal, que proporcione un modelo aproximado de la planta, obtenemos una *red neuronal modelo identificado*. Las redes neuronales modelo identificado son usadas en diseños de control neuronal directo. Para entender mejor el funcionamiento de un sistema de control neuronal, es necesario entender algunos fundamentos sobre redes neuronales.

2.3. Fundamentos de redes neuronales

La capacidad que muestran los sistemas biológicos para realizar las tareas más complicadas sin la necesidad aparente de hacer operaciones cualitativas complejas, y de aprender gradualmente a través del tiempo, los hacen atractivos como sistemas computacionales. En particular, las redes neuronales artificiales son atractivas como modelos computacionales, gracias a la capacidad que

tienen de aprender a partir de estímulos externos y generalizar, lo que permite que puedan ser utilizadas para procesar datos complejos. Las redes neuronales artificiales son modelos matemáticos inspirados en sistemas nerviosos biológicos, que son utilizados para establecer relaciones de entrada-salida de algún tipo de sistema. Han sido utilizadas para realizar funciones complejas en varios campos de aplicación, incluyendo el control. Están compuestas de elementos más simples, llamados neuronas, que operan en paralelo. Una neurona artificial es un modelo matemático simplificado de una neurona biológica. El modelo considera a las neuronas unidades elementales de procesamiento de la información. La información que ingresa a la red (en la forma de señales), es pasada a las neuronas a través de conexiones. Cada conexión tiene un valor llamado “peso” que multiplica a la señal transmitida. Cada neurona realiza un procesamiento interno, que aplica una función que opera con los valores recibidos de las neuronas de la capa anterior, tomando en cuenta el valor de las entradas x_i y los pesos sinápticos, w_i , de las conexiones. La activación o desactivación de la neurona depende de un valor llamado “bias¹³”. En la figura 2.5, se muestra el modelo de una neurona artificial¹⁴. El *bias* b puede ser tratado como otro peso, agregando un nodo de entrada x_0 , que siempre toma el valor de entrada $x_0 = +1$ y poniendo $w_0 = -b$. La función de activación es f .

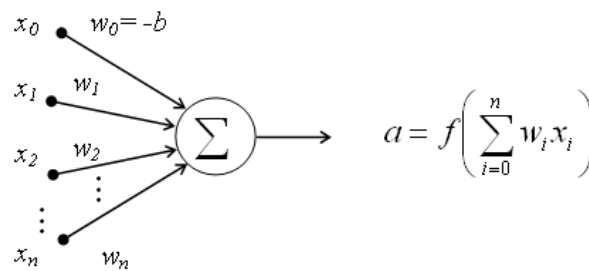


Figura 2.5. Neurona artificial con pesos y sesgo.

2.3.1. Arquitectura de redes

La arquitectura de una red neuronal se refiere a la forma en la que las neuronas están agrupadas e interconectadas dentro de la estructura de la red. Inicialmente, las neuronas se agrupan en capas, y una o más capas forman una red neuronal. De esta manera, las redes neuronales se pueden clasificar por el número de capas y por el sentido que tienen las conexiones entre las neuronas. Tomando en cuenta el número de capas, las redes neuronales se pueden clasificar

¹³ En inglés bias y traducido al español como umbral o sesgo. Si la neurona artificial es un perceptrón, el bias se comporta como un umbral de disparo. Si la neurona artificial es un ADALINE, el sesgo proporciona un grado más de libertad.

¹⁴ Es un perceptrón cuando la función de transferencia (también llamada de activación) de la neurona es un escalón. Es un ADALINE cuando la función de transferencia es una función lineal.

como redes *neuronales monocapa* y redes *neuronales multicapa*. En su forma básica, las redes monocapa tienen uniones laterales que las conectan con otras neuronas de la misma capa. Las redes con múltiples capas, tienen uniones que las conectan con neuronas de otra capa. El sentido que tienen las conexiones entre neuronas da origen a las redes neuronales con conexiones hacia delante (feedforward), y a las redes recurrentes o con retroalimentación (feedback). A continuación sólo se presentará a las redes con conexiones hacia delante.

2.3.2. Redes feedforward

Las redes *feedforward* son redes con múltiples capas que están organizadas e interconectadas en secuencia, y cada neurona dentro de una capa proporciona entrada solamente a las neuronas en la siguiente capa de la secuencia, no tiene conexiones hacia atrás o con neuronas de la misma capa. Las conexiones comienzan en la capa que recibe la señal exterior (capa de entrada), continuando con las capas intermedias (capas ocultas), siguiendo hasta la capa de salida. Por esta razón se les llaman redes con conexiones hacia adelante o *feedforward*. Antes de usar una red *feedforward*, se debe entrenar para que una entrada particular conduzca a un objetivo de salida específico. Para esto, se aplica una *regla de entrenamiento* que tiene como objetivo *adaptar* los valores de los pesos de las conexiones entre las neuronas, calculando los cambios deseados para los *pesos* y los *bias*, dado un vector de entrada y el error asociado. El proceso para hallar nuevos pesos y nuevos bias se repite hasta que el error se reduce al mínimo. Las redes *feedforward* se pueden entrenar principalmente bajo dos esquemas de aprendizaje: supervisado y no supervisado. A continuación únicamente se presenta el aprendizaje supervisado.

2.3.3. Aprendizaje supervisado

En el *aprendizaje supervisado*, la red neuronal debe aprender el mapeo entre un vector de entradas $X(t)$ y un vector de salidas deseadas $Y(t)$. La red neuronal que satisface el mapeo funcional entre estos dos vectores puede tomar muchas formas, sin embargo, es posible modelar este mapeo, tomando ciertas consideraciones. Según Werbos en [64], se puede hacer aprendizaje supervisado para alguna función o mapeo tal que:

$$\hat{Y} = f(x, y) \tag{2.3}$$

En donde el mapeo funcional entre $X(t)$ y $Y(t)$, para una red *feedforward* está modelado por:

$$x_i = X_i; \quad 1 \leq i \leq m \tag{2.4}$$

$$net_i = \sum_{j=1}^{i-1} W_{ij} x_j; \quad m < i \leq N + n \tag{2.5}$$

$$x_i = \frac{1}{1 + \exp(-net_i)} \quad m \prec i \leq N + n \quad 2.6$$

$$\hat{Y}_i = x_{i+N} \quad 1 \leq i \leq n \quad 2.7$$

Donde m es el número de entradas, n es el número de salidas, y N es la medida de que tan grande se elige hacer la red. $N - n$ es el número de “neuronas ocultas”. Estas ecuaciones también muestran cómo las salidas de la red, dependen de sus entradas y de los pesos.

En la década de 1960 se introducen los primeros algoritmos de entrenamiento para redes *feedforward*, con el algoritmo LMS¹⁵ (Widrow y Hoff) y la regla del perceptron (Rosenblatt, 1962). Pero el mayor avance toma lugar en 1971, cuando Paul Werbos desarrolla el algoritmo *backpropagation* para redes neuronales con múltiples capas. El resto del trabajo de Werbos permaneció casi desconocido por el mundo científico hasta 1986, cuando Rumelhart, Hinton y Williams descubrieron la técnica y la hicieron ampliamente conocida [66]. A continuación se describen los algoritmos *LMS* y *backpropagation*.

2.3.4. El algoritmo LMS

El algoritmo del error cuadrático medio menor (LMS), es un algoritmo de aprendizaje supervisado, que tiene como objetivo reducir el error entre la respuesta deseada, d_k , y la respuesta actual, s_k , en una red neuronal. La función de error más comúnmente utilizada es el *Error Cuadrático Medio* (MSE¹⁶), que está definido como:

$$mse = \frac{1}{N} \sum_{k=1}^N \varepsilon_k^2 = \frac{1}{N} \sum_{k=1}^N (d_k - s_k)^2 \quad 2.8$$

Las aproximaciones más populares para reducir el MSE, en redes de uno y de múltiples elementos, están basadas en el método del *gradiente descendiente*. La adaptación de una red por *gradiente descendiente* comienza con un valor arbitrario W_0 para el vector de pesos del sistema. El gradiente de la función del MSE es medida y el vector de pesos es alterado en dirección opuesta al gradiente medido. Este procedimiento es repetido para que el MSE sea sucesivamente reducido en promedio, causando que el vector de pesos se aproxime a un valor óptimo local. El método del gradiente descendiente puede ser descrito por la relación:

$$W_{k+1} = W_k + \mu(-\nabla_k) \quad 2.9$$

Donde μ es un parámetro que controla la *estabilidad y la tasa de convergencia*, y ∇_k es el valor del

¹⁵ De las siglas en inglés de Least Mean Square error .

¹⁶ De las siglas en ingles de Mean Square Error.

gradiente en un punto sobre la superficie del MSE correspondiente a $W = W_k$. El algoritmo LMS trabaja realizando aproximaciones descendientes importantes en el MSE en el espacio de pesos. Esta superficie es una función cuadrática de los pesos y por lo tanto convexa y tiene un único mínimo global. Un gradiente instantáneo basado en el cuadrado del error instantáneo es:

$$-\nabla_k = \frac{\partial \varepsilon_k^2}{\partial W_k} = \begin{Bmatrix} \frac{\partial \varepsilon_k^2}{\partial w_{0k}} \\ \vdots \\ \frac{\partial \varepsilon_k^2}{\partial w_{nk}} \end{Bmatrix} \quad 2.10$$

El LMS trabaja usando el gradiente instantáneo estimado $\tilde{\nabla}_k$ en lugar del gradiente ∇_k . Haciendo esta sustitución en la ecuación (2.9) se tendrá:

$$W_{k+1} = W_k + \mu(-\tilde{\nabla}_k) = W_k - \mu \frac{\partial \varepsilon_k^2}{\partial W_k} \quad 2.11$$

El error presente ε_k está definido como la diferencia entre la respuesta deseada d_k y la salida $s_k = W_k^T X_k$:

$$\varepsilon_k \stackrel{\Delta}{=} d_k - W_k^T X_k \quad 2.12$$

Realizando la diferenciación de la ecuación (2.11) y reemplazando el error lineal por la definición dada en la ecuación 2.12 obtenemos:

$$W_{k+1} = W_k - 2\mu \varepsilon_k \frac{\partial (d_k - W_k^T X_k)}{\partial W_k} \quad 2.13$$

Realizando la diferenciación en la ecuación (2.13), obtenemos:

$$W_{k+1} = W_k + 2\mu \varepsilon_k X_k \quad 2.14$$

Las ecuaciones 2.12, 2.13 y 2.14, expresan el algoritmo LMS.

2.3.5. *Backpropagation*

El algoritmo backpropagation es una generalización del algoritmo LMS, para redes con múltiples capas con funciones de transferencia diferenciables no lineales. Cuando se aplica el algoritmo backpropagation a redes feedforward, se ajusta los pesos en dirección opuesta al gradiente instantáneo de la suma del error cuadrado en el espacio de pesos. La suma instantánea del error

cuadrado \mathcal{E}_k^2 , es la suma de los cuadrados de los errores en cada una de las N_y salidas de la red (ecuación 2.15):

$$\mathcal{E}_k^2 = \sum_{i=1}^{N_y} \mathcal{E}_{ik}^2 \quad 2.15$$

En su forma simple el *entrenamiento backpropagation* comienza presentando un vector X de patrones de entrada a la red, propagándolo hacia delante a través de la red para generar un vector Y de salida y calculando el error en cada salida. Continúa propagando los efectos de los errores hacia atrás a través de la red para asociar una *derivada del error cuadrático* δ con cada neurona, calculando el gradiente por cada δ , y finalmente actualizando los pesos de cada neurona basándose en el correspondiente gradiente. El proceso se repite cuando se presenta un nuevo patrón a la red. Los valores iniciales de los pesos son puestos generalmente a pequeños valores aleatorios. Para una neurona de salida, con función de transferencia sigmoidea se tendrá:

$$\delta = \mathcal{E} \cdot sgm'(s) \quad 2.16$$

Donde \mathcal{E} es el error en la salida de la neurona y s es la sumatoria de salida de la misma unidad.

El procedimiento para hallar δ^l , la derivada del error cuadrático asociado a una neurona de la capa oculta \mathcal{L} , implica multiplicar cada derivada $\delta^{(l+1)}$ asociada a cada neurona en la capa por el peso W asociado correspondiente. Las derivadas del error cuadrático pesado, son entonces sumadas, produciendo un término de error $\mathcal{E}^{(l)}$, el cual en su momento es multiplicado por $sgm'(s^{(l)})$, la derivada con función sigmoidea. La δ correspondiente a la neurona j en la capa oculta \mathcal{L} está dada por:

$$\delta_j^{(l)} = sgm'(s_j^{(l)}) \sum_{i \in N^{(l+1)}} \delta_i^{(l+1)} w_{ij}^{(l+1)} \quad 2.17$$

Donde $N^{(l+1)}$ es un conjunto que contiene los índices de todas las neuronas en la capa $\mathcal{L}+1$ y $w_{ij}^{(l+1)}$ es el peso que conecta a la neurona i , en la capa $\mathcal{L}+1$ a la salida de la neurona j en la capa \mathcal{L} . Actualizando los pesos de la neurona, con el método del gradiente descendiente, usando el gradiente instantáneo se tiene:

$$W_{k+1} = W_k + \mu(-\tilde{\nabla}_k) = W_k + 2\mu\delta_k X_k \quad 2.18$$

En donde W es el vector de pesos de la neurona y X es el vector de entrada a la neurona. Después de retropropagar todas las derivadas del error cuadrado, se completa una iteración backpropagation agregando a cada vector de pesos, el vector de entrada correspondiente,

escalado por las derivadas del error cuadrático asociado. Las ecuaciones 2.16, 2.17 y 2.18 comprenden la regla general de actualización de pesos en el algoritmo backpropagation para redes neuronales con capas. Los pasos para el algoritmo backpropagation son los siguientes:

1. Se inicializan todos los pesos de las sinapsis con pequeños valores aleatorios.
2. Se aplica un vector X de entrada y se especifica la salida deseada, d , que debe proporcionar la red.
3. Se calcula la salida actual de la red. Para esto, es necesario calcular la salida en cada capa, comenzando en la primera capa y hasta llegar a la capa de salida, esta será la salida Y , de la red.

Para calcular la salida en la capa oculta se tendrá que:

- a) Calcular los valores de las entradas netas para las neuronas ocultas con:

$$net_{pj}^h = \sum_{i=1}^N w_{ji}^h x_{pi} + b_j^h$$

- b) Calcular las salidas de las neuronas ocultas:

$$y_{pj} = f_j^h(net_{pj}^h)$$

Para calcular la salida en las neuronas de salida se tendrá que:

- a) Calcular los valores de las entradas netas para las neuronas de salida con:

$$net_{pk}^o = \sum_{j=1}^L w_{kj}^o y_{pj} + b_k^o$$

- b) Calcular las salidas:

$$y_{pk} = f_k^o(net_{pk}^o)$$

4. El siguiente paso es calcular los términos de error para todas las neuronas, comenzando por las neuronas en la capa de salida.

1. Los términos de error para las neuronas de salida están dados por

$$\delta_{pk}^o = (y_{pk} - d_{pk}) f_k^{o'}(net_{pk}^o)$$

2. para calcular los términos de error para las unidades ocultas tendremos:

$$\delta_{pj}^h = f_j^{h'}(net_{pj}^h) \sum_k \delta_{pk}^o w_{kj}^o$$

Los términos de error en las unidades ocultas son calculados antes que los pesos de las conexiones para las unidades de la capa de salida han sido actualizados.

5. Actualización de los pesos en la capa de salida.

$$w_{kj}^o(t+1) = w_{kj}^o(t) + \eta \delta_{pk}^o y_{pj}$$

La actualización de los pesos en la capa oculta

$$w_{ji}^h(t+1) = w_{ji}^h(t) + \eta \delta_{pk}^h x_i$$

6. El proceso se repite hasta que el error, $E_p = \frac{1}{2} \sum_{k=1}^M \delta_{pk}^2$, es aceptablemente pequeño.

2.3.6. El tamaño de la red

Una vez que se ha elegido una red neuronal para resolver un problema, es necesario definir parámetros como: el *tamaño de la red*, la *función de activación de las neuronas*, el número de *entradas y salidas* y el número de *patrones para el entrenamiento*.

El *tamaño* se refiere al número de capas y al número de neuronas en cada capa. El número de neuronas en las capas de entrada y salida puede ser definido a partir del número de patrones en los vectores de entrada y salida, para una aplicación específica. El diseñador de la red, puede especificar el número de capas intermedias y el número de neuronas en cada capa, basándose en algunos *criterios*, *reglas heurísticas* o *métodos especializados* para acotar estos parámetros [21].

Los **criterios** toman en consideración algunos aspectos generales sobre la composición de la red; por ejemplo, se sabe que una red demasiado simple puede conducir a resultados poco precisos y una red muy grande puede complicar el aprendizaje. Y que una función de activación no lineal no es decisiva en la capacidad que tiene la red para calcular una correspondencia entrada-salida. Y que una red con una capa oculta cuyas neuronas tienen una función de activación no lineal y una capa de salida cuyas neuronas tienen una función de activación lineal, puede aproximar cualquier función con un error relativamente pequeño, es decir, se comporta como un *aproximador universal de funciones*. Aunque se considera que un *aproximador universal de funciones*¹⁷ puede tener hasta dos capas ocultas, en la mayoría de los casos una red con una única capa oculta resulta ser suficiente para obtener buenos resultados.

Por otra parte, las **reglas heurísticas** hacen consideraciones que no están comprobadas matemáticamente, pero que han demostrado ser eficientes en diversas aplicaciones prácticas. Por ejemplo, Lippmann [39] considera que una red (MLP) con una capa oculta es suficiente para resolver problemas relativamente complejos, si la capa intermedia tiene al menos $2m$ neuronas, siendo m el número de entradas. Hech-Nielsen [27] aplica el teorema de Kolmogorov para demostrar que una red con una capa oculta, con neuronas con función de activación no lineal, continua, monótona creciente, que tiene $2m + 1$ neuronas, es suficiente para aproximar cualquier función continua de m variables de entrada. La *regla de la pirámide geométrica*, considera que el número de neuronas en cada capa sigue una progresión geométrica. Para una red con una única

¹⁷ En las décadas de 1980 y 1990, diversos grupos de investigadores propusieron teoremas que demostraban matemáticamente que un MLP, con un máximo de dos capas ocultas constituía un aproximador universal de funciones. Una de las aportaciones más conocidas es el teorema de Funahashi (1989). Apéndice VI.

capa oculta, el número de neuronas se puede aproximar por $\sqrt{m \times n}$, en donde m es el número de variables de entrada y n es el número de variables de salida. Por último, la *regla de la capa oculta-capas de entrada*, propone que el número de neuronas en la capa oculta está relacionado con el número de neuronas en la capa de entrada. La regla más usada es 2×1 , que quiere decir que el número de neuronas en la capa oculta no deberá ser superior al doble de las neuronas en la capa de entrada.

Los **métodos especializados** para determinar el tamaño óptimo de la red se pueden clasificar en: *Métodos poda*, y *métodos constructivos*.

Los **métodos de poda**, parten de tener una red con una arquitectura compleja, que está entrenada con un algoritmo que añade un término de penalización a la función de error, de esta manera elimina los parámetros redundantes (pesos y/o neuronas) a través de un proceso conocido como poda de la red.

Los **métodos constructivos**, parten de una arquitectura mínima de red, que en el entrenamiento genera un error de aprendizaje medio. Dependiendo de este error, se van incorporando neuronas y/o conexiones para mejorar el ajuste del modelo, hasta que se alcanza un resultado óptimo.

En la tesis se considera el uso de una red neuronal con una sola capa oculta, y se aplica un algoritmo heurístico para determinar el número óptimo de neuronas en la capa oculta, también se aplica el método de prueba y error para encontrar el valor óptimo para el momento y la tasa de aprendizaje para el algoritmo backpropagation estándar.

2.4. El sistema de control propuesto

Se trata de un sistema de control no lineal basado en un microprocesador de propósito general, que se utiliza un esquema básico de aprendizaje por imitación para realizar el control de velocidad de un motor de corriente directa.

El esquema básico de aprendizaje por imitación está compuesto de tres elementos: percepción de una acción, procesamiento de la información percibida (aprendizaje) y reproducción de la acción percibida (acto motor).

Es necesaria la presencia de un motor que modele el movimiento y con esto proporcione los parámetros de entrada al sistema. El motor modelo no necesariamente deberá tener las mismas características dimensionales y rangos de operación, que el motor con el sistema de control propuesto.

En una primera etapa, en el procesamiento de la información percibida, una red neuronal artificial feedforward aprenderá el mapeo entre el movimiento percibido y el movimiento que debe ejecutar el motor que aprende, con lo cual se transferirá la regla de control para el motor.

Para determinar las características del movimiento percibido (velocidad y sentido), el sistema contará con un módulo de decodificación de señales. Para ejecutar el movimiento, el sistema contará con dos módulos que generan las señales de control.

Las señales de entrada (señales codificadas) y salida (señales de control), serán transmitidas al sistema vía un puerto paralelo estándar.

Después del aprendizaje el sistema podrá desempeñar cualquier perfil de movimiento.

2.5. Otros trabajos

Las aplicaciones de las redes neuronales en el control de motores de corriente directa han sido muy variadas, pero ninguna de estas aplicaciones se ha basado en un esquema de aprendizaje por imitación.

Por ejemplo en [9] Baruch et al utilizan una red neuronal recurrente junto con un algoritmo de entrenamiento “Back-propagation” aplicado a la identificación en tiempo real y control adaptativo de un motor de CD. En [25] Gouda y El-Samany, utilizan el algoritmo de entrenamiento “Back-propagation Levenberg Marquardt”, en donde la velocidad deseada, la corriente de armadura y el correspondiente voltaje de armadura de un motor de CD se usan para entrenar la red para generar un mapeo entre las entradas de la red (velocidad del motor y corriente de armadura) y el voltaje de armadura. En [54] Rubaai et al, diseñan un controlador de alto desempeño que realiza identificación y control simultáneamente, la dinámica del motor y la carga son modeladas en línea y son controladas usando dos diferentes redes neuronales basadas en los esquemas de control e identificación cuando el motor está en operación. En [55] Rubaai y Kotaru realizan el control de un motor, en el que utilizan una red neuronal artificial feedforward y un algoritmo de aprendizaje adaptativo, para capturar y emular mapeos no lineales, para implementar una regla de control no lineal completa.

Ventajas del sistema propuesto:

Las ventajas del sistema de control propuesto son:

- No es necesario establecer un modelo matemático exacto no lineal del sistema.
- Al aprender la red neuronal el mapeo entre el movimiento percibido del modelo y el que debe ejecutar el motor, se estará estableciendo la regla de control para el sistema.
- Aunque los motores no tengan las mismas dimensiones, se podrá transferir una regla de control para el motor con el sistema propuesto, siempre y cuando se puedan extraer los valores de velocidad y sentido de movimiento del motor modelo.

2.6. Resumen

El aprendizaje es la adquisición de una conducta duradera por medio de la práctica; la imitación es la capacidad de reconocer y reproducir las acciones de otros, y el aprendizaje por imitación es la capacidad de aprender por medio de la observación de las acciones que realizan otros. A través del tiempo, la habilidad de imitar ha sido estudiada para descubrir los mecanismos que implica; actualmente, en la psicología cognitiva, las teorías de la imitación tratan de responder al problema de correspondencia, el cual cuestiona ¿Cómo es que el imitador conoce qué patrones de activación motora harán su acción parecida a la del modelo? Pero esta es una respuesta que aún no ha sido hallada.

Hoy en día, las ventajas que proporciona la imitación para la transferencia de conocimiento y habilidades, han sido aplicadas en diversas áreas de la ciencia, en las que están incluidas las ciencias computacionales. Desde el punto de vista de las ciencias computacionales, la imitación es un problema complejo que requiere del mapeo de una acción percibida desde un marco de referencia externo a un marco de referencia interno en el observador. En particular, en el área de control, la imitación plantea la alternativa de transferir las habilidades de control de un sistema a otro por medio de ejemplos de comportamiento. En este contexto, se plantea un sistema de control para motores que tiene la capacidad de aprender la tarea de control por medio de un conjunto reducido de patrones de comportamiento, en donde el aprendizaje, se lleva a cabo por medio de una red neuronal artificial. En el capítulo siguiente se discuten algunas generalidades de los sistemas de control para motores de corriente directa, se plantea un esquema básico para el aprendizaje por imitación y también se plantea el esquema general de control, detallando cómo funciona cada una de las partes que lo conforman.

Capítulo 3

El sistema de control

3.1. Generalidades

Es un sistema de control digital para motores de corriente directa que está basado en un microprocesador de propósito general, y utiliza un esquema básico de imitación para aprender la regla de control. El sistema cuenta con un módulo que le permite percibir el movimiento realizado por otro motor (el cual se mueve gobernado por una regla de control ya establecida), y mapearlo a la estructura propia del motor para que éste aprenda la regla de control y pueda realizar su propio movimiento, esto quiere decir que aún cuando los motores (el que exhibe el movimiento y el que aprende) sean de distintas dimensiones, con el esquema planteado, el sistema que aprende podrá mapear el movimiento percibido a la estructura propia del motor y así este aprenderá la regla de control.

Para conocer la velocidad y sentido de movimiento del eje del motor que exhibe el movimiento, se utiliza un codificador óptico de cuadratura. Por otra parte, como es necesario variar la velocidad y el sentido del movimiento del eje del motor que aprende, se puede utilizar un método de modulación digital que permita manejar estos dos parámetros simultáneamente. El método más utilizado por su simplicidad y fácil implementación es la *modulación por ancho de pulso*, PWM, en este caso se utiliza un método de modulación llamado *PWM Signo-magnitud*.

A continuación, se describe brevemente como opera un codificador óptico de cuadratura y también se describe brevemente en qué consiste el método de modulación *PWM Signo-Magnitud*.

3.1.1. Codificador óptico

Para determinar la velocidad y sentido del movimiento del eje de un motor, se puede acoplar un codificador de movimiento rotatorio a su eje. Los codificadores más usados son los codificadores ópticos de cuadratura. Estos codificadores utilizan dos señales de salida (A y B),

en cuadratura¹⁸ para codificar la velocidad y sentido del movimiento. Para codificar la velocidad, ambas señales experimentan una variación en su periodo, de manera que a mayor velocidad el periodo de las señales disminuye, o bien la frecuencia de las señales aumenta. Para determinar el sentido de movimiento, las señales experimentan un desfaseamiento una con respecto a la otra, *A* con respecto a *B* o viceversa, con lo cual se codifica el sentido de movimiento. En la figura 3.1, se muestra las señales de salida de un codificador óptico. Cuando el eje del motor gira en el sentido de las manecillas del reloj, ocurre primero el pulso en la señal *A*, y sólo existen cuatro patrones de entrada únicos que corresponden a cada uno de los estados mostrados. En el sentido contrario ocurre primero el pulso de la señal *B*.

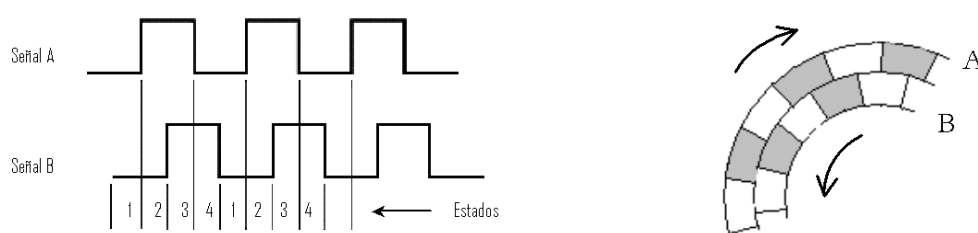


Figura 3.1. Señales proporcionadas por el codificador de cuadratura.

3.1.2. *PWM Signo-Magnitud*

El método de modulación *PWM Signo-Magnitud*, está formado por una señal de *signo* y una señal de *magnitud*. La señal de *signo* proporciona información sobre la dirección de giro del eje del motor. Como se muestra en la figura 3.2, cuando mantiene un nivel alto la corriente en el motor fluirá de manera que el eje del motor gire en sentido de las manecillas del reloj (sentido positivo) y cuando mantiene un nivel bajo, la corriente en el motor fluirá en el sentido contrario haciendo que el eje del motor gire en el sentido contrario a las manecillas del reloj (sentido negativo).

La señal de *magnitud* es una señal modulada en su ciclo de trabajo, por lo que la magnitud de la corriente entregada al motor para que pueda realizar un movimiento es proporcional a la duración del pulso de la señal.

Cuando se utiliza PWM signo-magnitud para controlar la velocidad y el sentido del movimiento del eje del motor, se debe utilizar adicionalmente un circuito que permita el manejo simultáneo de ambas señales, y que además permita separar la parte digital de la parte analógica del sistema, para esto se puede utilizar un circuito llamado *punte H*.

¹⁸ Dos señales están en cuadratura cuando están desfasadas 90° una con respecto de la otra.

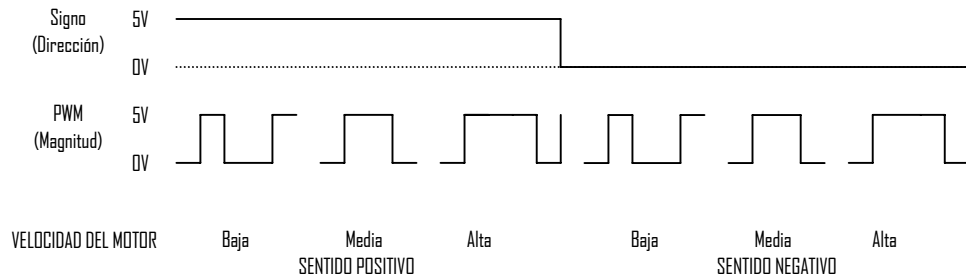


Figura 3.2. Señal PWM Signo-Magnitud

La figura 3.3 ilustra como cambia la configuración del *punte H* según el nivel de la señal *signo*. Cuando la señal de *signo* tiene un nivel alto, la corriente fluye de V_{cc} , pasa por *S1*, por el motor y por *S4*. Cuando la señal de *signo* tiene un nivel bajo, la corriente fluye en sentido contrario, esto es, parte de V_{cc} , pasa por *S2*, a través del motor y por *S3*. Es precisamente este cambio en el flujo de la corriente el que determina el cambio en el sentido del giro del eje del motor.

El ancho de pulso de la señal determina la velocidad de giro del eje del motor, la cual será proporcional al promedio de corriente que deberá entregar el *punte H* al motor para que pueda alcanzar la velocidad deseada (figura 3.4).

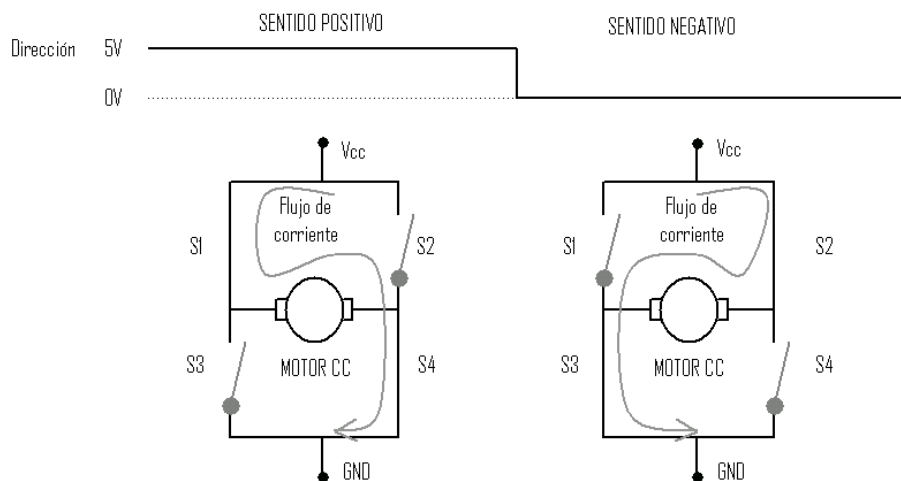


Figura 3.3. Funcionamiento básico de un puente H.

