INFORME TÉCNICO DE

RESIDENCIA PROFESIONAL:

INGENIERÍA EN ELECTRÓNICA

PRESENTAN:

BALLINAS BERMÚDEZ LUIS ENRIQUE

SANTIAGO SÁNCHEZ RIDER

CON EL TEMA:

“SISTEMA DE CONTROL PARA UN CONVERTIDOR CD-CA”

ASESOR:

ING. MA. CATALINA SALGADO GUTIÉRREZ

PERIODO:

JULIO - DICIEMBRE 2015

TUXTLA GUTIÉRREZ, CHIAPAS, ENERO DEL 2016

Índice

1 Introducción 1

1.1 Antecedentes…… 2
1.2 Planteamiento del problema……3
1.3 Objetivo general……3
1.4 Objetivos específicos……4
1.5 Justificación……4

2 Marco teórico 5

2.1 Circuito inversor de voltaje (VCS)……...5
2.2 Teoría de control……10
2.3 Redes neuronales artificiales……15
2.3.1 Redes neuronales biológicas……...16
2.3.2 Redes neuronales artificiales……...17
2.3.3 Generalización de una RNA……23
2.4 Tipos de neuronas y funciones de activación……24
2.5 Algoritmos de entrenamiento……27
2.5.1 El algoritmo de retro propagación……28
2.5.2 Algoritmo derivado del método de retro propagación……..33

3 Desarrollo 35

3.1 Procedimiento y descripción de las actividades realizadas……35
3.2 Resultados……59

4 Conclusiones 63

4.1 Conclusiones y recomendaciones……63
4.2 Competencias desarrolladas y/o aplicadas……64

Anexos 65

Anexo A……...65
Anexo b……67
Referencias……69

Capítulo 1

Introducción

Este proyecto se basa en la necesidad de realizar un control sobre un inversor. Hay muchos métodos conocidos para controlar estos dispositivos que son tan comunes en lineas de suministro de energía eléctrica y otras aplicaciones tanto industriales como para consumo doméstico. El planteamiento es adecuar la respuesta mediante un acondicionamiento de la señal de control que rige el comportamiento del convertidor.

El algoritmo presentado en este trabajo se basa en un sistema de control que utiliza una Red Neuronal Artificial (RNA) para controlar el voltaje de salida del convertidor, que a diferencia de los sistemas implementados en el control clásico, no requiere del conocimiento exacto de la función de transferencia del sistema, un hecho atractivo para la implementación de este dispositivo, sobre todo cuando no se tiene un conocimiento preciso del proceso que se desea controlar, ya que el control inteligente puede considerarse como una generalización del concepto del control, donde sus métodos se desarrollan para emular alguna característica importante del ser humano, que incluyen, adaptación y aprendizaje, planeación bajo gran incertidumbre y el trabajo con gran cantidad de datos.

Recientemente, se ha incrementado la atención a las redes de neuronas artificiales (RNA). Para modelar sistemas no lineales y diseñar controladores, la estructura de estos modelos se forma por un arreglo uniforme de unidades básicas de cálculo interconectadas entre sí, que después de seleccionar su estructura e interconexiones, se pondera mediante un entrenamiento, para aproximar el comportamiento entrada-salida de un sistema dinámico.

Las RNA son sistemas de procesamiento paralelo distribuido, compuestos por elementos de procesamiento no lineal, que se desempeñan de manera similar a las funciones más elementales de las neuronas biológicas. Las RNA tienen la habilidad de aprender de la experiencia, generalizar a partir de ejemplos previos y resumir información pertinente de muestras que contienen información irrelevante o incompleta. Las RNA no son adecuadas para tareas matemáticas simples, como puede ser, el cálculo de la caída de tensión en un alimentador, sin embargo, ayudan a resolver un gran número de problemas de reconocimiento de patrones, que son computacionalmente

difíciles o imposibles de resolver por medio de programas iterativos convencionales.

Las redes neuronales son capaces de identificar y controlar sistemas variantes en el tiempo con múltiples entradas - múltiples salidas. Estos modelos con un entrenamiento continuo en línea, pueden seguir la dinámica del sistema, siendo su identificación adaptable a los cambios en las condiciones y punto de operación.

1.1 Antecedentes

A continuación algunos trabajos importantes en los cuales se basó el proyecto desarrollado.

- Aplicación de un controlador Analógico Neuronal del Inversor de un UPS, implementado con Amplificadores Operacionales, con entrenamiento fuera de línea, usando patrones obtenidos de la simulación de un controlador, patrones que contienen referencias de la corriente de la carga, los resultados experimentales muestran un desempeño superior del controlador neuronal en la disminución de la distorsión armónica total, sobre todo con la operación en cargas no lineales (Sun y cols., 2002).

- Aplicación de un controlador analógico con la red neuronal para inversores de UPS. Los resultados de simulaciones realizadas por el controlador neuronal propuesto demostró un nivel bajo en la distorsión armónica total, en condiciones provocadas por cargas no lineales, se realiza una comparación con un controlador Proporcional Integral optimizado demostrando con los resultados obtenidos, el desempeño superior del controlador neuronal (Ionesco, Moscalu, y Moscalu, 2006).

- Desarrollo de controladores neuronales y Neuro-Fuzzy para inversor de fuente de voltaje interrumpible (UPS), para mejorar su respuesta transitoria y adaptabilidad a diversas cargas. Una retroalimentación de corriente de carga idealizada es utilizada para que el controlador pueda obtener los patrones de ejemplo para formación de las redes. Patrones de ejemplo en virtud de diversas condiciones de carga se utilizan en el entrenamiento fuera de línea de las redes neuronales y Neuro-Fuzzy, que se pone tan simple como sea posible para reducir el tiempo de cálculo. Los pesos y los bias de la Red Neuronal, se actualizan utilizando el algoritmo de retropropagación para hacer que el error cuadrático medio entre la salida deseada y de salida real sea menor

que el valor predefinido. Los resultados de la simulación muestran que los controladores neuronales y Neuro-Fuzzy pueden proporcionar un voltaje de salida senoidal con un bajo nivel de Distorsión Armónica Total (DAT) con diferentes condiciones de carga, y exhibe un buen rendimiento transitorio cuando los cambios de carga son efectuados (Bhoopal y Madhav, 2009).

- Control Neuronal para la Corrección de Disturbios en Redes de Energía Eléctrica se propone un esquema de control basado en Redes Neuronales Artificiales del Convertidor Estático de Voltaje (VSC) de un Restaurador Dinámico, mediante la estrategia SPWM. Este esquema puede reconocer el tipo de disturbio, por medio del nivel de voltaje en rms y corregir en forma rápida y precisa las variaciones en el voltaje de suministro con una técnica de compensación en fase, utilizada para los equipos no sensibles al salto de fase (*Zepeda Hernández, J.A.,* 2014).

1.2. Planteamiento del Problema

En la actualidad muchas industrias tanto privadas como públicas cuentan con más de un convertidor de voltaje y la mayoría de ellas dependen de que estos funcionen correctamente ya que sus equipos son muy sofisticados y cualquier variación de voltaje podría dañarlos parcial o totalmente.

1.3. Objetivo General

Diseñar e Implementar un Sistema de Control Neuronal para un Convertidor de Corriente Directa a Corriente Alterna.

1.4. Objetivos Específicos

- Diseñar e Implementar la instrumentación para la medición, control y protección del convertidor de CD-CA.
- Diseñar e Implementar el control SPWM.
- Diseñar e Implementar una red neuronal artificial para el control del convertidor CD-CA.
- Simular el Sistema de Control Neuronal en Simulink de MatLab
- Comparar los resultados obtenidos en la simulación, con el sistema implementado en la tarjeta de desarrollo ARDUINO.

1.5. Justificación.

Debido a la gran cantidad de convertidores usados en la actualidad en los diferentes ámbitos de la sociedad, tanto comerciales como industriales, requiere que los equipos a los que este alimentando se mantengan en un estado óptimo de desempeño, por lo cual se necesita de un sistema de control que pueda permitir que estos convertidores tengan una respuesta rápida ante cualquier perturbación es ahí donde las redes neuronales entran en acción.

Capítulo 2

Marco Teórico.

2.1. Circuito Inversor de Voltaje (VSC)

Los inversores son circuitos que convierten la corriente continua en corriente alterna, estos circuitos transfieren potencia desde una fuente continua a una carga en alterna, usando modulación SPWM (Hart, 2001) u otro tipo dependiendo de las necesidades de la carga a alimentar.

La modulación por ancho de pulsos (PWM), proporciona un método para disminuir el factor DAT (Distorsión Armónica Total) de la corriente de carga. La salida de un inversor PWM en conjunto con un sistema de filtrado, en general cumple con las especificaciones de DAT con más facilidad que el esquema de conmutación de onda cuadrada. La salida PWM sin filtrar tendrá un factor DAT relativamente elevado, pero los armónicos tendrán frecuencias mucho más altas, haciendo más sencillo el proceso de filtrado (Hart, 2001).

En la modulación PWM, la amplitud del voltaje de salida se puede controlar por medio de las forma de onda moduladora. Dos ventajas de la modulación PWM son la reducción de los requerimientos del filtro para reducir los armónicos y el control de la amplitud de salida. Entre las desventajas se puede citar que los circuitos de control de los interruptores son más complejos, así como la presencia de mayores pérdidas debidas a una conmutación más frecuente, sin embargo con la fabricación hoy en día de dispositivos de alta velocidad y baja resistencia de conmutación esta desventaja queda superada (Hart, 2001).

El control de los interruptores para la salida senoidal PWM (SPWM) requiere:

1. Una señal de referencia llamada señal de control o moduladora de tipo senoidal.

2. Una señal portadora de tipo triangular que controla la frecuencia de conmutación.

El convertidor en puente de onda completa

El convertidor de onda completa mostrado en la figura 2.1 es el circuito básico que se utiliza para convertir un voltaje continuo en alterno. A partir de una

entrada continua se obtiene una salida de alterna cerrando y abriendo interruptores en una determinada secuencia como se muestra en la tabla 2.1. El voltaje de salida v0 puede ser +Vcc, −Vcc o cero, dependiendo de que interruptores estén cerrados, el inversor funciona con una conmutación monopolar.

Tabla 2.1 Combinaciones de Interruptores

Interruptores cerrados	Voltaje de Salida
S_1 y S_2	$+V_{cc}$
S_3 y S_4	$-V_{cc}$
S_1 y S_3	0
S_2 y S_4	0

S1 y S4 así como S2 y S3, no deben estar cerrados al mismo tiempo, ya que se produciría un cortocircuito en la fuente de continua. Los interruptores reales no se abren y cierran instantáneamente, por tanto deben tenerse en cuenta los tiempos de transición de la conmutación al diseñar el control de los interruptores. El solapamiento de los tiempos de conducción de los interruptores resultaría en un cortocircuito, denominado en ocasiones fallo de solapamiento (shoot-through) en la fuente de voltaje continua. El tiempo permitido para la conmutación se denomina tiempo muerto (blanking time).

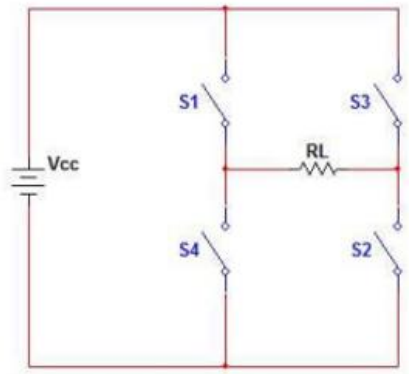


Figura 2.1 Puente Convertidor de Onda Completa

Conmutación monopolar

La figura 2.2 ilustra el principio de la modulación por anchura de pulsos monopolar con señal de referencia senoidal comparada con una señal

triangular. Cuando el valor instantáneo de la sinusoide de referencia es mayor que la portadora triangular, la salida está en +VCC, y cuando la referencia es menor que la portadora, la salida está en −VCC;

El esquema de conmutación que permitirá implementar la conmutación Monopolar utilizando el puente inversor de onda completa que se muestra en la figura 2.1 se determina comparando las señales instantáneas v_{seno} o señal de referencia y dos señales triangulares, v_{tri} denominada señal portadora, defasadas 180 grados entre sí, el resultado de este proceso proporciona la sincronía de los interruptores (Hart, 2001).

S1 conduce cuando $v_{seno} > v_{tri}$

S2 conduce cuando $-v_{seno} < v_{tri}$

S3 conduce cuando $-v_{seno} > v_{tri}$

S4 conduce cuando $v_{seno} < v_{tri}$

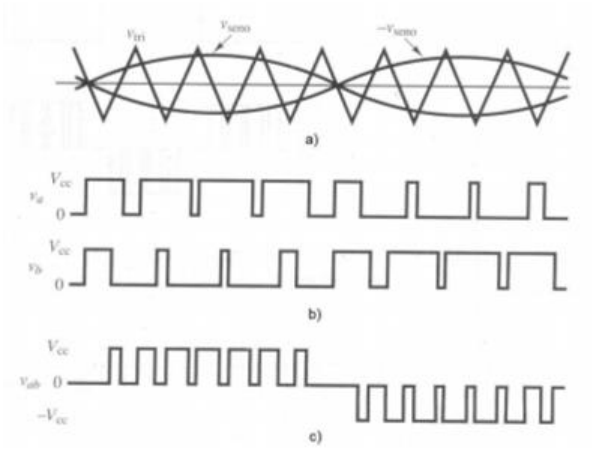


Figura 2.2 Modulación por anchura de Impulso Monopolar

Los interruptores mostrados (S1, S4) y (S2, S3) mostrados en la figura 2.1 del puente completo del inversor son complementarios, cuando un interruptor de uno de los pares está cerrado, el otro se encuentra abierto, El voltaje a través de la carga oscilan entre VCC y cero y viceversa.

Consideraciones relativas a la modulación SPWM

Cuando se utiliza la modulación por ancho de pulso es necesario tener en consideración los siguientes puntos:

1. Índice de Modulación de frecuencia m_f : La serie de Fourier del voltaje de salida PWM tiene una frecuencia fundamental que es la misma que la de la señal de referencia. Las frecuencias armónicas existen en y alrededor de los múltiplos de la frecuencia de conmutación. Los valores de algunos armónicos son bastante grandes, a veces mayores que la componente fundamental. Sin embargo, como estos armónicos se encuentran en frecuencias altas, para eliminarlos es suficiente con el apoyo de un filtro de paso bajo.

El índice de Modulación de frecuencia m_f se define como la relación entre las frecuencias de las señales portadora y de referencia de acuerdo a la ecuación 2.1.

$$m_f = \frac{f_{portadora}}{f_{referencia}} = \frac{f_{tri}}{f_{seno}} \tag{2.1}$$

Al aumentar la frecuencia de la portadora (aumento de m_f) aumentan las frecuencias a las que se producen los armónicos, una desventaja de las elevadas frecuencias de conmutación son las mayores pérdidas en los interruptores utilizados para implementar el inversor.

2. Índice de modulación de amplitud m_a: El índice de modulación de amplitud ma se define como la relación entre las amplitudes de las señales de referencia y portadora de acuerdo a la ecuación 2.2.

$$m_a = \frac{v_{m,referencia}}{v_{m,portadora}} = \frac{v_{m,seno}}{v_{m,tri}} \tag{2.2}$$

Si $m_a \leq 1$, la amplitud de la frecuencia fundamental del voltaje de salida, V_1, es linealmente proporcional a ma. Es decir:

$$V_1 = m_a V_{CC} \tag{2.3}$$

Por lo tanto la amplitud de la frecuencia fundamental de la salida PWM está controlada por ma. Esto resulta importante en el caso de un voltaje en corriente continua sin regular, porque el valor de m_a se puede ajustar para compensar las variaciones en la fuente de alimentación, produciendo una salida de amplitud constante. Por otra parte ma se puede variar para cambiar la amplitud de salida. Si m_a es mayor que uno, la amplitud de la salida aumenta al incrementarse el valor de m_a, aunque no de forma lineal.

3. Interruptores: Los interruptores en el circuito en puente de onda completa son capaces de soportar la corriente en cualquier dirección para la modulación PWM, cuando se usan componentes discretos para la conmutación tales como dispositivos BJT, FET o IGBT, no se abren o se cierran instantáneamente, de tal manera que es importante tener en cuenta los tiempos de conmutación en el control de los interruptores (Sigg, Turkes, y Kraus, 1997; Baliga, Adler, Love, Gray, y Zommer, 1984).

4. Voltaje de Referencia: El voltaje de referencia senoidal debe generarse dentro del circuito de control del inversor, o tomarse de una referencia externa, esta señal requiere poca potencia, ya que la potencia principal la proporciona la fuente de corriente continua.

Armónicos en la modulación SPWM

La serie de Fourier de la salida de modulación PWM Unipolar mostrada en la figura 2.2 se calcula examinando cada uno de los valores de los pulsos que la integran. La forma de onda triangular está sincronizada con la de referencia, si se elige una m f que sea un entero impar, entonces la salida PWM muestra una simetria impar y se puede expresar como:

$$v_o(t) = \sum_{n=1}^{\infty} V_n sen(n\omega_0 t) \tag{2.4}$$

Para el k-ésimo pulso de la salida PWM, el coeficiente de Fourier es:

$$v_{nk} = \frac{2}{\pi}\int_0^T V(t) sen(n\omega_0 t) d(\omega_0 t) \tag{2.5}$$

$$= \frac{2}{\pi}[\int_{\alpha_k}^{\alpha_k+\delta_k} V_{CC} sen(n\omega_0 t) d(\omega_0 t) + \int_{\alpha_k+\delta_k}^{\alpha_k+1} (-V_{CC}) sen(n\omega_0 t) d(\omega_0 t)] \tag{2.6}$$

Integrando,

$$V_{nk} = \frac{2V_{CC}}{n\pi}[cosn\alpha_k + cosn\alpha_{k+1} - 2cosn(\alpha_k + \delta_k)] \tag{2.7}$$

Cada coeficiente de Fourier Vn para la forma de onda PWM es la suma de Vnk para los p pulsos comprendidos en un periodo:

$$v_n = \int_{k=1}^{p} V_{nk} \tag{2.8}$$

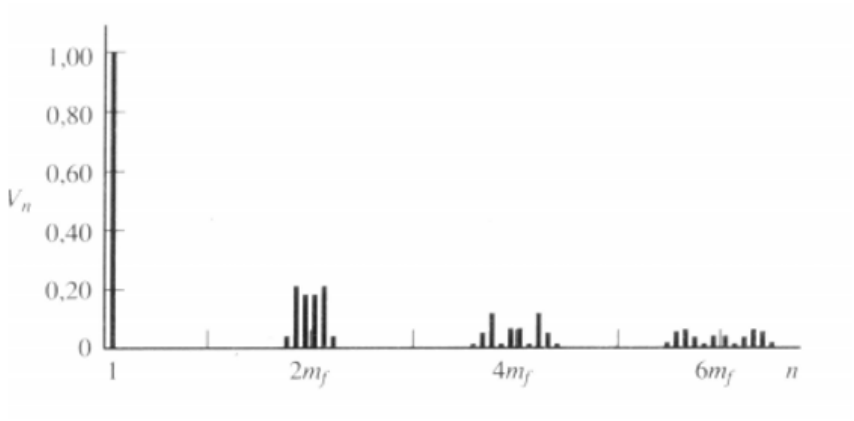


Figura 2.3 Espectro de frecuencia para PWM unipolar con ma=1

El espectro de frecuencia normalizado de la conmutación Monopolar para m_a = 1 se muestra en la figura 2.3. Las amplitudes de los armónicos son una función de ma, porque la anchura de cada pulso depende de las amplitudes relativas de la onda senoidal y triangular. Las primeras frecuencias armónicas en el espectro de salida están en y alrededor de m_f. En la tabla 2.2 se indican los primeros armónicos de salida para una conmutación PWM monopolar. Los coeficientes de Fourier no son una función de m_f si éste es elevado (≥ 9).

Tabla 2.2 Coeficientes de Fourier normalizados para PWM unipolar

	$m_a = 1$	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
$n = 1$	1.00	0.90	0.8	0.70	0.6	0.5	0.4	0.3	0.2	0.1
$n = 2m_f \pm 1$	0.18	0.25	0.31	0.35	0.37	0.36	0.33	0.27	0.19	0.10
$n = 3m_f \pm 3$	0.21	0.18	0.14	0.10	0.07	0.04	0.02	0.01	0.00	0.00

2.2. Teoría de Control

La función de un mecanismo de control es mantener de manera certera, propiedades esenciales de un sistema a valores deseados sujeto a perturbaciones. Históricamente, sistemas de control simple pero efectivos, han sido empleados en la regulación del flujo agua y control de nivel de líquido en los vasos de vino durante siglos. Algunos de estos conceptos todavía se utilizan hoy en día, por ejemplo, el sistema de flotador en el tanque de agua del inodoro. Sin embargo, los sistemas de control moderno utilizados en la

industria hoy en día son mucho más complejos y debe sus orígenes al desarrollo de la teoría de control. La importante labor temprana en el control automático moderno se puede remontar al diseño de James Watt, del gobernador fly-ball en 1788 para el control de la velocidad de una máquina de vapor. En 1868, Maxwell presentó su primer análisis matemático de un control retroalimentado. Fue durante este tiempo que los estudios sistemáticos en los sistemas de control y dinámicas de retroalimentación comenzaron. Un hecho significativo fue el bien conocido Criterio de estabilidad de Routh en 1877, que hizo merecedor a E.J. Routh al Premio Adam.

El principio del siglo XX vio el comienzo de lo que hoy se conoce como la teoría de control clásica. El trabajo de Minorsky en 1922, sobre la determinación de la estabilidad desde las ecuaciones diferenciales que describen el sistema (ecuación característica) y el desarrollo de Nyquist en 1932, procedimiento gráfico para determinar la estabilidad (respuesta en frecuencia), contribuyeron sustancialmente al estudio de la teoría de control. En 1934, Hazen introdujo el término “Servomecanismo” para describir la posición del sistema de control en su intento por desarrollar una teoría generalizada de servomecanismos. Dos años después, el desarrollo del controlador proporcional integral derivativo fue descrito por Callendar, año de 1936.

