ESCUELA POLITÉCNICA DEL EJÉRCITO

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA, AUTOMATIZACIÓN Y CONTROL

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TITULO EN INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ENTRENAMIENTO EN REDES NEURONALES UTILIZANDO EL SOFTWARE NEUROSYSTEMS DE SIEMENS

JORGE ENRIQUE SALAZAR CASTILLO

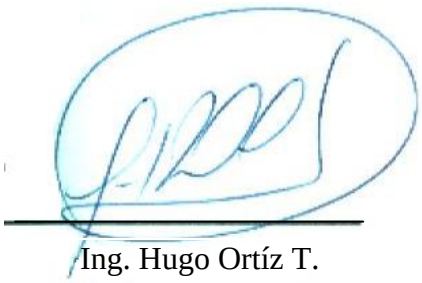
SANGOLQUÍ – ECUADOR

2009

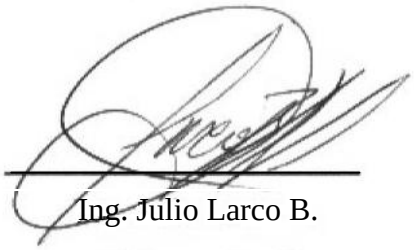
CERTIFICACIÓN

Certificamos que el presente Proyecto de Grado titulado "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE ENTRENAMIENTO EN REDES NEURONALES UTILIZANDO EL SOFTWARE NEUROSYSTEMS DE SIEMENS", ha sido desarrollado en su totalidad por el señor JORGE ENRIQUE SALAZAR CASTILLO con CI: 1718955238, y elaborado bajo nuestra dirección como requisito previo para la obtención del Título en Ingeniería Electrónica, Automatización y Control.

Atentamente



Ing. Hugo Ortíz T.
DIRECTOR



Ing. Julio Larco B.
CODIRECTOR

AGRADECIMIENTOS

A mis padres y hermanos, cuya fe en mí me enseño a tener fe en mí mismo. A ellos debo todo lo que soy.

A mis directores de tesis Ingenieros Hugo Ortíz y Julio Larco, por brindarme la oportunidad de recurrir a sus capacidades y experiencias en un marco de confianza, afecto y amistad, fundamentales para la concreción de este trabajo.

A todas las personas, compañeros y amigos, que de alguna u otra forma contribuyeron a la realización de este trabajo.

DEDICATORIA

Dedicado a todas aquellas personas

que hicieron posible terminar este

largo camino

PRÓLOGO

El presente proyecto se orienta hacia el análisis de las características y funcionalidad del software *NeuroSystems* de Siemens. Se diseña e implementa un controlador de temperatura usando una Red de Neuronas Artificiales, configurada y entrenada en *NeuroSystems*.

Se ha abordado una breve introducción de las redes neuronales artificiales, a fin de establecer los principios y fundamentos necesarios para la construcción y entrenamiento de dichas redes.

Se detallan las características del hardware requerido, así como de las herramientas de software a utilizar (*Step7*, *WinCC*, *NeuroSystems*), prestando principal atención a las características necesarias para la implementación del sistema de entrenamiento.

Se incluye una aplicación de control de temperatura a fin de utilizar las herramientas de configuración y análisis proporcionadas por *NeuroSystems* para la implementación de las redes neuronales artificiales. Se detalla la programación del software utilizado, la interfaz gráfica, así como su implementación.

Finalmente, los datos obtenidos de las pruebas realizadas al sistema implementado, son evaluados con el fin de exponer conclusiones que describan en forma precisa las ventajas y desventajas del software de control y el sistema de control obtenido.

ÍNDICE

CAPÍTULO I

INTRODUCCIÓN

1.1 ANTECEDENTES

Existe actualmente una gran tendencia a establecer un nuevo campo de la computación que integraría los diferentes métodos de resolución de problemas que no pueden ser descritos fácilmente mediante un enfoque algorítmico tradicional. Estos métodos tienen su origen en la emulación de sistemas biológicos. Se trata de una nueva forma de computación que es capaz de manejar las imprecisiones e incertidumbres de problemas relacionados con el mundo real (reconocimiento de formas, toma de decisiones, etc.). Para ello se dispone de un conjunto de metodologías como la Lógica Difusa, el Razonamiento Aproximado, la Teoría del Caos y las Redes Neuronales.