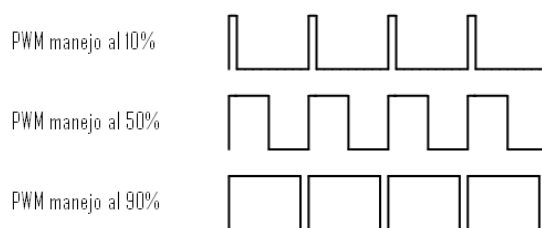


Figura 3.4. Ciclo PWM al 10%, 50% y 90%.

Por ejemplo, cuando el ciclo de trabajo de la señal es del 10%, el *punte H* deberá proporcionar la cantidad de corriente necesaria para que el motor gire al 10% de su máxima velocidad, entonces, cuanto más grande sea el ciclo de trabajo de la señal *PWM de Magnitud*, el *punte H* proveerá una cantidad de corriente de mayor proporción al motor.

3.2. Esquema básico para el aprendizaje por imitación

El esquema propuesto para el aprendizaje por imitación está dividido en tres partes: percepción, aprendizaje y acto motor (figura 3.5). La *percepción* permite determinar los parámetros característicos del movimiento que se percibe. El *aprendizaje* realiza el mapeo entre los parámetros que caracterizan al movimiento percibido y los parámetros que el acto motor utilizará para generar las señales de control. Y el *acto motor* genera las señales para el control.

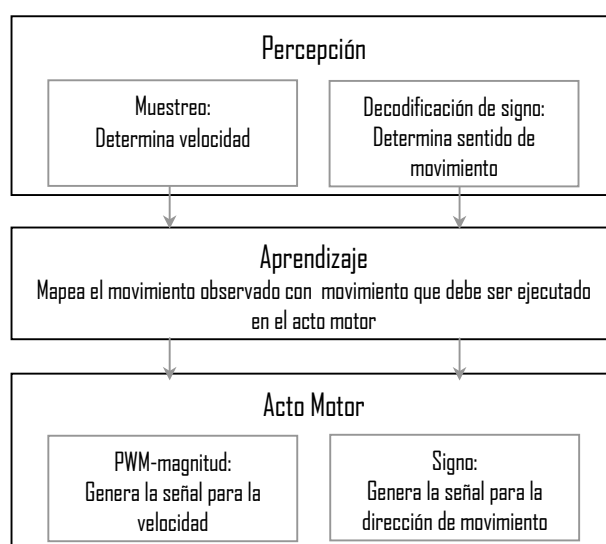


Figura 3.5. Esquema para el aprendizaje por imitación.

La percepción esta formada por dos módulos: *muestreo*¹⁹ y *decodificación de signo*. El *aprendizaje* se lleva a cabo por medio de una red neuronal artificial. Para realizar el *acto motor*, el sistema de control del motor cuenta con un módulo que genera la señal *PWM-magnitud* y un módulo de *signo* necesarias para llevar a cabo un movimiento.

3.3. El sistema de control con aprendizaje por imitación.

Una vez que se ha establecido el esquema básico para el aprendizaje por imitación, el sistema de control propuesto estará definido como se muestra en la figura 3.6.

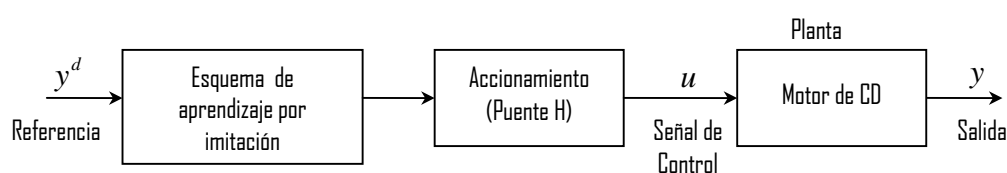


Figura 3.6. Diagrama a bloques del sistema de control con aprendizaje por imitación.

En el control, la referencia proporciona al sistema los datos sobre la acción de movimiento que debe reproducir el motor. La referencia es capturada por el sistema por medio de una interfaz de comunicación implementada a través de un puerto paralelo y la información proveniente de dichas señales es suministrada al esquema de *aprendizaje por imitación*, el cual procesa la información y proporciona los datos necesarios para generar las señales para el control del motor. Las señales generadas son suministradas al bloque de accionamiento (puente H), el cual proporciona la cantidad de corriente necesaria para que el motor realice el movimiento.

Para explicar cómo funciona el aprendizaje por imitación, se hará referencia nuevamente al esquema de la figura 3.5 y se explicará cómo funciona cada uno de los módulos que constituyen al sistema. Como se puede ver el esquema está compuesto por tres subsistemas: percepción, aprendizaje y acto motor

Percepción

La percepción cuenta con dos módulos que le permiten estimar las acciones a imitar: el *muestreo* y la *decodificación de signo*. Por medio de estos módulos, el sistema recibe la información necesaria para seguir la pista del movimiento que debe imitar.

El *muestreo* y la *decodificación de signo* están diseñados para determinar la velocidad y sentido de movimiento, por medio del periodo y el desfase que presentan dos señales en cuadratura

¹⁹ Se maneja el término muestreo para determinar sólo el periodo de la señal, ya que se trata de una señal digital cuya magnitud se mantiene constante y corresponde a la magnitud de una señal TTL.

como las mostradas en la figura 3.1. El período de las señales A y B, varía con respecto a la velocidad, mientras que el ciclo de trabajo se mantiene constante. La variación del periodo que caracteriza a las señales A y B se toma como parámetro para determinar el desempeño del perfil de movimiento que el motor percibe de otro motor en movimiento. Ambas señales son iguales y permanecen desfasadas 90° por estar en cuadratura. Para que el subsistema de percepción procese la información proporcionada por las señales, será necesario analizar las señales para determinar su periodo (*muestreo*), y el sentido de movimiento del eje del motor (*decodificación de signo*). A continuación, se describe como funciona cada uno de estos módulos.

Muestreo

Realiza un seguimiento de las señales que ingresan al sistema para determinar su periodo, la figura 3.7, muestra un ejemplo básico de este mecanismo. Una forma de determinar el periodo de una señal (periódica) como A o B, es verificar cuando han ocurrido tres transiciones en el nivel de la señal, ya que esto es indicativo de que ha transcurrido un periodo completo de la señal. Adicionalmente, se lleva una cuenta del número de muestras que se han tomado en el transcurso de un período de la señal, comenzando la cuenta cuando ocurre la primera transición de nivel en el periodo y terminándola cuando ocurre la tercera transición. Para determinar de manera numérica el periodo de la señal, se multiplica el número de muestras tomadas en un período de la señal por el tiempo que transcurre entre la toma de dos muestras consecutivas (el periodo de muestreo), esto es:

$$\text{Periodo_de_la_señal} = \text{Número_de_muestras} \times \text{Periodo_de_muestreo}$$

En el ejemplo de la Figura 3.7, el *Número_de_muestras* = 4 y el *Periodo_de_muestreo* = *duración del temporizador*.

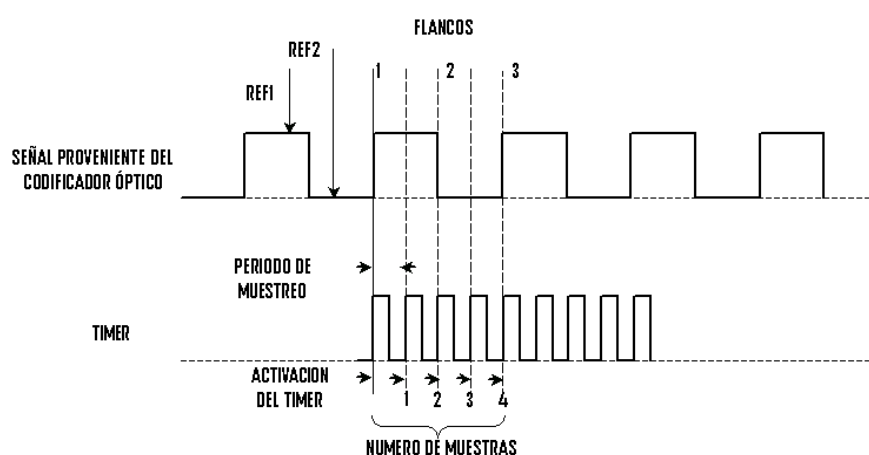


Figura 3.7. Mecanismo de muestreo del sistema sensorial.

En el mecanismo de muestreo es suficiente con hacer el seguimiento de una de las señales (A ó B), ya que, como se menciona anteriormente, ambas señales tienen características idénticas excepto por el desfase que manifiestan. De esta forma se evita que la información adquirida sea redundante.

Para comenzar la toma de muestras, la señal es examinada²⁰ para verificar cuándo ocurre una transición del nivel de la señal. Para asegurar que la primera muestra sea tomada cuando la señal experimenta una transición de nivel, se analizan dos casos, el primero de ellos es cuando se pretende hacer un muestreo de la señal y ésta se encuentra en un nivel alto, **REF1**, el segundo es cuando la señal se encuentra en un nivel bajo, **REF2**. Después de esto, cuando la transición es tomada por buena, comienza el muestreo. Para realizar el muestreo en sí, se activa el **TEMPORIZADOR** cuya función es la de producir el período del muestreo.

Para llevar la cuenta del número de transiciones de la señal, se hace un examen de la señal para determinar cuándo ocurre una transición (se verifica constantemente el nivel de la señal), en el diagrama de flujo de la figura 3.8 se muestra a esta acción de examinar como un elemento de decisión, en donde N_ANT es el nivel de la muestra anterior de la señal y N_ACT es el nivel de la muestra actual. Si $N_ANT \neq N_ACT$ ha ocurrido una transición que se contabiliza en la variable **FLANCOS**. Mientras, se sigue contabilizando la variable **MUEST**, que proporciona el número de muestras tomadas en un período de la señal que se muestrea.

Para producir el *periodo de muestreo* se utiliza el contador 2 del CI 8254 programado en *Modo 0*, el cual se encuentra en el sistema básico de temporización del sistema que aloja al sistema de control del motor.

Para que el contador opere debe ser cargado inicialmente con una palabra de control, la cual incluye el modo en el cual deberá operar y después se carga con una cuenta que corresponde al tiempo de duración de un período de muestreo. El formato de la palabra de control y el modo de operación se describieron en la sección V.5 del apéndice V. La manera en cómo funciona el contador cuando se le carga una cuenta se describe con más detalle en la sección V.5. del apéndice V. La rutina de la función periodo del programa BPNN.c que se encuentra en el Apéndice I, realiza el muestreo.

Decodificación de signo

El módulo de decodificación de signo determina el sentido del movimiento que se percibe. Para la decodificación de signo, se toman como referencia el desfaseamiento que existe entre las señales y cómo se presenta ese desfaseamiento, ya que esta característica es la que permite determinar el sentido del movimiento que tiene el eje del motor. En la figura 3.1, se muestran los

²⁰ En inglés a esta acción se le denomina “polling”.

cuatro estados que se pueden definir a partir del desfaseamiento que presentan las señales A y B. En el sentido positivo, cuando el eje del motor gira en el sentido de las manecillas de un reloj, ocurre primero el pulso en la señal A, y sólo existen cuatro patrones de entrada únicos que corresponden a cada uno de los estados mostrados.

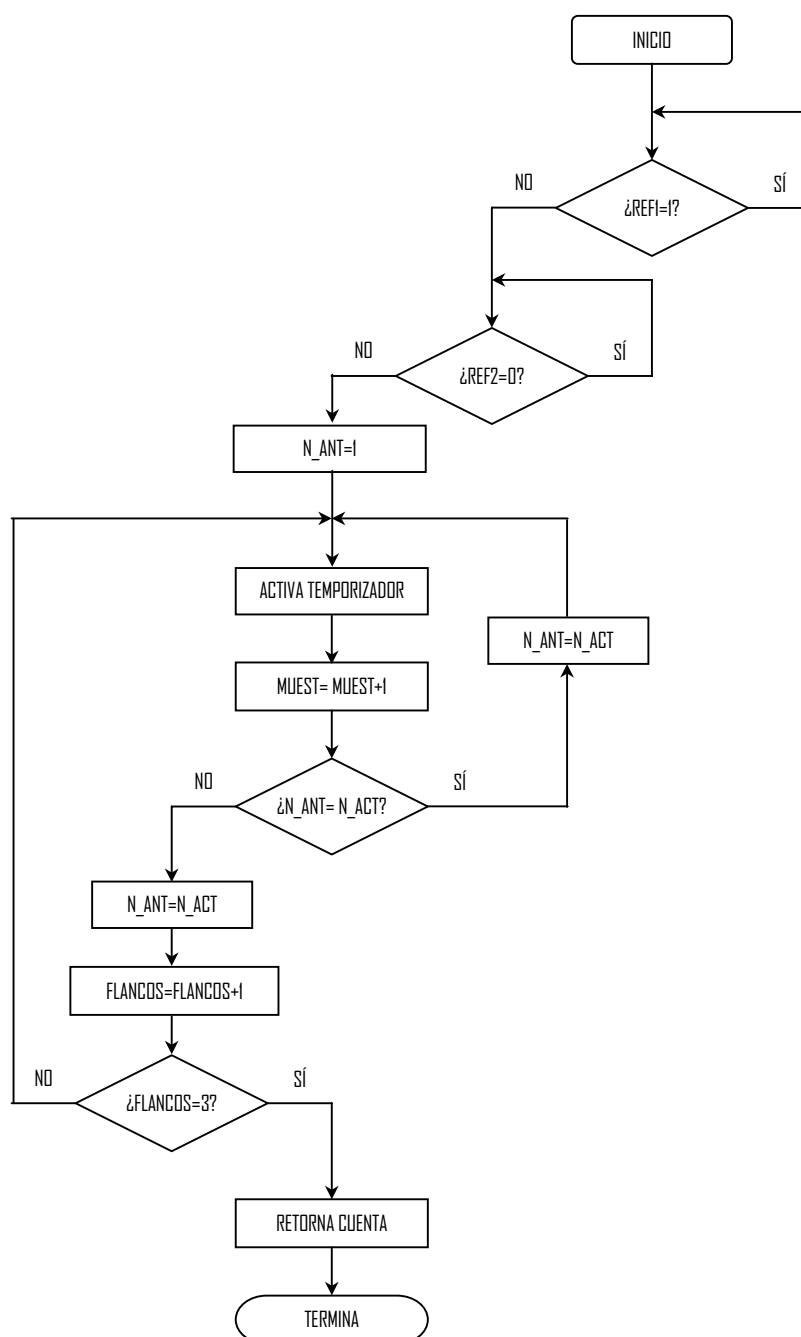


Figura 3.8. Diagrama de flujo del mecanismo de muestreo del sistema sensorial del motor.

Cada estado se puede interpretar como la combinación de dos bits, cada uno de los cuales representa el nivel que mantiene cada una de las señales en ese estado. La Tabla 3.1, muestra como es la combinación de los bits para cada uno de los cuatro estados cuando el eje del motor gira en sentido positivo y cuando lo hace en sentido negativo.

Tabla 3.1. Combinación de los bits para cada uno de los cuatro estados de las señales A y B.

GIRO DEL EJE DEL MOTOR EN EL SENTIDO POSITIVO				
Estados	1	2	3	4
A	0	1	1	0
B	0	0	1	1
GIRO DEL EJE DEL MOTOR EN EL SENTIDO NEGATIVO				
A	0	0	1	1
B	0	1	1	0

La figura 3.9 ilustra cada uno de los cuatro estados mostrados en la Tabla 3.1. Cada uno de los extremos de los ejes que dividen a la circunferencia en cuatro cuadrantes corresponde a cada uno de los cuatro estados. Cuando el eje del motor gira en el sentido positivo, las señales A y B pasan a través de los cuatro estados que forman la secuencia: 00, 10, 11 y 01; para el sentido negativo la secuencia será: 00, 01, 11 y 10. De esta manera, el mecanismo de *decodificación de signo* verifica de qué manera cambian los niveles de las señales, para determinar el sentido en el que gira el eje del motor. Como se puede ver en el diagrama de la figura 3.10, el movimiento en sentido negativo se decodifica como -1, y el movimiento en sentido positivo se decodifica como 1. La *decodificación de signo* se hace una vez, antes de hacer el muestreo de la señal. La rutina de la función *dirección_de_giro()* del programa BPNN.c que se encuentra en el Apéndice I, realiza la decodificación de signo.

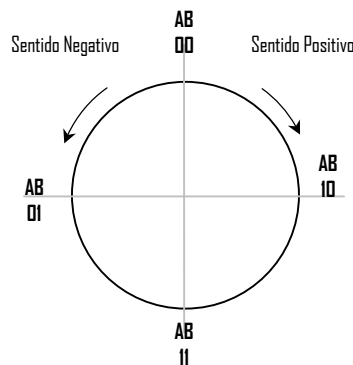


Figura 3.9. Esquema gráfico del sentido de giro del eje de un motor.

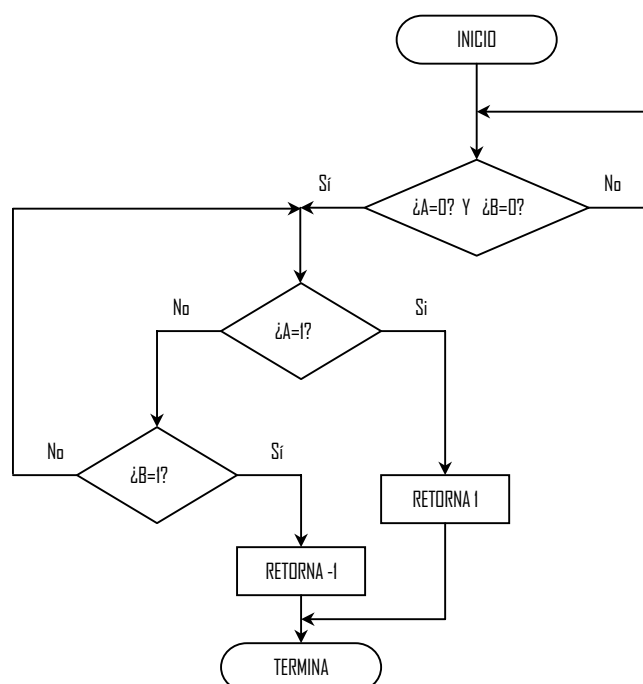


Figura 3.10. Diagrama de flujo del mecanismo de decodificación de signo.

El aprendizaje

Dado que el aprendizaje por imitación es una forma de aprendizaje inherente a algunas especies animales, en las cuales se ha encontrado evidencia de que existe un sistema de neuronas que están relacionadas con la imitación, se planteó que el aprendizaje se realizara a través de una red neuronal artificial, y así éste se asemejara más al mecanismo neuronal que en los organismos animales está relacionado con este tipo de aprendizaje.

En cuanto a aplicaciones de control, es justificable utilizar una red neuronal en el diseño de sistemas de control, sobre todo cuando la información que se tiene sobre el comportamiento del sistema a controlar está en la forma de datos de entrada y salida, y cuando el sistema presenta características no lineales, como es el caso, sobre todo cuando es necesario establecer un punto de “velocidad cero”, y movimiento tanto en sentido positivo como en sentido negativo.

Una vez que se ha elegido utilizar una red neuronal en una aplicación, el siguiente paso es definir el tipo de red neuronal mas adecuada en función de la tarea que ésta debe desempeñar dentro del sistema. También se debe definir el tamaño, el algoritmo de entrenamiento y otros parámetros que influyen en el desempeño de la red y del sistema de control en general. Pero antes es necesario definir los parámetros para el mapeo.

Determinación de los parámetros para el Mapeo

Los datos para los vectores de entrada se obtuvieron del motor que sirvió como modelo y los datos para los vectores de salida se obtuvieron del motor que aprende por imitación (por medición directa en el osciloscopio). En ambos casos, se adquirieron datos de las señales provenientes del codificador acoplado al motor y de las señales que controlan al motor. Los datos que se adquirieron del codificador fueron: *período (T)* y *ancho de pulso (AP)* de las señales en cuadratura. Los datos obtenidos de las señales de control son el *período (T)* y el *ancho de pulso (AP)* de la señal *PWM de magnitud* y *el signo*. Debido a la diferencia en el rango de velocidades que manejan los motores, los vectores de entrada para el mapeo se obtuvieron del MM (Motor Modelo) y los vectores de salida se obtuvieron del MA (Motor Aprendiz). De los datos obtenidos del MM, se puede observar que los valores obtenidos para el *período* varían conforme la velocidad del motor cambia, en tanto que el ciclo de trabajo se mantiene constante (aproximadamente en el 40%). Por lo que se escogió el *período* como dato relevante. El *signo* proporciona información sobre el sentido de giro del eje del motor, y se definió para tres estados: sentido negativo, punto de paro y sentido positivo. La Tabla 3.2 muestra una extracción de los datos de *período (T)*, *ancho de pulso (AP)* y *signo(SIGNO)*, del MM. Únicamente se incluyen los datos correspondientes a una de las señales del codificador óptico, ya que por tratarse de señales en cuadratura, ambas tienen características iguales en un mismo instante de tiempo. Los datos utilizados para el entrenamiento se muestran en las columnas sombreadas. Los datos de control relevantes para el MA son las señales *PWM de Magnitud* y *signo*. Como la señal *PWM de magnitud* experimenta una variación en el ciclo de trabajo, manteniéndose constante el *período* con respecto a la variación de la velocidad del motor, se tomó como parámetro relevante para el entrenamiento de la red neuronal el *ancho de pulso (AP)* de la señal. La señal *de signo* es un nivel alto para el movimiento en dirección positiva (definido como 1), o un nivel bajo para el movimiento en dirección negativa (definido como -1).

Tabla 3.2. Datos para el control obtenidos del MM y del MA. Los datos para el MM dentro de las columnas sombreadas con verde, corresponden al vector de entrada para el entrenamiento de la red neuronal. Los datos correspondientes al MA dentro de las columnas sombreadas con gris, corresponden a los datos del vector de salida para el entrenamiento de la red neuronal.

DATOS PARA EL MAPEO						
MM (Codificador Óptico)			Definido	MA (PWM)		
rps	T(μs)	AP (μs)	Signo	T(μs)	AP (μs)	Signo
Velocidad en sentido negativo						
0.5	6200	2480	-1	85.4	2.5	0
0.6	5575	2230	-1	85.4	3	0
0.7	4956	1982	-1	85.4	3.5	0
0.8	4339	1735	-1	85.4	4	0
0.9	3716	1486	-1	85.4	4.5	0
1	3100	1240	-1	85.4	5	0
2	1580	680	-1	85.4	6	0
3	1144	464	-1	85.4	7	0
4	820	343.8	-1	85.4	7.7	0

5	666	262.8	-1	85.4	8.7	0
6	542	227.2	-1	85.4	9.5	0
7	484	200	-1	85.4	10.7	0
8	410	170.2	-1	85.4	11.3	0
9	374	151.8	-1	85.4	12.2	0
10	332	134.4	-1	85.4	13.3	0
11	301	124	-1	85.4	14	0
12	279	113.6	-1	85.4	14.6	0
13	257	104.8	-1	85.4	16	0
14	240	98	-1	85.4	16.6	0
15	222	89.8	-1	85.4	18	0
16	210	83.28	-1	85.4	18.6	0
17	198	80	-1	85.4	19.4	0
18	186	75	-1	85.4	20	0
19	176	71.12	-1	85.4	21.4	0
20	168	68	-1	85.4	22.2	0
21	160	65.4	-1	85.4	23	0
22	152	60	-1	85.4	24	0
23	146	58.4	-1	85.4	24.8	0
24	140	56	-1	85.4	25.4	0
25	134	53.6	-1	85.4	26.8	0
26	129	51.6	-1	85.4	27.4	0
27	124	49.6	-1	85.4	28.4	0
Punto de paro						
0	0	0	0	0	0	0
Velocidad en sentido positivo						
0.5	6200	2480	1	85.4	2.5	1
0.6	5575	2230	1	85.4	3	1
0.7	4956	1982	1	85.4	3.5	1
0.8	4339	1735	1	85.4	4	1
0.9	3716	1486	1	85.4	4.5	1
1	3100	1408	1	85.4	5	1
2	1580	680	1	85.4	6	1
3	1144	464	1	85.4	7	1
4	820	343.8	1	85.4	7.7	1
5	666	262.8	1	85.4	8.7	1
6	542	227.2	1	85.4	9.5	1
7	484	200	1	85.4	10.7	1
8	410	170.2	1	85.4	11.3	1
9	374	151.8	1	85.4	12.2	1
10	332	134.4	1	85.4	13.3	1
11	301	124	1	85.4	14	1
12	279	113.6	1	85.4	14.6	1
13	257	104.8	1	85.4	16	1
14	240	98	1	85.4	16.6	1
15	222	89.8	1	85.4	18	1
16	210	83.28	1	85.4	18.6	1
17	198	80	1	85.4	19.4	1
18	186	75	1	85.4	20	1
19	176	71.12	1	85.4	21.4	1
20	168	68	1	85.4	22.2	1
21	160	65.4	1	85.4	23	1
22	152	60	1	85.4	24	1
23	146	58.4	1	85.4	24.8	1
24	140	56	1	85.4	25.4	1
25	134	53.6	1	85.4	26.8	1
26	129	51.6	1	85.4	27.4	1
27	124	49.6	1	85.4	28.4	1

El resto de los datos para el MA se muestra en el Apéndice II. La figura 3.11, muestra en (a) cómo varía el periodo de las señales provenientes del codificador óptico acoplado al MM con respecto a la velocidad del motor dada en revoluciones por segundo (rps). En (b) muestra cómo varía el ciclo de trabajo de la señal PWM de magnitud con respecto a la velocidad del motor para

el MA. Una vez que se han definido los parámetros para el mapeo, es necesario determinar la arquitectura de la red neuronal que realizará el *mapeo*.

Características de la red neuronal

Como es el caso, cuando la información sobre el comportamiento del sistema está en la forma de pares de datos de entrada y salida, será necesario encontrar una función que se ajuste a los datos y que además cumpla con un criterio de desempeño para que una entrada dada conduzca a una salida específica. En teoría la función que realiza el mapeo entre el acto motor percibido y el realizado por el motor, puede tomar muchas formas y el criterio de desempeño debe ser tal que los resultados obtenidos sean muy aproximados a los esperados. En tal caso la red neuronal debe comportarse como una función que realiza el mapeo entre los vectores de los datos de entrada a los vectores de salida, de manera que el desempeño de la red sea muy bueno para obtener valores muy próximos a los esperados, es decir, la red se comporta como una función de aproximación. En este caso podemos escoger una red feedforward, ya que una red de este tipo, adecuadamente entrenada puede cumplir con los requerimientos del sistema que se plantea, pues se sabe que puede aproximar cualquier función con un error relativamente pequeño, siempre y cuando la red tenga al menos una capa oculta (teorema de Funahashi [22]), por esta razón se escogió red neuronal con tres capas (entrada, oculta y salida).

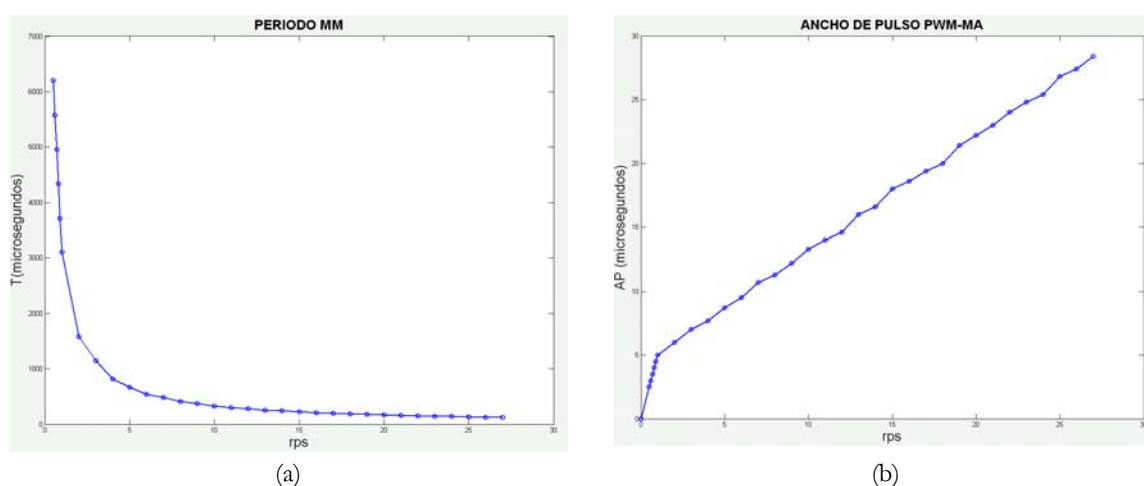


Figura 3.11. En (a) se muestra la gráfica de rps vs. periodo de la señal que proporciona la pista de movimiento del MM. En (b) se muestra la gráfica de rps vs. ancho de pulso de la señal PWM que controla al MA.

El número de neuronas en la red, se establece de acuerdo a las necesidades de diseño. Para el mapeo son necesarias dos entradas y dos salidas, entonces el número de neuronas en la entrada es igual al número de entradas y el número de neuronas en la capa de salida es igual al número

de salidas (2 en cada caso). Para el número de neuronas en la capa oculta, se estableció un criterio heurístico que establece que la capa intermedia debe tener al menos $2m$ neuronas, siendo m el número de neuronas en la capa de entrada [40]. Otros parámetros que influyen en el desempeño de la red son los pesos de las conexiones entre neuronas, la tasa de aprendizaje y el momento, los cuales se propusieron sobre la base de las experiencias expuestas por diversos autores de trabajos con redes neuronales feedforward. La tasa de aprendizaje se escogió en un intervalo de (0.1, 0.5), el momento se escogió en el intervalo (0,1). Estos criterios, junto con el algoritmo de entrenamiento de la red y los parámetros como la tasa de aprendizaje y momento, permitieron encontrar la mejor configuración de red para la aplicación.

En una etapa inicial, se realizó una simulación. Para ello se utilizó el programa Matlab 7.0 para implementar una red neuronal Back-propagation. El código para el entrenamiento y simulación se encuentra en el Apéndice I, sección I.3. La Figura 3.12 muestra la configuración de la red neuronal implementada y el origen de los patrones de entrenamiento.

La programación de la arquitectura de la red, el algoritmo de entrenamiento y el funcionamiento en línea del mecanismo de *Mapeo* para el sistema de aprendizaje por imitación se realizó a través del lenguaje de programación C. El código se muestra en el Apéndice I, sección I.