La teoría de control, al igual que otras ramas de la ingeniería, se sometieron a desarrollos significativos durante la Segunda Guerra Mundial. Con Base en el trabajo de Nyquist, H.W. Bode introdujo un método para el diseño de amplificadores retroalimentados, en la actualidad conocido como Diagrama de Bode. En 1948, el método del lugar de las raíces del diseño y análisis de estabilidad fue desarrollado por W.R. Evans.

Con la introducción de las computadoras digitales en la década de los 60´s, el uso de la respuesta de frecuencia y ecuaciones características comenzaron a dar paso a las ecuaciones diferenciales ordinarias (EDO), que funcionaban bien con las computadoras. Esto dio lugar al nacimiento de la teoría de control moderna. Mientras que el término teoría de control clásica se utiliza para describir los métodos de diseño de Bode, Nyquist, Minorsky y trabajos similares, la teoría de control moderna se basa en métodos de diseño con EDO que son más adecuados para la ingeniería asistida por computadora, por ejemplo, el enfoque de espacio de estado. Ambas ramas de la teoría de control se basan en una representación matemática de la planta de control de la que se desea obtener su rendimiento.

Para hacer frente a las cuestiones de no-linealidades y parámetros variables en el tiempo en los modelos de plantas, se introdujeron estrategias de control que se adaptan continuamente a las variaciones de las características de la planta, generalmente conocidas como sistemas de control

adaptativo, ellas incluyen técnicas tales como control autoajustable, control H-infinito, modelo referenciado de control adaptativo y control de modo deslizante. Estudios que incluyen el uso de observadores de estado no lineales que continuamente estiman los parámetros de la planta de control, pueden ser empleados para hacer frente a la cuestión de la no observabilidad, que es la condición por la cual no todos los estados requeridos están disponibles para la retroalimentación, esta es una solución económica, ya que no requiere muchos sensores, como por ejemplo en los variadores de velocidad, o porque es físicamente difícil o incluso imposible obtener los estados de retroalimentación, como puede suceder en un reactor nuclear. En muchos casos, el modelo matemático de la planta es simplemente desconocido o indefinido, lo que conduce a mayores complejidades en el diseño del sistema de control. La propuesta de los sistemas de control inteligentes da un mejor rendimiento en estos casos, a diferencia de las técnicas convencionales de control, los controladores inteligentes se basan en inteligencia artificial (IA) en lugar del modelo de la planta. Estos imitan el proceso de toma de decisiones humana y, a menudo se pueden implementar en sistemas complejos con más éxito que las técnicas de control convencionales. La Inteligencia Artificial puede ser clasificada en sistemas expertos, lógica difusa, redes neuronales artificiales y algoritmos genéticos. Con excepción de los sistemas expertos, estas técnicas se basan en métodos de computación suave. El resultado de estas técnicas es su capacidad de hacer aproximaciones y “conjeturas inteligentes” cuando es necesario, a fin de llegar con suficiente aproximación a un resultado bueno bajo un determinado conjunto de restricciones. En el diseño de los sistemas de control inteligente se pueden emplear una o más técnicas de Inteligencia Artificial (Cirstea, Dinu, McCormick, y Khor, 2002).

Técnicas Básicas

Entre las técnicas de Inteligencia Artificial usadas en Control Inteligente destacan:

- Los Sistemas Expertos, basados en el uso de las técnicas y herramientas de diseño de sistemas expertos de Inteligencia Artificial.

  En este tipo de sistemas la base de conocimientos de control se obtiene de un experto humano: el operador del proceso a controlar. En ella se recoge esencialmente dos tipos de información: reglas referentes a la interpretación del estado del proceso y reglas para la determinación de las actuaciones. Para aumentar la eficiencia las reglas se clasifican en grupos o metareglas (Shin y Xu, 2008).

- Lógica Difusa, control basado en reglas que utiliza técnicas para manejar la imprecisión. Cabría separar el estudio de los controladores difusos como alternativa al control adaptativo, predictivo u otros del control experto que utiliza incertidumbre (Panda, Mahapatra, y Bagarty and Behera, 2011) .

  Los controladores difusos tratan de implantar estrategias de control expresadas en términos lingüísticos por los operadores de proceso, para ello se basan en técnicas de lógica difusa.

  La lógica difusa ha alcanzado un notable desarrollo tanto en el estudio formal como en el de aplicaciones y herramientas para diseño disponibles.

- Redes Neuronales; una red neuronal es, siguiendo a Hecht-Nielsen, “una estructura de procesamiento de información paralela y distribuida”, formada por elementos de procesamiento interconectados mediante canales unidireccionales de información. Cada elemento de procesamiento tiene una conexión de salida con diferentes ramas portadoras de la misma señal. Esta señal de salida será de un tipo matemático cualquiera. Todo el procesamiento que se hace en un elemento debe ser completamente local, por ejemplo: dependerá solo de los valores actuales de las entradas al elemento y de posibles valores almacenados en memoria local.

  Las redes neuronales en control se utilizan por su capacidad de aprender el comportamiento no lineal de las variables de un proceso. Esta capacidad se puede utilizar para el diseño de sistemas que funcionen como simulador, identificador o controlador (Hagan y Beale, 1996).

- Algoritmos Genéticos, los algoritmos genéticos se están utilizando en control, entre otras aplicaciones, para depurar de forma automática las reglas que forman la base de conocimiento. Ésta se equipara a un conjunto de organismos vivos, capaces de evolucionar para adaptarse mejor al entorno. Esta adaptación se mide a partir de la tasa de fallos y aciertos de los individuos.

  Al igual que en la evolución de las especies, cuando se produce una variación del entorno, sólo los que se adaptan a esa variación sobreviven, mientras que los que no pueden adaptarse son eliminados.

A lo largo del tiempo van naciendo nuevos individuos que pasan a formar parte de la comunidad, con características genéticas que les hacen parecerse a sus padres, y permiten que la especie se mantenga. En algunas ocasiones se producen mutaciones que dan lugar a individuos mejor o peor adaptados.

Los algoritmos genéticos toman reglas buenas para crear nuevos individuos similares a ellas, que permitan al sistema de inferencia obtener mejores resultados.

El control Neuronal es una rama del campo general del control inteligente, el cual está basado en los conceptos de la inteligencia artificial (AI), ésta puede ser definida como la emulación en computadora del proceso del pensamiento humano. Las técnicas de Inteligencia Artificial son generalmente clasificadas como: Sistemas Expertos (SE), Lógica Difusa (FL) y Redes neuronales artificiales (ANN).

Los sistemas expertos clásicos están basados en el álgebra booleana y usan cálculos precisos, mientras los sistemas de lógica difusa involucran cálculos basados en razonamientos aproximados. La lógica difusa es un súper conjunto de lógica convencional booleana que ha sido extendido para manejar el concepto de parcialmente verdadero- verdadero, valores entre “completamente verdadero y completamente falso” (Sivanandam, Sumathi, y Deepa, 2007). Concepto introducido por el Dr. Lotfi Zadeh de la Universidad de Berkeley California en la década de los 60’s como un medio para modelar la incertidumbre del lenguaje natural. La verdad de una expresión lógica en lógica difusa es un numero entre [0,1]. La lógica difusa se ha convertido en una herramienta útil para el control de sistemas y procesos industriales complejos. Es usado en procesos que no tienen un modelo matemático simple, para procesos altamente no lineales, o cuando se va a realizar el tratamiento de los conocimientos lingüísticamente formulados. A pesar de que fue inventado en los Estados Unidos, el rápido crecimiento de esta tecnología se originó en Japón y ahora ha vuelto a alcanzar el EE.UU. y Europa. Los controladores basados en este enfoque matemático, se conocen como controladores difusos.

El uso de redes neuronales artificiales (RNAs) es el método más poderoso de la Inteligencia Artificial. Las RNAs son estructuras de procesamiento de información el cual emula la arquitectura y modo de operación de un tejido nervioso biológico (Guyton y Hall, 2000; Hecht-Nielsen, 1990).

Cualquier RNA es un sistema formado por varias entidades básicas (llamadas neuronas) que están interconectadas y operan en paralelo transmitiendo señales entre sí con el fin de lograr una determinada tarea de procesamiento (Hagan y Beale, 1996). Una de las características más sobresalientes de las RNA's es su capacidad para simular el proceso de aprendizaje. Se suministran como pares relacionados de señales de entrada y de salida de la cuales las reglas generales son automáticamente derivadas de modo que la RNA en determinadas condiciones será capaz de generar la salida correcta para una señal que no se había sido utilizado anteriormente. El enfoque neuronal puede ser combinado con la lógica difusa, generando sistemas neuro difusos que combinan las ventajas de los dos paradigmas.

2.3. Redes Neuronales Artificiales

Debido a que las computadoras modernas son computacionalmente muy poderosas y siguen incrementando su capacidad de procesamiento, los científicos e investigadores continúan diversificando su empleo, para tareas que son relativamente simples para los humanos. Basado en ejemplos, junto con algunas retroalimentaciones de un "maestro", se aprende fácilmente a reconocer la letra A o distinguir un objeto de otro, con más experiencia es posible refinar las respuestas y mejorar el desempeño. Este es un ejemplo típico que ilustra una tarea para la cual la solución por computadora puede ser difícil.

Tradicionalmente, la computadora digital basada en lógica secuencial tiene excelente desempeño en muchas áreas, aunque ha sido menos exitosa para otras. El desarrollo de las redes de neuronas artificiales comenzó aproximadamente en los años 50's, motivado por el deseo de entender el cerebro y emular algunas de sus actividades.

Se descubrió que la codificación en forma de pulsos digitales con cierta frecuencia proporciona calidad, seguridad y simplicidad en la transmisión de información. Los tres conceptos clave de los sistemas nerviosos, que se pretende emular en los artificiales, son: paralelismo de cálculo, memoria distribuida y adaptabilidad al entorno (Guyton y Hall, 2000). De esta manera, se puede hablar de las RNAs como sistemas paralelos, distribuidos y adaptativos. Estas redes se adaptan fácilmente al entorno modificando su sinapsis, y aprenden de la experiencia, con la posibilidad de generalizar conceptos a partir de casos particulares. En el campo de las redes neuronales se denomina a esta propiedad generalización a partir de ejemplos. Las características de las redes neuronales biológicas que sirven como inspiración para las redes de neuronas artificiales se exhiben en la figura 2.6.

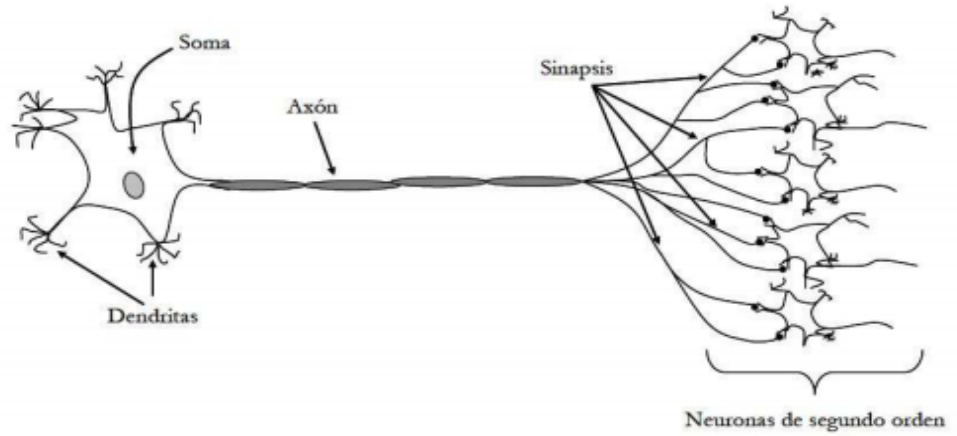


Figura 2.4 Estructura Típica de una Neurona Biológica

Las computadoras digitales de alta velocidad hacen factible la simulación del proceso neuronal. Las redes de neuronas artificiales son modelos computacionales en paralelo que comprenden unidades de procesamiento adaptable interconectadas. Estas redes son implementaciones en paralelo de sistemas dinámicos o estáticos no lineales. Una característica muy importante de estas redes es su naturaleza adaptativa, donde “aprenden mediante ejemplos” reemplazando la “programación” en la solución de problemas.

Esta característica hace de tales modelos computacionales adecuados en dominios de aplicación donde se tiene poco o incompleto entendimiento del problema a resolver, pero donde se tienen disponibles datos de entrenamiento. Otra característica importante es la arquitectura paralela intrínseca que permite un rápido cálculo de la solución cuando las redes se implementan en computadoras digitales en paralelo o, en su caso, cuando se implementan en hardware especializado. “La Neurona Artificial” es la unidad básica para la construcción de bloques de procesamiento de una red neuronal. Es necesario entender las capacidades computacionales de estas unidades de procesamiento como un prerrequisito para entender el funcionamiento de una red (Hecht-Nielsen, 1990).

2.3.1. Redes Neuronales Biológicas

La neurona es la unidad funcional básica del sistema nervioso. En la figura 2.4 se observa una neurona típica (del asta anterior de la médula espinal). Formada por tres partes principales: el soma, que es el cuerpo principal de la neurona; un solo axón, que parte del soma y se incorpora a un nervio periférico que sale de la médula espinal; y las dendritas, que son

numerosas prolongaciones ramificadas del soma, que se extienden hasta 1 mm en las áreas medulares circundantes. Las señales de entrada (aferentes) llegan a la neurona a través de las sinapsis que establecen, sobre todo, las dendritas, aunque también el cuerpo celular.

El número de conexiones sinápticas que mantienen las fibras de entrada se encuentra entre varios cientos a doscientos mil. Por el contrario, la señal de salida (eferente) se transmite por el único axón de la neurona. Dicho axón tiene muchas ramas separadas destinadas a otras partes del sistema nervioso o a la periferia del cuerpo. Un rasgo especial de la mayoría de las sinapsis es que la señal se transmite de ordinario solamente en la dirección hacia delante (del axón a las dendritas). Además las neuronas se disponen en un gran número de redes nerviosas con distinta organización que determinan las funciones del sistema nervioso.

Una de las funciones más importantes del sistema nervioso consiste en procesar la información aferente para elaborar respuestas mentales y motoras adecuadas. Solo una pequeña parte de la información sensorial importante desencadena una respuesta motora inmediata. Gran parte de los datos restantes se almacena para usarlos más tarde en la regulación de los actos motores y en los procesos mentales. El almacenamiento de información es el proceso que llamamos memoria. Cada vez que cierta clase de señales sensoriales atraviesan una serie de sinapsis, aumenta la capacidad de dichas sinapsis para transmitir las mismas señales la siguiente ocasión, proceso que se llama facilitación.

En el sistema nervioso central la información se transmite principalmente bajo la forma de potenciales de acción nervioso, que pasan uno tras otro por una serie de neuronas. Cada impulso puede además ser bloqueado al transmitirse desde una neurona a la siguiente, cambiar, y en vez de ser único convertirse en impulsos repetidos; o integrarse con los impulsos de otras neuronas para dar lugar a modelos complejos de impulsos en las neuronas sucesivas. Todas estas funciones se consideran funciones sinápticas de las neuronas. Sinapsis constituye el punto de unión entre una neurona y la siguiente y determina las direcciones de las señales nerviosas al propagarse por el sistema nervioso (Guyton y Hall, 2000).

2.3.2. Redes de Neuronas Artificiales

Una red de neuronas artificiales es un sistema de procesamiento de información que tiene ciertas características de desempeño similar con las redes neuronales biológicas. Las RNAs han sido desarrolladas como una generalización de modelos matemáticos del reconocimiento humano o

neuronas biológicas, basadas en las siguientes consideraciones (Fausett,1994; Hassoun, 1995):

1. El procesamiento de la información ocurre en muchos elementos sencillos denominados neuronas.

2. Las señales circulan entre neuronas a través de ligas de conexión.

3. Cada una de las ligas de conexión tiene un peso asociado, que en una red neuronal típica, modifica la señal transmitida.

4. Cada neurona aplica una función de activación (generalmente no-lineal) a su entrada (suma de señales de entrada ponderadas) para determinar su señal de salida.

Una red neuronal se caracteriza por: el patrón de conexiones entre neuronas (arquitectura), el método de determinar los pesos en las conexiones (algoritmo de entrenamiento o aprendizaje) y la función de activación.

Las RNAs consisten de un gran número de sencillos elementos de procesamiento llamadas neuronas, unidades, células o nodos. Cada una de las neuronas se conecta a otras neuronas por medio de ligas de comunicación directas, cada una con un peso asociado. Cada neurona tiene un estado interno, llamado activación o nivel de actividad, que es una función de la entrada que recibe. Típicamente, una neurona envía su activación como una señal a diferentes neuronas.

Un sistema neuronal está compuesto por los siguientes elementos esenciales:

- Un conjunto de procesadores elementales o neuronas artificiales.

- Un patrón de conectividad o arquitectura.

- Una dinámica de activaciones.

- Una regla o dinámica de aprendizaje.

- El entorno donde opera.

En general, la respuesta de las neuronas biológicas es de tipo no lineal, característica que se emula en las artificiales. La formulación de la neurona

artificial como dispositivo no lineal constituye una de sus características más destacables, y la que proporciona mayor interés en las RNAs, pues el tratamiento de problemas no lineales no suele ser fácil de abordar mediante técnicas convencionales.

Modelo General de las RNAs

Se denomina procesador elemental o neurona a un dispositivo simple de cálculo que, a partir de un vector de entrada procedente del exterior o de otras neuronas, proporciona una única respuesta o salida (Mirchandani y Cao, 1989; Hornik y cols., 1989). Los elementos que constituyen la neurona de etiqueta i, figura 2.5, son los siguientes:

- Conjunto de entradas, x_j (t).

- Pesos sinápticos de la neurona i, w_{ij} que representan la intensidad de interacción entre cada neurona presináptica j y la neurona postsináptica i.

- Regla de propagación $\sigma(w_{ij}, x_j$ (t) que proporciona el valor del potencial postsináptico h_i (t) = $\sigma(w_{ij}, x_j$ (t) de la neurona i en función de sus pesos y entradas.

- Función de activación fi (a_i (t −1), hi (t)), que proporciona el estado de activación actual a_i (t) = fi (a_i (t − 1), h_i (t)) de la neurona i, en función de su estado anterior a_i (t − 1) y de su potencial postsináptico actual.

- Función de salida F_i (a_i (t)), que proporciona la salida actual y_i (t) = F_i (a_i (t)) de la neurona i en función de su estado de activación.

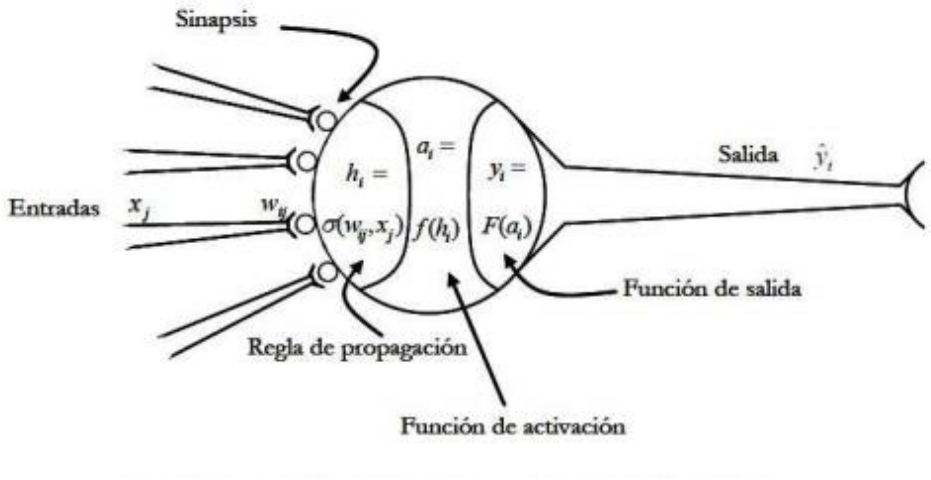


Figura 2.5 Modelo genérico de una Neurona Artificial

De este modo, la operación de la neurona i puede expresarse como:

$$y_i(t) = F_i(f_i(a_i(t\ 1), \sigma_i(w_{ij}, w_j(t))))$$

Este modelo de neurona formal se inspira en la operación de la neurona biológica, en el sentido de integrar una serie de entradas y proporcionar cierta respuesta, que se propaga por el axón (Guyton y Hall, 2000). Las variables de entrada y salida pueden ser binarias (digitales) o continuas (analógicas), dependiendo del modelo y aplicación. Con frecuencia se añade al conjunto de pesos de la neurona un parámetro adicional, b_i que se denomina umbral, figura 2.6.

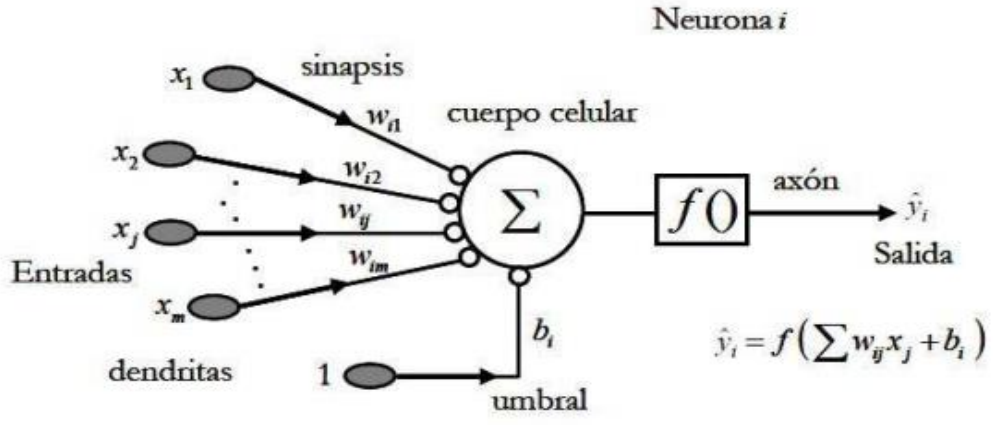


Figura 2.6 Modelo de Neurona Estándar

Arquitecturas de Redes Neuronales

Generalmente, las neuronas en la misma capa tienen idéntico comportamiento. Los principales factores que determinan el comportamiento de una neurona son: la función de activación y el patrón de los pesos en las conexiones sobre las cuales se envía y recibe la señal. El arreglo de neuronas en capas y los patrones de conexión dentro y entre capas se denomina arquitectura de la red. Muchas redes neuronales tienen una capa de entrada en la cual la activación de cada unidad es igual a una señal de entrada externa.