Con las Redes Neuronales se busca la solución de problemas complejos, no como una secuencia de pasos, sino como la evolución de unos sistemas de computación inspirados en el cerebro humano, y dotados por tanto de cierta "inteligencia", los cuales no son sino la combinación de elementos simples de proceso (neuronas) interconectados, que operando de forma paralela en varios estilos, consiguen resolver problemas relacionados con el reconocimiento de formas o patrones, predicción, codificación, control y optimización entre otras aplicaciones.

La herramienta de configuración *NeuroSystems* [2] es un simulador pensado para el desarrollo de redes neuronales con un gran nivel de detalle. Con la ayuda del *Simatic WinCC, NeuroSystems* está concebido para la visualización y manejo de redes neuronales de alto nivel que no sólo soporta el proceso de aprendizaje sino también la configuración y análisis de redes neuronales en el PC.

1.2 OBJETIVOS

Objetivo General

Diseñar e implementar un sistema de entrenamiento de Redes Neuronales utilizando la herramienta de configuración *NeuroSystems.*

Objetivos Específicos

1. Describir y analizar los principios y comportamiento de las redes neuronales
2. Analizar y detallar los paquetes de programación *SIMATIC Step7*, *SIMATIC WinCC*, *ActiveXControl* con los que trabaja *NeuroSystems*
3. Describir la estructura del sistema de automatización Siemens S7-300 así como sus módulos de entrada y salida
4. Diseñar el modelo y estructura de Redes Neuronales
5. Simular el comportamiento de la red neuronal y realizar la implementación
6. Realizar pruebas a la red y evaluar los resultados obtenidos
7. Elaborar un manual de usuario del sistema de entrenamiento.

1.3 TEORÍA DE REDES NEURONALES

Uno de los grandes enigmas que ha preocupado al hombre desde tiempos ancestrales es el de su propia naturaleza. Cuáles son las características que nos hacen humanos, qué tiene el hombre que no tienen el resto de los animales, qué nos hace únicos entre todos los seres vivos. Este enigma ha venido asociado con el de la inteligencia, pues dentro de la naturaleza humana está el ser inteligente, y ésta es una característica que discrimina absolutamente a nuestra especie. Es por esto por lo que el estudio de la inteligencia ha fascinado a filósofos y científicos desde siempre y ha sido un tema recurrente en tratados y libros, y sin embargo no se han producido avances significativos.

Uno de los retos más importantes a los que se enfrenta el ser humano de nuestra generación es el de la construcción de sistemas inteligentes, entendiéndose por sistema a cualquier dispositivo físico o lógico capaz de realizar la tarea requerida. Este es

precisamente el objetivo de la disciplina científica conocida con el nombre de Inteligencia Artificial.

Dentro de la Inteligencia Artificial se pueden distinguir dos grandes áreas. Una se ocupa de la construcción de sistemas con características que se puedan definir como inteligentes. A este campo se lo denomina Inteligencia Artificial Simbólica. En este caso, se define el problema a resolver y se diseña el sistema capaz de resolverlo siguiendo esquemas prefijados por la disciplina.

Frente a esta perspectiva se encuentra la otra gran área de la Inteligencia Artificial, la Subsimbólica. En este caso no se realizan diseños a alto nivel de sistemas capaces de resolver los problemas utilizando las técnicas de la disciplina, sino que se parte de sistemas genéricos que van adaptándose y construyéndose hasta formar por sí mismos un sistema capaz de resolver el problema. El mecanismo fundamental que capacita a los seres vivos para la realización de tareas sofisticadas no preprogamadas directamente es el sistema nervioso. Desde este punto de vista la perspectiva subsimbólica trata de estudiar los mecanismos de los sistemas nerviosos, del cerebro, así como su estructura, funcionamiento y características lógicas, con la intención de diseñar programas basados en dichas características que se adapten y generen sistemas capaces de resolver problemas. Es en este campo donde se encuadran las Redes de Neuronas Artificiales. [1]

Idealmente, el objetivo de las Redes de Neuronas Artificiales es llegar a diseñar máquinas con elementos neuronales de procesamiento paralelo, de modo que el comportamiento global de esa red “emule”, de la forma más fiel posible, los sistemas neuronales de los animales. Esto hace imprescindible el estudio profundo de los mecanismos que rigen el comportamiento de los sistemas neuronales.

1.3.1 Fundamentos Biológicos de las Redes Neuronales

El aparato de comunicación neuronal de los animales y del hombre, formado por el sistema nervioso y hormonal, en conexión con los órganos de los sentidos y los órganos efectores (músculos, glándulas), tiene la misión de recoger informaciones, transmitirlas y elaborarlas, en parte también almacenarlas y enviarlas de nuevo en forma elaborada.