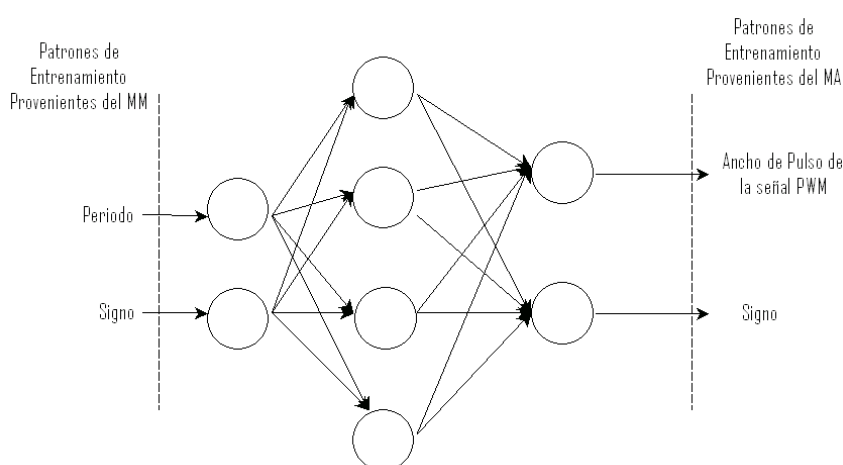


Figura 3.12. Configuración de la red neuronal que realiza el mapeo en el aprendizaje por imitación y el control del motor.

El acto motor

El *acto motor*, genera las señales para la ejecución de las acciones percibidas una vez que se ha realizado el *mapeo*. Esto corresponde a construir las señales de *Magnitud* y *Signo* a partir de los datos proporcionados por la red neuronal en el *mapeo*. Así, las señales construidas controlan la cantidad de corriente que debe fluir por el motor para que éste realice el mismo movimiento que el percibido.

Generación de la señal PWM de Magnitud

En la generación de la señal *PWM de Magnitud* uno de los factores importantes es el manejo del tiempo, ya que esto nos permite construir las señales de control con la exactitud adecuada para poder llevar a cabo el *acto motor* tal y como lo haría el MM. Para esto es necesario realizar una temporización de precisión para llevar a cabo el control del MA. Para generar la temporización necesaria se utiliza el CI temporizador-contador 8254. El 8254 cuenta con tres temporizadores-contadores²¹, que están conectados a un reloj que oscila a una frecuencia de 1.193180 MHz. En una PC, solamente el contador 2 está disponible para uso general, puede emplearse para temporizar y puede ser programado por el usuario. Para realizar temporizaciones el temporizador es cargado primero con una palabra de control y después con la cuenta que ha de efectuar. La cuenta es la unidad mínima de tiempo con la cual se podrá temporizar un evento. Esta unidad mínima de tiempo está en función de la frecuencia de reloj que alimenta al temporizador.

Como se menciona arriba el reloj que alimenta al temporizador tienen una frecuencia de 1.193180 Mz, por lo que podrá realizar temporizaciones en unidades de 1/1193180 segundos, de esta manera una cuenta equivale a 0.8380965152 μ seg. Así, para saber cuantas cuentas se deben cargar en el temporizador para realizar una temporización de X segundos, tendremos que:

$$Cuentas = \frac{Xseg.}{0.8380965152\mu seg.}$$

Por ejemplo, para realizar una temporización de 85.4 μ seg el número de cuentas que se cargará al temporizador-contador será de:

$$Cuentas = \frac{85.4\mu s}{0.8380965152\mu s.} = 101.897572 \approx 102$$

En teoría el número máximo de cuentas que se puede cargar al temporizador es de 56536 (2^{16}). La generación de la señal PWM de Magnitud comienza cuando el mecanismo de mapeo proporciona el dato numérico correspondiente al *ancho de pulso* de la señal *PWM de Magnitud* y el mecanismo que realiza el *acto motor* lo lee. El ancho de pulso es un valor numérico de tipo real entre 0 y 1 el cual se debe escalar multiplicando por cien para obtener su valor real en microsegundos. Por ejemplo, si un valor proporcionado por el mapeo es de 0.089, el valor real será de 8.9 μ seg. Lo que sigue es calcular el número de cuentas que se debe cargar al temporizador. Siguiendo con el ejemplo, el número de cuentas para 8.9 μ seg es:

²¹ Temporizador-contador 0, temporizador-contador 1 y temporizador-contador 2.

$$Cuentas = \frac{8.9\mu s}{0.8380965152\mu seg.} = 10.619301$$

Entonces el temporizador se puede cargar con una cuenta de 10 o de 11, esto es, redondeando la cifra a un número entero mayor o menor. Una vez realizado esto, se carga el temporizador con el valor de *cuenta-1*. Después, el temporizador se activa para iniciar la temporización y se envía un “1” a través del puerto paralelo para iniciar el pulso de la señal *PWM de magnitud*.

Para verificar cuándo termina la cuenta, se realiza un examen constante de la línea OUT del temporizador a través del bit 5 del puerto 61h, OUT baja cuando se carga la cuenta y sube cuando termina²², figura 3.13. En el Modo de Operación 0, la cuenta se carga en un ciclo de reloj y comienza a decrecer hasta el siguiente ciclo de reloj. Mientras OUT mantiene un nivel bajo, se continúa escribiendo “1” al puerto paralelo, para simular la continuidad de la señal. Cuando OUT sube, se carga el temporizador con el valor de cuenta correspondiente al resto del período de la señal *PWM de magnitud*. Para esto se realiza la siguiente operación:

$$Cuentas = \frac{(Periodo_{PWM} - Ancho_de_Pulso_{PWM})seg.}{0.8380965152\mu seg.}$$

Entonces, si $Periodo_{PWM} = 85.4\mu seg$ y $Ancho_de_Pulso_{PWM} = 8.9\mu seg$, tendremos que:

$$Cuentas = \frac{85.4\mu seg - 8.9\mu seg}{0.8380965152\mu seg} = 91.278269 \approx 91$$

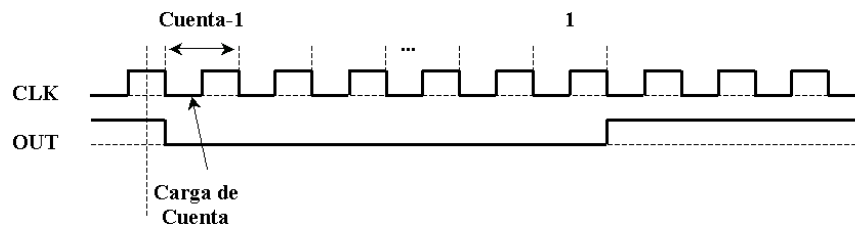


Figura 3.13. Modo de operación del temporizador utilizado en el acto motor.

Así, se carga el temporizador con el valor de *cuenta-1*. Después, el temporizador se activa para iniciar la temporización y se envía un “0” a través del puerto paralelo para continuar con el periodo de la señal PWM de magnitud. De nuevo, para verificar cuándo termina la cuenta, se realiza un examen constante de la línea OUT del temporizador a través del bit 5 del puerto 61h.

²² Termina cuando la cuenta llega a “1”.

Mientras OUT mantiene un nivel bajo, se continúa escribiendo “0” al puerto paralelo, para simular la continuidad de la señal. En esta ocasión, cuando OUT sube, el mecanismo del acto motor regresa a verificar si se inicia el procedimiento de generación de la señal PWM de Magnitud. El diagrama de bloques de la figura 3.14 esquematiza el procedimiento de construcción de la señal PWM de Magnitud.

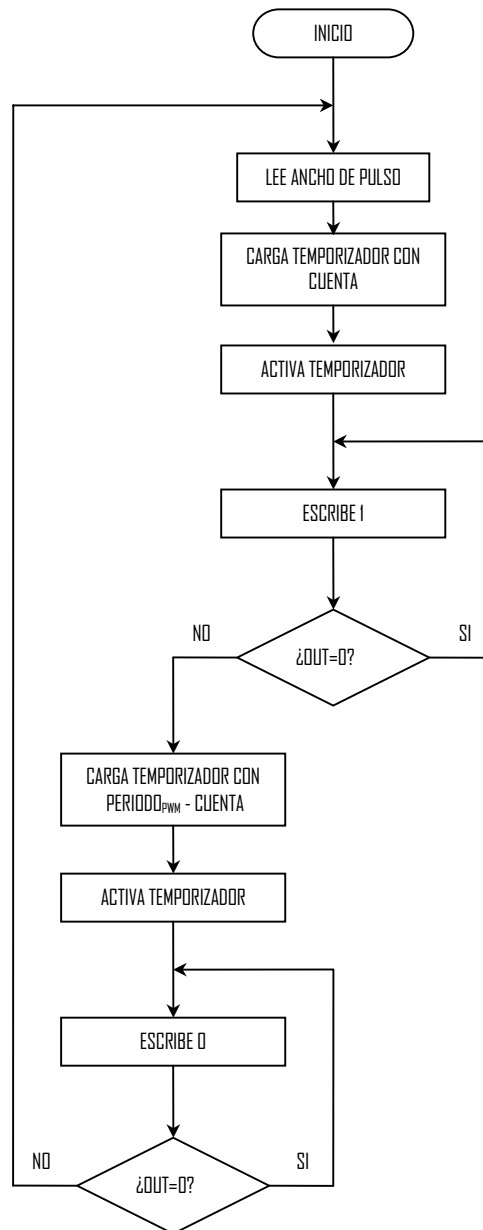


Figura 3.14. Generación de la señal PWM de magnitud.

La señal PWM de signo

En la generación de la señal de *Signo*, se toma el valor aportado por el mapeo, se escala y se envía a través del puerto paralelo como un bit 0 ó 1 que se puede ver como una señal que tiene un nivel alto o un nivel bajo. Como se recordará, cuando la señal de *Signo* tiene un nivel TTL alto (un “1”) indica que el eje del motor deberá moverse en el sentido positivo; y cuando tiene un nivel bajo (un “0”) indica que el eje del motor deberá moverse en sentido negativo.

Al igual que en la generación de la señal PWM de Magnitud, para la señal de Signo, inicialmente se tomaron datos de esta señal en un sistema de control tradicional, y se pudo observar que para todo rango de velocidades que mantiene el eje del MA, se mantiene constante siempre y cuando el sentido de giro de MA no cambie. Siendo característico que cuando el eje del motor gira en el sentido positivo esta señal mantiene un nivel alto (“1”), y cuando lo hace en el sentido contrario mantiene nivel bajo (“0”). La figura 3.15 muestra el diagrama de flujo para la generación de la señal PWM de signo. La función `pwm()` del programa `BPNN.C` muestra el código para generar la señal PWM de Magnitud y el Signo.

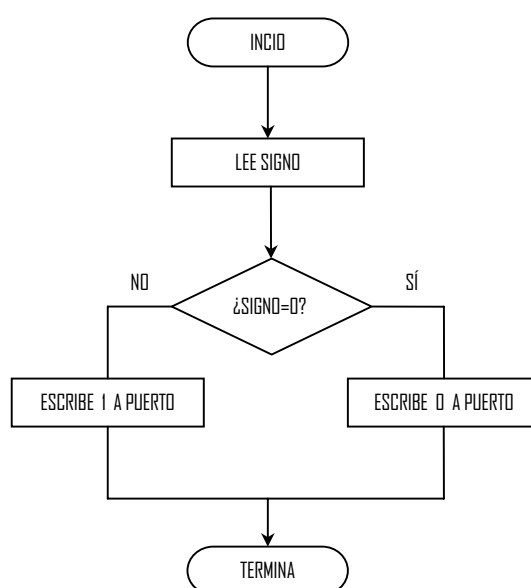


Figura 3.15. Generación de la señal de Signo.

3.4. La interfaz de comunicación

El SPP tiene 8 líneas de datos definidas en el registro de datos (físicamente en los pines 2 a 9 del conector DB-25), 4 líneas de control definidas en el registro de control (físicamente en los pines 1, 14, 16 y 17) y 5 líneas de estatus definidas en el registro de estatus (físicamente en los pines 10,

11, 12, 13 y 15, ver Tabla V.2 en el Apéndice V), todas unidireccionales. Las líneas de datos y de control son líneas de salida y las líneas de estado son de entrada. Esta configuración en el SPP permitió construir una interfaz de comunicación de entrada-salida; así, se utilizaron las líneas de datos para el envío de datos y las líneas de estados para la recepción de datos. Para el envío de datos se utilizaron 2 de las 8 líneas del registro de datos (pines 2 y 9), que corresponden a las señales PWM de Magnitud y de Signo respectivamente. Para la recepción de datos se utilizaron 2 de las cinco líneas del registro de estatus (pines 10 y 15), que corresponden a la recepción de las señales en cuadratura, canales A y B, del codificador óptico acoplado al MM. De modo que la configuración para la interfaz de comunicación de MA quedó como se muestra en la figura 3.16.

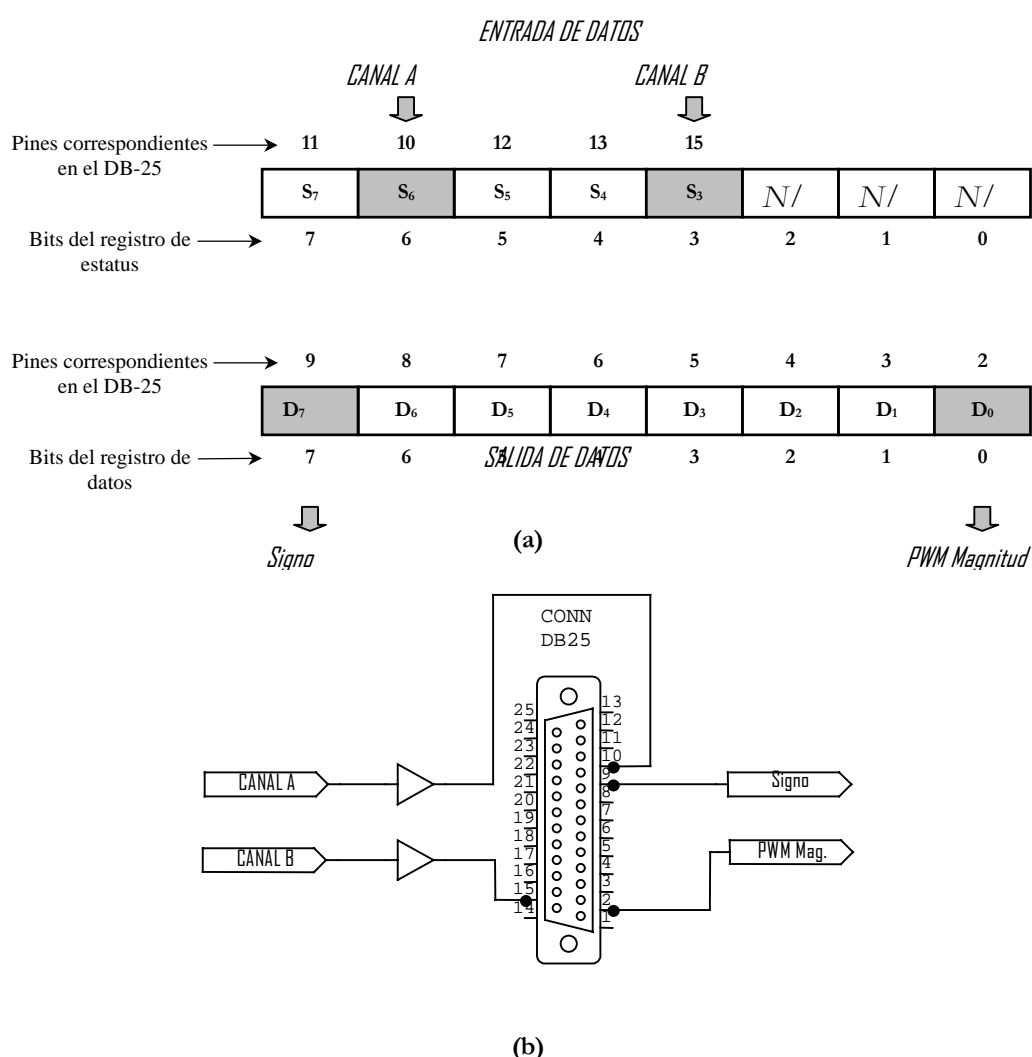


Figura 3.16. Interfaz de comunicación del MA. (a) Configuración de los registros de estatus para la entrada de datos, y de datos para la salida de datos. (b) Diagrama de conexión.

3.5. Resumen

El esquema básico de aprendizaje por imitación está constituido por tres partes importantes: percepción, aprendizaje y acto motor. La *percepción* permite determinar los parámetros característicos del movimiento que se percibe (velocidad y sentido). El *aprendizaje* realiza el mapeo entre los parámetros que caracterizan al movimiento percibido y los parámetros que el acto motor utilizará para generar las señales de control, y el *acto motor* genera las señales para el control.

Para determinar la velocidad y sentido del movimiento percibido, el sistema decodifica las señales provenientes de un codificador.

Para el aprendizaje, el sistema utiliza una red neuronal, sobre todo porque la información sobre el comportamiento del motor está primordialmente en la forma de datos de entrada-salida, y es necesario establecer movimiento en sentido positivo y negativo, y puntos de paro (velocidad cero). En tal caso la red neuronal debe comportarse como una función que realice el mapeo entre los datos del movimiento percibido y los datos que permitan generar las señales de control con un buen grado de exactitud, para que los resultados sean muy próximos a los esperados; por lo que, en este caso, la red neuronal se debe comportar como un aproximador de funciones. En este sentido, se escogió una red neuronal backpropagation con una capa oculta, ya que cumple con los requerimientos de desempeño.

El número de neuronas en la capa oculta, se escogió de acuerdo a un criterio heurístico que propone que la capa oculta debe tener al menos dos veces el número de neuronas en la capa de entrada. El acto motor aplica un método de modulación PWM signo-magnitud para generar las señales de control.

El capítulo siguiente presenta el método utilizado para el entrenamiento de la red neuronal, las pruebas y ajustes que se hicieron en los parámetros de la red neuronal y los resultados obtenidos a partir de las pruebas hechas al sistema. También se hace un análisis de complejidad del algoritmo back propagation.

Capítulo 4

Pruebas y resultados

Para realizar las pruebas al sistema, se utilizó como modelo un motor de corriente directa cuyo sistema de control está basado en un microcontrolador LM 629. Las pruebas se realizaron en dos etapas, la primera se realizó con perfiles de velocidad fija y se hicieron algunos ajustes a los parámetros de la red neuronal. La segunda etapa contempló el desempeño de perfiles de velocidades variables.

4.1. El control de motores con redes neuronales

Las primeras dificultades que se presentan cuando se diseña un sistema de control para motores de CD, son debidas a los efectos de las no linealidades del motor, que se dan principalmente por la saturación y la fricción, y que pueden degradar el desempeño del controlador. Adicionalmente debido a la dificultad que representa obtener un modelo no lineal exacto del motor, y cuando la información sobre su comportamiento está sólo en la forma de pares de datos de entrada y salida, es posible utilizar alguna técnica de control inteligente, que no requiera de un modelo exacto. Existen varias técnicas de control inteligente, entre las cuales se encuentra el control con redes neuronales o neurocontrol. Según Werbos [64], existen principalmente cinco criterios básicos de diseño en neurocontrol:

- Control supervisado, en donde las redes neuronales son entrenadas sobre una base de datos que contiene un conjunto de entradas $X(t)$ y las acciones correctas $u(t)$, usando el aprendizaje supervisado para que la red aprenda el mapeo de X a u .
- Control inverso directo, la red neuronal es entrenada para desarrollar un modelo inverso del sistema. La entrada de la red es el proceso de salida y la salida de la red es el correspondiente proceso de entrada. Entonces, cuando se aplica una entrada determinada, la red será capaz de calcular la acción de control correspondiente.

- Control neuronal adaptivo, donde las redes neuronales son usadas para realizar mapeos no lineales.
- Backpropagation de utilidad, el cual maximiza algunas medidas de utilidad o desempeño sobre el tiempo, pero no es eficiente para el ruido y no proporciona aprendizaje en tiempo real para grandes problemas.
- Métodos críticos adaptivos, los cuales pueden ser definidos como métodos de aproximación a la programación dinámica.

Aunque las aplicaciones son muy variadas, los métodos de control que utilizan redes neuronales en sus diseños se pueden clasificar principalmente en dos tipos: control neuronal directo y control neuronal indirecto.

En el control neuronal directo, la red neuronal actúa como controlador, el cual genera las señales de control para enviarlas directamente al sistema. En el control neuronal indirecto la red no envía las señales de control directamente al sistema, sino que sólo indica algunas características relevantes del sistema. Por otra parte el método más simple para obtener un controlador basado en la adaptación de parámetros a partir de datos de entrada y salida es por modelo inverso. El modelo inverso es básicamente la estructura de la red representando la inversa del sistema dinámico en la región de entrenamiento o identificación. Para obtener el modelo inverso la red neuronal se entrena con los patrones de entrada $u(t)$ y salida $y(t)$ del sistema, junto con las entradas y salidas pasadas para predecir la salida actual $\hat{u}(t)$. La entrada a la red neuronal son los valores de la salida del sistema; y la salida son los valores de entrada del sistema. La red entrenada de esta manera es el modelo inverso. En la figura 4.1 se muestra esquemáticamente la fase de entrenamiento para obtener el modelo inverso de un motor.

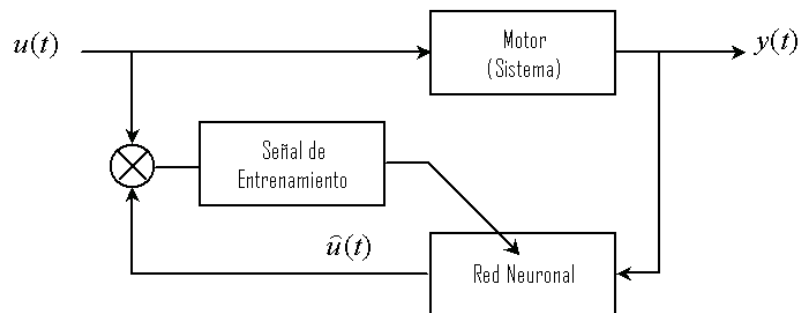


Figura 4.1. Fase de entrenamiento de la red por modelo inverso.

Una vez que se ha entrenado a la red bajo el esquema del modelo inverso, se implementa el controlador sustituyendo a $y(t)$ por $y^d(t)$, el controlador resultante queda como se muestra en la figura 4.2. Como se podrá observar el controlador es el modelo neuronal inverso o red neuronal inversa RNI.

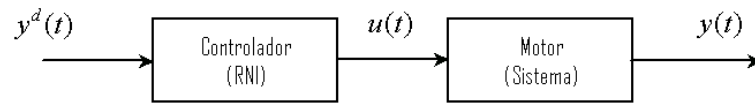


Figura 4.2. Control por modelo inverso.

En la tesis se aplica un esquema de control basado en un *modelo inverso modificado*, se llama así porque los patrones de entrenamiento $y(t)$ provienen del motor que se utiliza como modelo, que utiliza el mismo tipo de parámetros de control que el motor aprendiz, pero de distinta dimensión.

4.2. Complejidad computacional.

La complejidad computacional indica el esfuerzo que se requiere aplicar a un algoritmo, o qué tan costoso resulta en cuanto a tiempo o espacio. Para evaluar la eficiencia de un algoritmo se pueden utilizar unidades lógicas que expresan una relación entre el tamaño de un archivo y la cantidad de tiempo requerida para procesar los datos.

Un método muy recurrido para evaluar la eficiencia de un algoritmo es la *complejidad asintótica*, en este método, se utilizan sólo ciertos términos de una expresión para expresar la eficiencia del algoritmo. La complejidad asintótica mide el número de asignaciones y/o el número de comparaciones realizadas durante la ejecución de un programa, para evaluar la complejidad temporal de un algoritmo. En la tesis se evalúa la complejidad del programa desarrollado para el sistema, por medio de la complejidad asintótica, utilizando como criterio el número de asignaciones.

4.3. El control con aprendizaje por imitación

El aprendizaje de habilidades motoras por medio de la imitación tiene tres elementos básicos: la percepción del movimiento, aprendizaje (procesamiento interno de la información percibida), y la consecuente ejecución del movimiento percibido. En el sistema propuesto, mostrado en la figura 4.3, la percepción del movimiento se da a través del bloque de *percepción*. El procesamiento interno de la información percibida, se hace por medio del bloque de *aprendizaje* con una red neuronal inversa (RNI). Y la ejecución del movimiento se lleva a cabo por el bloque llamado *acto motor*. En este esquema de aprendizaje es importante la presencia de un modelo que muestre el movimiento a imitar, en este caso el modelo es un motor identificado como Motor Modelo (MM). Por otra parte, el motor que aprende el movimiento a través de la imitación, el Motor Aprendiz (MA), será capaz de reproducir los movimientos percibidos del modelo. En la figura 4.3, se muestra el sistema de control propuesto. Como se puede observar, el sistema percibe el

movimiento por medio del bloque de *Percepción*, el cual decodifica las señales A y B para obtener información sobre la velocidad y sentido del movimiento percibido. En este bloque se realiza el muestreo de una de las señales para determinar el periodo, el cual toma el valor numérico T_C . También se realiza una *Decodificación de Signo*, para determinar el sentido del movimiento percibido, esta variable puede tomar únicamente dos valores: -1 ó 1 y se identifica como $SIGNO_C$. En el bloque de *Aprendizaje* la red neuronal realiza el mapeo de los datos proporcionados por el bloque de *Percepción* (T_C y $SIGNO_C$), y los datos que servirán para generar las señales PWM de magnitud (AP_{PWM}) y de signo ($SIGNO$) para realizar el movimiento del motor.

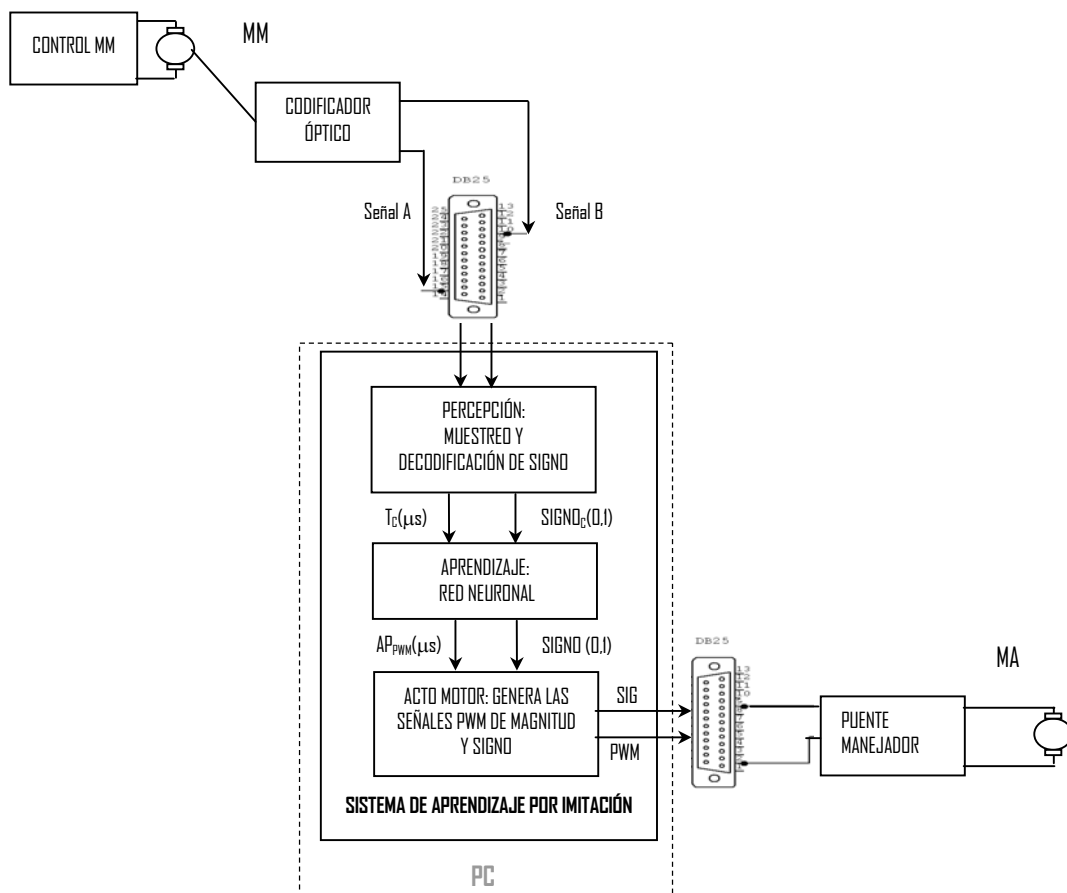


Figura 4.3. Diagrama del sistema de control con aprendizaje por imitación.

El bloque del *Acto Motor* genera las señales de control para el motor a partir de los datos AP_{PWM} y $SIGNO$. La señal PWM de Magnitud varía su ciclo de trabajo según se necesite variar la velocidad a la que se mueve el motor. La señal de $SIGNO$ es una señal con un nivel alto o bajo, según el sentido de giro del eje del motor. La comunicación para la entrada y salida de datos para el sistema de control del motor, se realiza por medio del puerto paralelo estándar. Se utilizan dos

líneas para la entrada de datos, hacia el sistema de control del motor, y dos líneas de salida de datos, del sistema de control hacia el Puente Manejador. Una vez que se ha establecido como está constituido el sistema propuesto, el siguiente paso es ajustar los parámetros propios de la red neuronal (*tasa de aprendizaje y el momento*).

4.3.1. Ajuste de parámetros y entrenamiento de la red neuronal

El ajuste de parámetros se hizo en la etapa de entrenamiento de la red neuronal. El entrenamiento se realizó varias veces, modificando cada vez los parámetros de la *tasa de aprendizaje y el momento*, hasta que la red mostró el comportamiento esperado. Los patrones de entrenamiento de entrada corresponden a los de *periodo* T_C y *Signo* para velocidades fijas dadas en revoluciones por segundo (rps). Los patrones de entrenamiento de salida corresponden a los valores de *ancho de pulso PWM* (AP_{PWM}) y *Signo*, que son los objetivos de respuesta que la red deberá alcanzar cuando se le presenta una entrada específica (ver tabla 4.1).

Tabla 4.1. Valores de los datos correspondientes a los vectores de entrada y salida para el entrenamiento de la red neuronal. Los datos dentro de las columnas sombreadas con azul, corresponden a los vectores de entrada. Los datos de las columnas sombreadas en color gris corresponden a los vectores de salida.