Las redes neuronales se pueden clasificar en unicapa o multicapa. Otra posible clasificación es en estáticas o dinámicas. Se puede definir el número de capas de la red como el número de capas de pesos interconectadas entre neuronas (Hagan y Beale, 1996). Esto se motiva por el hecho de que los pesos en una red contienen información extremadamente importante.

Se denomina arquitectura a la topología, estructura o patrón de conexión de una red neuronal. En una RNA los nodos se conectan por medio de sinapsis, esta estructura de conexiones sinápticas determina el comportamiento de la red. En general, las neuronas se suelen agrupar en unidades estructurales que se denominan capas. Las neuronas de una capa pueden agruparse, a su vez, formando grupos neuronales (clusters). Se distinguen tres tipos de capas: de entrada, de salida, y ocultas (Fausett, 1994):

- Una capa de entrada o sensorial está compuesta por neuronas que reciben datos o señales procedentes del entorno.

- Una capa de salida es aquella cuyas neuronas proporcionan la respuesta de la red neuronal.

- Una capa oculta es aquella que no tiene una conexión directa con el entorno.

Modos de Operación de las RNAs

Se distinguen dos modos de operación en los sistemas neuronales: el modo de recuerdo o ejecución, y el modo de aprendizaje o entrenamiento.

Además de la arquitectura, el método de ajustar el valor de los pesos (aprendizaje o entrenamiento) es una importante característica de las RNAs. Por conveniencia, se distinguen dos tipos de entrenamiento: supervisado y no supervisado (Guyton y Hall, 2000; Fausett, 1994). En general existe una útil

correspondencia entre los tipos de entrenamiento y el tipo de problema que se quiere resolver.

En la mayor parte de las ocasiones el aprendizaje consiste simplemente en determinar un conjunto de pesos sinápticos que permita a la red realizar correctamente el tipo de procesamiento deseado. Cuando se construye un sistema neuronal, se parte de un cierto modelo de neurona y de una determinada arquitectura de la red, estableciendo los pesos sinápticos iniciales como nulos o aleatorios. El proceso de aprendizaje es generalmente iterativo, actualizando los pesos una y otra vez, hasta que la red neuronal alcanza el desempeño deseado. Estos términos describen la forma en que se forman las redes neuronales.

El aprendizaje supervisado es el más común entre las redes neuronales (tanto el perceptrón de varios niveles como la función de base radial son sistemas de aprendizaje supervisado). Las técnicas de aprendizaje supervisado se emplean para la predicción, la clasificación y los modelos de series temporales. Las técnicas no supervisadas (Red de Kohonen) se utilizan para agrupar los casos similares.

La distinción entre aprendizaje supervisado y no supervisado depende de la forma en la cual el algoritmo de entrenamiento emplea la información proporcionada del entorno (patrón-clase).

El algoritmo supervisado asume la disponibilidad de un maestro o supervisor quien clasifica los ejemplos de entrenamiento en grupos, mientras el no supervisado no lo tiene. La red neuronal con entrenamiento supervisado compara sus predicciones con la respuesta objetivo y aprende de sus errores. En el aprendizaje no supervisado se debe identificar la información del entorno como parte de su proceso de entrenamiento. En general, la tarea del aprendizaje no supervisado es más abstracta y menos definida. La red, por medio de la regla de aprendizaje, puede reconocer regularidades en el conjunto de entradas, extraer rasgos, o agrupar patrones según su similitud. Tal como sucede en el medio de la educación, en el proceso de aprendizaje, el estudiante debe: enfocar su atención, observar la regularidad en el medio y realizar su hipótesis.

Muchos de los algoritmos de aprendizaje se basan en métodos numéricos iterativos que tratan de minimizar una función de costo (Fausett, 1994; Hassoun, 1995), lo que puede dar lugar en ocasiones a problemas de convergencia del algoritmo. La convergencia es una manera de comprobar si una determinada arquitectura, junto a su regla de aprendizaje, es capaz de resolver un problema. En el proceso de entrenamiento es importante distinguir entre el nivel de error alcanzado al final de la fase de aprendizaje para el

conjunto de datos de entrenamiento, y el error que la red ya entrenada comete ante patrones no utilizados en el aprendizaje, lo cual mide la capacidad de generalización de la red. Muchas veces interesa más una buena generalización que un error muy pequeño en la fase de entrenamiento.

2.3.3. Generalización de una RNA

Por generalización se entiende la capacidad de la red de dar una respuesta correcta ante patrones que no han sido empleados en su entrenamiento. Existe un error de generalización, que se puede medir empleando un conjunto representativo de patrones diferentes a los utilizados en el entrenamiento. De esta manera, se puede entrenar una red neuronal haciendo uso de un conjunto de aprendizaje y comprobar su eficiencia real, o error de generalización, mediante un conjunto de pruebas. Al principio la red se adapta progresivamente al conjunto de aprendizaje, acomodándose al problema y mejorando la generalización. Sin embargo, puede suceder que el sistema se ajuste demasiado a las particularidades de los patrones empleados en el entrenamiento, aprendiendo incluso del ruido presente, por lo que crece el error que comete ante patrones diferentes a los empleados en el entrenamiento (error de generalización). La validación cruzada consiste en entrenar y validar a la vez para detenerse en el punto óptimo; esta técnica se utiliza ampliamente en la fase de desarrollo de una red neuronal supervisada (Bishop, 1995).

Los modelos que implementan las redes neuronales son de elevada complejidad, por lo que pueden aprender (memorizar) casi cualquier cosa, motivo por el cual incurren fácilmente en sobre aprendizaje. Una buena consideración es ajustar el tamaño de la red a la complejidad del problema que se está tratando, debiendo limitar en lo posible su tamaño, evitando así la aparición de sobre entrenamiento. Se ha demostrado que el número efectivo de parámetros es generalmente menor que el número de pesos (Bishop, 1995). Ése es el motivo por el que la parada temprana del aprendizaje evita el sobre entrenamiento, pues es equivalente a limitar el número de parámetros de la red (sin modificar para nada la arquitectura actual, es decir, el número de pesos). La capacidad de generalización de la red la determinan en buena medida las siguientes tres circunstancias: la arquitectura de la red, el número de ejemplos de entrenamiento, y la complejidad del problema. En conclusión existen dos formas de luchar contra el fenómeno de sobre entrenamiento: la parada temprana y limitar el tamaño de la arquitectura de la red.

2.4. Tipos de Neuronas y Funciones de Activación

La operación de las Neuronas Artificiales está inspirado en su contraparte biológica. Cada neurona tiene varias entradas (correspondiente a la sinapsis de la neurona biológica) y una salida simple, el Axón. Cada entrada está caracterizada por un nivel de certeza que indica la influencia de la entrada correspondiente en la salida de la neurona. La neurona calcula un equivalente total de la suma de los pesos de las entradas individuales, ecuación 2.9.

$$net = \sum_{i=1}^{n} w_i x_i \tag{2.9}$$

El resultado es comparado con un valor constante llamado umbral y la señal de salida es calculada como una función de su diferencia (net − t). Esta función es llamada función de transferencia o función de activación. Los pesos de la entrada, el nivel de umbral y la función de activación son parámetros que describen completamente a la neurona artificial. Dependiendo del tipo de neurona artificial la función de activación puede tener diferentes formas. Hay neuronas analógicas que usan funciones de activación real continua y neuronas discretas las cuales usan funciones de activación que son discontinuas. Las neuronas bipolares pueden generar salidas positivas o negativas mientras que una unipolar genera solo valores positivos. En el caso de las neuronas bipolares analógicas, la función de activación más popular está dada por 2.10. La salida varía de manera continua entre -1 y 1, dependiendo de la señal de entrada, éste puede ser cualquier valor real, figura 2.7.

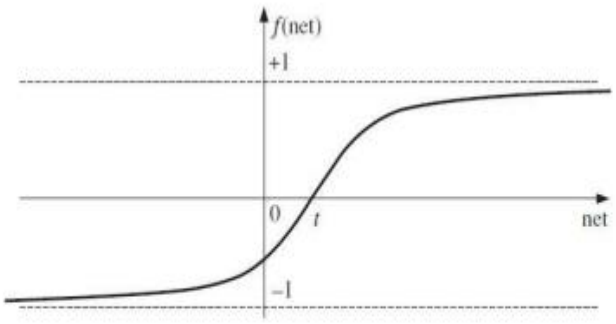


Figura 2.7 Función de activación sigmoidal para neurona analógica bipolar (λ = 1)

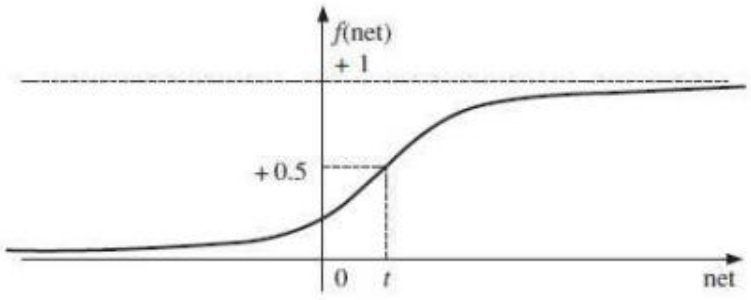
El parámetro λ es una constante que controla la pendiente de la gráfica de la función de activación. Algunos autores consideran el valor de λ = 1 para

simplificar los cálculos, mientras que otros operan con el formato más general presentado en 2.3, pero los resultados fundamentales y las propiedades de la RNA permanecen válidos en ambas situaciones. La función en 2.3 es parte de una clase de función de transferencia más grande llamada “funciones Sigmoidales”. Lo que tienen en común es la forma gráfica y la propiedad de ser derivable, lo cual es esencial en algunas aplicaciones.

$$f_1(net) = \frac{2}{1 + \lambda e^{-(net - t)}} - 1 \tag{2.10}$$

Una función de activación alternativa es la presentada en la ecuación 2.11 y es parte del grupo de las funciones sigmoidales como se muestra en la ecuación 2.12 ésta tiene los mismos valores límites como la función f1. Las neuronas unipolares analógicas son similares a las bipolares con la diferencia que las señales de salida solo toma valores entre 0 y 1, figura 2.8. Su función de activación está descrita por 2.13.

$$f_2(net) = \frac{1 - \lambda e^{-(net - t)}}{1 + \lambda e^{-(net - t)}} \tag{2.11}$$

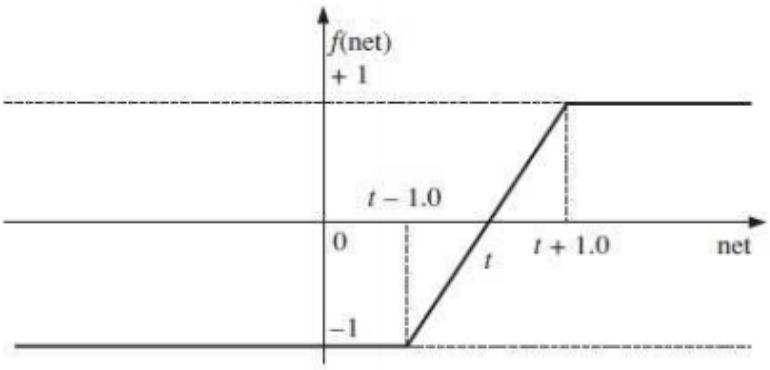


Fi **Figura 2.8** Función de activación sigmoidal para neurona analógica unipolar (λ = 1)

$$\begin{cases} \lim_{net \to \infty} f_2(net) = 1 \\ \lim_{net \to -\infty} f_2(net) = -1 \end{cases} \tag{2.12}$$

$$f_3(net) = \frac{1}{1 + \lambda e^{-(net - t)}} - 1 \tag{2.13}$$

No todas las funciones de activación son sigmoidales. La función de activación escalón lineal se presenta en 2.14 no derivable en 2 puntos: net = t − 1.0 y net = t + 1.0 figura 2.9.



Fi **Figura 2.9** Función de activación no sigmoidal

$$f_4(net) = \begin{cases} -1 & \text{if } net < t - 1.0 \\ net - t & \text{if } net \in [t - 1.0; t + 1.0] \\ 1 & \text{if } net > t + 1.0 \end{cases} \tag{2.14}$$

Las neuronas discretas utilizan funciones de activación de tipo umbral. La neurona discreta bipolar se asocia con la función de activación descrita en 2.15, mientras que la de tipo bipolar usa la función de activación mostrada en 2.16. Estas dos funciones pueden ser considerados casos limitados ($\lambda \rightarrow \infty$) de las funciones de transferencia sigmoidal presentados en 2.10 y 2.13.

$$f_5(net) = \begin{cases} 1 & net \geq t \\ -1 & net < t \end{cases} \tag{2.15}$$

$$f_6(net) = \begin{cases} 1 & net \geq t \\ 0 & net < t \end{cases} \tag{2.16}$$

En los últimos años, se han introducido tipos más sofisticados de neuronas y funciones de activación con el fin de resolver diferentes tipos de problemas prácticos. En particular, las neuronas de base radial han demostrado ser útiles para sistemas de control y aplicaciones en sistemas de identificación. Estas neuronas utilizan las denominadas funciones de activación de base radial. La ecuación 2.17 presenta la forma más utilizada para una función de este tipo, donde “x” es el vector n dimensional de señales

de entrada y “t” un vector constante de la misma dimensión mientras || • || es la norma euclidiana en el espacio dimensional n.

$$f_7(x) = exp(-||x - t||^2) \tag{2.17}$$

Prácticamente f7 nos muestra que tan cerca está el vector “x” del vector “t” en este espacio de n dimensiones. Entre más cerca este “x” de “t” más grande es f7(x); si x=t entonces f7 (x) =1. La campana de Gauss típica se obtiene para el caso unidimensional, mientras que para dos dimensiones es ilustrada en la figura 2.10. Obviamente, un tipo tal como está neurona está muy lejos del modelo biológico, pero esto es irrelevante, ya que resulta útil para ciertas aplicaciones técnicas.

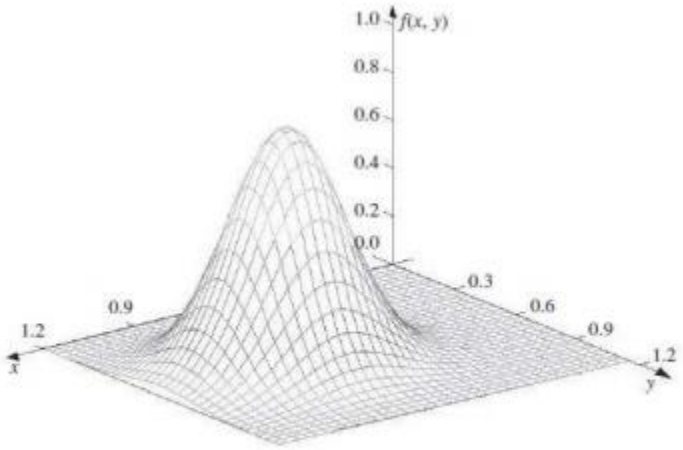


Figura 2.10 Función de activación radial para dos dimensiones

2.5. Algoritmos de Entrenamiento

Una de las características más importantes de las redes neuronales es su capacidad de aprender (ser entrenada) y mejorar su operación usando un conjunto de ejemplos llamados patrones de entrenamiento.

El proceso de entrenamiento está controlado por algoritmos matemáticos clasificados en: constructivos y no constructivos. Los algoritmos de entrenamiento no constructivos adaptan solo los pesos de las conexiones y los niveles de umbral, los algoritmos constructivos modifican las características de la red incluyendo su arquitectura (neuronas e incluso capas son adicionadas o eliminadas si es necesario). Todos los algoritmos modifican el peso y el umbral de la neurona basado en cálculos que analizan la respuesta

de la red a entradas particulares. Las modificaciones se realizan de tal manera que las salidas resulten más cercanas a lo esperado. Dependiendo de la naturaleza de los patrones de entrenamiento, hay dos categorías de algoritmos: supervisados y no supervisados (Haykin y Network, 2004). Los algoritmos supervisados usan patrones de entrenamiento compuestos de pares de entradas y salidas respectivas, los algoritmos no supervisados usan solo vectores de entrada. En el caso de algoritmos supervisados, el proceso de entrenamiento está controlado por una entidad externa (El "maestro") que es capaz de establecer si las salidas de la red son los adecuados para las entradas y cuál es el tamaño del error. Entonces los parámetros de la red son modificados de acuerdo al método de corrección particular definido por cada algoritmo de entrenamiento. En el caso de los métodos no supervisados (La regla Hebbian, algoritmo el "Ganador se lleva todo", etc.), no hay forma de conocer los resultados esperados. La red evoluciona de acuerdo a la experiencia ganada desde los vectores de entrada previos. Los valores de los pesos convergen a un ajuste de los valores finales dictados por los valores de las entradas usadas como patrones de entrenamiento en conjunción con el algoritmo de entrenamiento particular.

La familia de algoritmos de entrenamiento no supervisados son principalmente usados para procesamiento de señales e imágenes, donde la clasificación de patrones, la agrupación de datos o algoritmos de compresión son involucrados. Los problemas de ingeniería de control son mejor abordados por métodos de entrenamiento supervisado, como la relación entre las entradas y las salidas deseadas son mejor definidas y fáciles para controlar.

2.5.1. El Algoritmo de Retro Propagación

El algoritmo de entrenamiento supervisado más popular es el llamado "error de retropropagación", o simplemente "retropropagación". El algoritmo de entrenamiento implica una estructura de Red Neuronal Artificial hacia adelante (FFANN) por sus siglas en inglés, estructura de red construida con neuronas y funciones de activación sigmoidal. El algoritmo de retropropagación es un método de gradiente adecuado para minimizar el error de operación total de la red neuronal. El error total es una función definida por la ecuación 2.18 donde O_i^ref es el vector columna de la salida de referencia y O_i es el vector columna de la salida actual de la red correspondiente al patrón de entrada número "i". El error total Err es la suma de todos los errores correspondientes a los patrones de entradas np.

$$Err = \sum_{i=1}^{n_p} (O_i^{ref} - O_i)^T (O_i^{ref} - O_i) = \sum_{i=1}^{n_p} ||O_i^{ref} - O_i||^2 \quad (2.18)$$

Para cada paso de entrenamiento, el vector de todos los pesos de las neuronas y valores de umbrales (W) está siendo actualizada en cada uno de las iteraciones en la que el error total (Err) es disminuido. El vector W puede ser asociado a un punto en el espacio dimensional NW (parámetros de espacio), donde NW es el número total de pesos y umbrales en la red neuronal. La forma más eficiente para realizar la actualización es cambiar el punto W a lo largo de la curva indicada por el gradiente del error total (∇Err). Este principio está definido por la ecuación 2.19 donde W (t) es el parámetro vector durante el ciclo actual de entrenamiento, y W (t + 1) es el parámetro vector en el siguiente ciclo de entrenamiento y η es la constante del rango de entrenamiento. Idealmente el algoritmo se detiene cuando el error total (Err) es cero. En la práctica, éste se detendrá cuando el error total (Err) sea considerado despreciable.

$$W(t+1) = W(t) - \eta \nabla Err = W(t) - \eta \left(\frac{\partial Err}{\partial_{W_1}} \frac{\partial Err}{\partial_{W_2}} \frac{\partial Err}{\partial_{W_3}} \dots \frac{\partial Err}{\partial_{W_{N_W}}}\right)^T \quad (2.19)$$

Para el cálculo práctico del error del gradiente ∇Er r los componentes en el vector W usualmente son re-arreglados como una matriz de tres dimensiones. La matriz tiene un número de capas igual al número de capas de neuronas de la red neuronal. Cada capa rectangular es una matriz de dos dimensiones conteniendo una línea por cada neurona en la capa correspondiente de la red neuronal. Cada línea incluye los pesos de entrada y su nivel de umbral de la neurona. Por lo tanto, los elementos W_{jkm} en la matriz de tres dimensiones es el peso “m” de la neurona “k” situado en la capa “j” dentro de la red neuronal. El nivel de umbral corresponde al elemento final en cada línea y no se trata de manera diferente a los pesos de entrada, ya que puede ser considerado como un peso extra suministrado con una señal de entrada constante −1. En el caso de redes de una capa, los componentes del gradiente de error son calculados de acuerdo a la ecuación 2.20 donde la señal de entrada xm es la señal de entrada correspondiente.

$$\frac{\partial Err}{\partial_{1km}} = \frac{\partial}{\partial_{1km}} \sum_{i=1}^{n_p} ||O_i^{ref} - O_i||^2 = -\sum_{i=1}^{n_p} 2(o_{ik}^{ref} - o_{ik}) \frac{df}{d(net_{ik} - t_k)} x_m \quad (2.20)$$

Las funciones de activación sigmoidal bipolares dadas por 2.10 tienen la propiedad indicada en la ecuación 2.21 por tanto la ecuación 2.20 puede ser transformada en 2.22 (Zurada, 1992).

$$\frac{df}{d(net-t)} = \frac{1}{2}(1-f^2) \tag{2.21}$$

$$\frac{\partial Err}{\partial_{1km}} = -\sum_{i=1}^{n_p}(o_{ik}^{ref} - f_{i1k})(1-f_{i1k}^2)x_m \tag{2.22}$$

Si la red tiene más de una capa entonces la señal de entrada xm es generada por la neurona “m” en la capa 2 de tal manera que “xm” es remplazada por f2m. Las ecuaciones 2.24 y 2.25 ilustran los cálculos para las neuronas en la capa dos y tres.

$$\frac{\partial Err}{\partial_{1km}} = -\sum_{i=1}^{n_p}(o_{ik}^{ref} - f_{i1k})(1-f_{i1k}^2)f_{2m} \tag{2.23}$$

$$\frac{\partial Err}{\partial_{2km}} = -\sum_{i=1}^{n_p}\sum_{x}(o_{ix}^{ref} - f_{ilx})(1-f_{i1x}^2)W_{1xk}(1-f_{i2k}^2)f_{3m}) \tag{2.24}$$

$$\frac{\partial Err}{\partial_{3km}} = -\sum_{i=1}^{n_p}\sum_{x}(o_{ix}^{ref} - f_{ilx})(1-f_{i1x}^2)\sum_{y}W_{1xy}(1-f_{i2y}^2)W_{2yk}(1-f_{i3k}^2)f_{4m}) \tag{2.25}$$

Los cálculos anteriores pueden ser generalizados para cualquier número de capas de las redes neuronales.

$$\begin{cases} \frac{\partial Err}{\partial W_{jkm}} = -\sum_{i=1}^{n_p}\sum_{x}(o_{ix}^{ref} - f_{i1x})\delta_{i1x} \\ \delta_{i1x} = (1-f_{i1x}^2)\sum_y W_{1xy}\delta_{i2y} \\ \delta_{i2x} = (1-f_{i2x}^2)\sum_y W_{2xy}\delta_{i3y} \\ ..... \\ \delta_{i(k-1)x} = (1-f_{i(k-1)x}^2)\sum_y W_{((k-1)xy}\delta_{iky} \\ \delta_{ikx} = \begin{cases} 0 & x \neq m \\ 1-f_{i(k-1)x} & x = m \end{cases} \end{cases} \tag{2.26}$$

Cada componente del gradiente está determinado siguiendo los procesos iterativos descritos en 2.26. Resultados similares son obtenidos para todos los tipos de funciones de activación sigmoidales. Estas ecuaciones justifican el nombre del algoritmo de entrenamiento ya que el error de salida de la FFANN afecta los cálculos referidos de cualquier peso porque su influencia se propaga hacia atrás hacia la entrada desde una capa a la siguiente de acuerdo a 2.26.