El elemento estructural y funcional más esencial, en el sistema de comunicación neuronal, es la célula nerviosa o neurona. Un diagrama de una célula nerviosa típica es el que se muestra en la Figura 1.1.

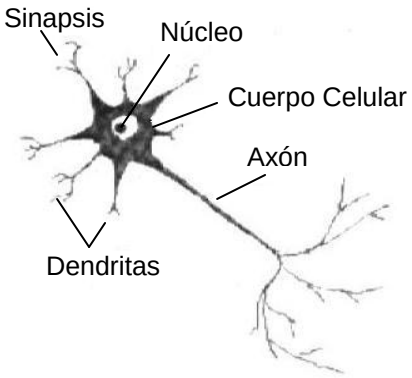


Figura 1.1. Descripción de una célula nerviosa típica

En este esquema se aprecia que la neurona consta de un cuerpo celular y un núcleo, como el resto de las células del organismo, pero cuenta también con algunos elementos específicos. En primer lugar está el axón, que es una ramificación de salida de la neurona. Además, la neurona cuenta con un gran número de ramificaciones de entrada, las dendritas, que propagan la señal al interior de la neurona.

Las sinapsis recogen información electro-química procedente de las células vecinas a las que la célula en cuestión está conectada; esta información llega al núcleo donde es procesada hasta generar una respuesta que es propagada por el axón. Más tarde, la señal única propagada por el axón se ramifica y llega a dendritas de otras células a través de lo que se denomina sinapsis. Las sinapsis son los elementos de unión entre el axón y dendritas. Es un espacio líquido donde existen determinadas concentraciones de elementos ionizados. Estos iones hacen que el espacio intersináptico posea ciertas propiedades de conductividad que activen o impidan, en cierto grado, el paso del impulso eléctrico.

En algunas de las células nerviosas, los receptores, reciben información directamente del exterior; a esta información se le da el nombre de estímulo. Existen terminaciones nerviosas en casi todas las partes del organismo que se encargan de recibir la

información visual, auditiva, táctil, etc. Esta información, una vez elaborada, pasa a ser tratada como el resto de la información del sistema nervioso y convertida en impulsos electro-químicos.

Son estos impulsos los que, básicamente, ponen en funcionamiento la red neuronal del sistema nervioso, la cual propaga todas las señales asíncronamente, como se ha descrito, hasta que llega a los efectores, los órganos, glándulas, músculos; que son capaces de transformar la información recibida en acciones motoras, hormonales, etc. Así es como un organismo recibe, procesa y reacciona ante la información recibida del exterior. [1]

Nuestro comportamiento, inteligente o no, y el del resto de los animales superiores, siguen un esquema de este tipo. Y son estos mecanismos los que tratan de incorporar los modelos artificiales que han ido apareciendo a lo largo de la historia de las Redes de Neuronas Artificiales.

1.3.2 Principios de las Redes Neuronales Artificiales

Las Redes de Neuronas Artificiales tienen un gran potencial para la construcción de sistemas de computación que no necesitan ser programados, debido a que el aprendizaje es por ejemplos. No se trata de la aplicación ciega y automática de un algoritmo, el proceso de elaboración de la información recibida depende de las distintas características, tanto estructurales como funcionales de la red.

Existen modelos muy diversos de redes de neuronas en los cuales se siguen filosofías de diseño, reglas de aprendizaje y funciones de construcción de las respuestas muy distintas. Una primera clasificación se hace en función del recorrido que sigue la información dentro de la red, y así se distinguen redes alimentadas hacia adelante y redes con retroalimentación.

1.3.2.1 La neurona artificial

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

estado. El nivel de activación de una célula depende de las entradas recibidas y de los valores sinápticos.

Para calcular el estado de activación se ha de calcular en primer lugar la entrada total ***E*** a la célula. Este valor se calcula como la suma de todas las entradas ponderadas por ciertos valores. [1]