PARÁMETROS PARA EL ENTRENAMIENTO DE LA RED NEURONAL										
Movimiento en sentido positivo						Movimiento en sentido negativo				
rps	Vector No.	Vectores de Entrada (MM)		Vectores de Salida (MA)		Vector No.	Vectores de Entrada (MM)		Vectores de Salida (MA)	
		$T_C(\mu s)$	Valor Signo	$AP_{PWM}(\mu s)$	Valor Signo		$T_{PWM}(\mu s)$	Valor Signo	$AP_{PWM}(\mu s)$	Valor Signo
0	1	0	0	0	0					
0.5	2	6200	1	2.5	1	34	6200	-1	2.5	0
0.6	3	5575	1	3	1	35	5575	-1	3	0
0.7	4	4956	1	3.5	1	36	4956	-1	3.5	0
0.8	5	4339	1	4	1	37	4339	-1	4	0
0.9	6	3716	1	4.5	1	38	3716	-1	4.5	0
1	7	3100	1	5	1	39	3100	-1	5	0
2	8	1580	1	6	1	40	1580	-1	6	0
3	9	1144	1	7	1	41	1144	-1	7	0
4	10	820	1	7.7	1	42	820	-1	7.7	0
5	11	666	1	8.7	1	43	666	-1	8.7	0
6	12	542	1	9.5	1	44	542	-1	9.5	0
7	13	484	1	10.7	1	45	484	-1	10.7	0
8	14	410	1	11.3	1	46	410	-1	11.3	0
9	15	374	1	12.2	1	47	374	-1	12.2	0
10	16	332	1	13.3	1	48	332	-1	13.3	0
11	17	301	1	14	1	49	301	-1	14	0
12	18	279	1	14.6	1	50	279	-1	14.6	0
13	19	257	1	16	1	51	257	-1	16	0
14	20	240	1	16.6	1	52	240	-1	16.6	0
15	21	222	1	18	1	53	222	-1	18	0
16	22	210	1	18.6	1	54	210	-1	18.6	0
17	23	198	1	19.4	1	55	198	-1	19.4	0
18	24	186	1	20	1	56	186	-1	20	0
19	25	176	1	21.4	1	57	176	-1	21.4	0
20	26	168	1	22.2	1	58	168	-1	22.2	0
21	27	160	1	23	1	59	160	-1	23	0
22	28	152	1	24	1	60	152	-1	24	0
23	29	146	1	24.8	1	61	146	-1	24.8	0
24	30	140	1	25.4	1	62	140	-1	25.4	0
25	31	134	1	26.8	1	63	134	-1	26.8	0

26	32	129	1	27.4	1	64	129	-1	27.4	0
27	33	124	1	28.4	1	65	124	-1	28.4	0

Se utilizaron 65 vectores de entrada, uno correspondiente al estado de paro del motor, 32 correspondientes a cada una de las velocidades fijas con movimiento en el sentido positivo (dextrógiro), y 32 que corresponden al mismo rango de velocidades pero en sentido negativo (levógiro); y 65 vectores de salida, uno correspondiente al estado de paro del motor; 32 proporcionan información para generar las señales de control para un movimiento en sentido positivo, y 32 para generar las señales de control para un movimiento en sentido negativo. Una vez que se estableció un conjunto óptimo de parámetros (cuando las salidas de la red fueron más próximos a los objetivos), se concluyó con el entrenamiento de la red. En este punto el sistema es capaz de hacer el *Mapeo* entre los datos que dan una pista del movimiento que se debe ejecutar (T_C y *Signo*); y los datos que proporcionan información para generar los señales de control para el motor (AP_{PWM} y *Signo*). Una vez que el sistema ha aprendido los parámetros para reproducir un movimiento, se puede decir que está capacitado para ejecutar cualquier perfil de movimiento.

El siguiente paso fue realizar varias pruebas con perfiles de velocidad fija y perfiles de movimiento que tienen velocidades y aceleraciones variables, puntos de paro e inversión del sentido movimiento, como el que se muestra en la figura 4.4, para verificar que el bloque del *Acto Motor* generara correctamente las señales de control (*SIG* y *PWM*), para el movimiento del motor y verificar que el sistema funcionara adecuadamente.

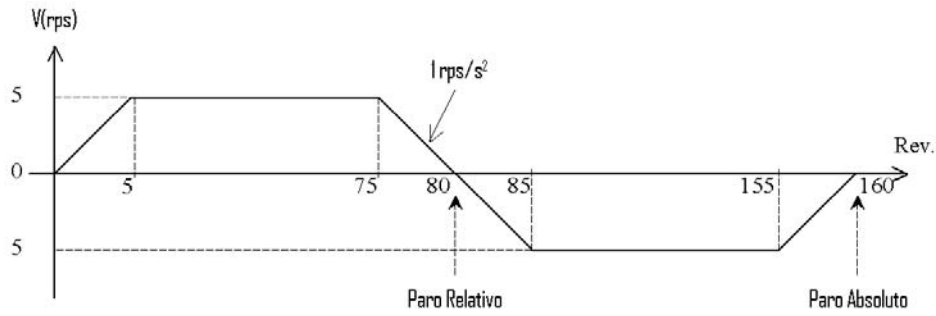


Figura 4.4. Perfil de movimiento para probar el sistema de control con Aprendizaje por Imitación.

4.3.2. Análisis de complejidad

El análisis de complejidad del algoritmo de control propuesto está basado principalmente en el análisis del algoritmo backpropagation. Para estimar la eficiencia de este algoritmo, se evaluará el número de instrucciones de asignación realizadas durante la ejecución del algoritmo

(complejidad asintótica). Se consideran como instrucciones de asignación a la inicialización y actualización de variables.

El algoritmo backpropagation consta de varios pasos: inicialización de los pesos, la propagación de un vector de entrada, el cálculo de la salida actual de la red, el cálculo de los términos de error para todas las neuronas, la actualización de los pesos y la comparación del error global. Para hacer un análisis inicial se escogió a la función forward () que propaga al vector de entradas para obtener la salida de la red. El código se muestra a continuación:

```

1. void forward() {
2.   /* Calcular los x2 (capa intermedia) */
3.   x2[0]=-1.0;
4.   for(j = 1 ;j<= N_OCULTA; j++){
5.     h = 0.0;
6.     for( k = 0 ;k<= N_ENTRADA;k++)
7.       h += w1[j][k] * x1[k];
8.     x2[j] = sigmoid(h);          /* Función de activación sigmoidea */
9.   }
10.  /* Calcular los x3 (capa de salida) */
11.  for(i = 1 ;i<= N_SALIDA;i++){
12.    h = 0.0;
13.    for(j = 0 ;j<= N_OCULTA;j++)
14.      h += w2[i][j] * x2[j];
15.    x3[i] = h ;                  /* Función de activación lineal */
16.  }
17.  if(CONTROL>3) {
18.    printf("Concluye forward\n");
19.    fflush(NULL);
20.  }
21. }
```

Considerando, por la estructura de la red, que el número de neuronas en la capa oculta (N_OCULTA), es del doble de neuronas en la capa de entrada (N_ENTRADA), podemos hacer:

$$N_ENTRADA = n$$

$$N_OCULTA = 2n$$

Iniciando el análisis en la línea 3, antes del ciclo for externo, se inicializa $x2[0]$. En el ciclo for externo (línea 4), antes de iniciar el ciclo, se inicializa j . Dentro del ciclo for externo se ejecutan $2n$ veces la actualización de j , la inicialización de h y k , y el ciclo for interno. En el ciclo for interno se actualizan k , h y se asignan valores a $x2[j]$, n veces. De esta manera, la estimación del número de operaciones en esta parte del algoritmo es el siguiente:

$$2 + 2n(3 + 3n) = 2 + 6n + 6n^2 = O(n) + O(n^2) \quad 4.1$$

Siguiendo en la línea 11, antes de iniciar el ciclo se inicializa **i**. Dentro del ciclo *for externo*, se ejecuta n veces la inicialización para **j** y **h**, se actualiza el valor de **i**, y se ejecuta el ciclo *for interno*. En el ciclo *for interno* se actualiza **j** y **h**, y se asignan valores a $x3[i]$, $2n$ veces. Entonces la estimación del número de operaciones es:

$$1 + n(3 + 3(2n)) = 1 + 3n + 6n^2 = O(n) + O(n^2) \quad 4.2$$

Sumando las expresiones de las ecuaciones 4.1 y 4.2, tendremos:

$$(2 + 6n + 6n^2) + (1 + 3n + 6n^2) = 3 + 9n + 12n^2 = cte + O(n) + O(n^2)$$

Entonces la complejidad se puede expresar como $O(n^2)$.

De igual forma, la inicialización de los pesos, el cálculo de la salida actual de la red, el cálculo de los términos de error para todas las neuronas, la actualización de los pesos y la comparación del error global, tienen a lo más dos ciclos anidados, por lo que la complejidad es del mismo orden. Cabe señalar que el tiempo de ejecución en el entrenamiento, puede variar dependiendo de aspectos como: el número de patrones de entrenamiento, la tasa de aprendizaje, el momento, y el error, entre otros. Pero, una vez que la red neuronal está entrenada adecuadamente, esta responde con bastante exactitud, a las necesidades de control.

4.4. Resultados

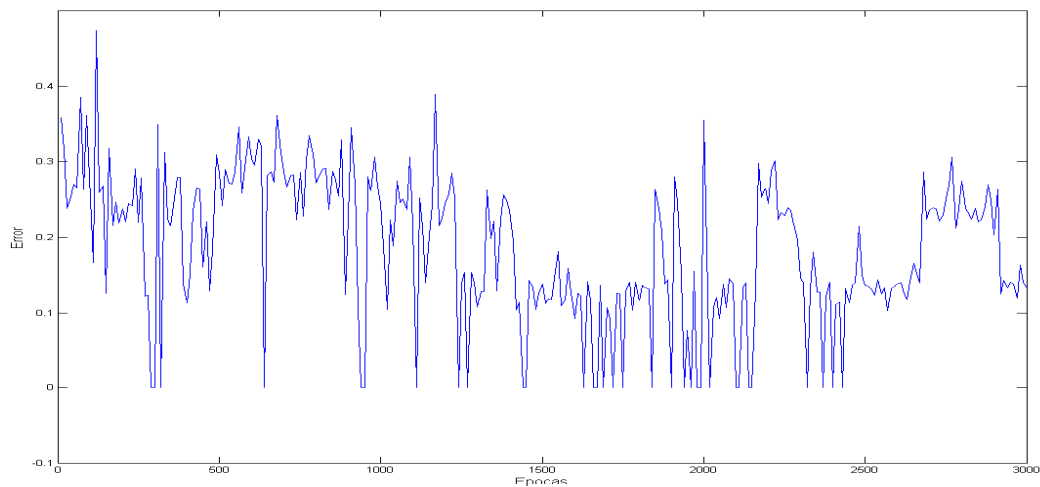
En la etapa de entraenamiento de la red neuronal y ajuste de los parámetros, se encontró que un conjunto de valores óptimos para los parámetros de *tasa de aprendizaje* y *momento*, con los cuales la red responde de manera más precisa son:

- Tasa de aprendizaje: 0.17
- Momento: 0.5

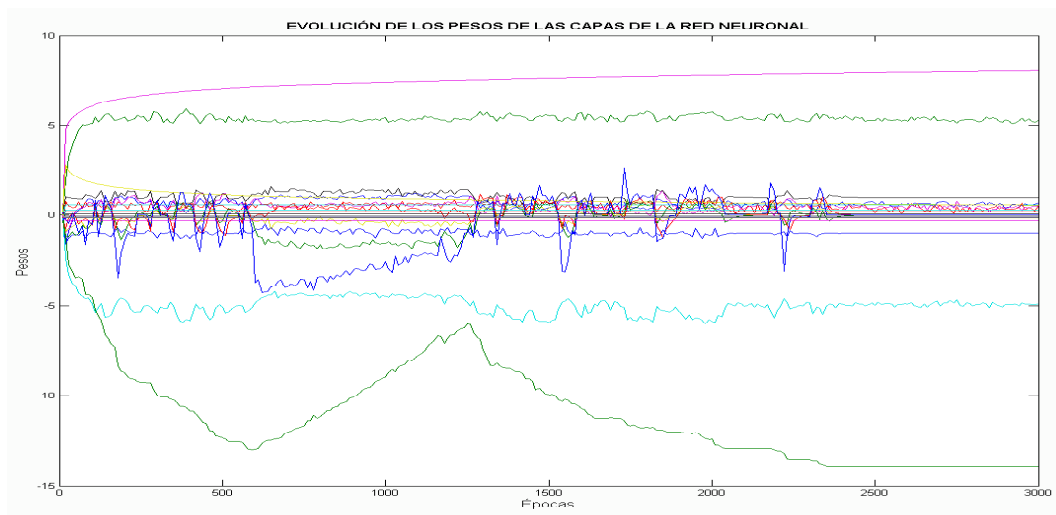
La figura 4.5 muestra la curva de desempeño del error y de la evolución de los pesos de la red neuronal durante el entrenamiento. En donde el error es la diferencia entre la salida deseada y la generada por la red neuronal. El error de entrenamiento fue de 1.0×10^{-7} , alcanzado en la época 1110. Ajustada la *red neuronal* de esta manera mostró el comportamiento esperado, cuando se le presentaron los datos para perfiles de movimiento a velocidades fijas y cuando se le presentó un perfil de movimiento variante.

La tabla 4.2 muestra los valores de los objetivos de salida para AP_{PWM} y *Signo* utilizados para el entrenamiento (mostrados en las columnas sombreadas con verde), los obtenidos durante las pruebas hechas al sistema y los valores que se obtienen cuando se utiliza un control PID (columnas en gris), para perfiles de velocidad fija, para el movimiento en sentido positivo y negativo. Como se puede observar, la mayoría de los valores de AP_{PWM} que son utilizados para el

entrenamiento y los que son obtenidos durante la etapa de pruebas tienen una diferencia máxima de $0.4 \mu s$. Por ejemplo, los datos en los renglones sombreados con color verde correspondientes a los valores de AP_{PWM} obtenidos para 10 y 20 rps, difieren en 0.1 y $0.2 \mu s$ respectivamente. En este caso, la diferencia no es crítica, pues a partir de estos datos se consigue generar con suficiente exactitud las señales de control para el motor. Por otra parte, los datos obtenidos para 1 y 27 rps, son exactamente iguales a los valores utilizados en el entrenamiento de la red.



(a)



(b)

Figura 4.5. (a) Desempeño del error y (b) evolución de los pesos de las capas de la red neuronal durante el entrenamiento.

La gráfica de la figura 4.6, muestra los valores de los objetivos para AP_{PWM} , utilizados en el entrenamiento en color rojo, mientras que los resultados obtenidos a partir de las pruebas se muestran en color azul, y los obtenidos con el control PID se muestran en cian. Los valores en la columna: *Parámetros obtenidos a partir de las pruebas*, son utilizados por el *Acto Motor* para generar las señales *PWM de Magnitud y Signo*.

Tabla 4.2. Tabla comparativa entre los valores de salida (objetivos) que se utilizaron para el entrenamiento de la red neuronal, los valores obtenidos a partir de las pruebas hechas al sistema, y los valores obtenidos cuando se emplea un control PID.

VALORES COMPARATIVOS														
Movimiento en sentido positivo								Movimiento en sentido negativo						
rps	Vector	Entrenamiento		Pruebas		PID		Vector	Entrenamiento		Pruebas		PID	
		AP_{PWM}	Signo	AP_{PWM}	Signo	AP_{PWM}	Signo		AP_{PWM}	Signo	AP_{PWM}	Signo	AP_{PWM}	Signo
0	1	0	0	0	0	0	0							
0.5	2	2.5	1	2.6	1	2.4	1	34	2.5	0	2.5	0	2.4	0
0.6	3	3	1	3	1	3.1	1	35	3	0	3.1	0	3.1	0
0.7	4	3.5	1	3.6	1	3.5	1	36	3.5	0	3.5	0	3.5	0
0.8	5	4	1	3.9	1	3.9	1	37	4	0	4.2	0	3.9	0
0.9	6	4.5	1	4.7	1	4.5	1	38	4.5	0	4.6	0	4.5	0
1	7	5	1	5	1	5	1	39	5	0	5.2	0	5	0
2	8	6	1	6	1	5.9	1	40	6	0	6	0	5.9	0
3	9	7	1	7.2	1	6.8	1	41	7	0	7.2	0	6.8	0
4	10	7.7	1	7.6	1	7.4	1	42	7.7	0	7.6	0	7.4	0
5	11	8.7	1	8.8	1	8.6	1	43	8.7	0	8.8	0	8.6	0
6	12	9.5	1	9.6	1	9.5	1	44	9.5	0	9.6	0	9.5	0
7	13	10.7	1	10.8	1	10.4	1	45	10.7	0	10.8	0	10.4	0
8	14	11.3	1	11.5	1	11.3	1	46	11.3	0	11.5	0	11.3	0
9	15	12.2	1	12.4	1	12	1	47	12.2	0	12.4	0	12	0
10	16	13.3	1	13.2	1	13.1	1	48	13.3	0	13.2	0	13.1	0
11	17	14	1	14.2	1	14	1	49	14	0	14.2	0	14	0
12	18	14.6	1	14.8	1	14.9	1	50	14.6	0	14.8	0	14.9	0
13	19	16	1	16	1	15.8	1	51	16	0	16	0	15.8	0
14	20	16.6	1	16.8	1	16.7	1	52	16.6	0	16.8	0	16.7	0
15	21	18	1	18	1	17.6	1	53	18	0	18	0	17.6	0
16	22	18.6	1	18.8	1	18.5	1	54	18.6	0	18.8	0	18.5	0
17	23	19.4	1	19.2	1	19.4	1	55	19.4	0	19.2	0	19.4	0
18	24	20	1	20	1	20.3	1	56	20	0	20	0	20.3	0
19	25	21.4	1	21.2	1	21.2	1	57	21.4	0	21.2	0	21.2	0
20	26	22.2	1	22.4	1	22.1	1	58	22.2	0	22.4	0	22.1	0
21	27	23	1	23.2	1	22.8	1	59	23	0	23.2	0	22.8	0
22	28	24	1	24.4	1	23.9	1	60	24	0	24.4	0	23.9	0
23	29	24.8	1	24.8	1	24.6	1	61	24.8	0	24.8	0	24.6	0
24	30	25.4	1	25.6	1	25.7	1	62	25.4	0	25.6	0	25.7	0
25	31	26.8	1	26.4	1	26.6	1	63	26.8	0	26.4	0	26.6	0
26	32	27.4	1	27.2	1	27.5	1	64	27.4	0	27.2	0	27.5	0
27	33	28.4	1	28.4	1	28.6	1	65	28.4	0	28.4	0	28.6	0

Para probar la capacidad de respuesta del sistema con vectores de datos no aprendidos, se le presentó un perfil de movimiento con cambios de velocidad con aceleraciones, desaceleraciones y puntos de paro, en donde el sistema tiene que proporcionar una respuesta adecuada al transitar de una velocidad a otra (por ejemplo al pasar de una velocidad de 5 a 0 revoluciones por segundo, en 1segundo), y responder a valores de velocidad intermedia que no aprendió. Se pudo comprobar que el sistema tiene la capacidad de reproducir el perfil de movimiento.

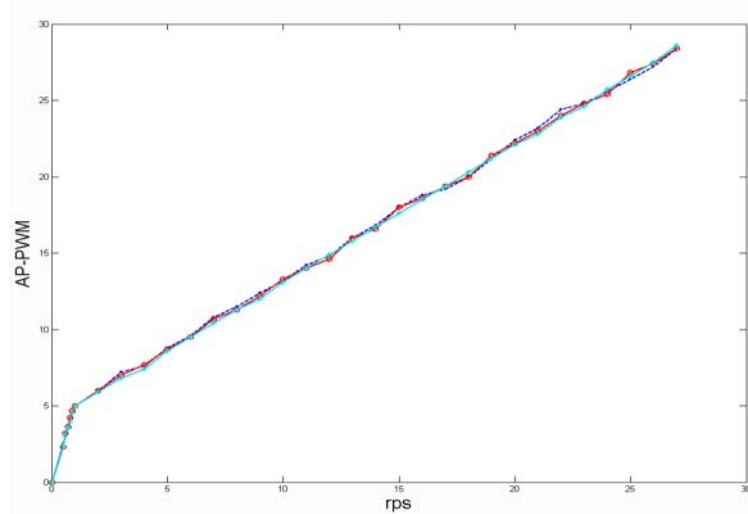
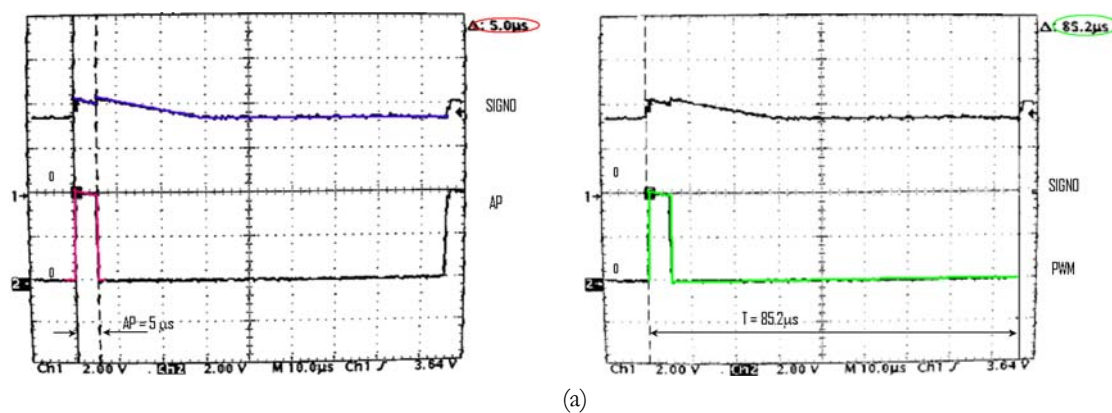


Figura 4.6. Comparativo entre los objetivos de entrenamiento, las salidas obtenidas durante las pruebas, y las salidas obtenidas con un control PID, para perfiles de velocidad fija para la variable AP_{PWM} .

Señales de control generadas por el Acto Motor

Las señales de control generadas por el *Acto Motor* a partir de los datos proporcionados por el Mapeo son la señal PWM y la señal de *Signo*. En la figura 4.7, se muestran la forma de las señales de control producidas por el *Acto Motor* para perfiles de velocidad fija para (a) 1 rps, (b) 10 rps, (c) 20 rps y 27 rps.

En los cuatro incisos, la gráfica de la izquierda muestra la señal de signo en color azul, en este caso es un nivel alto (en términos de la tecnología TTL), por lo que el eje del motor gira en sentido positivo, en la misma gráfica en color rojo, se muestra el pulso de la señal PWM, AP_{PWM} , y en la gráfica de la derecha en verde, se muestra el periodo completo de la señal PWM.



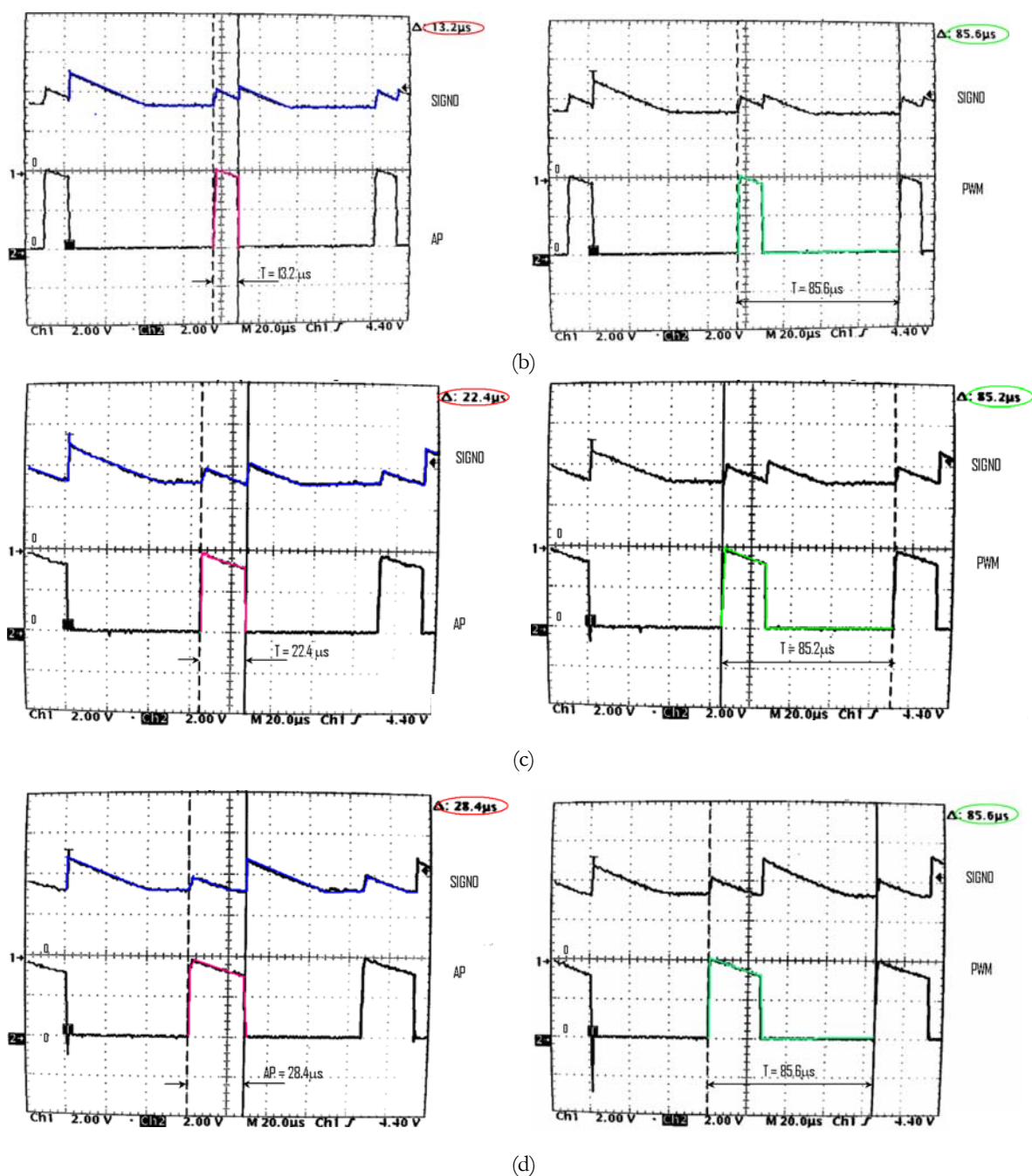


Figura 4.7. Señales de control proporcionadas por el *Acto Motor* a partir de los datos proporcionados para el *Mapeo*. Se pueden observar las señales de *SIGNO*, Ancho de Pulso (*AP*) de la señal PWM y el período completo de la señal PWM para: (a) 1 rps, (b) 10 rps, (c) 20 rps y (d) 27 rps. Las señales de control son específicamente la señal de *Signo* (marcada en azul) y la señal *PWM* (marcada en verde).

Las señales generadas por el *Acto Motor* son enviadas al puente manejador, el cual las traduce en la cantidad de corriente y sentido de flujo de la corriente para que el motor comience la ejecución del movimiento.

En cuanto a los resultados obtenidos a partir de las pruebas hechas con perfiles de movimiento variable, se pudo constatar que, dada la capacidad de generalizar de la red neuronal, el sistema puede controlar al motor para ejecutar perfiles de movimientos variables, y que al hacer pruebas en donde el motor que modela el perfil de movimiento tiene un control PID, ambos motores ejecutan el mismo movimiento, simultáneamente a pesar de que tienen diferencias sustanciales tanto en el rango de velocidades que manejan y torque.

4.5. Resumen

Cuando se utilizan redes neuronales para control, el método más simple para obtener un controlador basado en la adaptación de parámetros a partir de datos entrada-salida, es por modelo inverso. Para obtener el modelo inverso, la red neuronal se entrena con los patrones de entrada y salida, junto con las entradas y salidas pasadas para predecir la señal actual de control. En este caso, los valores de entrada a la red neuronal, son los valores de salida del motor y la salida son los valores entrada. En el caso concreto del sistema de control que se plantea, se aplica un esquema de control basado en un modelo inverso modificado, en donde los valores de entrada a la red provienen de la salida del MM. Adicionalmente, cuando se utiliza una red neuronal, será necesario ajustar ciertos parámetros como la tasa de aprendizaje y el momento, para que el desempeño de la red sea lo más óptimo posible.

Por otra parte, cuando se desea saber que tan eficiente resulta un algoritmo, se puede hacer un análisis de complejidad, el cual utiliza unidades lógicas que expresan el tamaño de un archivo y la cantidad de tiempo requerida para procesar los datos, que en el caso del algoritmo backpropagation esta complejidad no va más allá del orden cuadrático.

Los resultados mostrados, se probaron para motores de corriente directa de distintas dimensiones y se hace una comparación entre los objetivos de salida (patrones de salida), que se utilizaron para el entrenamiento, la respuesta del sistema propuesto y la respuesta cuando el motor está controlado por un PID.

En las secciones siguientes, se presentan las conclusiones, los trabajos futuros, los artículos publicados y trabajos relacionados con la tesis. También se presenta la bibliografía completa y apéndices.

Conclusiones

Partiendo de un esquema básico de aprendizaje por imitación se implementó un sistema básico de control para motores de corriente directa, con herramientas de hardware y de software de propósito general, que tiene la capacidad de aprender la regla de control por medio de ejemplos de comportamiento.

Haber usado una *Red Neuronal* para la implementación del *aprendizaje* en el *sistema de control*, tuvo que ver principalmente con la forma en que se presenta la información sobre el comportamiento del sistema, por la dificultad que representa obtener un modelo matemático no lineal del motor, y por la idea de plantear un mecanismo “artificial”, que funcionara de manera similar a como lo hace el cerebro, en los humanos y algunos animales, cuando aprenden por imitación, lo cual resultó ser una forma muy natural para la implementación. Para la adaptación de la red neuronal a partir de los datos de entrada-salida, se utilizó dinámica inversa.

Se escogió una *Red Neuronal Back-propagation*, principalmente por su capacidad de generalización y por la capacidad de respuesta que tienen cuando la aplicación implica relaciones no lineales, como las que se presentan cuando el sistema trabaja a bajas velocidades, dando la red neuronal resultados satisfactorios. Esto se pudo comprobar durante la segunda etapa de pruebas cuando después de entrenada la red, se le presentó un perfil de movimiento que incluía intervalos en donde el movimiento presentaba aceleraciones o desaceleraciones, puntos de paro, y cambios en la velocidad y en el sentido de movimiento, ya que la *red neuronal* pudo seguir con buena precisión el perfil.

En cuanto a la complejidad del software, ésta es de $O(n^2)$ y se mantiene constante. El tiempo de ejecución en el proceso de aprendizaje (entrenamiento de la red neuronal), puede variar debido a factores como la tasa de aprendizaje, el momento y la cantidad de patrones utilizados para el entrenamiento, pero la complejidad no aumentará en gran medida, aún con el aumento de motores a controlar. Este es uno de los beneficios que proporciona haber usado una red neuronal en el proceso de aprendizaje.

Los resultados obtenidos a partir de las pruebas experimentales hechas al sistema, mostraron resultados satisfactorios, ya que se puede garantizar que al implementar este tipo de control en un sistema de mayor dimensión, la complejidad del algoritmo utilizado no se disparará.

Trabajos futuros

En lo que se refiere a trabajos futuros, el sistema de control con el esquema presentado, puede ampliarse para controlar varios motores con alguna aplicación específica, por ejemplo, para entrenar un brazo mecánico.

Una vez que se comprobó que el sistema de control con aprendizaje por imitación funcionó como se esperaba, podría implementarse una aplicación embebida para controlar el movimiento de prótesis en el cuerpo humano, esto implicaría emplear las técnicas adecuadas para el censado de movimiento de la parte del cuerpo a la que corresponde dicha prótesis.

También podría utilizarse para entrenar partes con el propósito de sustituir partes dañadas o deterioradas dentro un sistema de mayor dimensión.

En general el rango de aplicaciones futuras para el sistema puede ser muy amplio. En términos generales se puede aplicar a sistemas automatizados que tengan que ejecutar cambios continuos de posición o velocidad, ya que el sistema entrenado adecuadamente respondería satisfactoriamente a dichos cambios.

Artículos publicados y trabajos relacionados con la tesis

Durante el desarrollo de este trabajo de tesis, se ha tenido la oportunidad de realizar algunos trabajos relacionados a ésta, entre los que están dos publicaciones en congresos y un proyecto de investigación.

Publicaciones en Congresos.