El algoritmo de retropropagación o propagación hacia atrás se enfrenta al problema bien conocido de cualquier algoritmo de optimización no lineal utilizando el método del gradiente, éste puede atascarse en un mínimo local de la función objetivo (La función de “Err” en este caso), por lo tanto, el algoritmo de propagación hacia atrás no está garantizado para generar una solución satisfactoria en todos los problemas de asociación de entrada-salida y las arquitecturas FFANN.

El resultado de entrenamiento dependerá de múltiples factores (Zurada, 1992):

- Arquitectura de la red (número de capas y número de neuronas en cada capa).
- Valores de parámetros iniciales W (0).
- Detalles del mapeo entrada-salida.
- Selección de patrones de entrenamiento (pares de entradas y sus correspondientes salidas deseadas).
- La constante del rango de entrenamiento η.

La retropropagación no es un algoritmo constructivo; la arquitectura de red tiene que ser elegido de antemano. Desafortunadamente, no hay un conjunto claramente definido de normas que deben seguirse a fin de decidir cuál es la arquitectura más adecuada para un problema. La elección de la arquitectura es el resultado de un proceso de ensayo y error apoyada por la experiencia previa. Sin embargo, se ha demostrado matemáticamente que cualquier asignación de entrada-salida puede ser aprendido por un FFANN con una sola capa oculta, siempre que el número de neuronas en la capa oculta sea lo suficientemente grande para el problema a resolver (Mirchandani y Cao, 1989). Esto significa que si una red neuronal demuestra incapacidad para aprender cómo realizar una determinada tarea, la solución es posible aumentando el número de neuronas en la capa o capas ocultas.

Una solución diferente es el de reiniciar el algoritmo con otro conjunto de parámetros iniciales W (0). Esta solución se basa en la suposición de que se ha generado el fallo anterior debido a que se detuvo el proceso al encontrar un mínimo local. La trayectoria del vector W en el espacio de parámetros depende de su punto de partida W (0), por lo tanto, la situación puede ser evitada cambiando los pesos y umbrales iniciales.

Otro aspecto importante es la elección de un conjunto de patrones de entrenamiento adecuado, de modo que si el número de diferentes valores de entrada es finito, el conjunto de patrones de entrenamiento cubre todas las posibilidades. Por otro lado, si este número es infinito (por ejemplo, cuando las entradas son señales analógicas), o si el número es demasiado grande, entonces sólo una selección de combinaciones de entrada se puede utilizar para entrenar a la red neuronal.

La calidad del proceso de entrenamiento está influenciada por la forma en que se genera el conjunto de datos de entrenamiento. Si los datos de entrenamiento son establecidos adecuadamente y cubren todos los aspectos de la correspondencia de entrada y salida, entonces, la red será capaz de generar respuestas correctas para las entradas que no fueron utilizadas durante el período de entrenamiento, a esta propiedad se le conoce como “generalización” y se hace posible por el hecho de que cualquier FFANN realiza una interpolación en un espacio n-dimensional (donde “n” es la longitud de los vectores de entrada) (Hornik y cols.,1989). La interpolación se realiza en base a la información proporcionada por los vectores de entrada utilizados durante el período de entrenamiento. Si la asignación de entrada-salida es continua y suave, entonces la red se generaliza fácilmente y da respuestas correctas, como resultado de un entrenamiento realizado con sólo unos vectores de entrada. Si la asignación de entrada-salida es robusto y complicado, entonces se requiere un gran número de vectores de entrada para un proceso de entrenamiento adecuado. El proceso de entrenamiento puede requerir cientos, miles o incluso millones de pasos del tipo descrito por 2.19. El número real depende de la naturaleza de la asignación de entrada-salida y de la constante de la tasa de aprendizaje (η). Un valor grande para η acelera el proceso de formación, pero también aumenta la probabilidad de que el vector W oscile en torno a la solución final sin poder llegar a ella. Un pequeño valor de η aumenta las posibilidades de obtener la solución deseada, sin embargo también aumenta el número de ciclos para el entrenamiento.

2.5.2. Algoritmo derivado del Método de Retro Propagación

Una serie de nuevos algoritmos se han derivado en las últimas dos décadas a partir del método de retropropagación clásico. Estos traen mejoras en el proceso de entrenamiento mediante la aceleración de la convergencia y la mejora de las posibilidades de encontrar una buena solución para este tipo de aplicaciones particulares.

Las mejoras propuestas se pueden resumir como:

- La tasa de aprendizaje η varía después de cada ciclo de entrenamiento. Se inicia con un valor grande y después disminuye progresivamente durante el proceso de entrenamiento. Por lo tanto, el proceso de entrenamiento es rápido al principio, y se evitan las oscilaciones finales porque el valor de η disminuye durante el proceso.

- Todos los parámetros de red son ajustables y tienen su propia tasa de aprendizaje ηi en particular. El algoritmo de retropropagación puede ser lento, porque el uso de un parámetro de tasa de aprendizaje única puede no satisfacer todas las variaciones de error complicados en el espacio de parámetros NW − dimensional.

- Por lo tanto, un valor de tasa de aprendizaje que es apropiado para el ajuste de un peso no está necesariamente adecuados para el ajuste de otro. Por lo tanto, el algoritmo de aprendizaje se describe por la ecuación 2.27.

$$W(t+1) = W(t) - (\eta_1 \frac{\partial Err}{\partial_{W_1}} \eta_2 \frac{\partial Err}{\partial_{W_2}} \eta_3 \frac{\partial Err}{\partial_{W_3}} ... \eta_{N_W} \frac{\partial Err}{\partial_{W_{N_W}}})^T \quad (2.27)$$

- Si una tasa de aprendizaje está asociado con cada parámetro de red, entonces, todas las tasas de aprendizaje se les permiten variar de un ciclo de entrenamiento a la siguiente. Métodos más sofisticados pueden calcular las constantes de tasa de aprendizaje basadas en las derivadas parciales de la función de error. Por lo tanto, si ηi es grande la influencia de W_i sobre el error es pequeño y viceversa ecuación 2.28.

$$\eta_i = \frac{1}{\frac{\partial Err}{\partial w_i} + K} \text{donde} K > 0 \quad (2.28)$$

- Si el signo de la derivada del error ∂Err/∂W_i oscila durante varias iteraciones consecutivas, el correspondiente parámetro ηi, tasa de aprendizaje, se reduce.

- La convergencia del entrenamiento se acelera por el que se completa el ajuste de peso actual con una fracción del ajuste de peso anterior, como se muestra en la ecuación 2.29. Este algoritmo se llama el método de impulso o momentum (Zurada, 1992) y al segundo término que indica la fracción del peso de ajuste más reciente se le llama plazo de impulso. El término momentum α es una constante seleccionada por el usuario con valores de entre (0.1 y 0.8).

$$W(t+1) = W(t) - \eta \nabla Err + \alpha [W(t) - W(t-1)] \quad (2.29)$$

- Las redes neuronales recurrentes de tiempo real deben ser entrenados de tal manera que aprendan una correlación temporal entre entradas y salidas. Un método de entrenamiento prometedor aplicable a este tipo de situaciones se le conoce como entrenamiento de retropropagación dinámica (Hecht-Nielsen, 1990) y se ha derivado de la clásica. La principal característica del nuevo método es que los vectores de entrada no se aplican al azar, sino de una manera rigurosamente definida. Los resultados esperados dependen tanto de la entrada actual como de las entradas anteriores, mientras que el cálculo de errores se lleva a cabo a nivel global para toda la serie temporal de los vectores de entrada.

Capítulo 3

Desarrollo

3.1. Procedimiento y Descripción de las Actividades Realizadas.

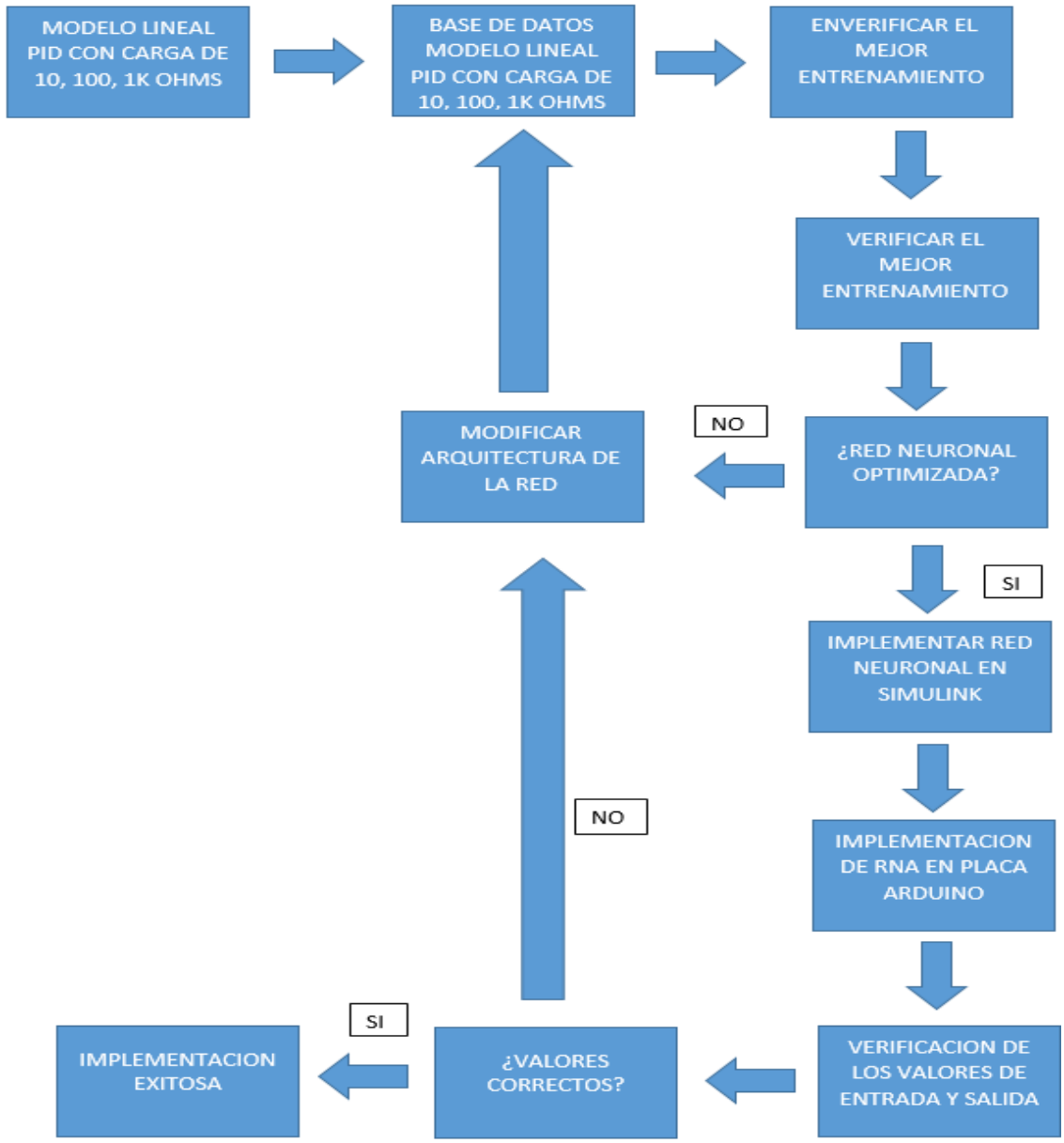


Figura 3.1 Procedimiento para la implementación del control neuronal

- **Modulación SPWM.**

La modulación spwm se realizó en la tarjeta Arduino Mega, primeramente se compararon las dos señales a modular las cuales fueron una señal triangular y una señal de media onda, mediante el código programado en la placa Arduino se obtuvo en la salida un tren de pulsos.

A continuación mostramos el código con el cual la modulación spwm fue hecha posible

```
#include <TimerThree.h>
int valor_y=0;
int valor_z=0;
int x=0;
int cruce=0;

void setup()
{
Serial.begin(9600);
analogReference(EXTERNAL);
pinMode(11,OUTPUT);
pinMode(10,OUTPUT);
pinMode(4,INPUT);
Timer3.initialize();
Timer3.setPeriod(100);
}

void loop()
{
valor_y=analogRead(A0);//onda triangular
valor_z=analogRead(A1);// senoidal rectificada
cruce=digitalRead(4);

if(cruce==HIGH)
{
  negativo();
}

if(cruce==LOW)
{
  positivo();
}
}

void positivo()
```

```
{

if((valor_z-valor_y)>0)
{
digitalWrite(11,HIGH);
delay(0.00008);// 80 nanosegundos
digitalWrite(10,LOW);
delay(0.0005);// 500 nanosegundos
}

if((valor_z-valor_y)<0)
{
digitalWrite(11,LOW);
delay(0.0005);
digitalWrite(10,HIGH);
delay(0.00008);
}
}

void negativo()
{

if((valor_z-valor_y)>0)
{
digitalWrite(10,HIGH);
delay(0.00008);
digitalWrite(11,LOW);
delay(0.0005);
}

if((valor_z-valor_y)<0)
{
digitalWrite(10,LOW);
delay(0.0005);
digitalWrite(11,HIGH);
delay(0.00008);
}
}
```

- **Diseño y elaboración del Rectificador de Media Onda con cruce por cero**

El circuito de la figura 3.2 se realizó mediante el software proteus, el cual nos proporcionara una señal de media onda la cual tiene un cruce por cero que nos permite detectar cuando la señal hace el cruce de positivo a negativo.

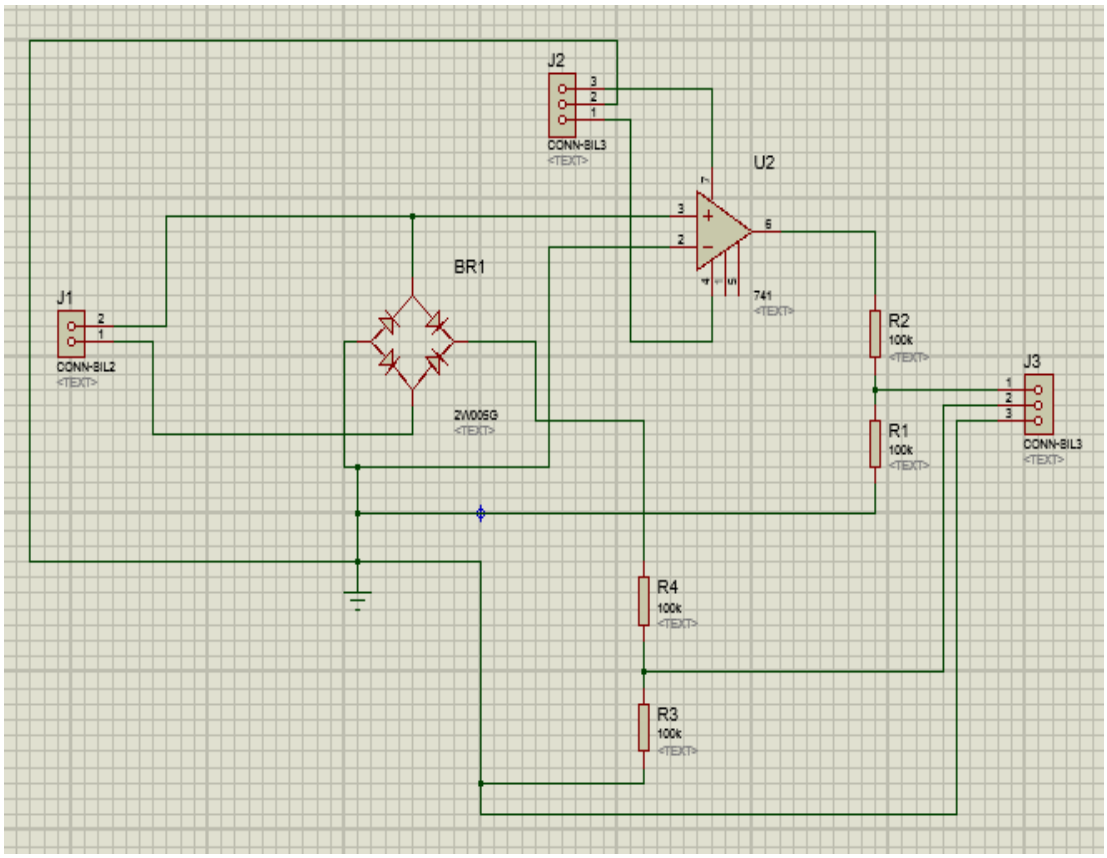


Figura 3.2 Circuito rectificador de media onda con cruce por cero

También se realizó el diseño en PCB para que este circuito pudiera ser elaborado en placa. Figura 3.3

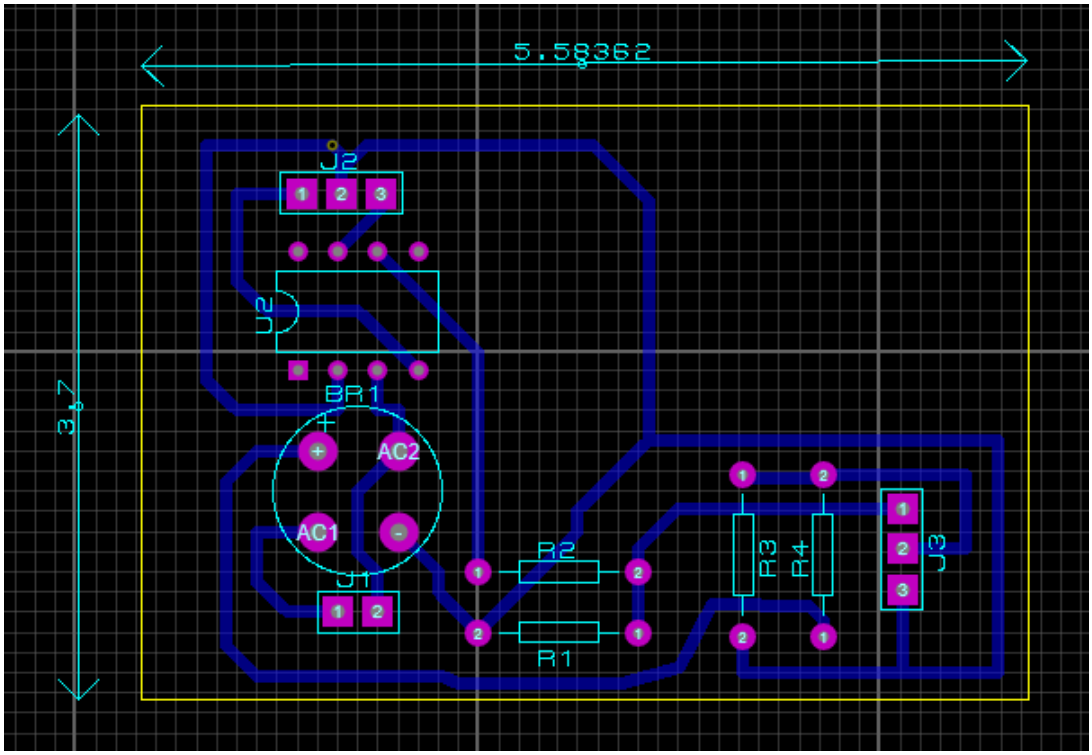


Figura 3.3 PCB Rectificador de media onda con cruce por cero

Una vez hecho el PCB, se procedió a planchar y a soldar cada uno de los componentes figura 3.4

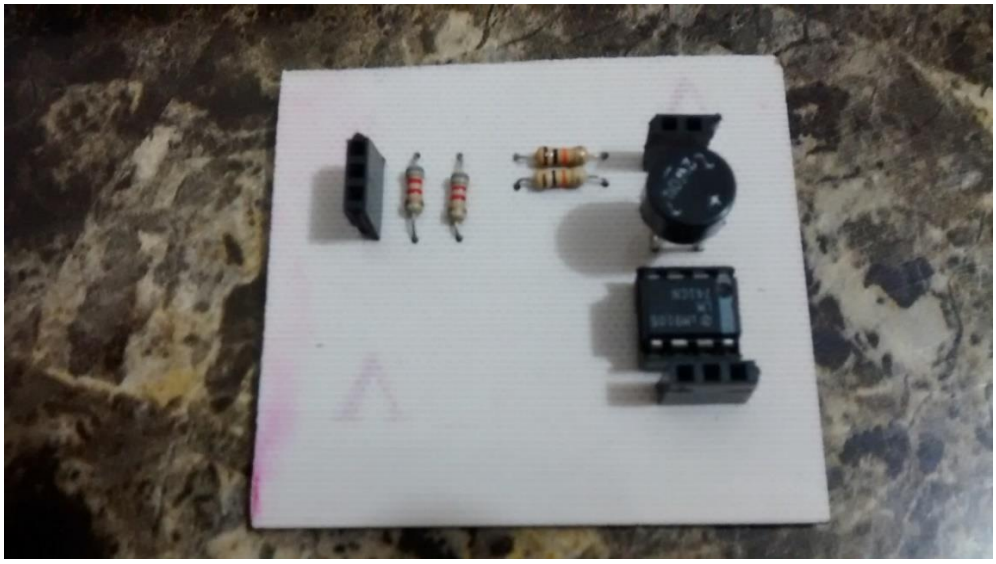


Figura 3.4 Rectificador de media onda con cruce por cero en placa

- **Diseño y elaboración del circuito Diente de Sierra.**

Para el diseño de este circuito se utilizó el software Multisim, ya que con otro software de simulación, el circuito presentaba problemas de ejecución a diferencia de Multisim con el cual se ejecutó sin problemas. Figura 3.5

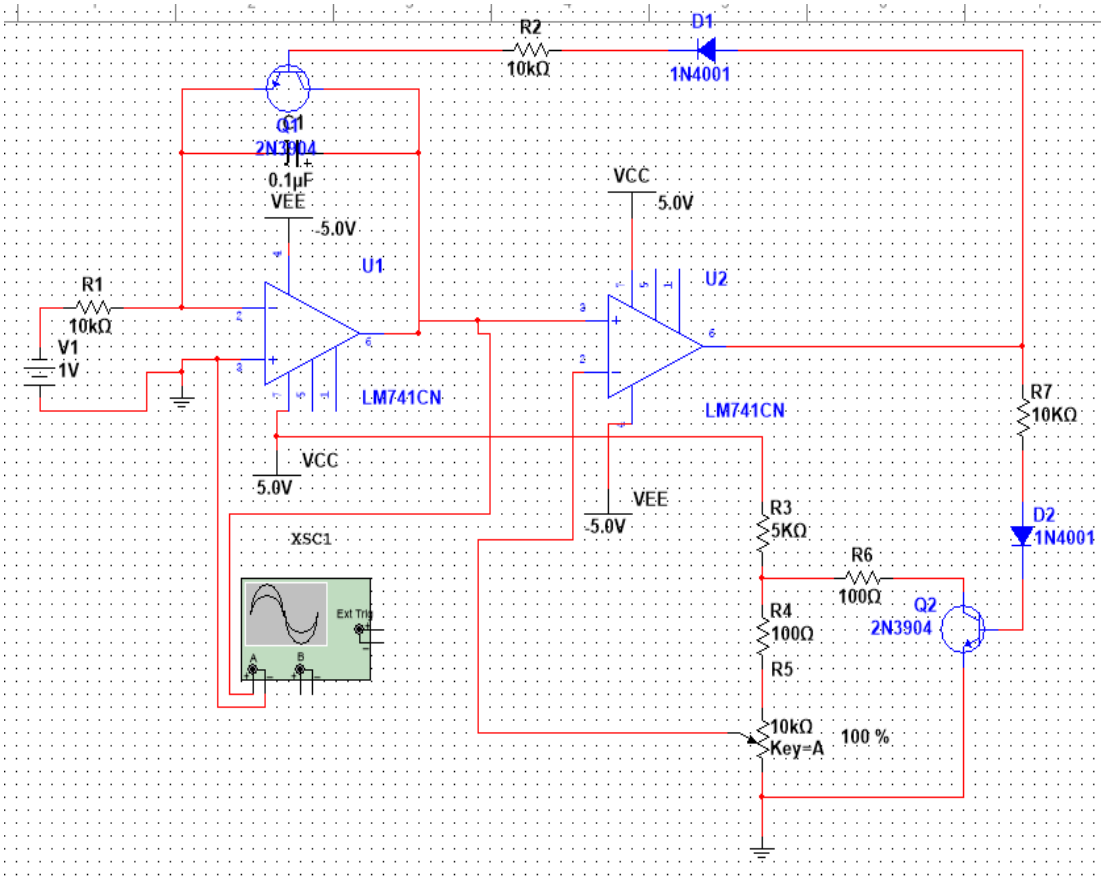


Figura 3.5 Circuito diente de sierra

El diseño en PCB tuvo que ser realizado en el software Proteus, ya que Multisim no cuenta con la opción de diseñar placas PCB figura 3.6

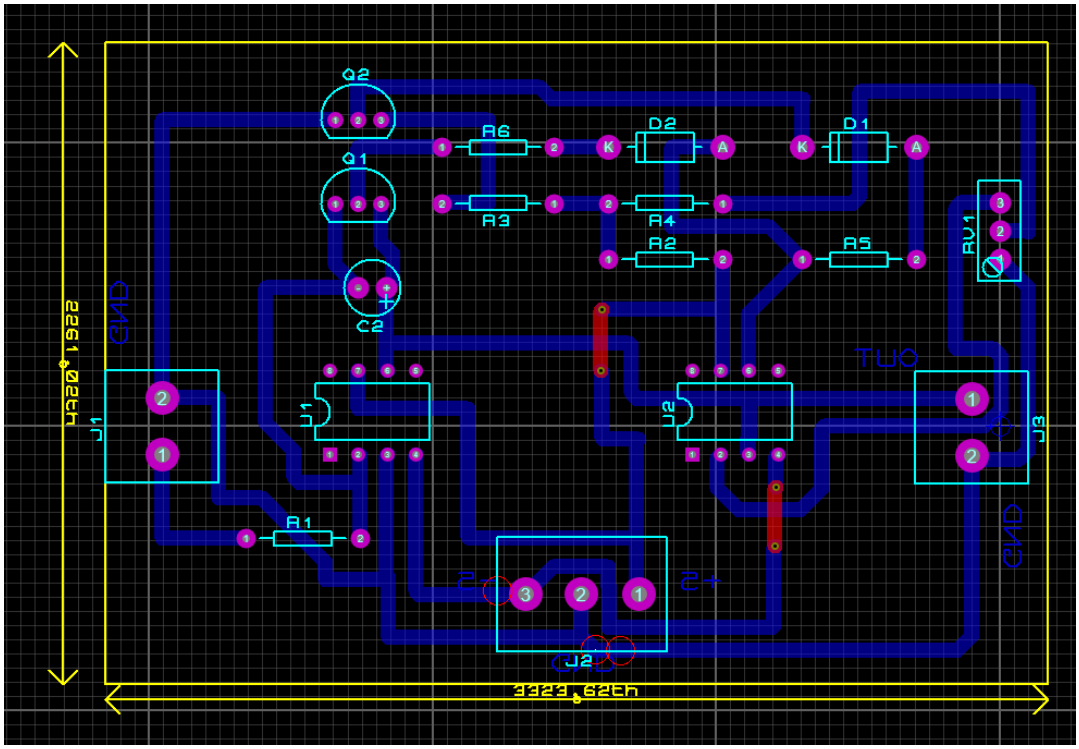


Figura 3.6 PCB Circuito diente de sierra

Una vez impreso el PCB se transfirió sobre la placa fenolica y a continuación se soldaron cada uno de los componentes figura 3.7

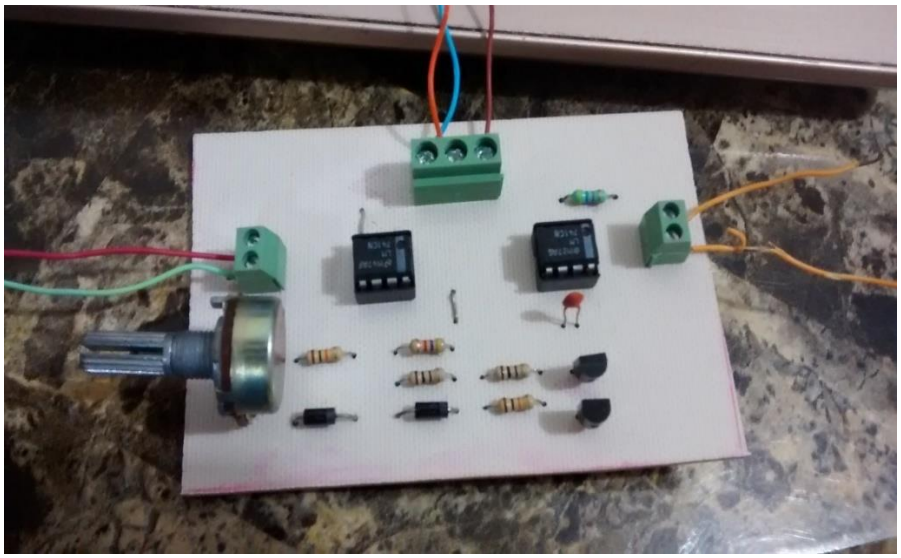


Figura 3.7 Circuito diente de sierra en placa

Implementación de los circuitos antes mencionados (circuito Rectificador de Media Onda con cruce por cero, circuito diente de sierra) y Arduino Mega figura 3.8 de esta manera ya se puede realizar la modulación spwm.

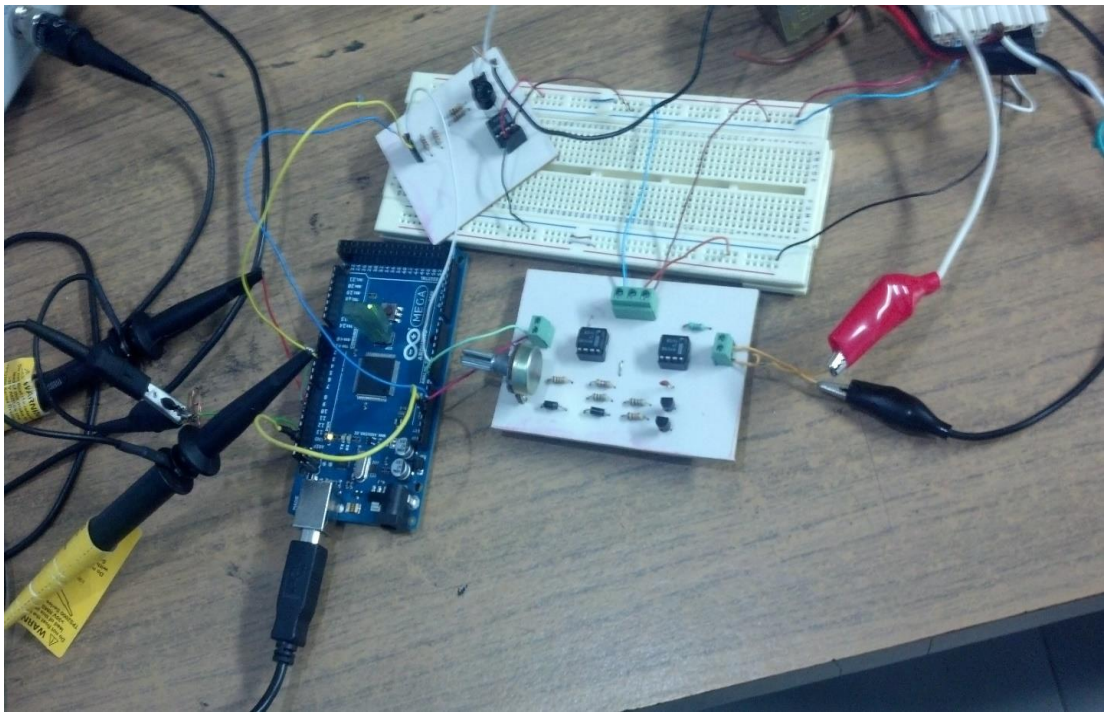


Figura 3.8 Implementación de todos los circuitos

- **Entrenamiento de la Red Neuronal.**

Para obtener los datos de entrenamiento se diseño el circuito figura 3.9 mediante el software Matlab con ayuda de la herramienta Simulink.

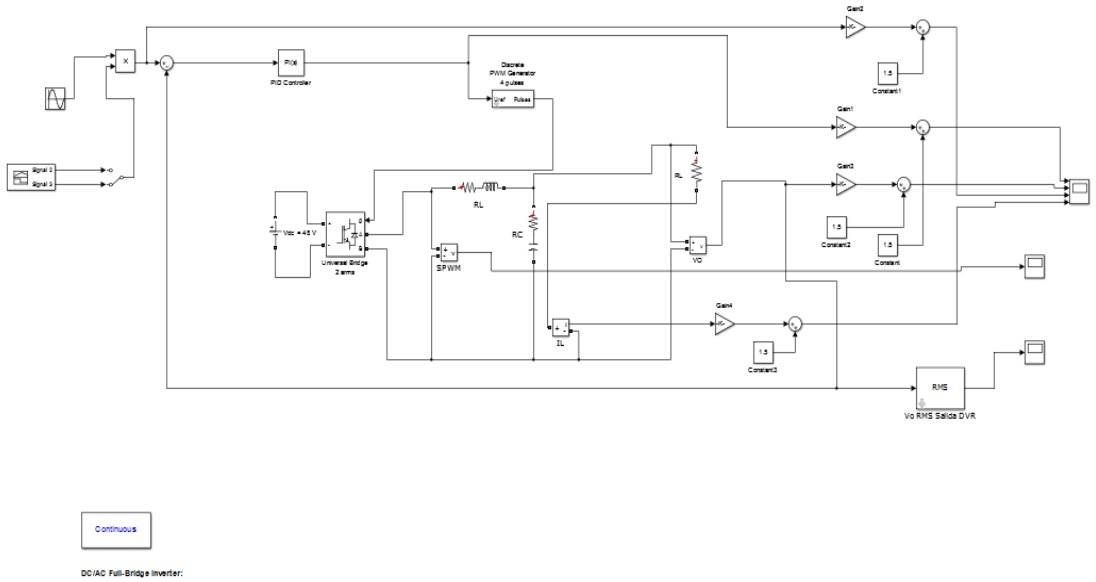


Figura 3.9 Modelo lineal PID para entrenamiento de la red neuronal

Utilizando Matlab se entrenó la red neuronal, se realizaron varias pruebas con el fin de que la red lograra obtener su máximo desempeño y así obtener mejores resultados.

Prueba 1.

La estructura de la red es 3, 2, 2 contando con tres entradas y una salida figura 3.10

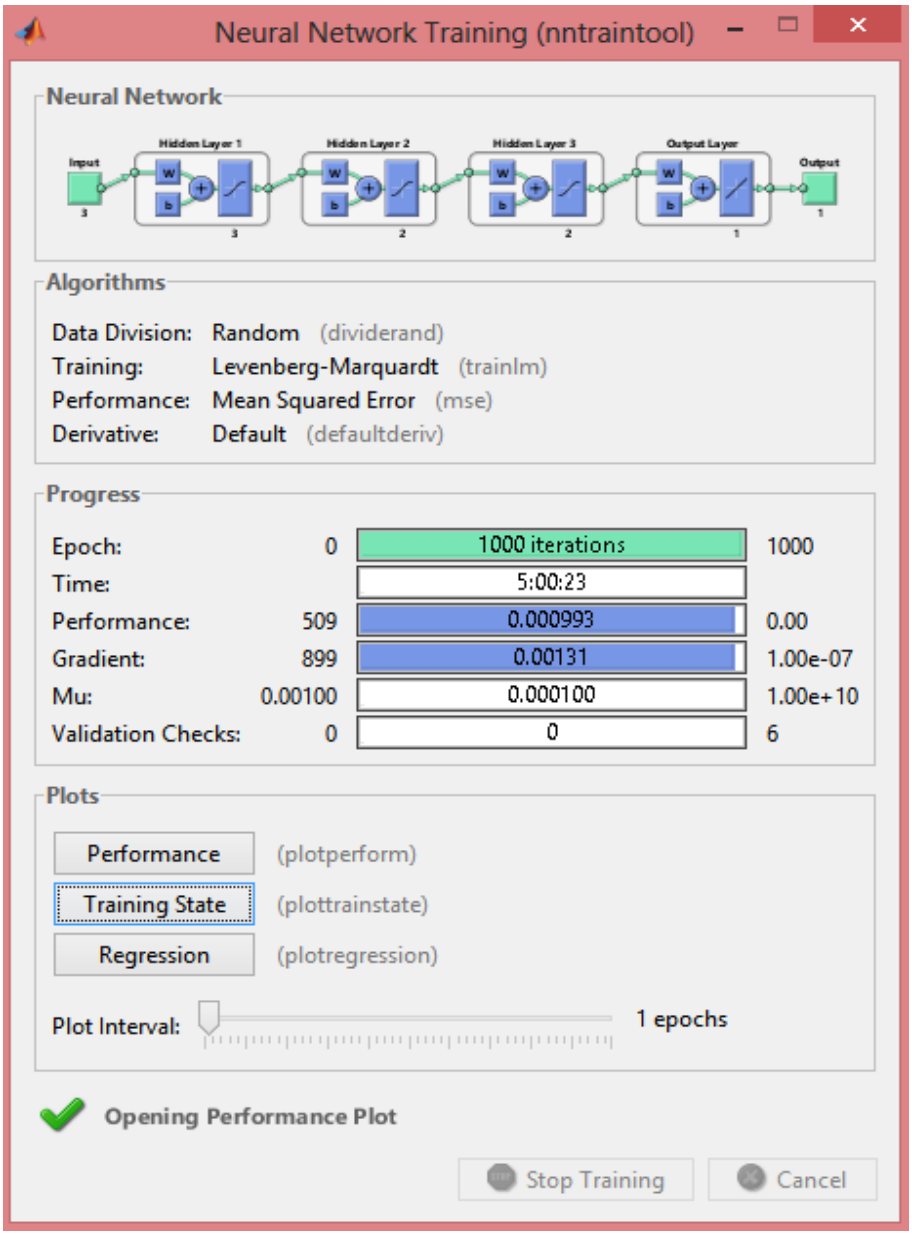


Figura 3.10 Interfaz de entrenamiento prueba uno.

Grafica de desempeño de la red neuronal figura 3.11

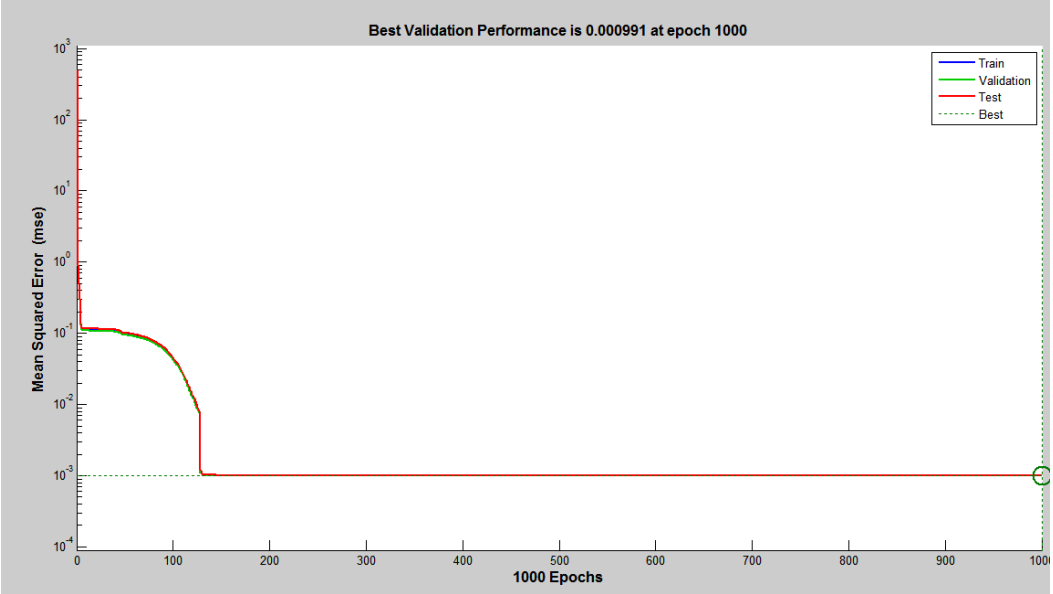


Figura 3.11 Grafica performance prueba uno.