- “Control de Motores de C. D. con Aprendizaje por Imitación Basado en Redes Neuronales”, presentado en el XVI Congreso Nacional y II Congreso Internacional de Informática y Computación de la ANIEI, celebrado en octubre del 2003 en la Ciudad de Zacatecas, Zacatecas, México. El artículo se publicó en la memoria: CNyCIIC 2003, volumen II, pp. 445-450, ISBN 970-36-0102-2.
- “Aprendizaje por Imitación: Un Esquema con Red Neuronal Aplicado al Control de Motores de CD”, presentado en la XII Convención de Ingeniería Eléctrica, celebrada en junio del 2007 en la ciudad de Santa Clara, Cuba.

Proyecto de investigación.

Se desarrolló un proyecto CGPI titulado “Control de Motores en tiempo real con aprendizaje por imitación utilizando redes neuronales”, clave 20031493, a cargo del Dr. Ricardo Barrón Fernández.

Bibliografía

- [1] **Acosta, C. A., and H. Hu. 2003.** “Goals and Actions: Learning by Imitation”. *Proceedings of AISB 2003, Second International Symposium on Imitation in Animals and Artifacts*: 179-182.
- [2] **Acosta, C. A., and H. Hu. 2003.** “Robotic Societies elements of Learning by Imitation”. *Proceedings of the 21st LASTED International Conference. Applied Informatics*:. 315-320.
- [3] **Alissandrakis, A., C. L. Nehaniv, and K. Dautenhahn. 2000.** “Learning How to Do Things with Imitation”. *AAAI Fall Symposium on Learning How to Do Things, American Association for Artificial Intelligence*: 1-6.
- [4] **Alissandrakis, A., C. L. Nehaniv, and K. Dautenhahn 2002.** “Do as I Do: Correspondences Across Different Robotic Embodiments”. *German Workshop on Artificial Life (GWAL5)*. In D. Polani, J. Kim, T. Martinetz (Eds): 143-152.
- [5] **Aström, K. J., and B. Wittenmark. 1988.** *Sistemas controlados por computador*. Madrid: Paraninfo
- [6] **Axelson, J. 1996.** *Parallel port complete programming, interfacing & using the PC's parallel printer port*. Madison, WI: Lakeview Research.
- [7] **Bakker, P., and Y. Kuniyoshi. 1996.** “Robot See, Robot Do: An Overview of Robot Imitation”. *AISB'96 Workshop on Learning in Robots and Animals*: 3-11.
- [8] **Bandura, A., and R. Walters. 1963.** *Social learning and personality development*. New York: Holt, Rinehart and Winston.
- [9] **Baruch, I., R. Garrido, J.-M. Flores, and J.-C Martínez. 2001.** “An adaptive neural control of a DC motor”. *IEEE International Symposium on Intelligent Control*, 2001. (ISIC '01), pp. 121-126.
- [10] **Belpaeme, T., B. De Boer, B. De Vylder, B. Jansen. 2003.** “The role of population dynamics in imitation”. *Proceedings of the Second International Symposium on Imitation in Animals and Artifacts*: 171-175.
- [11] **Billard, A. 2001.** “Learning Motor Skills by Imitation: A Biologically Inspired Robotic Model”. *Cybernetics and Systems* 32 (1/2): 155-193.
- [12] **Billard, A. 2003.** “Imitation”. In *Handbook of Brain Theory and Neural Networks*. Edited by Michael A. Arbib. 566-569. Cambridge, Mass: MIT Press. 2003.
- [13] **Billard, A., and M. Matarić. 2000.** “A biologically inspired robotic model for learning by imitation”. *Proceedings of the Fourth International Conference on Autonomous Agents*”: 373-380.
- [14] **Billard, A., and R. Siegwart. 2004.** “Robot learning from demonstration”. *Robotics and Autonomous Systems*. 47 (2): 65.
- [15] **Bird, G., M. Osman, A. Saggerson, and C. Heyes. 2005.** “Sequence learning by action, observation an action observation”. *British Journal of Psychology* 96. 96 (3): 371-378.

-
- [16] **Bolton, W. 2003.** *Mechatronics electronic control systems in mechanical and electrical engineering*. Harlow, England: Pearson/Prentice Hall.
 - [17] **Brass M, and C. Heyes. 2005.** "Imitation: is cognitive neuroscience solving the correspondence problem?" *Trends in Cognitive Sciences*. 9 (10): 489-95.
 - [18] **Brey, B., and J. J. del Arco Perez. 2001.** *Los microprocesadores Intel arquitectura, programación e interfaz de los procesadores 8086/8088, 80186/80188, 80286, 80386, 80486 Pentium, Pentium Pro y Pentium II*. México: Pearson educación.
 - [19] **Buchsbaum, D., and B. Blumberg, B. 2005.** "Imitation as a First Step to Social Learning in Synthetic Characters: A Graph-based Approach". *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*: 9-18.
 - [20] **Demiris J., and G. Hayes. 1996.** "Imitative learning mechanisms in robots and humans". D.A.I. research paper, no. 814. Edinburgh: University of Edinburgh, Dept. of Artificial Intelligence.
 - [21] **Flores López R, et all. 2008.** "Redes neuronales artificiales. Fundamentos teóricos y aplicaciones prácticas". España. Netbiblo S. L.
 - [22] **Funahashi, K.-I. 1989.** "On the Approximate Realization of Continuous Mappings by Neural Networks". *Neural Networks*, 2(3), 183-192.
 - [23] **Gonzalez, S., and J. Fernández de Cañete. 2004.** "Neural-network-based stable control by using harmonic analysis. Application to a nonlinear DC motor drive". *Journal Neural Computing & Applications*. Publisher: Springer London. 13 (4).
 - [24] **Gorman, B., and M. Humphrys. 2006.** "Towards Integrated Imitation of Strategic Planning and Motion Modeling in Interactive Computer Games". *Computers in Entertainment (CIE)*. 4 (4).
 - [25] **Gouda M., and A. El-Samahy. 2004.** "Artificial Neural Network based Control of High Performance DC Motor Drive Systems". *Proceeding of (442) European Power and Energy Systems-2004*.
 - [26] **Gupta, M., and N. K. Sinha. 1996.** *Intelligent control systems theory and applications*. Piscataway, NJ: IEEE Press.
 - [27] **Hayes, G., and J. Demiris. 1994.** "A Robot Controller Using Learning by Imitation". *International Symposium on Intelligent Robotic Systems*: 198-204.
 - [28] **Hecht-Nielsen, R. 1989.** "Theory of the backpropagation neural network". *IJCNN. International Joint Conference on Neural Networks*. 1, 593-605.
 - [29] **Hersch, M., and, A. Billard. 2006.** "A Model for Imitating Human Reaching Movements". *Proceeding of the 1st ACM SIGCHI/SIGART conference on Human-Robot-Interaction HRI '06*: 341 – 342.
 - [30] **Heyes, C. 1996.** "Introduction: Identifying and Defining Imitation". *Social Learning in Animals: The Roots of Culture*: 211-220. Academic Press, Inc. 1996.
 - [31] **Heyes, C. 2001.** "Causes and consequences of imitation". *Trends in Cognitive Sciences*. 5 (6): 253-261.
 - [32] **Heyes, C. 2003.** "Four routes of cognitive evolution". *Psychological Review*. 110 (4): 713-27.
 - [33] **Heyes, C. and E. Ray. 2000.** "What Is the Significance of Imitation in Animals?" *Advances in the Study of Behavior*. 29: 215-246.
 - [34] **Hilera González, J. R., and V. J. Martínez Hernando. 2000.** *Redes neuronales artificiales fundamentos, modelos y aplicaciones*. México: Alfaomega.
 - [35] **Iñigo Madrigal, R., and E. Vidal Idiarte. 2004.** *Robots industriales manipuladores*. México: Alfaomega.
 - [36] **Jackson, P.L., A. Meltzoff, and J. Decety. 2006.** "Neural circuits involved in imitation and perspective-taking". *NeuroImage*. 31 (1): 429-39.
-

-
- [37] **Jain, L. C., and W. Clarence. W. De Silva. 1999.** *Intelligent adaptive control industrial applications*. International series on computational intelligence. Boca Raton, Fla: CRC Press.
 - [38] **Jung, B., H. Ben Amor, and G. Heumer, M. Weber. 2006.** "From motion capture to action capture: a review of imitation learning techniques and their application to VR-based character animation". *Proceedings of the ACM symposium on Virtual reality software and technology*: 145 – 154.
 - [39] **Kelly, R., and V. Santibáñez. 2003.** *Control de movimiento de robots manipuladores*. Madrid: Pearson Educación.
 - [40] **Lippman. 1987.** "An introduction to computing with Neural Networks", *IEEE ASSP Magazine*, 3, 4, 4-22.
 - [41] **Lu, Yongxiang, Ying Chen, and Li Xu. 2001.** *Fluid power transmission and control proceedings of the Fifth International Conference on Fluid Power Transmission and Control: (ICFP'2001)*. Beijing: International Academic Publishers/World Pub. Corp.
 - [42] **Martín del Brio, B., and A. Sanz Molina. 2002.** *Redes neuronales y sistemas difusos*. Textos universitarios. México, D. F.: Alfaomega.
 - [43] **Masters, T. 1993.** "Practical Neural Network Recipes in C++". Academic Press, 1993.
 - [44] **Meltzoff, A. N. 2002.** "Imitation as Mechanism of Social Cognition: Origins of Empathy, Theory of Mind, and the Representation of Action". *Blackwell Handbook of Childhood Cognitive Development*: 6-25, Oxford: Blackwell Publishers Ed. U. Goswami.
 - [45] **Meltzoff A. N., and J. Decety. 2003.** "What imitation tells us about social cognition: a rapprochement between developmental psychology and cognitive neuroscience". *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*. 358 (1431): 491-500.
 - [46] **Meltzoff, A. N., and M. K. Moore. 1989.** "Imitation in Newborn Infants: Exploring the Range of Gestures Imitated and the Underlying Mechanisms". *Developmental Psychology*. 25 (6): 954-62.
 - [47] **Meltzoff, A. N., and M. K. Moore. 1998.** "Explaining Facial Imitation: A Theoretical Model". *Annual Progress in Child Psychiatry and Child Development*. 5.
 - [48] **Nguyen, Hung T. 2003.** *A first course in fuzzy and neural control*. Boca Raton, FL: Chapman & Hall/CRC Press.
 - [49] **Ogata, Katsuhiko. 2001.** *Modern control engineering*. Upper Saddle River, NJ: Prentice Hall.
 - [50] **Ozcalik. H. R., and A. Kucuktufekci. 2002.** "An Efficient Neural Controller for a DC Servo Motor by Using ANN and PLR identifiers". *IEEE International Conference on Artificial Intelligence Systems, (ICAIS'02)*, p. 224-227.
 - [51] **Pozo Municio, J. I. 1989.** *Teorias cognitivas del aprendizaje*. Colección Psicología manuales. Madrid: Morata.
 - [52] **Riley, M., A. Ude, K. Wade, and C. G. Atkeson. 2003.** "Enabling Real-Time Full-Body Imitation: A Natural Way of Transferring Human Movement to Humanoids". *Proceedings IEEE International Conference on Robotics and Automation*. 2: 2368-2374.
 - [53] **Rozental, M., and L. Fedorovich. 1965.** *Diccionario filosófico*. Montevideo: Ediciones Pueblos Unidos.
 - [54] **Rubaai, A., R. Kotaru, and M. D. Kankam. 2000.** "A Continually Online-Trained Neural Network Controlled for Brushless DC Motor Drive". *IEEE transactions on industry applications*. 36 (2): 475.
 - [55] **Rubaai, A., and R. Kotaru. 2000.** "Online identification and control of a DC motor using learning adaptation of neural networks". *IEEE transactions on industry applications*. 36 (3): 935.
-

-
- [56] **Schaal, S. 1999.** "Is imitation learning the route to humanoid robots?" *TRENDS IN COGNITIVE SCIENCES*. 3 (6): 233-242
 - [57] **Sciavicco, L., and B. Siciliano. 1996.** *Modeling and control of robot manipulators*. New York: McGraw-Hill Companies, Inc.
 - [58] **Simmons, G., and Y. Demiris. 2004.** "Imitation of human demonstration using a biologically inspired modular optimal control scheme". *Proceedings of the 2004 fourth IEEE-RAS/RSJ international conference on humanoid robots*. 1: 215-234.
 - [59] **Sung, Ha-Gyeong, and Jin Hur. 2004.** "High-Speed Position Control of Linear DC Motor for Carrier with Fast Response Using Seek Algorithm and Neural Network". *Electric Power Components and Systems*. 32 (1): 109-120.
 - [60] **Torres, F. 2002.** *Robots y sistemas sensoriales*. Madrid: Prentice Hall.
 - [61] **Vargas Montoya, S. 1977.** *Tratado de psicología*. México: Editorial Porrúa.
 - [62] **Vielma, E. y M. L. Salas. 2000.** "Aportes de las teorías de Vygotsky, Piaget, Bandura y Bruner. Paralelismo en sus posiciones en relación con el desarrollo". *EDUCERE*, 3 (9).
 - [63] **Weber, S., M. Matarić, and O. Chadwicke. 2000.** "Experiments in Imitation Using Perceptuo-Motor Primitives". *Proceedings of the Fourth International Conference on Autonomous Agents*: 136-137.
 - [64] **Werbos, P. J.** Backpropagation through time: what it does and how to do it. *Proc. IEEE*, Vol. 78, No. 10, October 1990.
 - [65] **Werbos, P. J. 1992.** "Neurocontrol and Supervised Learning: An overview and evaluation". In *Handbook of Intelligent Control. Neural, Fuzzy and Adaptive Reinhold. New York*, 1992.
 - [66] **Werbos, P. J. 1996.** Neurocontrol and Elastic Fuzzy Logic. Capabilities, concepts, and applications. *Intelligent control systems theory and applications*. 327-344. Piscataway, NJ: IEEE Press.
 - [67] **Widrow, B and M.A. Lehr. "Perceptrons, Adalines and y backpropagation".** *Handbook of Brain Theory and Neural Networks*. MIT Press: 566-569.
 - [68] **Wormuth, F. D. 1967.** Matched-Dependent Behavioralism: The Cargo Cult in Political Science. *The Western Political Quarterly*, 20 (4): 809-840.
 - [69] **Wyatt, A. 1992.** *Using assembly language*. Carmel, IN: Que.
 - [70] **Zazueta S., G. Sierra, J. Murillo, F. Quiroz, L. Gutiérrez, J. Valdez, B. Martínez, E. López, y A. Córdoba. 2004.** *La Electrónica de Control del Guiador del Telescopio de 2.12 m. del OAN*. Reporte Técnico en preparación. Instituto de Astronomía.
 - [71] **Zentall, T., and B.G. Galef. 1988.** *Social learning psychological and biological perspectives*. Hillsdale, N.J.: Lawrence Erlbaum Associates.

Manuales de referencia.

- [72] **Frock Heidi. 1995.** "IEEE 1284 – Updating the PC Parallel Port". *National Instruments. The Software is the Instruments*. Application Note 062.
- [73] **National Semiconductor Corporation. 1993.** "LM628/629 User Guide". *Application Note 706*.
- [74] **National Semiconductor. 1999.** "LM628 Programming Guide". *Application note 693*.
- [75] **National Semiconductor. 2000.** "LMD 18201, 3A, 55 V H-Bridge, General Description".
- [76] **National Semiconductor Corporation. 2003.** "LM628/LM629 Precision Motion Controller. General description"

Recursos en línea.

- [77] **Diccionario de la Real Academia Española en línea** . Vigésima segunda edición.
<http://www.rae.es>
- [78] **García de Celis C. 1997.** *El Universo Digital del IBM PC, AT y PS/2. 4ª Edición Electrónica.* Valladolid, España. <http://www.gui.uva.es/udigital>
- [79] **Nortecnica.** Nota técnica: Teoría de sensores inductivos.
http://www.nortecnica.com.ar/pdf/teoria_inductivos_1_2.pdf
- [80] **CRS Robotics Corporation.** www.crsrobotics.com
- [81] **MOTOMAN ROBOTICS IBÉRICA, S.L.** <http://www.motoman.es/aplicaciones.asp>
- [82] **Espacio logopédico.com** <http://www.espaciologopedico.com/recursos/glosariodet.php?Id=266>
- [83] **WorldCat. 2001-2007.** Provided by OCLC Online Computer Library Center, Inc.
<http://worldcat.org>
- [84] **The Chicago manual of style Online, Fifteenth Edition . 2006-2007.** University of Chicago.
<http://www.chicagomanualofstyle.org>

Apéndice I

Programas del sistema

I.1.Código del programa BPNN.C

```
/* **** */
/* Programa para el control de un motor de corriente directa. El control utiliza una red neuronal de 3 capas, con */
/* algoritmo de entrenamiento backpropagation. Con función de transferencia sigmoidea en la capa intermedia y */
/* función de transferencia lineal en la capa de salida. La inicialización de los pesos de las sinapsis se hace de */
/* manera aleatoria. */
/* **** */
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>
#define sigmoid(h) 1.0/(1.0+exp(-h)) /* Definición de la función sigmoidea */

/* Parámetros de la red y el entrenamiento */
int N_ENT; /* Número de neuronas en la capa de entrada */
int N_OC; /* Número de neuronas en la capa intermedia */
int N_SAL; /* Número de neuronas en la capa de salida */
int PTOT; /* Cantidad total de patrones en el archivo */
int PENT; /* Cantidad de patrones de entrenamiento */
int PPRU; /* Cantidad de patrones de prueba */
int ITER; /* Total de Iteraciones */
int GE; /* Para grabar error cada GE iteraciones */
int WTS; /* Número de archivo de sinapsis inicial */
float TASAA; /* Taza de aprendizaje */
float MOMENTO;

/* Matrices globales */
int *sec; /* Secuencia de presentación de los patrones */
float **datoentrena; /* Dato de entrenamiento */
float **datoprueba; /* Dato de prueba */
float **w1,**w2; /* Pesos entre neuronas */
float *grad2,*grad3; /* Gradiente en cada unidad */
float **dw1,**dw2; /* Corrección a cada peso */
float *x1,*x2,*x3; /* Activaciones de cada capa */
float **pred; /* Salidas predichas */
float *objetivo; /* Valor correcto de la salida */

/* Variables globales auxiliares */
char archpat[100];
int i,j,k;
float h;
```

```

int error;
    /* Declaración de prototipos */
int ini_pesos(void);
int guarda_pesos(int WTS);
int lee_pesos(int WTS);
int define_matriz();
int arquitect(char *nomarchivo);
int lee_datos(char *nomarchivo);
void mezclar(int hasta);
void mezclar(int hasta);
void propaga();
float actualiza_error(float **S,int pat_ini,int pat_fin,int usar_sec);
int entrena(char *nomarchivo);
unsigned int _far direccion_puerto_paralelo();
void limpia_puerto(unsigned int _far direccion_puerto);
int direccion_de_giro(unsigned int _far direccion_puerto);
unsigned int periodo(unsigned int _far direccion_puerto);
void pwm(unsigned int _far direccion_puerto, unsigned int cuentaa, unsigned int cuentab, unsigned int signo);

/*****
/*                               Rutina principal                               */
*****/
int main(int argc, char **argv)
{
    unsigned int _far *direccion;
    int i,nu,pa,pb,signo;
    char opcion;

    if(argc!=2){
        printf("Modo de uso: bpm <nomarch>\ndonde nomarch es el nombre del archivo (sin extension)\n");
        return 0;
    }

    /* Abrir puerto paralelo para muestreo y generación de señales */
    *direccion = direccion_puerto_paralelo();
    if (*direccion==0)
        printf("No se encontró puerto asignado a LPT' %d\n", i);
    else
        printf("La dirección asignada a LPT' %d es 0x%Xh\n", i, *direccion);
    limpia_puerto(*direccion);
    /* Definir la arquitectura de la red e inicializar los pesos */
    error=arquitect(argv[1]);
    if(error){
        printf("Error en la definicion de la red\n");
        return 1;
    }

    /* Leer los datos de entrenamiento */
    error=lee_datos(argv[1]);
    if(error){
        printf("Error en la lectura de datos\n");
        return 1;
    }

    /* Entrenar la red */
    error=entrena(argv[1]);
    if(error){
        printf("Error en el entrenamiento\n");
        return 1;
    }
    do {
        printf ("Esperando datos de prueba:\n");
        /* Espera el inicio de movimiento del MM */
        x1[1] = ((float)periodo(*direccion)*1.4)/100; /* Adquiere dato de magnitud */
    } while (1);
}

```

```

printf("\nMuestreando y reproduciendo movimiento\n");
printf("Oprima una tecla al terminar la reproducción del movimiento\n");

do {
    x1[1] = ((periodo(*direccion))*1.4)/100; /* Adquiere dato de magnitud */
    x1[2] = (float)direccion_de_giro(*direccion); /* Adquiere dato de signo */
    /* Propagar la red */
    propaga();
    /* Generar señal PWM */
    pa=ceil((x3[1]*10)*1.13);
    pb=80-pa;
    if (x3[2]>0.5)
        signo=128;
    else
        signo=0;
    for(i=0; i<3513; i++)
        pwm(*direccion,pa,pb,signo);
    } while(1);
do {
    printf("\nDesea llevar a cabo otro perfil de movimiento s/n:");
    opcion=getchar();
    fflush(stdin);
}
while (opcion!='n' && opcion!='N' && opcion!='s' && opcion!='S');
}
while (opcion=='s' || opcion=='S');
return(0);
}
/*****
/* Rutina para inicializar aleatoriamente los valores de todos los pesos de las sinapsis. Los valores están */
/* entre (-1,1) */
*****/
int ini_pesos(void)
{
    int j,i, intervalo_i, intervalo_f, peso_entero, largo;
    float x;
    float peso;
    time_t t;
    FILE *fp;
    char p[13];
    sprintf(p, "%d", WTS);
    largo=strlen(p);
    p[largo]='.';
    p[largo+1]='w';
    p[largo+2]='t';
    p[largo+3]='s';
    p[largo+4]='\0';

    if((fp=fopen(p,"a"))==NULL)
        return 1;
    intervalo_i=-10000;
    intervalo_f=10000;
    clrscr();
    srand((unsigned) time(&t));

    /* Primer capa */
    for (j = 1; j <= N_OC; j++)
        for (i = 0; i <= N_ENT; i++) {
            peso_entero= rand()%(intervalo_f-intervalo_i+1)+intervalo_i;
            peso= (float)peso_entero/RAND_MAX;
            x=peso;

```

```

        if (x<=0.5 | x>=-0.5){
            w1[j][i] = x;
            fprintf(fp,"%f\n",w1[j][i]);
            printf("%d %d peso %f\n", j, i, x);
        }
    }

    /* Segunda capa */
    for (j = 1; j <= N_SAL; j++)
    for (i = 0; i <= N_OC; i++) {
        peso_entero= rand()%(intervalo_f-intervalo_i+1)+intervalo_i;
        peso= (float)peso_entero/RAND_MAX;
        x=peso;
        if (x<=0.5 | x>=-0.5){
            w2[j][i] = x;
            fprintf(fp,"%f\n",w2[j][i]);
            printf("%d %d peso %f\n", j, i, x);
        }
    }
    fclose(fp);
    return 0;
}

/*****
/* Rutina para guardar valores de los pesos en un archivo, el nombre del archivo de salida es X.wts. */
*****/
int guarda_pesos(int WTS)
{
    FILE *fp;
    int largo;
    char p[13];

    sprintf(p, "%d", WTS);
    largo=strlen(p);
    p[largo]='.';
    p[largo+1]='w';
    p[largo+2]='t';
    p[largo+3]='s';
    p[largo+4]='\0';

    if((fp=fopen(p,"a"))==NULL)
        return 1;
    /* Primer capa */
    for(j=1;j<=N_OC;j++)
    for(i=0;i<=N_ENT;i++)
        fprintf(fp,"%f\n",w1[j][i]);

    /* Segunda capa */
    for (i=1;i<=N_SAL;i++)
    for(j=0;j<=N_OC;j++)
        fprintf(fp,"%f\n",w2[i][j]);
    fclose(fp);
    return 0;
}

/*****
/* Rutina para leer valores de los pesos desde un archivo con extensión .wts */
*****/
int lee_pesos(int WTS)
{
    FILE *fp;
    int largo;
    char p[13];

```

```

    sprintf(p, "%d", WTS);
    largo=strlen(p);
    p[largo]='.';
    p[largo+1]='w';
    p[largo+2]='t';
    p[largo+3]='s';
    p[largo+4]='\0';
    if((fp=fopen(p,"r"))==NULL) return 1;
    /* Primera capa */
    for(j=1;j<=N_OC;j++)
    for(i=0;i<=N_ENT;i++){
    fscanf(fp,"%f",&w1[j][i]);
    /* Segunda capa */
    for (j=1;j<=N_SAL;i++){
    for(i=0;j<=N_OC;j++){
    fscanf(fp,"%f",&w2[j][i]);
    printf("%f",w2[j][i]);
    }
    }
    fclose(fp);
    return 0;
    }
    /*****
    /* Rutina que reserva espacio en memoria para todas las matrices, las dimensiones son leídas del archivo */
    /* con extensión net */
    /*****
    int define_matriz()
    {
    int max;
    if(PTOT>PPRU) max=PTOT;
    else max=PPRU;
    sec=(int *)calloc(max,sizeof(int));
    x1=(float *)calloc(N_ENT+1,sizeof(float));
    x2=(float *)calloc(N_OC+1,sizeof(float));
    x3=(float *)calloc(N_SAL+1,sizeof(float));
    grad2=(float *)calloc(N_OC+1,sizeof(float));
    grad3=(float *)calloc(N_SAL+1,sizeof(float));
    objetivo=(float *)calloc(N_SAL+1,sizeof(float));
    if(sec==NULL || x1==NULL || x2==NULL || x3==NULL || grad2==NULL || grad3==NULL || objetivo==NULL)
    return 1;
    w1=(float **)calloc(N_OC+1,sizeof(float *));
    w2=(float **)calloc(N_SAL+1,sizeof(float *));
    dw1=(float **)calloc(N_OC+1,sizeof(float *));
    dw2=(float **)calloc(N_SAL+1,sizeof(float *));
    datoentrena=(float *)calloc(PTOT,sizeof(float *));
    if(PPRU) datoprueba=(float **)calloc(PPRU,sizeof(float *));
    pred=(float **)calloc(max,sizeof(float *));
    if(w1==NULL || w2==NULL || dw1==NULL || dw2==NULL || datoentrena==NULL || (PPRU&&datoprueba==NULL) ||
    pred==NULL)
    return 1;
    for(i=0;i<=N_OC;i++){
    w1[i]=(float *)calloc(N_ENT+1,sizeof(float));
    dw1[i]=(float *)calloc(N_ENT+1,sizeof(float));
    if(w1[i]==NULL || dw1[i]==NULL) return 1;
    }
    for(i=0;i<=N_SAL;i++){
    w2[i]=(float *)calloc(N_OC+1,sizeof(float));
    dw2[i]=(float *)calloc(N_OC+1,sizeof(float));
    if(w2[i]==NULL || dw2[i]==NULL) return 1;
    }

    for(i=0;i<PTOT;i++){

```

```

datoentrena[i]=(float *)calloc(N_ENT+N_SAL+1,sizeof(float));
if(datoentrena[i]==NULL)
return 1;
}
for(i=0;i<PPRU;i++){
datoprueba[i]=(float *)calloc(N_ENT+N_SAL+1,sizeof(float));
if(datoprueba[i]==NULL)
return 1;
}
for(i=0;i<max;i++){
pred[i]=(float *)calloc(N_SAL+1,sizeof(float));
if(pred[i]==NULL)
return 1;
}
return 0;
}
/*****
/* Rutina que establece la arquitectura de la red neurona y establece los par metros de entrenamiento, */
/* en base a los datos leídos del archivo imita.net */
*****/
int arquitec(char *nomarchivo)
{
FILE *b;
/* abrir archivo para lectura*/
sprintf(archpat,"%s.net",nomarchivo);
b=fopen(archpat,"r");
error=(b==NULL);
if(error){
printf("Error al abrir el archivo de parametros\n");
return 1;
}
/* lee estructura de la red */
fscanf(b,"%d",&N_ENT);
fscanf(b,"%d",&N_OC);
fscanf(b,"%d",&N_SAL);
/* lee datos para entrenamiento y para validacion */
fscanf(b,"%d",&PTOI);
fscanf(b,"%d",&PENTI);
fscanf(b,"%d",&PPRU);
/* lee parametros para el entrenamiento */
fscanf(b,"%d",&ITER);
fscanf(b,"%f",&TASAA);
fscanf(b,"%f",&MOMENTO);
/* lee dato para la salida de resultados */
fscanf(b,"%d",&GE);
/* nombre archivo sinapsis inicial */
fscanf(b,"%d",&WTS);
fclose(b);
/* Definir matrices para datos y pesos */
error=define_matriz();
if(error){
printf("Error en la definicion de matrices\n");
return 1;
}
error=ini_pesos();
if(error){
printf("Error en la inicializacion de los pesos\n");
return 1;
}
/* Imprime datos en pantalla */
printf("\nArquitectura de la red: %d:%d:%d",N_ENT,N_OC,N_SAL);

```

```

printf("\nArchivo de patrones: %s",nomarchivo);
printf("\nCantidad total de patrones: %d",PTOT);
printf("\nCantidad de patrones de entrenamiento: %d",PENT);
printf("\nCantidad de patrones de validacion: %d",PTOT-PENT);
printf("\nCantidad de patrones de prueba: %d",PPRU);
printf("\nEpocas: %d",ITER);
printf("\nTaza de aprendizaje: %f",TASAA);
printf("\nMomentum: %f",MOMENTO);
printf("\nFrecuencia para grabar resultados: %d EPOCAS",GE);
printf("\nArchivo con pesos iniciales: %d.WTS",WTS);
printf("\nPESOS INICIALES:\n");
for(j=1;j<=N_OC;j++) {
    for(i=0;i<=N_ENT;i++)
        printf("%f\n",w1[j][i]);
}
for(j=1;j<=N_SAL;j++) {
    for(i=0;i<=N_OC;i++)
        printf("%f\n",w2[j][i]);
}
return 0;
}
/*****
/* Función que lee los patrones de entrenamiento desde un archivo con extension .dat */
*****/
int lee_datos(char *nomarchivo)
{
    FILE *fpat;
    float valor;
    int separador;
    sprintf(archpat,"%s.dat",nomarchivo);
    fpat=fopen(archpat,"r");
    error=(fpat==NULL);
    if(error){
        printf("Error al abrir el archivo de datos\n");
        return 1;
    }
    printf("\n\nDatos de entrenamiento:");

    for(k=0;k<PTOT;k++){

        printf("\nP%d:\t",k);
        datoentrena[k][0]=-1.0;
        for(i=1;i<=N_ENT+N_SAL;i++){
            fscanf(fpat,"%f",&valor);
            datoentrena[k][i]=valor;
            printf("%f\t",datoentrena[k][i]);
            separador=getc(fpat);
            if(separador!='\n') ungetc(separador,fpat);
        }
        fclose(fpat);
        return 0;
    }
    /*****
    /* Rutina que genera aleatoriamente los índices para acceder a los patrones de entrenamiento. */
    *****/
    void mezclar(int hasta)
    {
        float x;
        int tmp;
        int top,select;

```

```

top=hasta-1;
while (top > 0) {
x = (float)rand();
x /= (RAND_MAX+1.0);
x *= (top+1);
select = (int)x;
tmp = sec[top];
sec[top] = sec[select];
sec[select] = tmp;
top--;
}

fflush(NULL);
}
/*****
/* Rutina que propaga el vector de entrada X1[] a través de la estructura de la red neuronal */
*****/
void propaga()
{
/* Calcular los X2 (capa intermedia) */
x2[0]=-1.0;
for(j = 1 ;j<= N_OC;j++){
h = 0.0;
for(k = 0 ;k<= N_ENT;k++)
h += w1[j][k] * x1[k];
x2[j] = sigmoid(h); /* Función de activación sigmoidea */
}

/* Calcular los x3 (capa de salida) */
for(i = 1 ;i<= N_SAL;i++){
h = 0.0;
for(j = 0 ;j<= N_OC;j++)
h += w2[i][j] * x2[j];
x3[i] = h ; /* Función de activación lineal */
}
fflush(NULL);
}
/*****
/* Rutina que actualiza el error estimado. */
*****/

float actualiza_error(float **S,int pat_ini,int pat_fin,int usar_sec)
{
float mse=0.0;
int patron,np;

for (patron=pat_ini; patron < pat_fin; patron++) {
if(usar_sec)
np = sec[patron]; /* numero de patrón */
else
np = patron;
for (i = 0; i <= N_ENT; i++)
x1[i] = S[np][i]; /* carga el patrón en x1 */
propaga(); /* lo propaga en la red */
for(i = 1; i <= N_SAL; i++) {
pred[np][i]=x3[i];
/* y actualizar error estimado */
mse += (x3[i]-S[np][N_ENT+i])*(x3[i]-S[np][N_ENT+i]);
}
}
mse /= ((float)(pat_fin-pat_ini));

```

```

fflush(NULL);
return mse;
}
/*****
/* Rutina de que entrena a la red neuronal, con el algoritmo backpropagation. El error calculado (prueba, */
/* al final de la ,poca, de validación y de prueba se guardan en el archivo con extensión mse. Los pesos */
/* finales de las sinapsis se guardan en el archivo con extension wts. */
*****/
int entrena(char *nomarchivo)
{
    int nu, nu1;
    int iter,epocas_del_minimo,largo;
    float taa,suma, cota;
    float mse,mse_entrena,mse_valid,mse_prueba,minimo_valid;
    char p[13];
    FILE *ferror,*fpredic, *fpesos;

    /* Inicializar archivos de control */
    sprintf(archpat,"%s.predic",nomarchivo);
    fpredic=fopen(archpat,"w");
    error=(fpredic==NULL);
    if(error){
        printf("Error al abrir archivo para guardar predicciones\n");
        return 1;
    }
    sprintf(archpat,"%s.mse",nomarchivo);
    ferror=fopen(archpat,"a");
    error=(ferror==NULL);
    if(error){
        printf("Error al abrir archivo para guardar curvas\n");
        return 1;
    }
    sprintf(p, "%d", WTS);
    largo=strlen(p);
    p[largo]='.';
    p[largo+1]='w';
    p[largo+2]='t';
    p[largo+3]='s';
    p[largo+4]='\0';
    if((fpesos=fopen(p,"a"))==NULL)
        return 1;
    fprintf(fpesos,"-----\n");

    /* Inicializacion de todas las matrices */
    for (j = 1; j <= N_OC; j++) {
        grad2[j] = 0.0;
        for (k = 0; k <= N_ENT; k++)
            dw1[j][k]=0.0;
    }
    for (i = 1; i <= N_SAL; i++) {
        grad3[i] = 0.0;
        for (j = 0; j <= N_OC; j++)
            dw2[i][j]=0.0;
    }
    for(k=0;k<PTOT;k++)
        sec[k]=k; /* Inicializacion del indice de acceso a los datos */
    x1[0]=-1.0; /* bias de las unidades de cada hilera */
    x2[0]=-1.0;
    minimo_valid=1000000.0;
    taa=TASAA;
    mezclar(PTOT);

```

```

for (iter=1; iter<=ITER; iter++) {
mse = 0.0;
cota = 0.000001;
mezclar(PENT);
/* Barrido sobre los patrones de entrenamiento */
for(nu1=0;nu1<PENT;nu1++) {
nu = sec[nu1];

for( k=1 ;k<=N_ENT;k++)      /* Carga el patron en x1 */
x1[k] = datoentrena[nu][k];
propaga();                    /* lo propaga a trav,s de la red */

for( k=1 ;k<=N_SAL;k++)      /* carga los objetivos */
objetivo[k] = datoentrena[nu][k+N_ENT];
for(i = 1 ;i<= N_SAL;i++)
{
/* calcula el gradiente en la salida */
grad3[i] = (objetivo[i] - x3[i]); /* Corresponde a lineal */
}
for(j = 1 ;j<= N_OC;j++)
{
/*calcula el gradiente en la capa oculta*/
suma = 0.0;
for(i = 1 ;i<= N_SAL;i++)
suma += grad3[i] * w2[j][i];
grad2[j] = suma * (1.0- x2[j]) * x2[j];
}
/* calcula dw2 y corrige w2 */
for( i = 1 ;i<= N_SAL;i++)
for( j = 0 ;j<= N_OC;j++){
dw2[j][i]= MOMENTO * dw2[j][i] + taa * grad3[i] * x2[j];
w2[j][i] += dw2[j][i];
fprintf(fpesos,"%f\n",w2[j][i]);
}
/* calcula dw1 y corrige w1 */
for( j = 1 ;j<= N_OC;j++)
for( k = 0 ;k<= N_ENT;k++){
dw1[j][k]= MOMENTO * dw1[j][k] + taa * grad2[j] * x1[k];
w1[j][k] += dw1[j][k];
fprintf(fpesos,"%f\n",w1[j][k]);
}
fprintf(fpesos,"-----\n");
/* Actualizar el mse */
for(i = 1 ;i<= N_SAL;i++)
mse += (x3[i] - objetivo[i]) * (x3[i] - objetivo[i]);
if (mse<=cota)
break;
}
/* Grabar error cada NERROR iteraciones */
if ((iter/GE)*GE == iter) {
mse /= ((float)PENT);
mse_entrena=actualiza_error(datoentrena,0,PENT,1);
/* Calcular mse de validacion; si no hay, usar mse_entrena */
if(PENT==PTOT){
mse_valid=mse_entrena;
} else{
mse_valid=actualiza_error(datoentrena,PENT,PTOT,1);
}
/* Calcular mse de prueba (si hay) */
if (PPRU>0){
mse_prueba =actualiza_error(datoprueba,0,PPRU,0);
}
}

```

```

} else mse_prueba = 0;
fprintf(ferror, "%f\t%f\t%f\t", mse, mse_entrena, mse_valid, mse_prueba);
fflush(NULL);
if(mse_valid < minimo_valid) {
    guarda_pesos(WTS+1);
    minimo_valid = mse_valid;
    epocas_del_minimo = iter;
}
}
fflush(NULL);
}
/* Mostrar resumen del entrenamiento */
printf("\nFin del entrenamiento.\n\n");
printf("Error final:\nEntrenamiento(est):%f\nEntrenamiento(med):%f\n", mse, mse_entrena);
printf("Validacion:%f\nPrueba:%f\n", mse_valid, mse_prueba);

/* Calcular y guardar predicciones sobre el archivo de prueba */
/* Leer pesos del m nimo de validaci n desde archivo */
lee_pesos(WTS);
mse_prueba = actualiza_error(datoprueba, 0, PPRU, 0);

for(k=0; k < PPRU; k++) {
    for(i = 1; i <= N_ENT; i++)
        fprintf(fpredic, "%f\t", datoprueba[k][i]);
    for(i=1; i <= N_SAL; i++)
        fprintf(fpredic, "%f\t", pred[k][i]);
    fprintf(fpredic, "\n");
}
printf("\nError minimo en
validacion:\nEpoca:%d\nValidacion:%f\nPrueba:%f\n\n", epocas_del_minimo, minimo_valid, mse_prueba);
fclose(fpredic);
fclose(ferror);
fclose(fpesos);
return 0;
}
/*****
/* Rutina que identifica la direcci n del puerto paralelo. El puerto paralelo es utilizado para recibir datos del */
/* codificador de cuadratura del MM, enviarla se al PWM del MA y recibir la se al de inicio de movimiento */
/* del MM, para sincronizar la operaci n del sistema */
*****/
unsigned int _far direccion_puerto_paralelo()
{
    unsigned int _far *puntero_a_direccion; /* Debe ser de al menos 2 bytes */
    puntero_a_direccion = (unsigned int _far*)0x00000408; /* volcado de memoria */

    if (*puntero_a_direccion == 0) /* No se encontro puerto paralelo */
        return(0);
    else
        return(*puntero_a_direccion);
}
/*****
/* Rutina que limpia el puerto paralelo */
*****/
void limpia_puerto(unsigned int _far direccion_puerto)
{
    asm push ax
    asm push dx
    asm mov dx, direccion_puerto
    asm mov ax, 0
    asm out dx, ax
    asm pop dx

```

```

    asm pop ax
}
/*****
/* Rutina para determinar la dirección de giro del eje del MM */
*****/
int direccion_de_giro(unsigned int _far direccion_puerto)
{
    unsigned int mascara=72;
    unsigned int direccionmasuno;
    direccionmasuno=direccion_puerto+1;

    asm push ax
    asm push dx
    asm mov ax,0 /* Limpiamos registros */
    asm mov dx,0
    asm mov dx, direccionmasuno /* Dirección del registro de estado */
caso_1:
    asm in al,dx /* Espera a que ambos bits sean cero */
    asm and ax,mascara
    asm cmp ax,0
    asm jne
caso_1
    lee_otra_vez:
    asm in al,dx
    asm and ax,mascara
    asm cmp al,0
    asm je lee_otra_vez
    asm cmp ax,64
    asm je salto1 /* En dirección de las manecillas del reloj */
    asm cmp ax,8
    asm je salto2 /* En dirección opuesta a las manecillas del reloj */
salto1:
    asm pop dx
    asm pop ax
    return(1);
salto2:asm pop dx
    asm pop ax
    return(-1);
}
/*****
/* Rutina que determina el periodo de una señal en cuadratura */
*****/
unsigned int periodo(unsigned int _far direccion_puerto)
{
    unsigned int cuentas;
    unsigned int direccion_mas_uno;

    direccion_mas_uno=direccion_puerto+1;
    asm push ax
    asm push bx
    asm push cx
    asm push dx
    asm mov ax,0
    asm mov bx,0
    asm mov cx,0
    asm mov dx,0
referencia1:
    asm mov dx, direccion_mas_uno
    asm in al,dx
    asm and al,64
    asm cmp al,64

```

```

asm je referencia1
referencia2:
asm mov dx, direccion_mas_uno
asm in al,dx
asm and al,64
asm cmp al,0
asm je referencia2
asm mov ax,0
asm mov dx,0
vuelve_a_leer:
asm inc cx
asm push ax
asm push dx
asm mov ax,0
asm mov dx,direccion_mas_uno
asm in al,dx
asm mov bl,al
asm pop dx
asm pop ax
asm and bl,64
asm cmp bl,bh
asm je vuelve_a_leer
asm mov bh,bl
asm inc dx
asm cmp dx,3
asm je fin
asm jmp vuelve_a_leer
fin:
asm mov cuentas,cx
asm pop dx
asm pop cx
asm pop bx
asm pop ax
return(cuentas);
}
/*****
/* Rutina que genera la señal PWM */
*****/
void pwm(unsigned int _far direccion_puerto, unsigned int cuentaa, unsigned int cuentab, unsigned int signo)
{
asm push ax
asm push dx
asm mov ax,0
asm mov dx,0

asm mov dx, 43h /* Direcciona la palabra de control */
asm mov al, 10110000b /* contador 2, modo 0 */
asm out dx,al
asm mov dx,42h /* Direcciona la cuenta al timer 2 */
asm mov ax,cuentaa /* Carga la cuenta */
asm out dx,al /* Carga el byte bajo */
asm xor al,al
asm out dx,al /* Carga el byte alto */
asm in al,61h
asm xor al,1
asm out 61h, al
asm mov dx,direccion_puerto /* Direcciona el puerto paralelo */
asm mov ax,signo
asm or al,1 /* y escribe 1 */
asm out dx,al

```

```

verifica_out:                                     /* Verifica cuando el timer termina */
    asm in al, 61h      /* la cuenta */
    asm and al,00100000b
    asm cmp al,0
    asm je verifica_out
    asm mov al,0
    asm out 61h,al
    asm mov ax,signo
    asm or  al,0        /* y escribe 0 */
    asm out dx,al
    asm mov dx,42h      /* Direcciona la cuenta al timer 2 */
    asm mov ax,cuentab  /* Carga la cuenta*/
    asm out dx,al       /* Carga el byte bajo */
    asm xor al,al
    asm out dx,al       /* Carga el byte alto */
    asm in al,61h
    asm xor al,1
    asm out 61h, al
    asm mov dx,direccion_puerto /* Direcciona el puerto paralelo*/
    asm mov ax,signo
    asm or  al,0        /* y escribe 0 */
    asm out dx,al
verifica_out2: /* Verifica cuando el timer termina */
    asm in al, 61h      /* la cuenta */
    asm and al,00100000b
    asm cmp al,0
    asm je verifica_out2
    asm mov al,0
    asm out 61h,al
    asm pop dx
    asm pop ax
}

```

I.2. Código del programa PERFIL1.C

```

/*****
/* Programa que envía comandos a una tarjeta que controla el movimiento del Motor Maestro y que
/* genera un perfil de movimiento predeterminado.
*****/

#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <dos.h>

typedef struct {          /* Desglosa un número hexadecimal de 8 bytes en cuatro componentes */
int bytehh;
int bytehl;
int bytelh;
int bytell;
} doublebyte;

/*Declaración de prototipos */

void ini629(int portcom, int portdato);
void inifiltro(int portcom, int portdato);
void inibreakpoint(int portcom, int portdato);
void breakpointrel(int portcom, int portdato, doublebyte dato);
void breakpointabs(int portcom, int portdato, doublebyte dato);
void inimodo(int portcom, int portdato, doublebyte dato);
void cambiaacel(int portcom, int portdato, doublebyte dato);

```

```

void cambiavel(int portcom, int portdato, doublebyte dato);
void cambiapos(int portcom, int portdato, doublebyte dato);
void inimov(int portcom);
void ocupado(int pcomando);
void esperaint(int portcom);
void paramov(int portcom, int portdato);
doublebyte decahex(unsigned long numerodec);
void retardo(void);
unsigned long escalaivel(float valor);
unsigned long escalaivel(float valor);
unsigned long escalacuentas(float valor);
unsigned long escalapos(float valor);

/*****
/* Rutina que genera un perfil de movimiento determinado */
*****/

void main(void)
{
doublebyte acelera, veloci, posi, modo, cuentas;
unsigned long dato;
int i;

clrscr();
outportb(0x302,0x00);          /* reinicio del 629 */
ocupado(0x302);
ini629(0x302, 0x303);          /* inicializa al 629 */
inifiltro(0x302, 0x303);        /* inicializa al filtro del CI 629 */  inibreakpoint(0x302, 0x303);      /* Inicializa la
interrupción de breakpoint */
dato = escalacuentas(80);        /* Dato para breakpoint relativo */
cuentas = decahex(dato);         /* Transforma dato a numero hexadecimal */
breakpointrel(0x302, 0x303, cuentas); /* Se envía dato de breakpoint al 629 */
modo = decahex(6184); /* Convierte dato a numero hexadecimal para seleccionar modo */
inimodo(0x302, 0x303, modo);      /* Se envía dato del nuevo modo al 629 */
dato = escalaivel(1*3);          /* Dato para aceleración (rev/seg^2) */
acelera = decahex(dato); /* Convierte dato a número hexadecimal para seleccionar aceleracion */
cambiaivel(0x302,0x303, acelera); /* Se envía la nueva aceleración al 629 */
dato = escalaivel(5*3); /* Dato para velocidad (rev/seg) */
veloci = decahex(dato); /* Transforma dato a numero hexadecimal para seleccionar velocidad */
cambiavel(0x302, 0x303, veloci); /* Se envía la nueva velocidad al 629 */
inimov(0x302); /* Indica al 629 que inicie movimiento */
printf("Esperc, ejecutando perfil de movimiento\n");

retardo();
ini629(0x302, 0x303);
dato = escalacuentas(160); /* Dato para breakpoint absoluto */
cuentas = decahex(dato); /* Transforma dato a numero hexadecimal */
breakpointabs(0x302, 0x303, cuentas); /* Se envía dato de breakpoint al 629 */
esperaint(0x302); /* espera a que se genere una interrupción de breakpoint */
paramov(0x302, 0x303); /* Indica hacer un paro a una desaceleración uniforme */
sleep(6);
ini629(0x302, 0x303);
inifiltro(0x302, 0x303);
modo = decahex(42);
inimodo(0x302, 0x303, modo);
dato = escalaivel(1*3); /* Dato para aceleración (rev/seg^2) */
acelera = decahex(dato); /* Transforma dato a numero hexadecimal para seleccionar aceleracion */
cambiaivel(0x302,0x303, acelera); /* Se envía la nueva aceleración al 629 */
dato = escalaivel(5*3); /* Dato para velocidad (rev/seg) */
veloci = decahex(dato); /* Transforma dato a numero hexadecimal para seleccionar velocidad */
cambiavel(0x302, 0x303, veloci); /* Se envía la nueva velocidad al 629 */
dato = escalapos(10); /* Dato para nueva posición (giro en sentido contrario) */

```

```

posi = decahex(dato); /* Transforma dato a numero hexadecimal para seleccionar posición */
cambiapos(0x302, 0x303, posi); /* Se envia la nueva posición al 629 */
inimov(0x302); /* Indica al 629 que inicie movimiento */
esperaint(0x302); /* espera a que se genere una interrupcion de breakpoint */
clrscr();
}

/*****
/* Rutina para inicializar al CI 629 */
*****/
/*****
void ini629(int portcom, int portdato)
{
    outportb(portcom, 0x1d); /* rsti */
    ocupado(portcom);

    outportb(portdato, 0x00); /* se limpian banderas */
    outportb(portdato, 0x00);
    ocupado(portcom);
}

/*****
/* Rutina para cargar los parametros del filtro del CI 629 */
*****/
/*****
void inifiltro(int portcom, int portdato)
{
    outportb(portcom, 0x1e); /* LFIL se cargan los parámetros del filtro */
    ocupado(portcom);

    outportb(portdato, 0x7f); /* valores tabla IV */
    outportb(portdato, 0x08);
    ocupado(portcom);

    outportb(portdato, 0x00); /* coeficiente para KP */
    outportb(portdato, 0xff);
    ocupado(portcom);

    outportb(portcom, 0x04); /* se actualizan los datos del filtro */
    ocupado(portcom);
}

/*****
/* Rutina para indicar el modo de funcionamiento del CI 629 */
*****/
/*****
void inimodo(int portcom, int portdato, doublebyte dato)
{
    outportb(portcom, 0x1f); /* comando LTRJ */
    ocupado(portcom);

    outportb(portdato, dato.bytelh); /* se cargan los parámetro de modo: */
    outportb(portdato, dato.bytell); /* modo posición o modo velocidad */
    ocupado(portcom);
}

/*****
/* Rutina para cargar al CI 629 el valor de aceleracion del motor */
*****/
/*****
void cambiaaccel(int portcom, int portdato, doublebyte dato)
{
    outportb(portdato, dato.bytehh);
    outportb(portdato, dato.bytehl);
}

```

```

ocupado(portcom);
outportb(portdato,dato.bytelh);          /* dato para la aceleración */
outportb(portdato,dato.bytell);
ocupado(portcom);
}

/*****
/* Rutina para cargar al CI 629 el valor de velocidad del motor */
*****/
void cambiavel(int portcom, int portdato, doublebyte dato)
{
outportb(portdato,dato.bytehh);
outportb(portdato,dato.bytehl);
ocupado(portcom);

outportb(portdato,dato.bytelh);          /* dato para la velocidad */
outportb(portdato,dato.bytell);
ocupado(portcom);
}

/*****
/* Rutina para cargar al CI 629 el valor de posicion del motor */
*****/
void cambiapos(int portcom, int portdato, doublebyte dato)
{
outportb(portdato,dato.bytehh);
outportb(portdato,dato.bytehl);
ocupado(portcom);

outportb(portdato,dato.bytelh);          /* dato para la posición */
outportb(portdato,dato.bytell);
ocupado(portcom);
}

/*****
/* Rutina que indica al CI 629 iniciar el movimiento del motor */
*****/
void inimov(int portcom)
{
outportb(portcom,0x01);                  /* inicio del control de movimiento */
ocupado(portcom);
}

/*****
/* Rutina que verifica el estado del bit de ocupado en el byte de status */
/* del CI 629 */
*****/
void ocupado(int pcomando)
{
unsigned char dato,dato1;
int sal=0;

do
{
dato=inportb(pcomando);
dato1=dato & 0x01;                      /* se checa el bit 0 del status */
if (dato1==0)
sal=1;
}
while(sal==0);
}

```

```

/*****
/* Rutina que convierte un numero decimal a hexadecimal y lo desglosa en      */
/* cuatro components.                                                         */
*****/
doublebyte decahex(unsigned long numerodec)
{

// ldiv_t numx;
unsigned long numx;
unsigned long numerod;
int i;
unsigned long numerohex[8];
doublebyte numhexa;

for (i=0;i<8;i++)
    numerohex[i]=0;
numerod = numerodec;
i--;
do

/* bucle para convertir un numero decimal a hexadecimal */
{
    numerohex[i] = numerod%16;
    numx = numerod/16;
    numerod = numx;
    i--;
}
while(numerod>=1);

/* Desglose del numero en cuatro componentes... */
numhexa.bytell = numerohex[7] + numerohex[6] * 16;
numhexa.bytelh = numerohex[5] + numerohex[4] * 16;
numhexa.bytehl = numerohex[3] + numerohex[2] * 16;
numhexa.bytehh = numerohex[1] + numerohex[0] * 16;

return(numhexa);
}

/*****
/* Rutina que inicializa al CI 629 para poder ejecutar breakpoints          */
*****/
/*****
void inibreakpoint(int portcom, int portdato)
{
    outportb(portcom,0x1c); /* se enmascaran interrupciones */
    ocupado(portcom);

    outportb(portdato,0x00); /* se desenmascara el bit de int. de breakpoint */
    outportb(portdato,0x40);
    ocupado(portcom);
}

/*****
/* Rutina para cargar al CI 629 el valor de un breakpoint relativo          */
*****/
/*****
void breakpointrel(int portcom, int portdato, doublebyte dato)
{
    outportb(portcom,0x21); /* se inicia cargando un breakpoint relativo */
    ocupado(portcom);
}

```

```

outportb(portdato,dato.bytehh); /* se carga el breakpoint relativo */
outportb(portdato,dato.bytehl);
ocupado(portcom);

outportb(portdato,dato.bytehh);
outportb(portdato,dato.bytehl);
ocupado(portcom);
}

/*****
/* Rutina para cargar al CI 629 el valor de un breakpoint absoluto */
*****/
/*****
void breakpointabs(int portcom, int portdato, doublebyte dato)
{
outportb(portcom,0x20); /* se inicia cargando un breakpoint absoluto */
ocupado(portcom);

outportb(portdato,dato.bytehh); /* se carga el breakpoint absoluto */
outportb(portdato,dato.bytehl);
ocupado(portcom);

outportb(portdato,dato.bytehh);
outportb(portdato,dato.bytehl);
ocupado(portcom);
}

/*****
/* Rutina para esperar una interrupcion por breakpoint del CI 629. Se realiza una */
/* verificación constante hace uso del pooleo del byte de status. */
*****/
void esperarint(int portcom)
{
int estado, estado2;
int sal=0;

do
{
estado = inportb(portcom); /* se poolea el byte de status para verificar cuando se */
estado2 = estado & 0x40; /* activa la interrupción */
if (estado2!=0)
sal = 1;
}
while(sal==0);
}

/*****
/* Rutina que indica al CI 629 detener el movimiento del motor */
*****/
void paramov(int portcom, int portdato)
{
outportb(portcom,0x1f); /* carga buffers de parametros de trayectoria */
ocupado(portcom);
outportb(portdato,0x04); /* se indica parar motor y que no se daran - */
outportb(portdato,0x00); /* parametros de trayectoria */
ocupado(portcom);
outportb(portcom,0x01);
ocupado(portcom);
}

```

```

/*****
/*          Rutina que genera un retardo          */
/*****
void retardo(void)
{
    long int i=0,j=0;

    for (j<299;j++);
    for (i<7000000;i++);
}

/*****
/* Rutina que genera el valor de velocidad interpretable por el CI 629          */
/*****
unsigned long escalavel(float valor)
{
    float veloci;
    unsigned long velround;
    float residuo;

    veloci = 8946.97472*valor;
    velround = veloci;
    residuo = veloci-velround;
    if (residuo>=0.5)
        velround = velround + 1;
    return (velround);
}

/*****
/* Rutina que genera el valor de aceleracion interpretable por el CI 629          */
/*****
unsigned long escalaacel(float valor)
{
    float acel;
    unsigned long acelround;
    float residuo;

    acel = 3.0536*valor;
    acelround = acel;
    residuo = acel-acelround;
    if (residuo>=0.5)
        acelround = acelround + 1;
    return (acelround);
}

/*****
/* Rutina que genera el numero de cuentas interpretable por el CI 629          */
/*****
unsigned long escalacuentas(float valor)
{
    unsigned long cuenta;
    cuenta = 400 * valor;
    return(cuenta);
}

/*****
/* Rutina que genera el numero de rotaciones interpretable por el CI 629          */
/*****
unsigned long escalapos(float valor)
{

```

```
unsigned long posi;
posi=4294967295-(400*valor);
return (posi);
}
```

I.3 Código del programa de simulación en Matlab 7.0

```
net=newff(minmax(p1), [4,2], {'logsig','purelin'}, 'trainrp');
net.trainParam.show=1e3;
net.trainParam.epochs=1e6;
net.trainParam.goal=1e-5;
[net, tr]= train(net, p1, t);
a=sim(net,p1)
```

Donde:

- **newff** crea un red Backpropagation feed-forward
- **minmax (p1)** toma el argumento p1 y regresa una matriz Rx2 de valores mínimos y máximos para cada renglón de p1.
- **[4,2]** quiere decir que la red neuronal tiene 4 neuronas en la capa oculta y 2 neuronas en la capa de salida.
- **logsig** es la función de transferencia en la capa oculta de la red neuronal y es del tipo sigmoidea logarítmica. Logsig toma como entrada una matriz de SxQ elementos y retorna cada elemento de la matriz comprimido entre 0 y 1.
- **purelin** es la función de transferencia en la capa de salida y es una función de transferencia del tipo lineal, esta toma una matriz de entrada de SxQ elementos y retorna los mismos valores.
- **trainrp** es una función de entrenamiento de la red que actualiza los valores de los pesos y “bias” de acuerdo al algoritmo Backpropagation resilient.
- **net.trainParam.show=1e3;** es el número de épocas entre despliegue.
- **net.trainParam.epochs=1e6;** número máximo de épocas en el entrenamiento.
- **net.trainParam.goal=1e-5;** desempeño del resultado

Apéndice II

Tabla completa de valores obtenidos para el desarrollo del proyecto

TABLA DE DATOS PARA EL MOTOR MODELO (MM)						
SENTIDO DE GIRO EN DIRECCIÓN POSITIVA						
	CODIFICADOR			PWM		
rps	T(μs)	AP(μs)	SIGNO	T(μs)	AP(μs)	SIGNO
1	2790	1130	1	85,4	6	1
	3100	1240	1	85,4	7,4	1
2	1580	600	1	85,4	9,4	1
	1810	710	1	85,2	10	1
3	1100	400	1	85,4	13,4	1
	1144	444	1	85,4	13,4	1
4	864	344	1	85,4	15,4	1
	820	332	1	85,4	15,4	1
5	666	270	1	85,2	18	1
	678	266	1	85,6	19,2	1
6	542	218	1	85,4	22,7	1
	548	214	1	85,4	21,3	1
7	484	190	1	85,4	24,7	1
	462	184	1	85,4	24,7	1
8	404	156	1	85,4	26,6	1
	410	158	1	85,4	29,3	1
9	372	150	1	85,4	30,1	1
	374	148	1	85,4	30,7	1
10	332	130	1	85,4	34	1
	331	131	1	85,4	33,3	1
11	300	119	1	85,4	36,7	1
	301	121	1	85,4	37,3	1
12	279	110	1	85,4	41,4	1
	277	108	1	85,4	40,1	1
13	256	101	1	85,4	43,4	1
	257	101	1	85,4	42,7	1
14	240	97	1	85,4	46	1
	237	93	1	85,4	46	1
15	221	87	1	85,4	48,8	1
	218	84	1	85,4	47,4	1
16	210	83	1	85,4	52	1
	208	81	1	85,4	51,4	1
17	197	79	1	85,4	54,6	1
	198	78	1	85,4	54	1
18	186	71	1	85,4	57,2	1
	184	74	1	85,4	58,2	1
19	175	68	1	85,4	60	1
	176	68	1	85,4	60,8	1
20	168	67	1	85,4	63,4	1
	167	67	1	85,4	64	1
21	160	64	1	85,4	66,2	1
	159	62	1	85,4	66,8	1

22	152	59	1	85.4	70.2	1
	152	59	1	85.4	70	1
23	146	57	1	85.4	72	1
	144	57	1	85.4	72	1
24	139	55	1	85.4	74.6	1
	140	56	1	85.4	75.2	1
25	133	52	1	85.4	78	1
	134	52	1	85.4	78	1
26	128	50	1	85.4	82.2	1
	129	50	1	85.4	80.8	1
27	124	49	1	85.4	83.4	1
	124	47	1	85.4	84.6	1

SENTIDO DE GIRO DEL EJE EN DIRECCIÓN NEGATIVA

CODIFICADOR				PWM		
rps	T(μs)	AP(μs)	SIGNO	T(μs)	AP(μs)	SIGNO
1	2790	1130	0	85,4	6	0
	3100	1240	0	85,4	7,4	0
2	1580	600	0	85,4	9,4	0
	1810	710	0	85,2	10	0
3	1100	400	0	85,4	13,4	0
	1144	444	0	85,4	13,4	0
4	864	344	0	85,4	15,4	0
	820	332	0	85,4	15,4	0
5	666	270	0	85,2	18	0
	678	266	0	85,6	19,2	0
6	542	218	0	85,4	22,7	0
	548	214	0	85,4	21,3	0
7	484	190	0	85,4	24,7	0
	462	184	0	85,4	24,7	0
8	404	156	0	85,4	26,6	0
	410	158	0	85,4	29,3	0
9	372	150	0	85,4	30,1	0
	374	148	0	85,4	30,7	0
10	332	130	0	85,4	34	0
	331	131	0	85,4	33,3	0
11	300	119	0	85,4	36,7	0
	301	121	0	85,4	37,3	0
12	279	110	0	85,4	41,4	0
	277	108	0	85,4	40,1	0
13	256	101	0	85,4	43,4	0
	257	101	0	85,4	42,7	0
14	240	97	0	85,4	46	0
	237	93	0	85,4	46	0
15	221	87	0	85,4	48,8	0
	218	84	0	85,4	47,4	0
16	210	83	0	85,4	52	0
	208	81	0	85,4	51,4	0
17	197	79	0	85,4	54,6	0
	198	78	0	85,4	54	0
18	186	71	0	85,4	57,2	0
	184	74	0	85,4	58,2	0
19	175	68	0	85,4	60	0
	176	68	0	85,4	60,8	0
20	168	64	0	85,4	63,4	0
	167	67	0	85,4	64	0
21	160	64	0	85,4	66,2	0
	159	62	0	85,4	66,8	0
22	152	59	0	85,4	70,2	0
	152	59	0	85,4	70	0
23	146	57	0	85,4	72	0
	144	57	0	85,4	72	0
24	139	53	0	85,4	74,6	0