Las siguiente grafica muestra el estado de la red durante el entrenamiento asi como sus validaciones figura 3.12

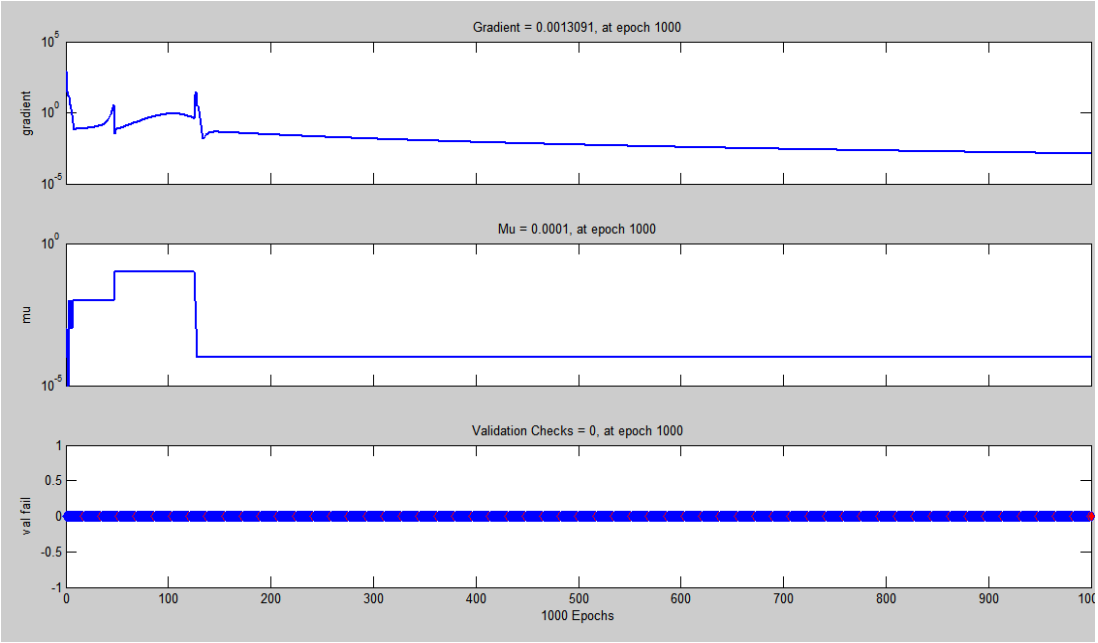


Figura 3.12 Grafica training state prueba uno

La grafica posterior nos indica que tan cerca estuvo la red neuronal de acercarse al objetivo (tarjet) figura 3.13

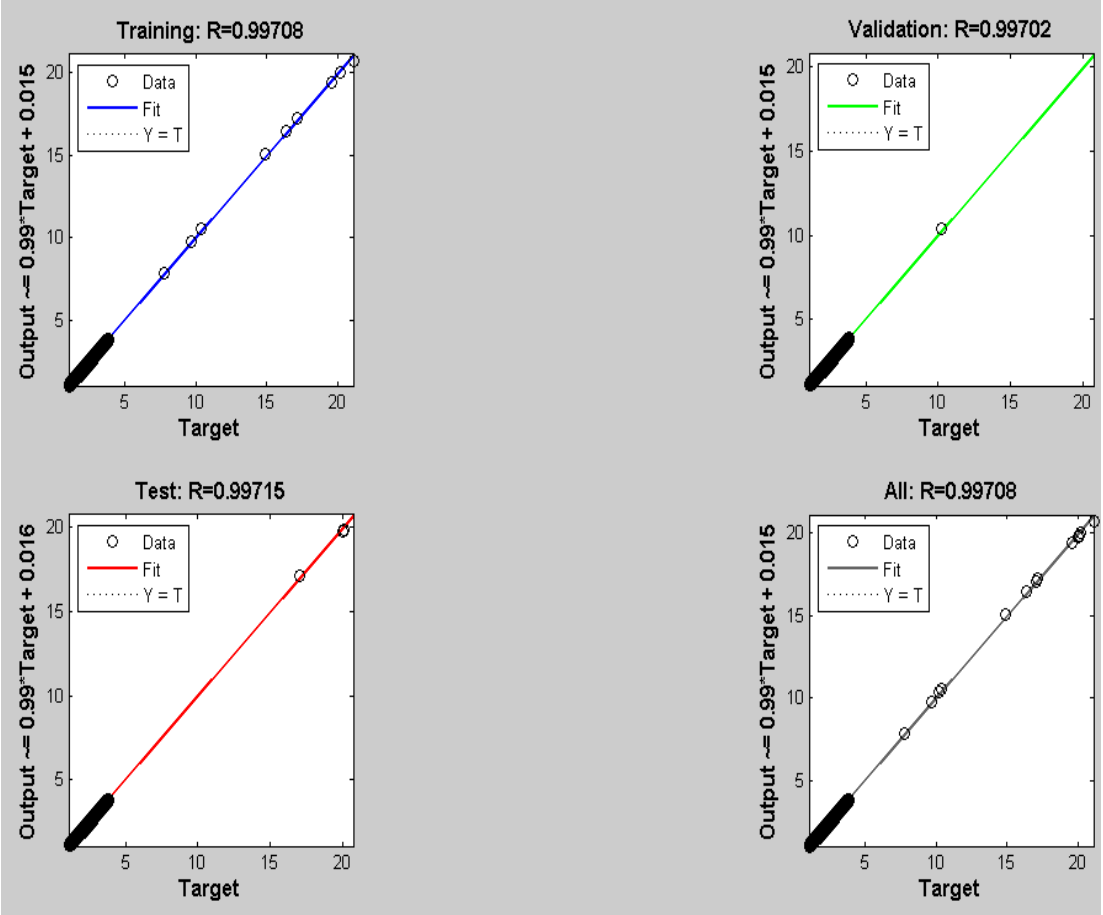


Figura 3.13 Grafica de regresión prueba uno

Prueba 2.

La estructura de la red es 3, 2, 2 contando con dos entradas y una salida figura 3.14

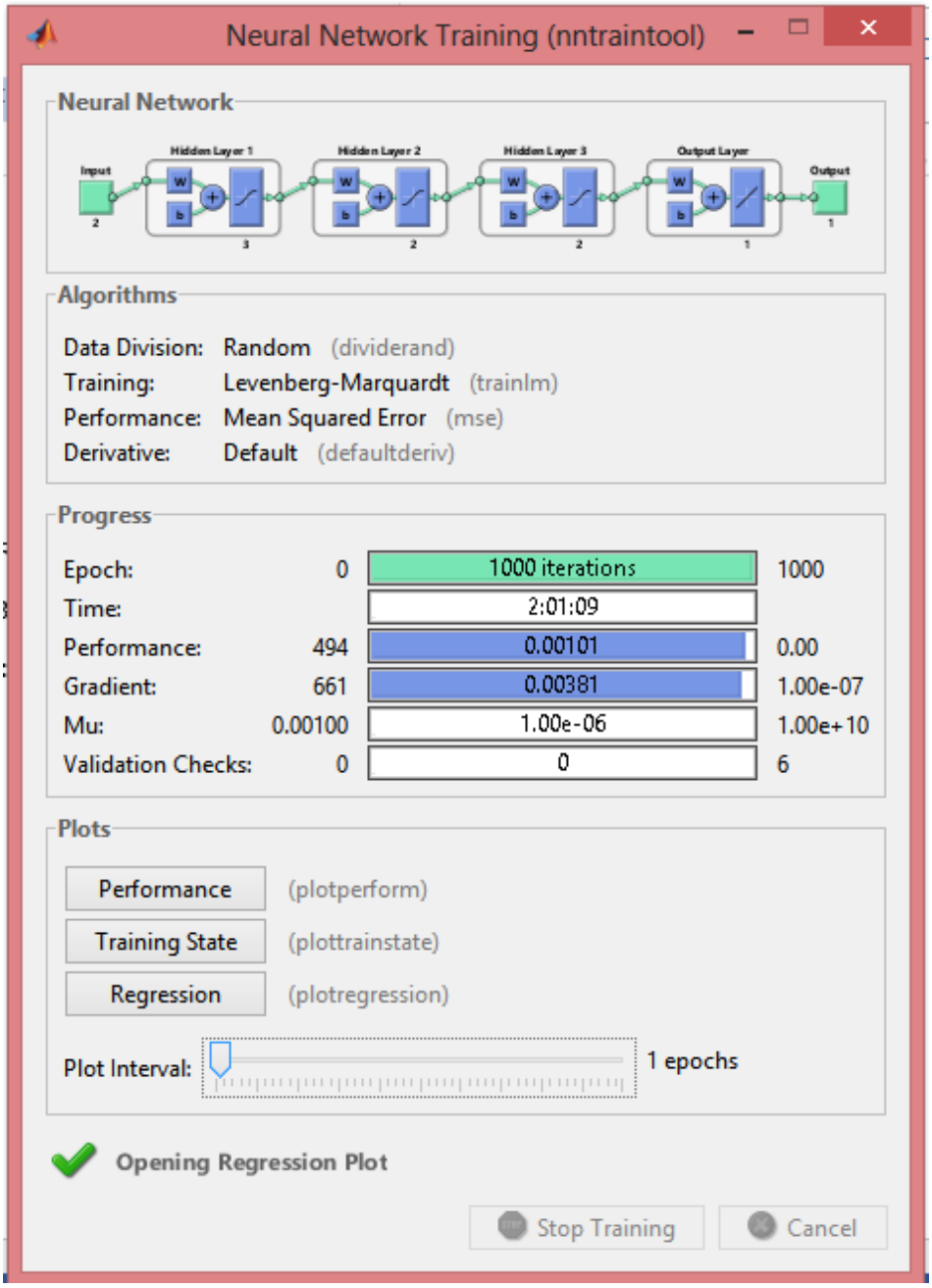


Figura 3.14 Interfaz de entrenamiento prueba dos.

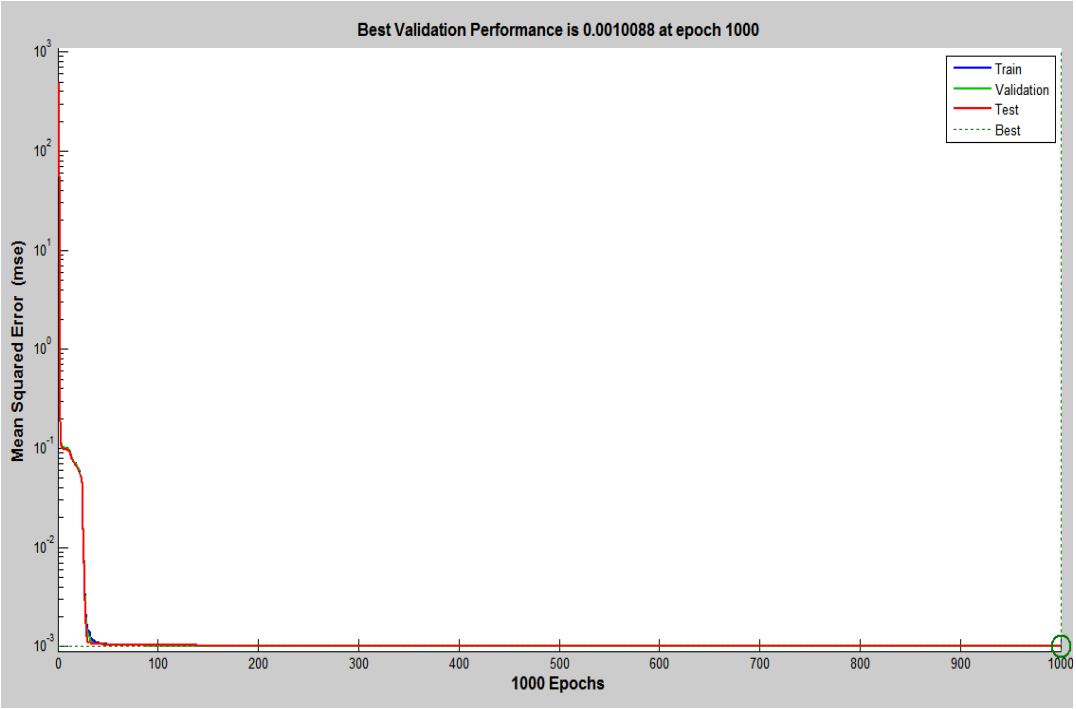


Figura 3.15 Grafica performance prueba dos

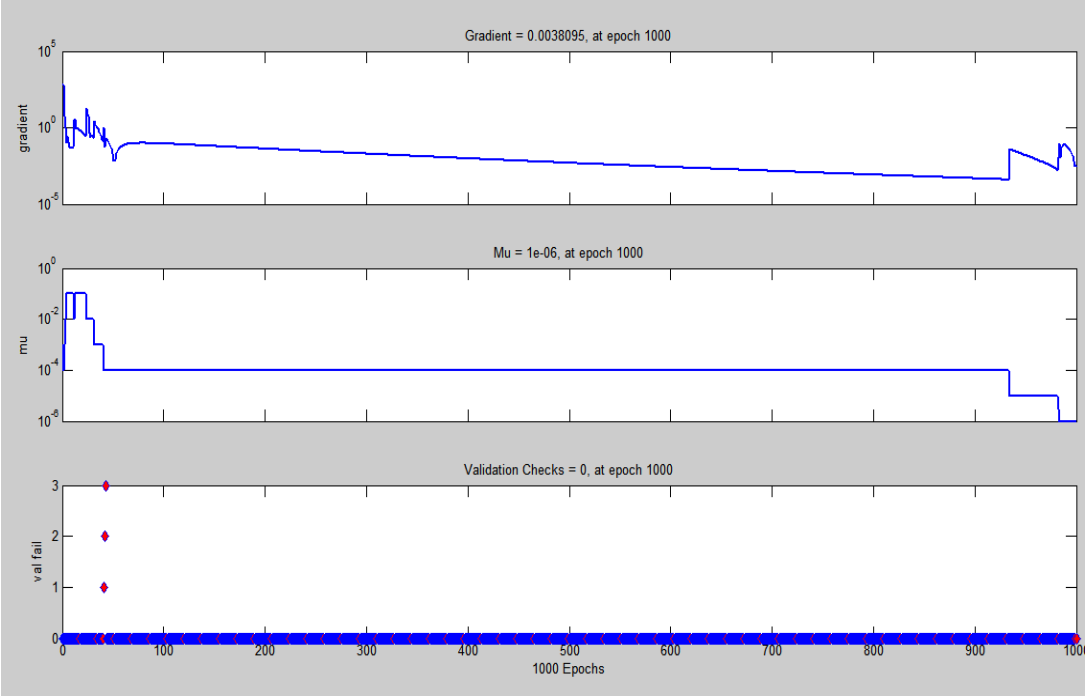


Figura 3.16 Grafica training state prueba dos

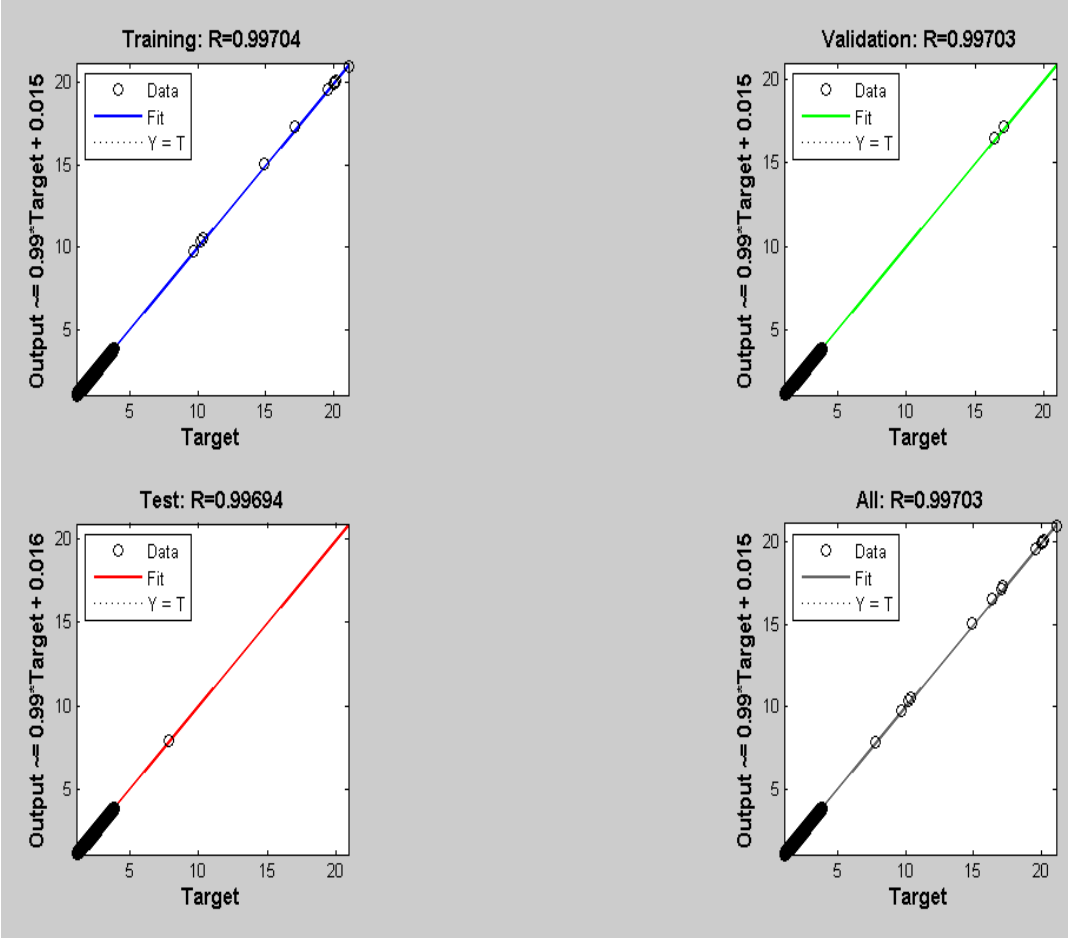


Figura 3.17 Grafica regression prueba dos

Prueba 3.

La estructura de la red es 3, 2, 2 contando con tres entradas y una salida figura 3.18

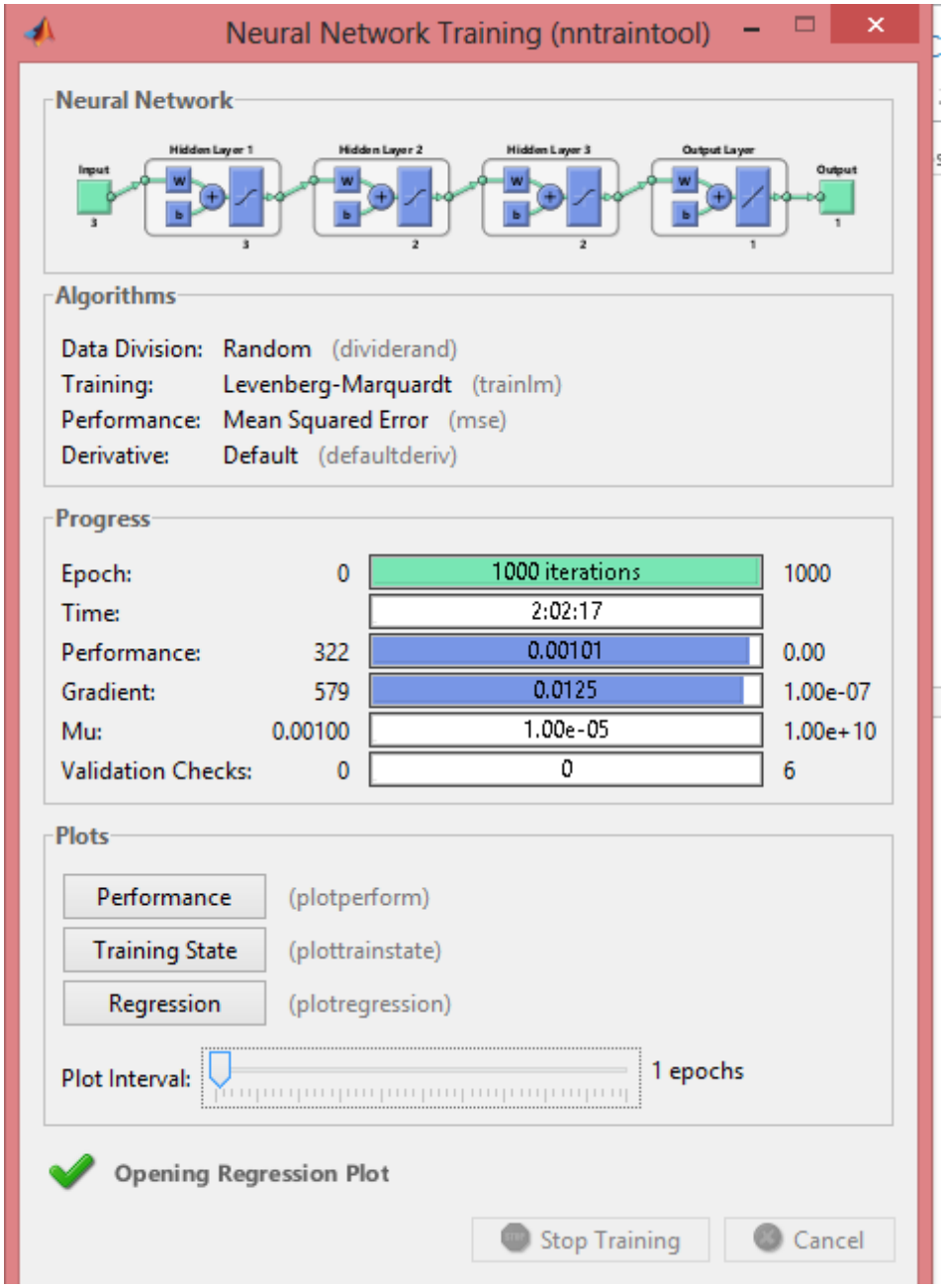


Figura 3.18 Interfaz de entrenamiento prueba tres.

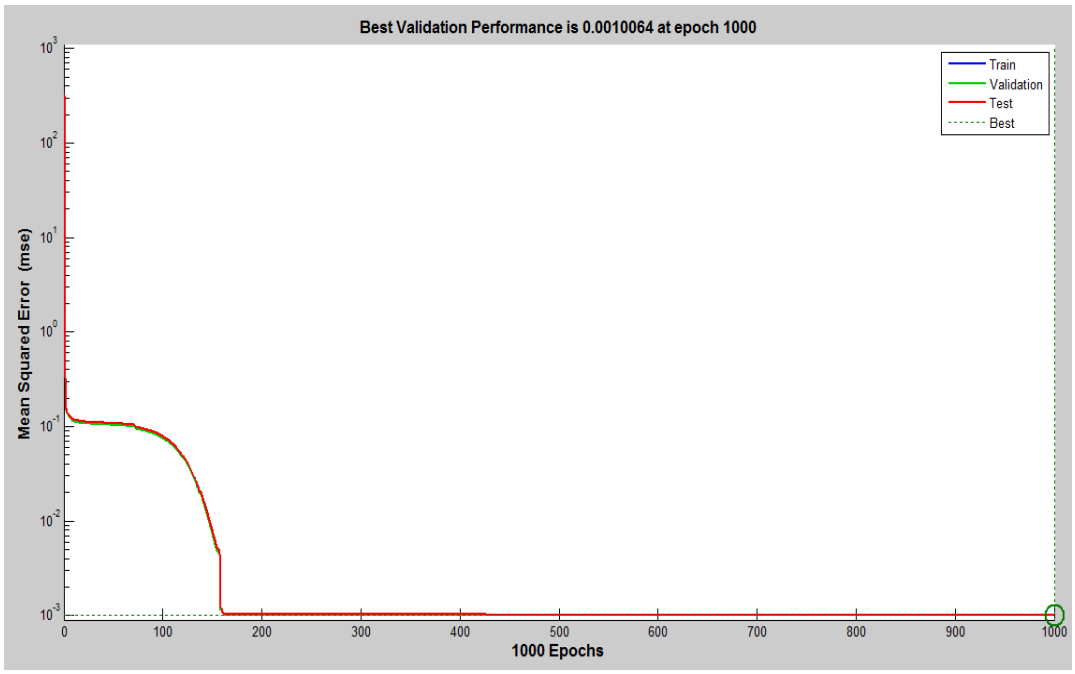


Figura 3.19 Grafica performance prueba tres

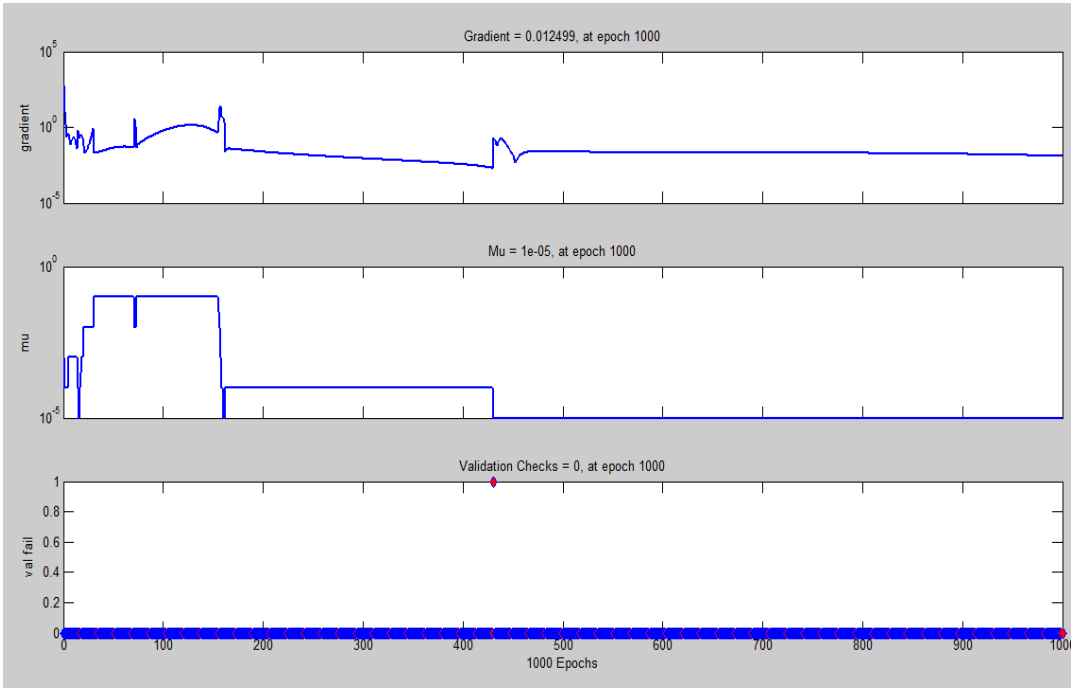
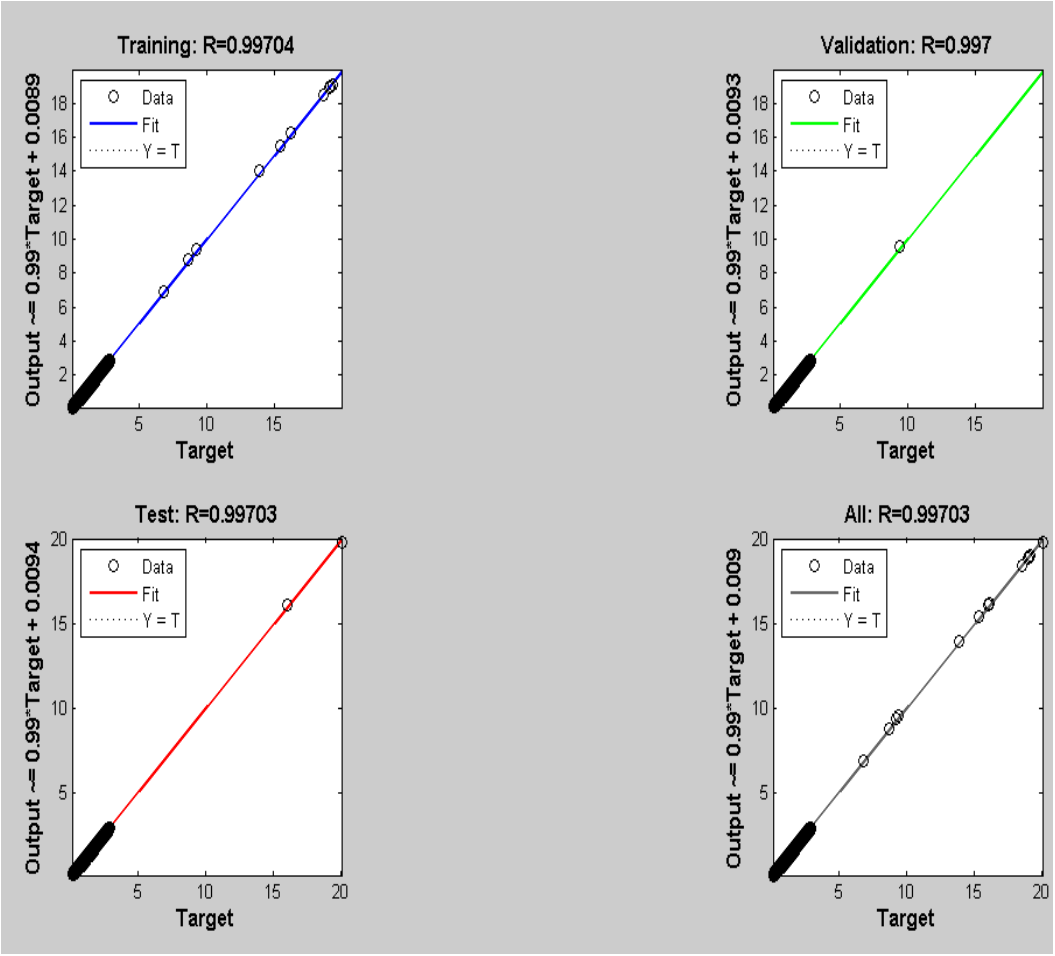


Figura 3.20 Grafica training state prueba tres



3.21 Grafica regresión prueba tres

Prueba 4.

La estructura de la red es 3, 2 contando con tres entradas y una salida figura 3.22

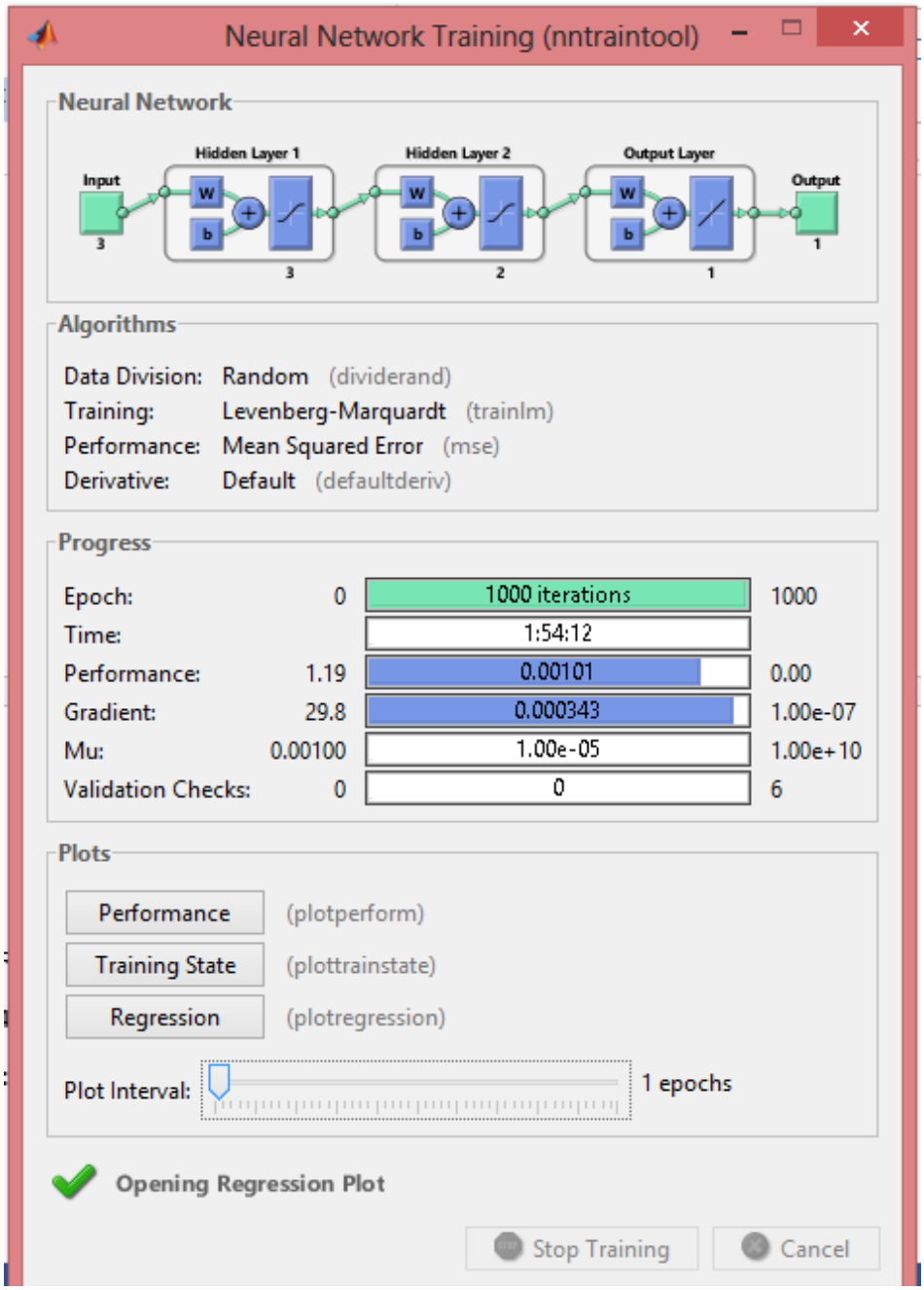


Figura 3.22 Interfaz de entrenamiento prueba cuatro.

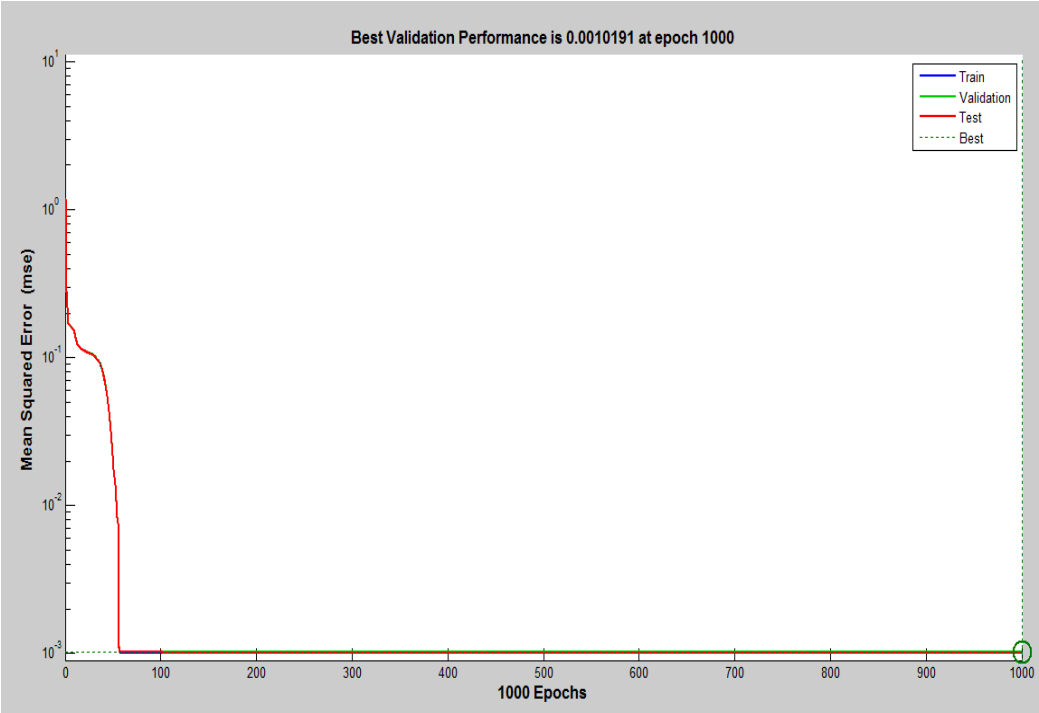


Figura 3.23 Grafica performance prueba cuatro

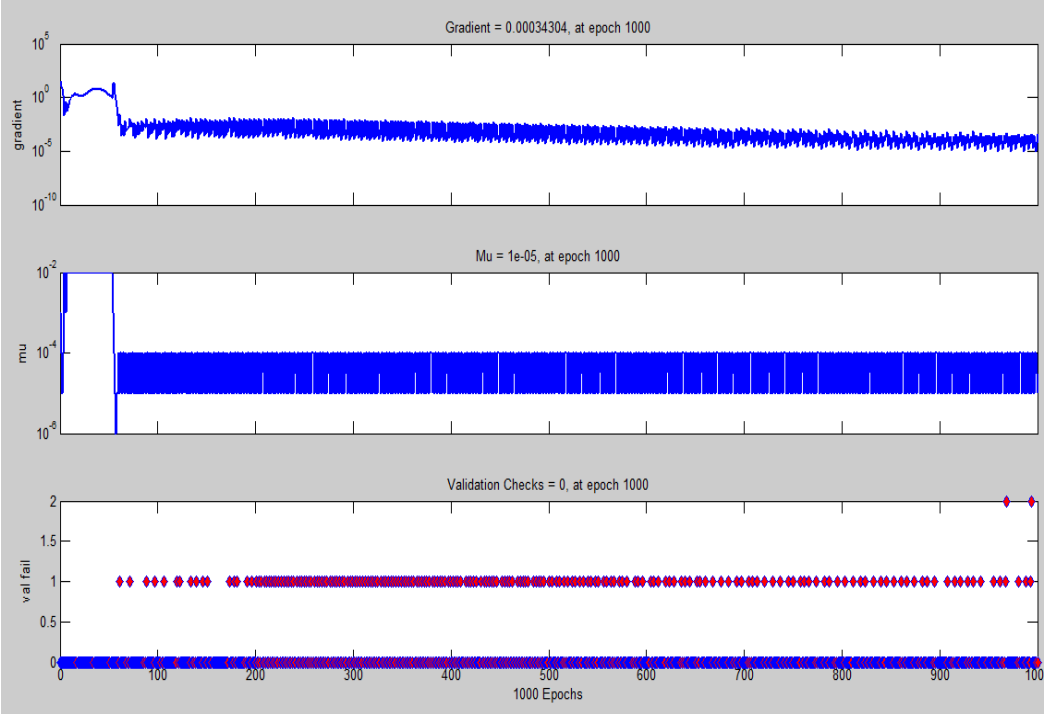


Figura 3.24 Grafica training state prueba cuatro

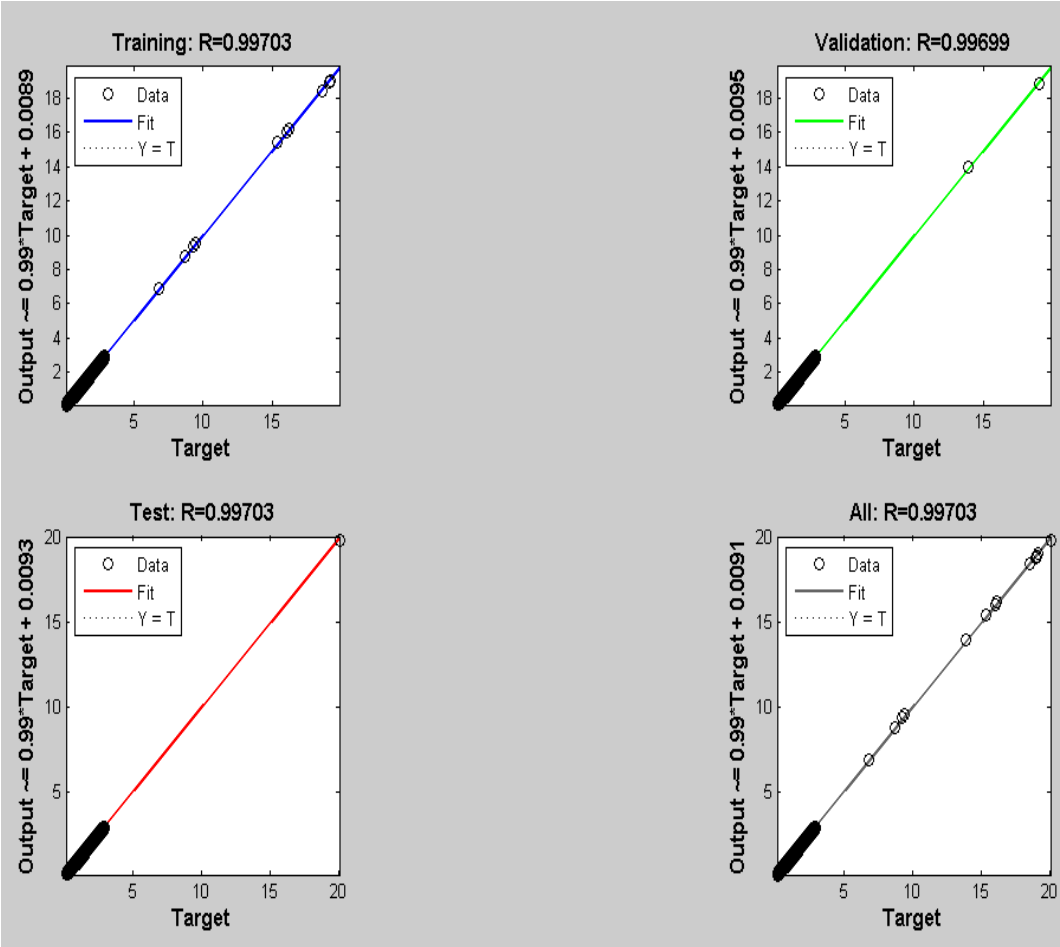


Figura 3.25 Grafica regresión prueba cuatro

Prueba 5.

La estructura de la red es 3, 2 contando con tres entradas y una salida figura 3.26

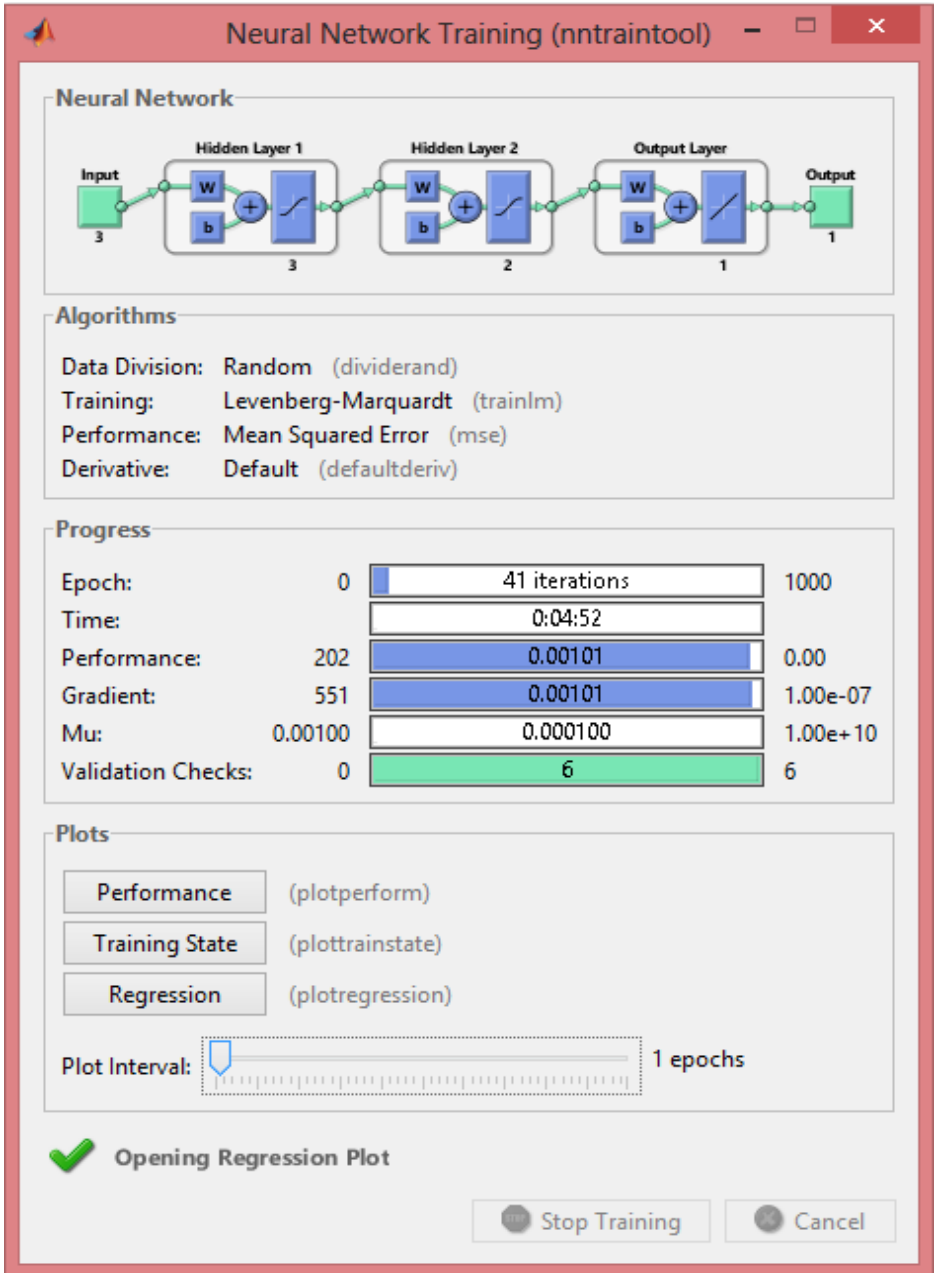


Figura 3.26 Interfaz de entrenamiento prueba cinco.