	140	56	0	85,4	75,2	0
25	133	52	0	85,4	78	0
	134	53	0	85,4	78	0
26	128	50	0	85,4	82,2	0
	129	50	0	85,4	80,8	0
27	124	49	0	85,4	83,4	0
	124	47	0	85,4	84,6	0
28	-	-	-	-	-	-

TABLA DE DATOS PARA EL MOTOR APRENDIZ (MA)

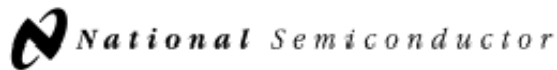
SENTIDO DE GIRO EN DIRECCIÓN POSITIVA						
CODIFICADOR			PWM			
rps	T(μs)	AP(μs)	SIGNO	T(μs)	AP(μs)	SIGNO
1	7520	3640	1	85.2	5	1
	7280	3800	1	85.4	5	1
2	4220	2140	1	85.4	5.9	1
	3860	1490	1	85.4	6	1
3	3360	1720	1	85.4	6.8	1
	3210	1650	1	85.4	7	1
4	2440	1240	1	85.4	7.7	1
	2490	1270	1	85.4	7.4	1
5	1970	1000	1	85.4	8.6	1
	2060	1040	1	85.4	8.7	1
6	1640	840	1	85.4	9.5	1
	1740	890	1	85.4	85.4	1
7	1430	720	1	85.4	10.4	1
	1490	450	1	85.4	10.7	1
8	1270	720	1	85.4	11.3	1
	1240	630	1	85.4	11.3	1
9	1104	564	1	85.4	12.2	1
	1108	564	1	85.4	12	1
10	996	504	1	85.4	13.1	1
	1004	512	1	85.4	13.3	1
11	928	472	1	85.4	14	1
	912	464	1	85.4	14	1
12	828	420	1	85.4	14.9	1
	836	424	1	85.4	14.6	1
13	772	392	1	85.4	15.8	1
	780	396	1	85.4	16	1
14	716	364	1	85.4	16.7	1
	718	356	1	85.4	16.6	1
15	668	340	1	85.4	17.6	1
	668	340	1	85.4	18	1
16	624	316	1	85.4	18.5	1
	624	316	1	85.4	18.6	1
17	560	280	1	85.4	19.4	1
	588	300	1	85.4	19.4	1
18	556	282	1	85.4	20.3	1
	556	282	1	85.4	20	1
19	528	268	1	85.4	21.2	1
	526	268	1	85.4	21.4	1
20	502	254	1	85.4	22.1	1
	500	252	1	85.4	22.2	1
21	478	242	1	85.4	23	1
	478	242	1	85.4	22.8	1
22	456	232	1	85.4	23.9	1
	460	234	1	85.4	24	1
23	432	222	1	85.4	24.8	1
	434	220	1	85.4	24.6	1
24	418	212	1	85.4	25.7	1
	418	212	1	85.4	25.4	1
25	402	204	1	85.4	26.6	1
	400	202	1	85.4	26.8	1
26	386	196	1	85.4	27.5	1
	386	194	1	85.4	27.4	1
27	370	188	1	85.4	28.4	1

	370	188	1	85.4	28.6	
TABLA DE DATOS PARA EL MOTOR APRENDIZ (MA)						
SENTIDO DE GIRO EN DIRECCIÓN NEGATIVA						
	CODIFICADOR			PWM		
rps	T(μs)	AP(μs)	SIGNO	T(μs)	AP(μs)	SIGNO
1	7520	3640	0	85.2	5	0
	7280	3800	0	85.4	5	0
2	4220	2140	0	85.4	5.9	0
	3860	1490	0	85.4	6	0
3	3360	1720	0	85.4	6.8	0
	3210	1650	0	85.4	7	0
4	2440	1240	0	85.4	7.7	0
	2490	1270	0	85.4	7.4	0
5	1970	1000	0	85.4	8.6	0
	2060	1040	0	85.4	8.7	0
6	1640	840	0	85.4	9.5	0
	1740	890	0	1	85.4	0
7	1430	720	0	85.4	10.4	0
	1490	450	0	85.4	10.7	0
8	1270	720	0	85.4	11.3	0
	1240	630	0	85.4	11.3	0
9	1104	564	0	85.4	12.2	0
	1108	564	0	85.4	12	0
10	996	504	0	85.4	13.1	0
	1004	512	0	85.4	13.3	0
11	928	472	0	85.4	14	0
	912	464	0	85.4	14	0
12	828	420	0	85.4	14.9	0
	836	424	0	85.4	14.6	0
13	772	392	0	85.4	15.8	0
	780	396	0	85.4	16	0
14	716	364	0	85.4	16.7	0
	718	356	0	85.4	16.6	0
15	668	340	0	85.4	17.6	0
	668	340	0	85.4	18	0
16	624	316	0	85.4	18.5	0
	624	316	0	85.4	18.6	0
17	560	280	0	85.4	19.4	0
	588	300	0	85.4	19.4	0
18	556	282	0	85.4	20.3	0
	556	282	0	85.4	20	0
19	528	268	0	85.4	21.2	0
	526	268	0	85.4	21.4	0
20	502	254	0	85.4	22.1	0
	500	252	0	85.4	22.2	0
21	478	242	0	85.4	23	0
	478	242	0	85.4	22.8	0
22	456	232	0	85.4	23.9	0
	460	234	0	85.4	24	0
23	432	222	0	85.4	24.8	0
	434	220	0	85.4	24.6	0
24	418	212	0	85.4	25.7	0
	418	212	0	85.4	25.4	0
25	402	204	0	85.4	26.6	0
	400	202	0	85.4	26.8	0
26	386	196	0	85.4	27.5	0
	386	194	0	85.4	27.4	0
27	370	188	0	85.4	28.4	0
	370	188	0	85.4	28.6	0

Apéndice III

Hojas de especificación.

III.1 Descripción General del LMD 18201.



October 2000

LMD18201 3A, 55V H-Bridge

General Description

The LMD18201 is a 3A H-Bridge designed for motion control applications. The device is built using a multi-technology process which combines bipolar and CMOS control circuitry with DMOS power devices on the same monolithic structure. The H-Bridge configuration is ideal for driving DC and stepper motors. The LMD18201 accommodates peak output currents up to 6A. Current sensing can be achieved via a small sense resistor connected in series with the power ground lead. For current sensing without disturbing the path of current to the load, the LMD18200 is recommended.

Features

- Delivers up to 3A continuous output
- Operates at supply voltages up to 55V
- Low $R_{DS(ON)}$ typically 0.33 Ω per switch

- TTL and CMOS compatible inputs
- No "shoot-through" current
- Thermal warning flag output at 145°C
- Thermal shutdown (outputs off) at 170°C
- Internal clamp diodes
- Shorted load protection
- Internal charge pump with external bootstrap capability

Applications

- DC and stepper motor drives
- Position and velocity servomechanisms
- Factory automation robots
- Numerically controlled machinery
- Computer printers and plotters

Functional Diagram

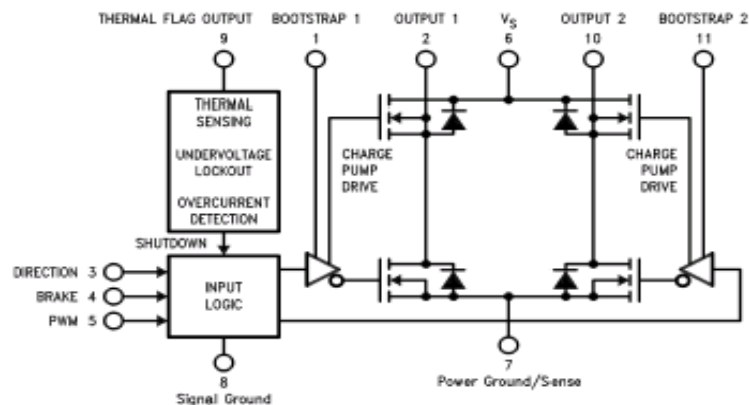


FIGURE 1. Functional Block Diagram of LMD18201

DS018201-1

III.2 Descripción del 74LS 244.

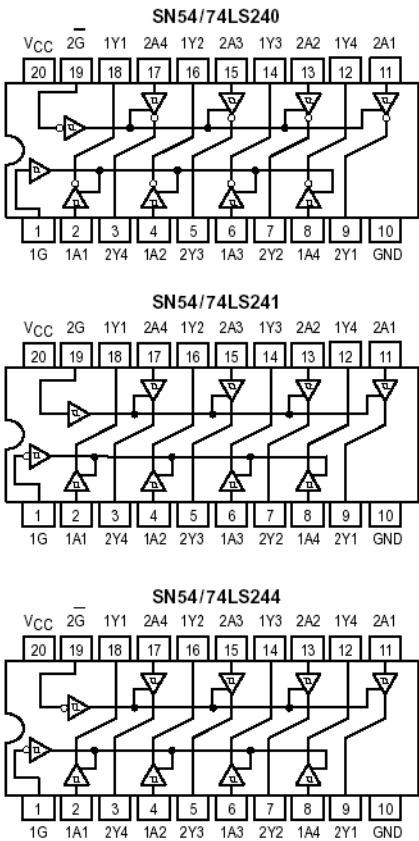


OCTAL BUFFER/LINE DRIVER
WITH 3-STATE OUTPUTS

The SN54/74LS240, 241 and 244 are Octal Buffers and Line Drivers designed to be employed as memory address drivers, clock drivers and bus-oriented transmitters/receivers which provide improved PC board density.

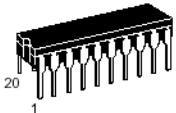
- Hysteresis at Inputs to Improve Noise Margins
- 3-State Outputs Drive Bus Lines or Buffer Memory Address Registers
- Input Clamp Diodes Limit High-Speed Termination Effects

LOGIC AND CONNECTION DIAGRAMS DIP (TOP VIEW)

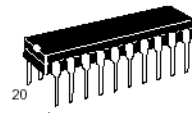


SN54/74LS240
SN54/74LS241
SN54/74LS244

OCTAL BUFFER/LINE DRIVER
WITH 3-STATE OUTPUTS
LOW POWER SCHOTTKY



J SUFFIX
CERAMIC
CASE 732-03



N SUFFIX
PLASTIC
CASE 738-03



DW SUFFIX
SOIC
CASE 751D-03

ORDERING INFORMATION

SN54LSXXXJ Ceramic
SN74LSXXXN Plastic
SN74LSXXXDW SOIC

Apéndice IV

Mapa anatómico del cerebro humano

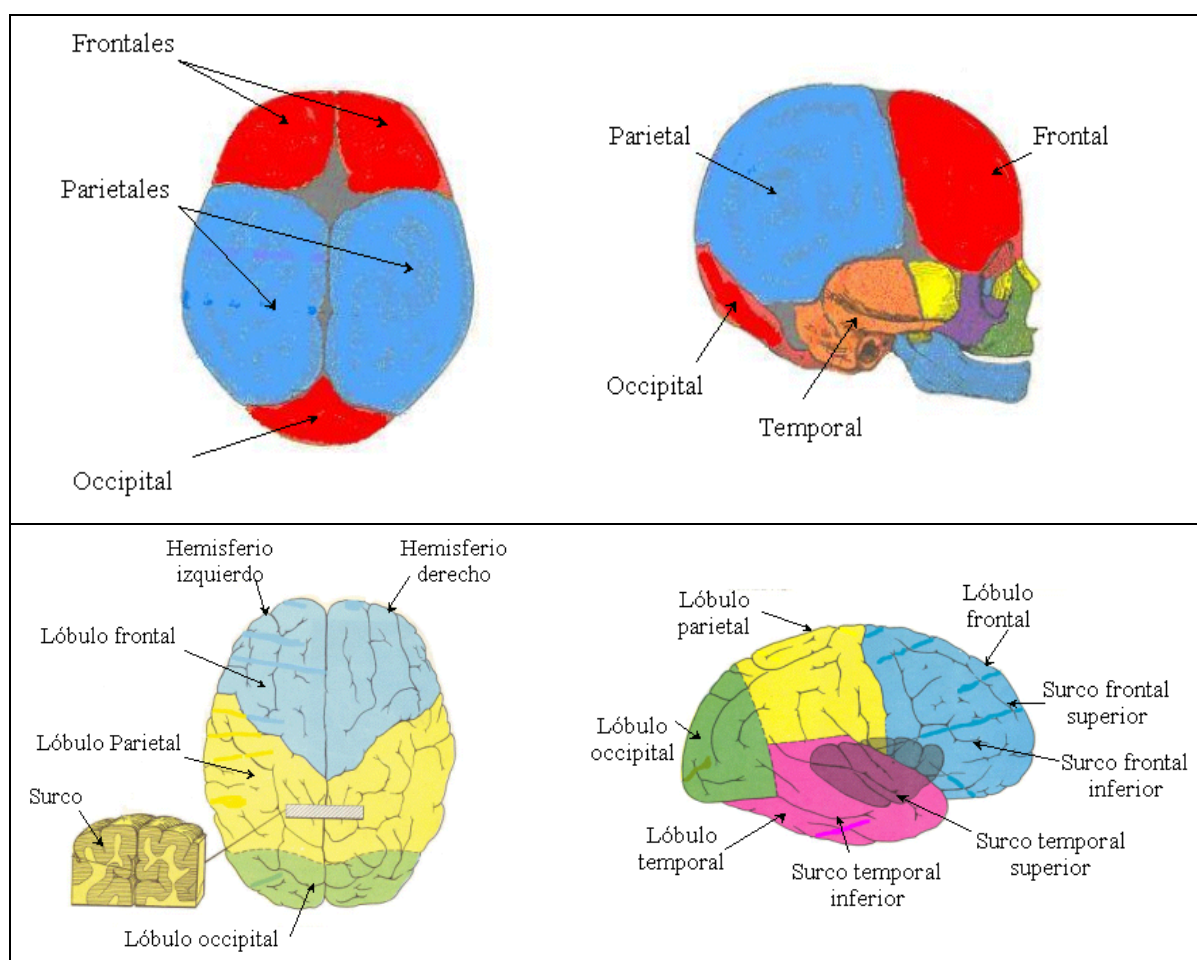


Figura IV.1. Mapa anatómico del cerebro.

Apéndice V

Fundamentos del hardware del sistema

El hardware de un sistema de control de motores de corriente directa, cuenta con varios dispositivos que son necesarios para realizar la tarea de control, entre los más importantes están: los propios motores, los dispositivos de censado de movimiento, los dispositivos de control de corriente, el microprocesador ó microcontrolador que realiza el control, y los dispositivos de comunicación entre el sistema de control y los motores. A continuación se describen brevemente los más importantes.

V.1. Motores de Corriente Directa.

Los motores eléctricos convierten la energía eléctrica en energía mecánica principalmente a través de movimiento rotatorio. Son usados frecuentemente en sistemas de control de posición o velocidad y se pueden clasificar en dos grandes grupos: los motores de Corriente Directa (CD) y los motores de Corriente Alterna (CA). Actualmente los motores de CD son los más usados en los sistemas de control modernos, gracias a su bajo costo y fácil control de velocidad y posición.

V.1.1. Principios básicos

El principio básico de funcionamiento de un motor de CD se puede explicar de la siguiente manera: cuando una corriente fluye a través de un circuito de alambre conductor que puede rotar y se encuentra dentro de un campo magnético, el circuito experimentará una fuerza en aquellas partes que se encuentran en ángulo recto al campo magnético, haciendo que el circuito gire hasta alcanzar una posición vertical (Figura V.1). Para que el circuito continúe su giro, es necesario que la corriente que fluye a través de él invierta su flujo.

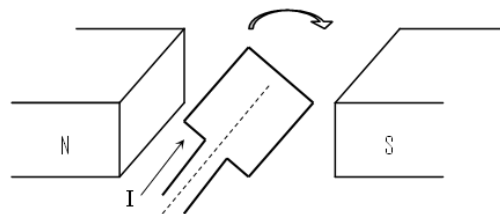


Figura V.1. Principio básico de funcionamiento de un motor de CD.

En un motor de CD real el circuito de alambre, se transforma en un conjunto de bobinas montadas en las ranuras de un material magnético cilíndrico llamado armadura o rotor, Figura V.2 (a). La armadura, se coloca dentro del campo magnético producido por los polos del campo y se encuentra libre para girar. Cada bobina de la armadura es conectada a través de contactos eléctricos a los segmentos adyacentes de un anillo segmentado llamado conmutador. Los contactos eléctricos se hacen a través de contactos de carbón llamados escobillas. El conmutador invierte la corriente de cada bobina, según la armadura se mueva entre los polos del campo, esto para asegurar que la armadura siga girando en la misma dirección. La dirección de rotación de un motor de CD se puede invertir invirtiendo la corriente de la armadura o la corriente del campo. Un motor que utiliza esta tecnología se llama motor de CD con escobillas. El problema que se tiene cuando se utiliza un motor de CD con escobillas es que las escobilla deben ser cambiadas con frecuencia y la armadura debe ser renovada continuamente. Para evitar este problema, se han diseñado los motores de CD sin escobillas (Figura V.2 (b)).

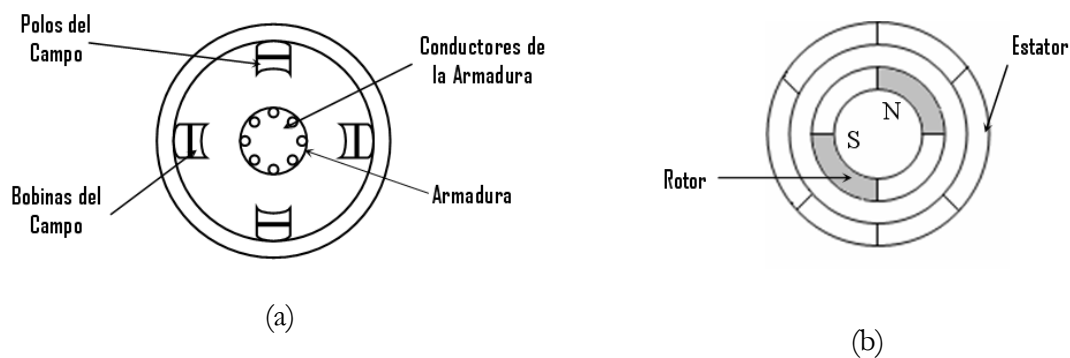


Figura V.2. (a) Motor de cuatro polos, el campo magnético es producido por la corriente que fluye en las bobinas.
(b) Motor de CD de imán permanente sin escobillas.

Los motores de CD sin escobillas con imán permanente están constituidos por dos partes principalmente: un imán permanente o rotor y una serie de bobinas llamadas estator. En un motor de este tipo, la corriente que llevan las bobinas es fija y la del imán es variable. La corriente para el estator es electrónicamente conmutada por transistores en secuencia a través de las bobinas. La conmutación será controlada por la posición del rotor dado que el estator es siempre forzado a actuar sobre el imán causando que este gire en la misma dirección. Los sensores de efecto Hall son usados generalmente para censar la posición del rotor e iniciar la conmutación.

Control de motores de CD

La velocidad de un motor depende de la corriente que fluye por las bobinas de la armadura (o del rotor, en el caso de un motor sin escobillas), de esta manera el control de velocidad se puede hacer variando de alguna manera el voltaje aplicado a la armadura. Para realizar esto, una

alternativa es utilizar un circuito electrónico que permita realizar el manejo del voltaje para obtener dicha variación.

Algunas veces el control de velocidad es hecho a través de señales provenientes de algún microprocesador, ya sea de propósito general o de propósito específico. En este caso la señal analógica para el manejo del motor deberá provenir de alguna forma de Convertidor Digital-Analógico (DAC). Aunque pareciera ser una buena alternativa, esto agregaría más costo al diseño y un alto consumo de potencia, reduciendo su fiabilidad. Para evitar esta situación, comúnmente se utiliza PWM, para variar el ancho del pulso de una señal digital de entrada al circuito que maneja el voltaje de entrada al motor.

Control de motores por medio de microprocesadores

Cuando se utiliza un microprocesador para realizar el control de movimiento de un motor de CD, se puede escoger entre un microprocesador de propósito general o uno de propósito específico. De cualquier manera, en un sistema de control de éste tipo, el microprocesador funciona como el cerebro del sistema, ya que puede digitalmente controlar la velocidad del motor mediante el monitoreo de las líneas de retroalimentación, si existen, y el manejo de las líneas de salida.

El esquema para el control de un motor de CD por medio de un microprocesador se muestra en la Figura V.3. En (a) se muestra el esquema para un sistema de control de lazo abierto usando un DAC para transformar la señal digital de control proveniente del microprocesador en una señal analógica para el circuito que maneja la corriente. También se muestra el esquema usando PWM. En (b) se muestra el esquema para un sistema de control de lazo cerrado en donde se usan un DAC y un Convertidor Analógico Digital (ADC) para hacer la correspondiente transformación de las señales involucradas en el control del sistema, de digital a analógica y de analógica a digital. En (c) se utiliza un DAC para realizar la transformación de la señal de control, mientras que en la retroalimentación se utiliza un codificador, el cual como sabemos proporciona señales digitales al microprocesador. (d) muestra el sistema de control con elementos que manejan señales puramente digitales: La señal PWM y el codificador.

En el mercado existen microcontroladores como el LM629 de National Semiconductor, y el 83CF51FA de Intel, que son microcontroladores de propósito específico para el control de Motores de CD. Para el control de la corriente que fluye por el motor se puede utilizar el CI LMD 18201, que es un puente H. En la siguiente sección se da una breve descripción del microcontrolador LM629, así como del CI LMD 18201.

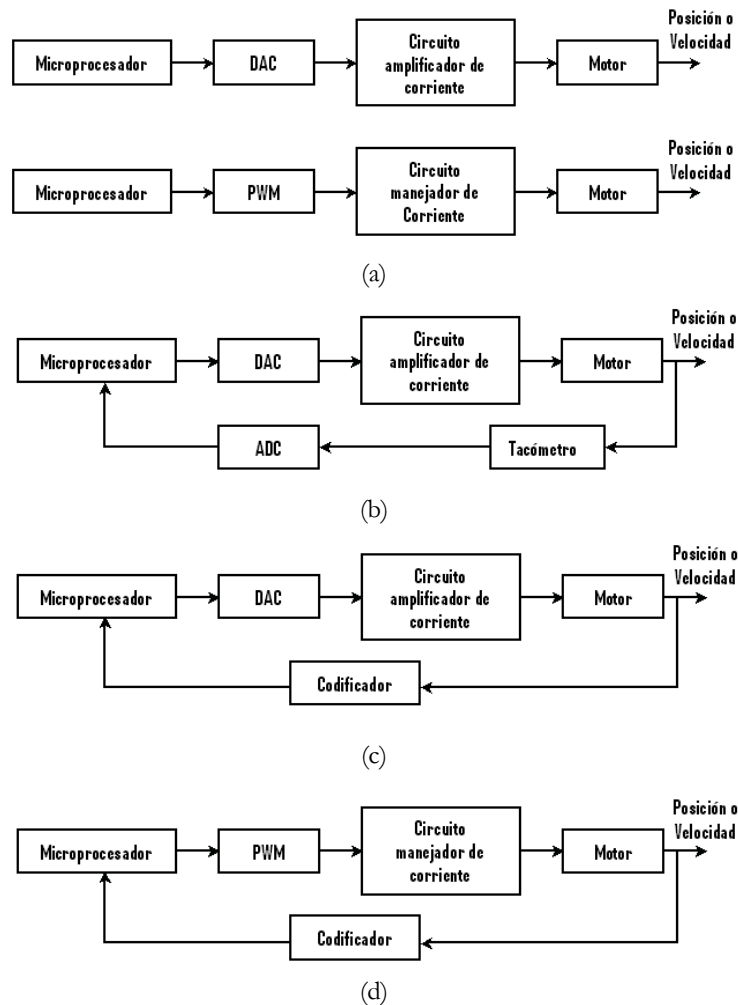


Figura V.3. Esquemas de control para un motor de CD.

V.2. El Microcontrolador LM629.

El LM629 es un microprocesador de propósito específico para el control de movimiento de motores de CD con y sin escobillas (Figura V.4). Es un microcontrolador periférico que incorpora en un dispositivo todas las funcionalidades de un sistema de control como control de movimiento y muestreo. Tiene un decodificador de posición, una sumatoria, un filtro digital de ciclo de compensación PID, y un generador de perfil de trayectoria. Para iniciar un movimiento, el microprocesador anfitrión descarga los valores de aceleración, velocidad y posición objetivo al generador de trayectoria del LM629. En cada intervalo de muestra, estos valores son usados para calcular una nueva posición, la cual es alimentada a la sumatoria. La posición actual del motor es determinada a partir de las señales de salida de un codificador óptico incremental acoplado al eje del motor que controla. La posición actual del motor es decodificada por el decodificador de posición del LM629, y es alimentada a la otra entrada de la sumatoria y

sustraída de la posición demandada para formar la señal de error de entrada para el circuito compensador de control. El compensador es de la forma de un filtro PID de tres términos (proporcional, integral, derivativo), implementado por un filtro digital.

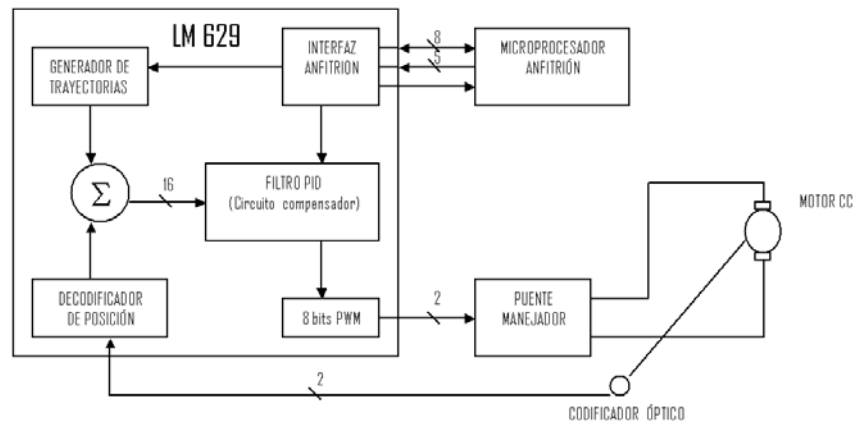


Figura V.4. Diagrama del control de un motor de CD a través de un microcontrolador LM629.

V.3. El circuito LMD 18201

El LMD18201 es un puente-H de 3A, que opera con fuentes de voltaje de hasta 55 V, y tiene entradas compatibles con CMOS y TTL. Está diseñado para aplicaciones en control de movimiento y es ideal para manejo de motores de CD y de pasos. Adapta picos por arriba de 6A. Además de ser ideal en el manejo de motores, el LM18201 se usa en procesos de automatización, control numérico de maquinaria, en impresoras y en plotters.

V.4. Sensores

Un sensor es un dispositivo eléctrico y/o mecánico que transforma una magnitud física en un valor que puede ser medido. La información que es suministrada al sensor debe ser preparada de manera que pueda ser tratada y procesada. Así, el sensor provee una señal que es interpretable para el usuario o el sistema que lo utiliza. Las características de construcción de un sensor lo harán idóneo para una aplicación determinada y por ello existe una gran variedad de tipos de sensores dentro de los cuales se encuentran:

- Sensores de desplazamiento y rotación
- Sensores de velocidad
- Sensores de proximidad

-
- Sensores de fuerza
 - Sensores de aceleración
 - Sensores de luz
 - Sensores neumáticos
 - Sensores ópticos

En lo que se refiere a este capítulo, sólo se describirán los dos primeros tipos de sensores.

Sensores de desplazamiento y rotación

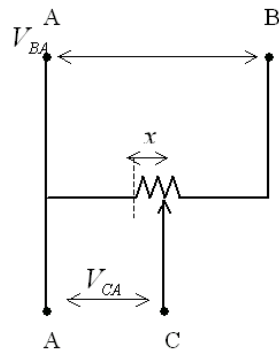
Los sensores de desplazamiento y rotación permiten medir, con respecto a un punto de referencia, la distancia lineal o el ángulo que se ha movido un elemento de un sistema. Para ello cuentan con un sistema de medición que les permite convertir un desplazamiento, ya sea lineal o rotacional, a un nivel de tensión que permitirá evaluar la cantidad de desplazamiento. Dentro de los sensores de desplazamiento y rotación más conocidos se encuentran:

- Potenciómetros
- Sensores capacitivos
- Transformadores diferenciales
- Sensores inductivos
- Sensores basados en efecto Hall
- Codificadores Ópticos

A continuación se describe el principio de funcionamiento de cada uno, así como sus características de aplicación.

Potenciómetros

Los potenciómetros utilizados como sensores, pueden determinar movimientos lineales o angulares. Para un potenciómetro lineal, como el mostrado en la Figura V.5, la diferencia de potencial en V_{Ac} está en función de la posición de x . Para determinar un desplazamiento se debe calibrar el potenciómetro, de manera que, por cada distancia de desplazamiento exista una diferencia de potencial proporcional al desplazamiento en x . Las ecuaciones (1) y (2) muestran la relación de la diferencia de potencial, con respecto a la variación de la resistencia, en un potenciómetro lineal y rotacional respectivamente.



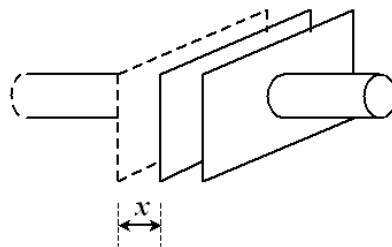
$$\frac{V_{CA}}{V_{BA}} = \frac{R_{CA}}{R_{BA}} \quad (1)$$

$$V_S = \frac{\theta}{\theta_{Max}} \quad (2)$$

Figura V.5. Principio de funcionamiento de un potenciómetro lineal.

Sensores capacitivos

Los capacitores que se utilizan como sensores deben ser de tipo variable. Medir un desplazamiento con un dispositivo de éste tipo, provocará un desplazamiento en algún elemento del capacitor influyendo directamente en la capacidad del mismo. En un sensor capacitivo como el mostrado en la Figura V.6, la variación de la capacidad estará fundamentada en la propiedad de que el potencial que almacena entre sus placas es inversamente proporcional a la distancia que las separa. La ecuación (3) muestra esta propiedad, en donde ϵ_a es la constante dieléctrica que separa a las dos placas, S es el área de cada una de las placas del condensador y d la distancia que separa a las placas.



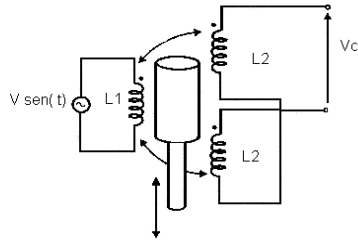
$$C = \epsilon_a \frac{S}{d} \quad (3)$$

Figura V.6. Sensor Capacitivo.