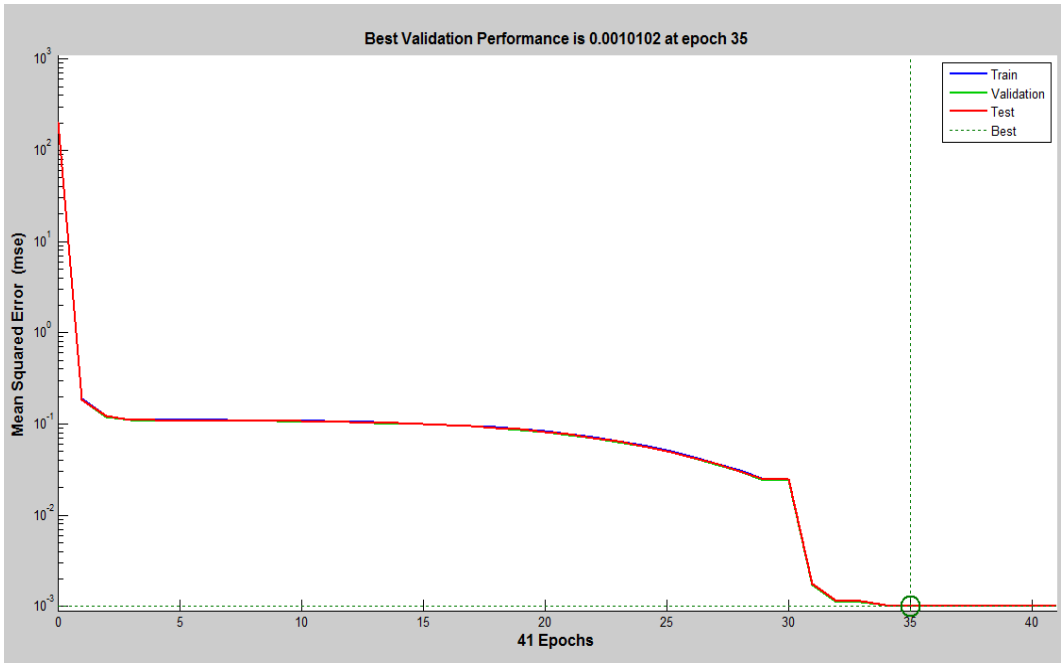


Figura 3.27 Grafica performance prueba cinco

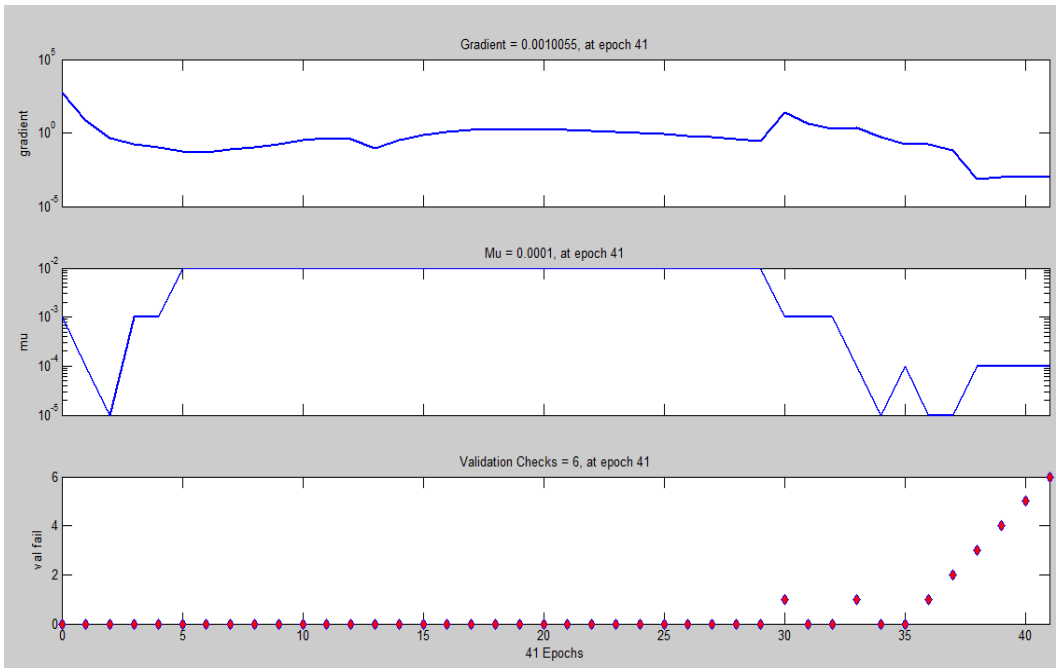


Figura 3.28 Grafica training state prueba cinco

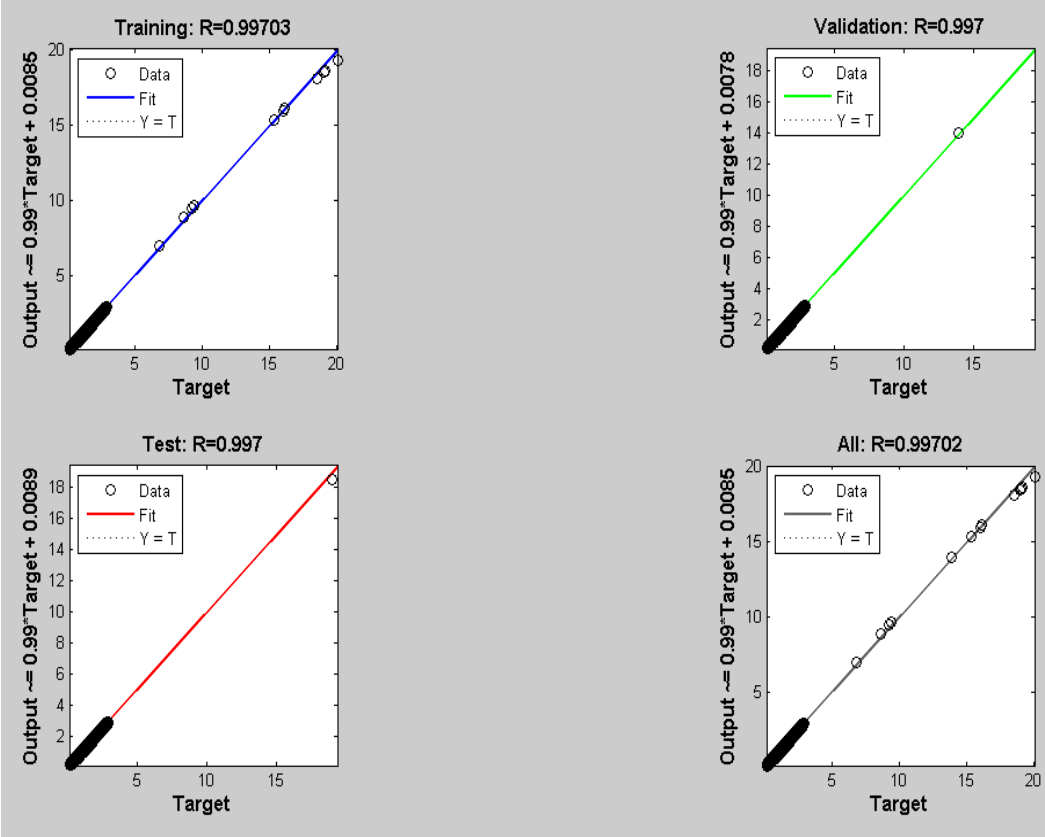


Figura 3.29 Grafica regression prueba cinco.

PRUEBA	ESTRUCTURA	ENTRADAS	SALIDAS	TRAINING	VALIDATION	TEST	ALL	NUMERO DE EPOCAS	VALIDACIONES
1	3,2,2	3	1	0.99708	0.99702	0.99715	0.99708	1000	0
2	3,2,2	2	1	0.99704	0.99703	0.99694	0.99703	1000	0
3	3,2,2	3	1	0.99704	0.997	0.99703	0.99703	1000	0
4	3,2	3	1	0.99703	0.99699	0.99703	0.99703	1000	0
5	3,2	3	1	0.99703	0.997	0.997	0.99702	41	6

Tabla 3.1 Resultados de los entrenamientos realizados.

- **Implementación de la Red Neuronal en la placa Arduino Mega.**

Para la implementación de la red neuronal en la placa Arduino Mega se utilizó la herramienta Simulink la cual podemos encontrar en el software Matlab figura 3.30.

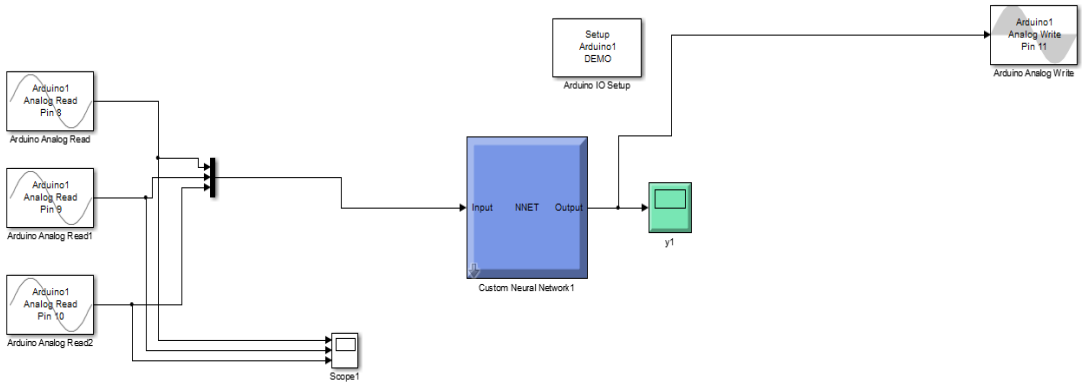


Figura 3.30 simulación red neuronal en Arduino

4.2. Resultados

En este apartado se presentan los resultados de las pruebas realizadas al control neuronal del convertidor CD-CA, sometiéndolo a diferentes voltajes de entrada para verificar si el diseño propuesto cumple con las especificaciones requeridas.

Se realiza la simulación de operación del control neuronal utilizando Simulink de Matlab, para obtener los resultados de salida. Figura 3.31.

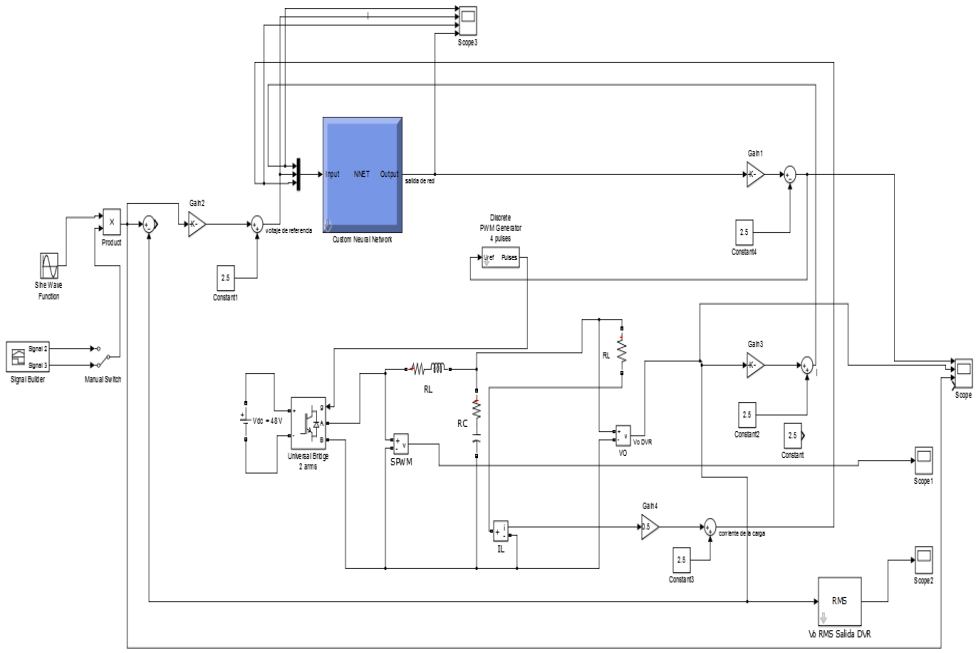


Figura 3.31 Diagrama de Control Neuronal para Convertidor CD-CA

Los resultados de la simulación del control neuronal arrojan un correcto funcionamiento de este logrando la estabilidad del convertidor CD-CA. Figura 3.32.

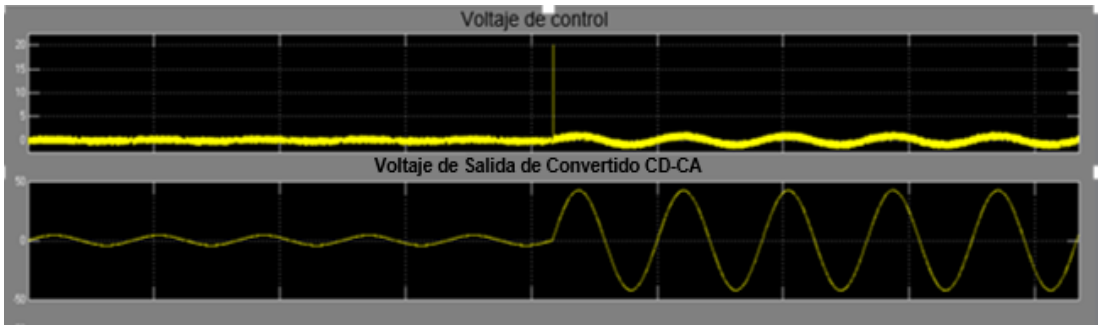


Figura 3.32 Formas de onda de voltaje de control y voltaje de salida del convertidor CD-CA

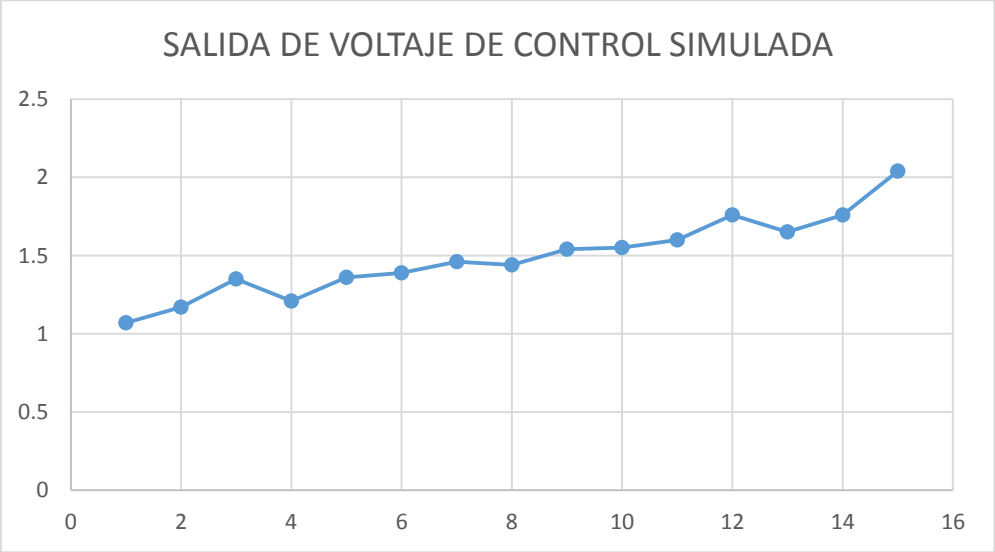
Se seleccionaron 15 valores de entrada y de salida obtenidos de la simulación del control neuronal esto con la finalidad de probar el funcionamiento de este en estado físico cargado en la placa arduino mega.

Se ingresaron todos los valores de entrada y se obtuvieron valores de salida muy similares a los de la simulación. Tabla 3.2.

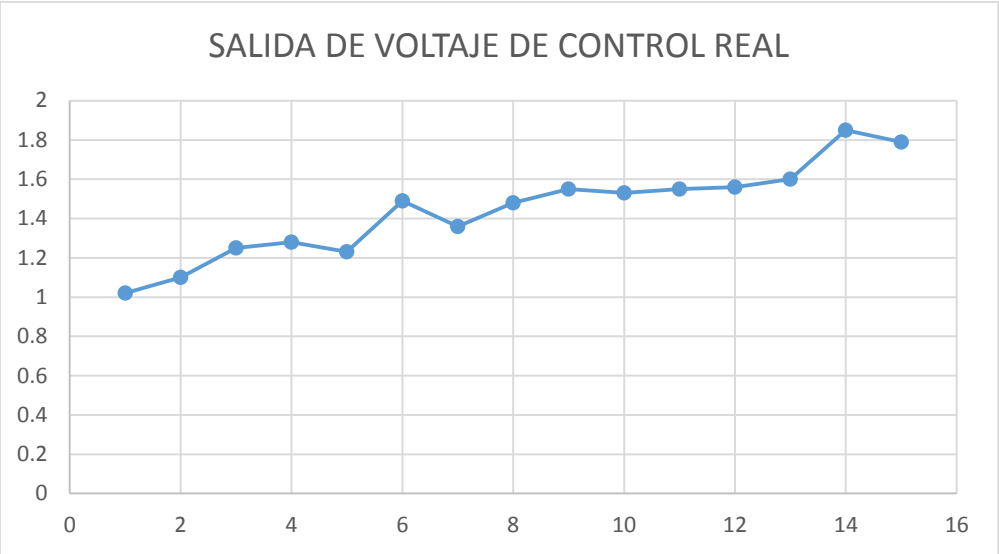
ENTRADAS DE LA RED NEURONAL			SALIDA	
VOLTAJE DE CARGA	VOLTAJE DE REFERENCIA	CORRIENTE DE CARGA	SALIDA SIMULADA	SALIDA REAL
0.06	0.06	0.02	1.07	1.02
0.2	0.2	0.17	1.17	1.1
0.4	0.4	0.37	1.35	1.25
0.6	0.6	0.58	1.21	1.28
0.8	0.8	0.78	1.36	1.23
1	1	0.99	1.39	1.49
1.36	1.36	1.36	1.46	1.36
1.50	1.50	1.50	1.44	1.48
1.64	1.64	1.64	1.54	1.55
1.9	1.9	1.91	1.55	1.53
2.2	2.2	2.22	1.6	1.55
2.4	2.4	2.42	1.76	1.56
2.6	2.6	2.63	1.65	1.6
2.8	2.8	2.84	1.76	1.85
2.94	2.94	2.98	2.04	1.79

Tabla 3.2 Valores de entrada y de salida

Las gráficas de comportamiento 3.1 y 3.2 nos muestran de manera lineal cual es la diferencia entre el resultado de las pruebas físicas con respecto a la simulación.



Grafica 3.1 Voltaje de Control (simulación)



Grafica 3.2 Voltaje de Control (real)

Capítulo 4

Conclusiones

4.1. Conclusiones y recomendaciones

El sistema de control para un convertidor DC/AC propuesto y desarrollado en esta investigación y de acuerdo a los resultados obtenidos en la simulación, demostró un buen desempeño al ser implementado en la placa Arduino Due, así como la implementación de la modulación SPWM a la placa Arduino Mega.

El uso de la plataforma Arduino nos fue de gran ayuda, ya que es un entorno agradable para programar además, de que es un lenguaje en el que ya hemos trabajado con anterioridad por lo que nos facilitó un poco más la realización del proyecto también podemos decir que es un lenguaje muy versátil lo cual nos ayudó a poder comunicarlo con el software Matlab.

Las redes neuronales fueron una buena propuesta para usarse como sistema de control en el convertidor, ya que se planea usarlo en un futuro como parte de otro proyecto en el cual la velocidad de respuesta es un factor muy importante.

Como recomendación para futuros trabajos que puedan mejorar lo ya realizado en este proyecto seria, buscar mejores sistemas digitales los cuales puedan ofrecer mejores prestaciones para que el desempeño del equipo se mucho mejor.

4.2. Competencias Desarrolladas y/o Aplicadas

- Trabajo en equipo
- Habilidad para trabajar de manera autónoma
- Capacidad de generar nuevas ideas
- Destrezas lingüísticas principalmente escrita
- Capacidad crítica y autocritica
- Capacidad para comunicarse con profesores de otras áreas
- Programación
- Diseño de circuitos
- Construcción

Anexos

Anexo a

Programación SPWM

```
#include <TimerThree.h>
int valor_y=0;
int valor_z=0;
int x=0;
int cruce=0;

void setup()
{
Serial.begin(9600);
analogReference(EXTERNAL);
pinMode(11,OUTPUT);
pinMode(10,OUTPUT);
pinMode(4,INPUT);
Timer3.initialize();
Timer3.setPeriod(100);
}

void loop()
{
valor_y=analogRead(A0);//onda triangular
```

```
valor_z=analogRead(A1);// senoidal rectificada
cruce=digitalRead(4);

if(cruce==HIGH)
{
  negativo();
}
if(cruce==LOW)
{
  positivo();
}
}
void positivo()
{
if((valor_z-valor_y)>0)
{
digitalWrite(11,HIGH);
delay(0.00008);// 80 nanosegundos
digitalWrite(10,LOW);
delay(0.0005);// 500 nanosegundos
}
if((valor_z-valor_y)<0)
{
digitalWrite(11,LOW);
delay(0.0005);
digitalWrite(10,HIGH);
delay(0.00008);
```

```
}
}

void negativo()
{
if((valor_z-valor_y)>0)
{
digitalWrite(10,HIGH);
delay(0.00008);
digitalWrite(11,LOW);
delay(0.0005);
}
if((valor_z-valor_y)<0)
{
digitalWrite(10,LOW);
delay(0.0005);
digitalWrite(11,HIGH);
delay(0.00008);
}
}
```

Anexo b

Código de entrenamiento Red Neuronal

```
net=newff(entradas,salidas,[3,2],{'tansig','tansig'});
net=init(net);
net=train(net,entradas,salidas);
a=sim(net,entradas);

w1=net.IW{1,1}
```

```
b1=net.b{1}

w2=net.LW{2,1}
b2=net.b{2}

w3=net.LW{3,2}
b3=net.b{3}
```

Referencias bibliográficas

X., Sun, H.L., M., Leung, F. H., Xu, D., Wang, Y., y Lee, Y.-S, “Analogue Implementation of a Neural Network Controller for UPS Inverter Applications”, *IEEE Transactions on Power Electronics*, 2002.

M., Jazayeri, y H., Abdollahzadeh, “A Novel DVR Control System Design for Compensating all Types of Voltage Sags Based on Pre-Fault Method”, *European Journal of Scientific Research*, 2009.

J.A. Zepeda Hernández, “Control Neuronal para la Corrección de Disturbios en Redes de Energía Eléctrica”, M.C. tesis, Instituto Tecnológico de Tuxtla Gutiérrez, Tuxtla Gutiérrez, México, 2014

J.S. Roger Jang, C. Tsai Sun, Eiji Mizutami, *Neuro-Fuzzy and soft computing*, Prentice Hall, 1997.

El-Gammal, A.M., Abou-Ghazala, A. Y., y El-Shennawy, T. I. “Dynamic Voltage Restorer (DVR) for Voltage Sag Mitigation”, *International Journal on Electrical Engineering and Informatics*, 2011

Hart, D.W.*Electrónica, de potencia*. España: Prentice Hall, 2001.