Transformadores diferenciales

Los transformadores diferenciales, aprovechan el efecto de inducción entre bobinas para la detección de movimientos lineales de precisión. Como se puede ver en la Figura V.7, este tipo de sensor cuenta con una bobina primaria L_1 , dos bobinas secundarias L_2 y un núcleo ferromagnético. Cuando circula una corriente alterna a través de la bobina primaria con una intensidad y tensión determinada, se induce una corriente alterna de intensidad y tensión distinta en las bobinas secundarias, lo que hará que el núcleo ferromagnético se mueva, éste movimiento

es precisamente el que permitirá a este tipo de sensores determinar un movimiento.



$$\frac{V_1}{V_2} = \frac{i_1}{i_2} \quad (3)$$

Figura V.7. Esquema de un transformador diferencial.

Este tipo de sensor es de rápido tiempo de respuesta, baja histéresis, alta resolución y linealidad, aunque no se pueden medir desplazamientos muy grandes.

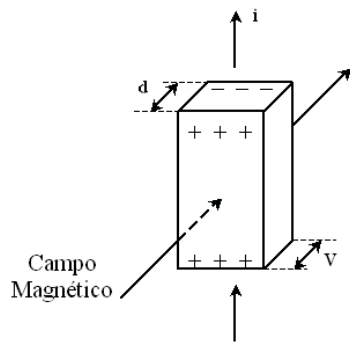
Sensores inductivos

El principio de funcionamiento de los sensores inductivos está basado en la interacción de conductores metálicos con un campo magnético. Este tipo de sensores está formado por una bobina, un objeto ferromagnético y un imán permanente. Cuando se sitúa un imán cerca de una bobina y se desplaza un objeto ferromagnético entre ambos, ocasionará una variación en el flujo magnético a través de la bobina originándose una fuerza electromotriz inducida. Como ejemplo de aplicación de éste principio están los sensores “sincros” y los “resolvers”. Los sensores inductivos son utilizados como sensores de posición, desplazamiento y presencia.

Sensores basados en efecto Hall

Los sensores de efecto Hall utilizan la presencia de un campo magnético para generar cambios en la corriente que circula por un semiconductor y así proporcionar un determinado nivel de tensión. En la figura V.8, se muestra una placa de metal semiconductor tipo “n” que conduce una corriente eléctrica. Cuando el semiconductor es inmerso en un campo magnético, se genera un campo eléctrico, debido a que las cargas en el semiconductor se polarizan. El campo eléctrico generado compensa al campo magnético que se ejerce sobre el semiconductor. La polarización de cargas en los extremos del semiconductor genera una diferencia de potencial V , cuyo valor está dado por la ecuación (5). K_H es una constante denominada coeficiente Hall, B_f es la densidad de flujo magnético, i es la intensidad de la corriente que fluye por el semiconductor y d es el grosor del conductor.

En este tipo de sensores la variación en la intensidad del flujo magnético en presencia de una intensidad de corriente constante, provocará una variación en la diferencia de potencial a consecuencia del efecto Hall, pudiendo determinar de ésta manera un desplazamiento o proximidad de un objeto.



$$V = K_H \frac{B_f \cdot i}{d} \quad (5)$$

Figura V.8. Efecto Hall en un semiconductor tipo “n”.

Codificadores ópticos

Un codificador es un dispositivo que convierte un desplazamiento lineal o angular en señales digitales. Un codificador óptico utiliza información óptica para determinar la magnitud de un desplazamiento. Está formado por una fuente de luz, un diodo fotorreceptor y un disco óptico con una serie de marcas, ver figura V.9 (a). Cuando gira el disco, se generan una serie de pulsos de luz que son detectados por el fotorreceptor, el número de pulsos generados determinan la cantidad de grados girados. En consecuencia, la resolución del codificador estará dada por el número de marcas en el disco. Se tienen principalmente dos tipos de codificadores ópticos:

- Codificadores Absolutos
- Codificadores Incrementales.

Codificador absoluto

Un codificador absoluto genera un patrón de palabra único para toda posición. Las pistas de un codificador absoluto de disco, generalmente 4 o 6, son codificadas para generar a su salida código binario, BCD ó Gray. En la Figura V.9 (b), el disco óptico está marcado con zonas transparentes y opacas de manera que para cada posición angular, el fotorreceptor detectará una señal luminosa única que corresponde a esa posición. Los codificadores absolutos son más usados en aplicaciones donde los dispositivos estarán inactivos durante largos períodos de tiempo, existe un riesgo de caída de corriente o la posición de inicio es desconocida.

Codificador incremental

Un codificador incremental genera un pulso, por cada paso incremental, de manera que se puede determinar la cantidad de grados girados contando el número de pulsos generados por el codificador. Los codificadores incrementales se utilizan principalmente para determinar la velocidad de giro de, por ejemplo, del eje de un motor. Aunque también se pueden usar para medir velocidades lineales realizando la conversión necesaria de movimiento angular a lineal. Existen principalmente dos tipos de codificadores incrementales:

- Tacómetro.

- Codificador incremental de cuadratura.

Ambos son sensores específicos de movimiento angular y se describen a continuación.

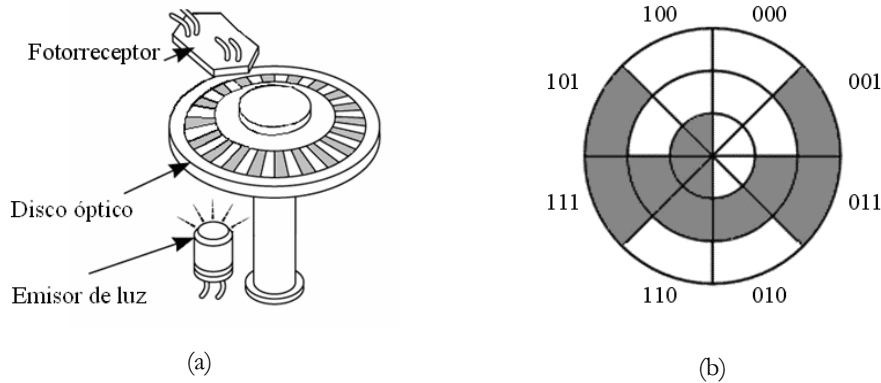


Figura V.9. (a) Partes principales de un codificador óptico. (b) Codificador absoluto.

Tacómetros

Los tacómetros son codificadores incrementales con una sola pista de marcas que generan una única señal de pulsos cuya frecuencia indica la velocidad de desplazamiento. La estructura básica de un tacómetro es parecida a la de un motor de corriente continua, pero se podría decir que presenta un comportamiento opuesto, ya que convierte movimiento rotacional (energía de rotación) en señales eléctricas (energía eléctrica).

Codificador incremental de cuadratura

Un codificador incremental de cuadratura codifica las rotaciones de un determinado eje como pulsos eléctricos. Cuenta con dos pistas y sectores desfasados 90°, ver Figura V.10. Por ello, utilizan dos canales de salida (generalmente nombrados A y B) para determinar una posición. Los dos canales en cuadratura indican posición y dirección de movimiento. De ésta manera, determinando el número de pulsos y la fase relativa de las señales A y B, se puede mantener una pista de la posición y dirección de movimiento. Algunos codificadores de cuadratura incluyen un tercer canal de salida, llamado “cero” o señal de referencia, la cual provee un sólo pulso por revolución. Este pulso puede ser usado para la determinación precisa de una posición. La resolución de un codificador incremental de cuadratura se especifica como un número de líneas, que indica el número de ciclos de la señal de salida por cada revolución del eje.

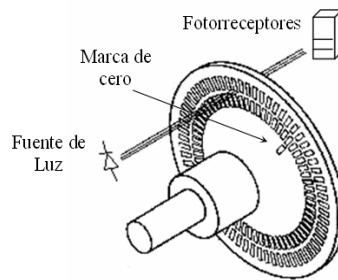


Figura V.10. Codificador incremental de cuadratura.

V.5. El Temporizador de Intervalos Programable 8254

El 8254 es un CI que tiene dos funcionalidades, es contador y temporizador. Fue diseñado²³ por Intel, para resolver algunos problemas comunes de control de tiempo en el diseño de Microcomputadoras. Al igual que otros dispositivos periféricos de la familia Intel es tratado por el software del sistema como un arreglo de puertos de entrada y salida. Además de resolver problemas en el control de tiempo, se pueden implementar otras funcionalidades con el 8254 como: reloj de tiempo real, contador de eventos, generador de tasa programable, generador de onda cuadrada, multiplicador de tasa binaria y generador de forma de onda compleja, entre otras aplicaciones.

Descripción general

El 8254 está constituido por tres contadores independientes de 16 bits (ver figura V.11), cada uno con la capacidad de manejar entradas de reloj de hasta 10MHz y de contar en binario o decimal codificado en binario (BCD).

Cada contador tiene una entrada de reloj (CLK), una entrada de compuerta (GATE) y una salida (OUT). La entrada CLK proporciona la frecuencia de reloj básica para el funcionamiento del contador, la entrada GATE permite controlar al contador en algunos modos de operación y la terminal OUT permite leer la salida del contador.

²³ El diseño original es el CI 8253, con las mismas características de construcción pero es un diseño menos versátil que el 8254. Los equipos PCXT están equipados con un CI 8253.

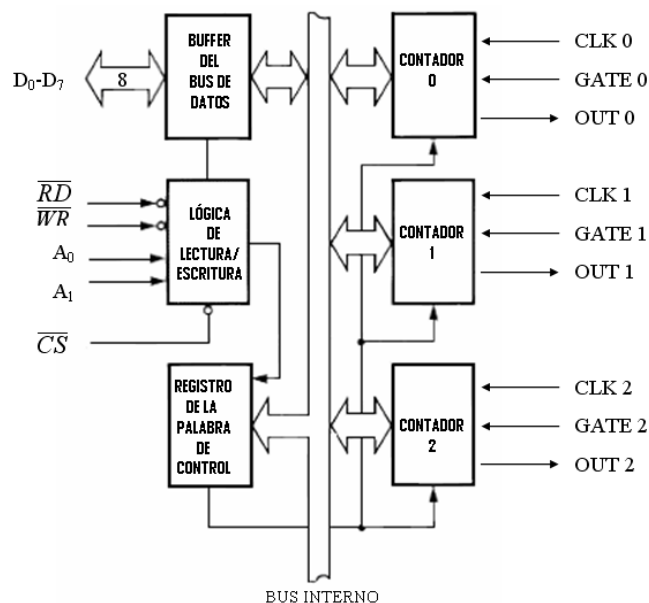


Figura V.11 Diagrama de Bloques del CI 8254

El *Buffer del Bus de Datos* es un buffer de 8 bits bidireccional de tres estados que comunica al 8254 con el bus del sistema. El bloque de *Lógica de Lectura/Escritura* acepta entradas del bus y genera señales de control para otros bloques funcionales del 8254. Las líneas A_0 y A_1 , selecciona uno de los tres contadores o el registro de la palabra de control para ser “leído de” o “escrito en” (Tabla V.1). Cuando la línea de entrada \overline{RD} está en nivel bajo el CPU anfitrión está leyendo uno de los tres contadores. Cuando la línea de entrada \overline{WR} está en nivel bajo el CPU está escribiendo una palabra de control o una cuenta inicial. \overline{RD} y \overline{WR} son ignorados si \overline{CS} tiene un nivel alto.

Tabla V.1. Entradas A_1 y A_0 para seleccionar alguno de los tres contadores o el registro de la palabra de control en el CI 8254.

A_1	A_0	Selecciona
0	0	Contador 0
0	1	Contador 1
1	0	Contador 2
1	1	Registro de la Palabra de Control

El *Registro de la Palabra de Control* es seleccionado cuando $A_1, A_0 = 11$, si el CPU hace una operación de escritura al 8254, el dato es almacenado en el *Registro de la Palabra de Control* y es interpretada como una palabra de control usada para definir la operación de los contadores.

Programación del 8254

Al inicio de su funcionamiento el 8254 está en un estado indefinido, esto es, el modo de operación, el valor de cuenta y la salida de todos los contadores contienen valores aleatorios.

Para que cada *contador* opere es necesario programarlo de manera individual escribiendo una *palabra de control* y una *cuenta inicial*.

La palabra de control

La *palabra de control* (ver Figura V.12) permite seleccionar al contador, el tipo de operación (lectura o escritura), el modo de operación y el tipo de cuenta (binaria o BCD).

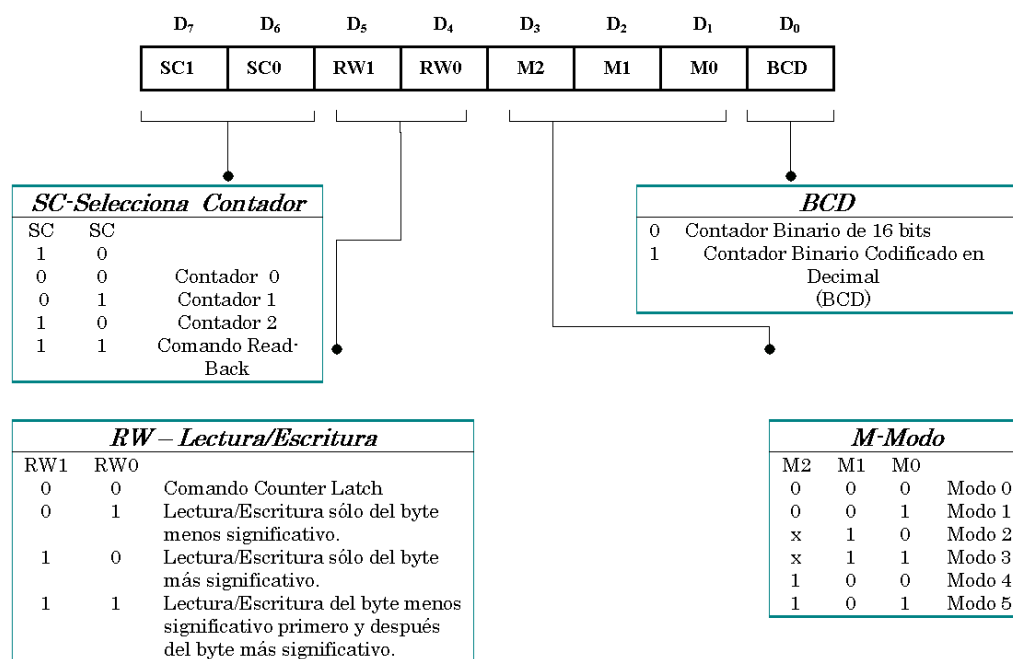


Figura V.12. Formato de la palabra de control para el 8254. A₁, A₀ = 11; \overline{CS} = 0; \overline{RD} = 1; \overline{WR} = 0.

Operaciones de lectura

Existen tres posibles métodos para leer los contadores del 8254: una operación simple de lectura, el comando *Counter Latch* y el comando *Read-Back*.

Operación simple de lectura

El primer método consiste en una simple operación de lectura al Contador seleccionado por las entradas A₁, A₀. La entrada CKL del Contador seleccionado deberá ser inhibida usando la entrada GATE, o un circuito lógico externo, para evitar que el Contador sea leído en medio del proceso de actualización de la cuenta, lo cual daría un valor de cuenta erróneo.

Comando Counter Latch

Al igual que una palabra de control, este comando es escrito al *Registro de la Palabra de Control*, el cual es seleccionado cuando A₁, A₀ = 11, también como en una palabra de control, se selecciona uno de los tres contadores por medio de los bits SC0 y SC1 pero para distinguir a este comando de una palabra de control, los bits D₅ y D₄ deben estar en cero. El resto de los bits van a cero para mejorar la compatibilidad con futuras versiones del CI (ver Figura V.13).

Comando Read-Back

El comando Read-Back permite al usuario leer el valor actual de la cuenta, el modo programado y el estado actual de la salida OUT y de la bandera de Cuenta Nula del Contador seleccionado. El comando es escrito en el *Registro de la Palabra de Control* y tiene el formato mostrado en la Figura 4.13. El comando aplica a los contadores seleccionados. Para seleccionar alguno de ellos o todos se pone el bit correspondiente al contador a 1.

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
	SC2	0	0	0	0	0	0

SC0	SC1	Contador
0	0	0
0	1	1
1	0	2
1	1	Comando Read-Back

$A_1, A_0 = 11$; $\overline{CS} = 0$; $\overline{RD} = 1$; $\overline{WR} = 0$.
 SC1, SC0: Especifican el contador involucrado.
 D₅, D₄: 00 elige al comando Counter Latch.

Figura V.13. Formato del comando Counter Latch del 8254.

Operaciones de escritura

Debido a que la programación del 8254 es muy flexible, para realizar operaciones de escritura se debe tener en cuenta dos cosas: 1) Para cada contador se debe escribir primero la *Palabra de Control*, después la cuenta inicial. 2) La cuenta inicial deberá seguir el formato de cuenta especificado en la *Palabra de Control*.

Dado que el registro de la palabra de control y los tres contadores tienen direcciones separadas, y cada *Palabra de Control* especifica al Contador implicado, no se requiere de una secuencia especial de instrucciones.

Un nuevo valor de cuenta inicial se puede cargar en un Contador en cualquier momento, sin que esto afecte el modo en que ha sido programado.

$A_1, A_0 = 11$ $\overline{CS} = 0$ $\overline{RD} = 1$ $\overline{WR} = 0$

D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
1	1	\overline{COUNT}	\overline{STATUS}	CNT 2		CNT 0	0

D₅=0, mantener la cuenta del (de los) contador(es) seleccionado(s).
 D₄=0, mantener el Status del (de los) contador(es) seleccionado(s).
 D₃=1, selecciona el contador 2
 D₂=1, selecciona el contador 1
 D₁=1, selecciona el contador 0
 D₀, está reservado para expansión futura; puede ser cero.

Figura V.14. Formato del comando Read-Back del 8254.

Modos de operación.

Existen seis modos de operación, del *Modo 0* al *Modo 5*, con los cuales cada uno de los contadores del 8254 puede ser programado.

Modo 0. Interrupción al final de la cuenta.

El *Modo 0* (Figura 4.14), es usado típicamente para contar eventos. Cuando la palabra de control y la cuenta inicial son escritas, OUT toma el valor 0 lógico y permanece así hasta que el contador llega a cero, en ese momento OUT toma el valor 1 lógico, y no vuelve a bajar sino hasta que una nueva cuenta o una nueva Palabra de Control en Modo 0 es escrita en el Contador. Después de que la Palabra de control y la cuenta inicial han sido escritas al Contador, la cuenta inicial será cargada hasta el siguiente pulso de reloj. Dado que este pulso de reloj no decrementa la cuenta inicial, OUT permanecerá en un nivel bajo N+1 ciclos de reloj.

Modo 1. Multivibrador monoestable hardware.

En este modo la entrada GATE dispara un pulso único que activa al contador, haciendo que OUT permanezca en nivel bajo hasta que el contador termine la cuenta, OUT tomará de nuevo un nivel alto hasta un pulso de reloj después del siguiente disparo.

Modo 2. Generador de tasa.

Este modo funciona como un contador dividido por N. Esto es usado típicamente para generar una interrupción de Reloj de Tiempo Real. OUT inicialmente está en nivel alto. Cuando la cuenta inicial se ha decrementado hasta llegar a uno OUT toma un nivel bajo por un ciclo de reloj, y enseguida vuelve a tomar un nivel alto. El Contador recarga la cuenta inicial y el proceso se repite indefinidamente.

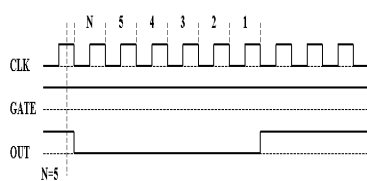


Figura V.15. Modo de operación 0

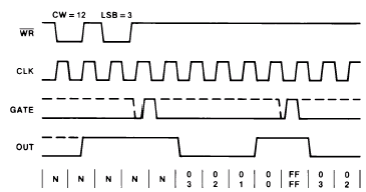


Figura V.16 Modo de operación 1

Modo 3. Generador de Onda Cuadrada.

En este modo OUT inicialmente está en nivel alto, al transcurrir la primera mitad de la cuenta, OUT toma un nivel bajo hasta que termina la cuenta. El Modo 3 es periódico y la secuencia anterior es repetida indefinidamente. Una cuenta inicial N resulta en una onda cuadrada de N ciclos de reloj.

Modo 4. Pulso Strobe Iniciado por software.

OUT permanece inicialmente en nivel alto. Cuando la cuenta inicial expira OUT toma un nivel bajo por un ciclo de reloj y vuelve a tomar un nivel alto. La secuencia de conteo es disparada escribiendo la cuenta inicial. GATE=1, habilita la cuenta; GATE=0, deshabilita la cuenta. GATE no tiene efecto sobre OUT.

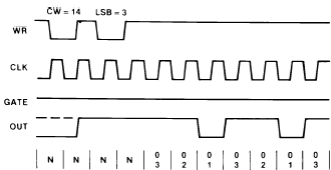


Figura V.17 Modo de operación 2.

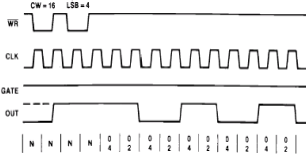


Figura V.18 Modo de operación 3.

Modo 5. Pulso Strobe Iniciado por Hardware.

En este modo, OUT inicialmente está en un nivel alto. La cuenta inicial es disparada por un flanco de subida de GATE. Cuando la cuenta expira, OUT toma un nivel bajo por un ciclo de reloj y luego sube. Después de escribir la palabra de control y la cuenta inicial el contador no es cargado hasta un pulso después del disparo.

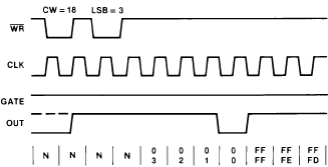


Figura V.19. Modo de operación 4

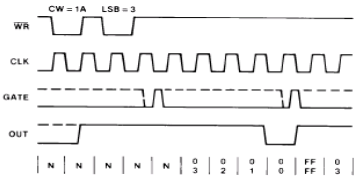


Figura V.20. Modo de operación 5

El 8254 en la computadora personal

Como se planteó al principio de ésta sección, el 8254 es un CI que fue diseñado para resolver algunos problemas de temporización en el diseño de microcomputadoras. La familia de microcomputadoras IBM PC utiliza al CI 8254 en los equipos AT y PS/2 para controlar ciertas funciones del sistema básicas como:

- Controlar las funciones de temporización internas del BIOS, como actualizar el reloj del sistema (aproximadamente cada 18.2 veces por segundo).
- Temporizar operaciones de refresco de la memoria.

-
- Proporcionar una fuente de temporización a los dispositivos del sistema.

El 8254 posee una interfaz de comunicación a través de los puertos de direcciones 40h al 43h, por lo que cada uno de los tres contadores puede ser accedido por medio de estos puertos. El puerto 40h es usado como canal de entrada/salida para el contador 0, el puerto 41h es usado como canal de entrada/salida para el contador 1, el puerto 42h es usado como canal de entrada/salida del contador 2 y el puerto 43h es usado en modo de control para los tres contadores, la *Palabra de Control* se envía a través de éste puerto.

Los contadores del 8254 están conectados a un reloj que oscila a una frecuencia de 1.193180 MHz. Cada uno de los tres contadores es usado para propósitos diferentes. El BIOS programa por defecto al contador 0 en Modo 3, para generar la interrupción básica de tiempo que actualiza al reloj del sistema; al contador 1 en Modo 2 para controlar el refresco de memoria; y el contador 2 está disponible para uso general, puede emplearse para temporizar y puede ser programado por el usuario.

V.6. El puerto paralelo

Los puertos de comunicación (entrada/salida) son de especial importancia en el diseño de una PC, ya que son el medio por el cual el CPU se comunica con otros dispositivos. Esto nos permite que a través de una PC podamos hacer tareas que requieran de transferencia de datos, como por ejemplo: adquisición de datos, automatización de procesos, comunicación de todo tipo de dispositivos electrónicos para realizar tareas de control, etc. Una PC puede direccionar hasta 64K puertos de comunicación, por lo que cada puerto tiene asignado un número que le da una ubicación dentro de un rango de memoria.

Existen dos formas básicas para la transmisión de datos en una PC: serial y paralela. En la transmisión de datos serial, los datos se envían en forma de bits que se transmiten uno a la vez desde o hacia la PC a través de un sólo cable. En la transmisión de datos paralela, los datos son transmitidos enviando múltiples bits a la vez a través de múltiples cables. La transmisión de datos puede ser unidireccional o bidireccional.

El Puerto Paralelo Estándar

El Puerto Paralelo Estándar (SPP)²⁴ es también conocido como el Puerto Paralelo Centronics. Centronics Data Computer Corporation desarrollo una interfaz de 36 pines a mitad de la década de los años 60 del siglo pasado, para hacer una conexión computadora-a-impresora paralela unidireccional de 8 bits. A partir de entonces la interfaz comenzó a ser usada ampliamente y su implementación sufrió múltiples variaciones, hasta la introducción de la IBM PC en 1981 que incluía una interfaz paralela con un conector de 25 pines, 8 líneas de datos unidireccionales, 4

²⁴ Siglas en inglés de Standard Parallel Port.

líneas de control y 5 líneas de estados. Esta implementación del puerto paralelo se convertiría con el tiempo en el estándar de facto del puerto paralelo en la industria de las computadoras.

Aunque originalmente el SPP fue diseñado para comunicación con impresoras, debido a su gran aceptación se utiliza también para conectar dispositivos cada vez más rápidos.

Desde su introducción, el puerto paralelo ha sufrido varias modificaciones para hacerlo más veloz. Actualmente se conocen cuatro tipos de puertos paralelos:

- Puerto Paralelo Standard.
- Puerto Paralelo PS/2.
- El EPP (Enhanced Parallel Port).
- El ECP (Extended Capability Port).

Direccionamiento

En principio, la dirección base del puerto paralelo depende del tipo de adaptador que incorpora el PC. En forma, al momento del arranque el BIOS crea en la memoria principal una tabla de datos con las direcciones base de cuatro posibles puertos paralelos, que se almacenan como 4 bytes contiguos a partir de la dirección de memoria 408 h. Durante el arranque el BIOS comprueba si hay puertos paralelos en las direcciones 3BC h, 378 h y 278 h y almacena la dirección base del cualquiera de ellos que haya sido encontrado. El Puerto Paralelo usa tres direcciones continuas usualmente dentro de éstos tres rangos de direcciones:

- 278 h-27F h
- 378 h-37F h
- 3BC h-3BF h

La primera dirección dentro del rango es la dirección base del puerto, por ejemplo, en el primer inciso la dirección base es 278h. Esta dirección también es llamada dirección del puerto y corresponde al registro de datos del puerto paralelo. Siguiendo con el ejemplo, la segunda dirección dentro del rango (dirección *base + 1*) es 279 h, y corresponde al registro de estados del puerto. La tercera dirección dentro del rango (dirección *base + 2*) corresponde al registro de control y su valor en 27A h. El procedimiento es el mismo para todo puerto paralelo localizado dentro de alguno de los rangos de memoria establecidos.

Registros

El SPP define tres registros²⁵ para manipular las líneas de datos, control y de estatus del puerto paralelo: el *registro de datos*, el *registro de estados* y el *registro de control* (Tabla V.2).

El *registro de datos* (D0-D7), ubicado en la dirección base es un registro de sólo escritura y mantiene el byte de datos escrito en la salida.

El *registro de estatus* es de sólo lectura y mantiene los estados lógicos de los bits S7 a S3 (BUSY, ACK, PError, Select Y FAULT) en el byte de entrada. S1 y S0 no se usan. S2 en algunos puertos

²⁵ Los registros del puerto paralelo también son llamados puertos.

indica el estado de la interrupción del puerto paralelo, cuando IRQ=0 ha ocurrido una interrupción y cuando IRQ= 1, no ha ocurrido ninguna interrupción. Debido a la configuración y al modo de operación, el *registro de estatus* puede ser usado para transferir datos desde un dispositivo periférico a la PC.

Tabla V.2. Registros del Puerto Paralelo Estándar

Registro	Bits del Registro							
Registro de Datos	7	6	5	4	3	2	1	0
Dirección Base	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀
Registro de Estatus	S ₇	S ₆	S ₅	S ₄	S ₃	S ₂	S ₁	S ₀
Dirección Base + 1	BUSY	ACK	PError	Select	FAULT	IRQ	R	R
Registro de Control	C ₇	C ₆	C ₅	C ₄	C ₃	C ₂	C ₁	C ₀
Dirección Base + 2	R	R	R	IRQEN	SelecIn	INIT	AUTOFD	STROBE

El *registro de control*, mantiene los estados de los bits C0 a C3 (STROBE, AUTOFD, INIT y SelectIn). C4 (IRQEN) en algunos puertos habilita el requerimiento de interrupción. Es un registro de sólo escritura.

Interfaz física

El Estándar IEEE 1284 define tres conectores para el puerto paralelo: 1284-A, 1284-B y 1284-C. El 1284-A, es equivalente al conector DB-25 comúnmente usado en la mayoría de las computadoras. El 1284-B al conector de 36 pines usado comúnmente para conectar a los dispositivos periféricos. El 1284-C es un nuevo conector centerline 0.050 de 36 pines. En la Figura V.20, se muestra el conector DB-25 de una PC.

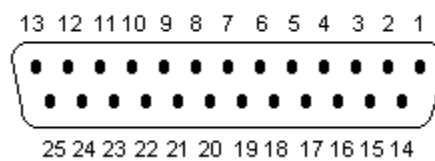


Figura V.21. Conector DB25 hembra del PC.

La tabla V.3. Muestra la descripción operacional de los conectores para el puerto paralelo definidos en el estándar IEEE-1284

Tabla V.3. Descripción operacional para el puerto paralelo definidos en el estándar IEEE 1284.

Número de Pin	1284-A 25 pin Dsub	Número de Pin	1284-A 25 pin Dsub
1	STROBE	14	AUTOFD
2	DATO 1	15	FAULT
3	DATO 2	16	INIT
4	DATO 3	17	SelectIn
5	DATO 4	18	Tierra
6	DATO 5	19	Tierra
7	DATO 6	20	Tierra
8	DATO 7	21	Tierra
9	DATO 8	22	Tierra
10	ACK	23	Tierra
11	BUSY	24	Tierra
12	PError	25	Tierra
13	Select		

Apéndice VI

Teorema de Funahashi.

Dada una función $f(x)$ no constante, acotada y monótona creciente. Sea X un subconjunto compacto de \mathfrak{R}^n . Sea un número real $\varepsilon > 0$ y k un entero $k \geq 3$. Entonces se tiene que para algún mapeo $g: X \rightarrow \mathfrak{R}^m$ cuyos componentes g_i son sumables en X , pueden ser aproximados en el sentido de la topología de L^2 en X por el mapeo de entrada/salida representado por una red neuronal de k capas ($k-2$ ocultas), con $f(x)$ como función de transferencia para las neuronas ocultas y funciones lineales para aquellas en las capas de entrada y salida. Esto es: $\forall \varepsilon > 0 \exists$ una red multicapa con las características dadas, definidas por el mapeo $g': X \rightarrow \mathfrak{R}^m$ de componentes g'_i , de manera que:

$$\|g - g'\|_{2X} = \left(\sum_{i=1}^m \int_X \left| g'_i(x) - g_i(x) \right|^2 dx \right)^{\frac{1}{2}} < \varepsilon,$$

Entonces si tenemos una red con una capa oculta, equivale a tomar $k = 3$. En resumen, una red neuronal con al menos una capa oculta puede aproximar hasta el nivel deseado dentro de un conjunto compacto cualquier función. Uno de los aspectos que el teorema no resuelve es la determinación del número de neuronas necesarias en la capa oculta.