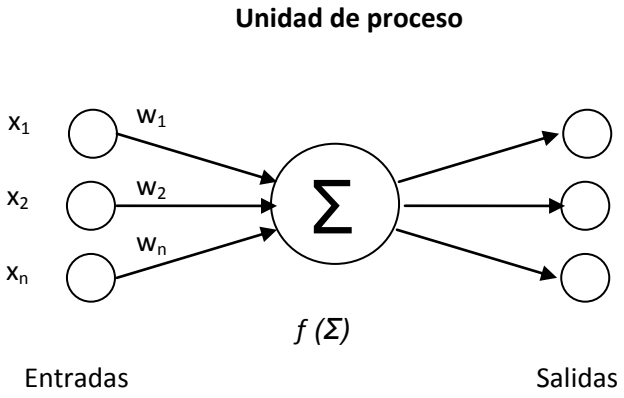


Figura 1.2. Esquema de una unidad de proceso típica

La Figura 1.2 muestra un modelo que representa esta idea. Aquí un grupo de entradas $x_1, x_2, ..., x_n$ son introducidas en una neurona artificial. Estas entradas, definidas por un vector ***x***, corresponden a las señales de la sinapsis de una neurona biológica. Cada señal se multiplica por un peso asociado $w_1, w_2, ..., w_n$ antes de ser aplicado el sumatorio. Cada peso corresponde a la fuerza de una conexión sináptica, es decir el nivel de concentración iónica de la sinapsis, y se representa por un vector ***w***.

El sumatorio, que corresponde al cuerpo de la neurona, suma todas las entradas ponderadas algebraicamente, produciendo una salida que se denomina ***E***, así:

$$E = x_1 w_1 + x_2 w_2 + ... + x_n w_n$$

Las señales ***E*** son procesadas además por una función llamada función de activación o de salida ***F***, que produce la señal de salida de la neurona. Más adelante se hará una descripción más detallada de los tipos de funciones de activación y de su importancia en un modelo de Red de Neuronas Artificial. [1]

1.3.2.2 Estructura básica de la red

En la Figura 1.3 se muestra un esquema de la interconexión de una red multicapa. El primer nivel lo constituyen las células de entrada, estas unidades reciben los valores de entrada de la red. A continuación hay una serie de capas intermedias, llamadas ocultas, cuyas unidades responden a las señales que reciben, cambiando su nivel de activación. El último nivel es el de salida. La salida de estas unidades sirve como salida de toda la red.

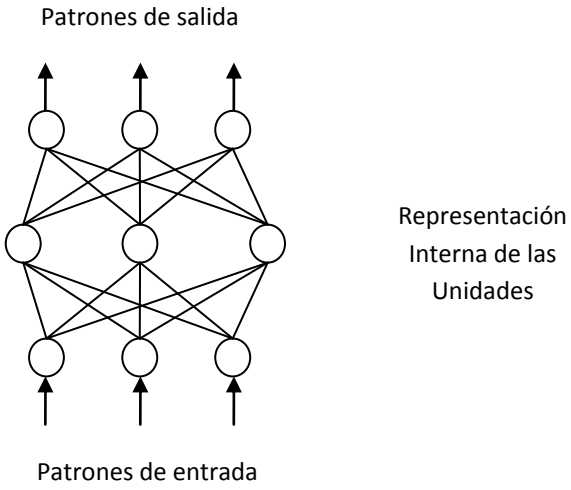


Figura 1.3. Esquema de una red de tres capas totalmente interconectadas

Para cada vector de entrada, éste es introducido en la red copiando cada valor de dicho vector en la célula de entrada correspondiente. Cada célula de la red, una vez recibida la totalidad de sus entradas, las procesa y genera una salida que es propagada a través de las conexiones entre células, llegando como entrada a la célula destino. Una vez que la entrada ha sido completamente propagada por toda la red, se producirá un vector de salida, cuyos componentes son cada uno de los valores de salida de las células de salida.

1.3.2.3 Aprendizaje

La parte más importante de una Red de Neuronas Artificial es el aprendizaje. El esquema de aprendizaje de una red es lo que determina el tipo de problemas que será capaz de resolver. Las Redes de Neuronas Artificiales son sistemas de aprendizaje basados en ejemplos. La capacidad de una red para resolver un problema estará ligada

de forma fundamental al tipo de ejemplos de que dispone en el proceso de aprendizaje. Desde el punto de vista de los ejemplos, el conjunto de aprendizaje debe poseer las siguientes características:

- **Ser significativo**. Debe haber un número suficiente de ejemplos. Si el conjunto de aprendizaje es reducido, la red no será capaz de adaptar sus pesos de forma eficaz.

- **Ser representativo**. Los componentes del conjunto de aprendizaje deberán ser diversos. Si un conjunto de aprendizaje tiene muchos más ejemplos de un tipo que del resto, la red se especializará en dicho subconjunto de datos y no será de aplicación general.

El aprendizaje en una Red de Neuronas Artificial consiste en la determinación de los valores precisos de los pesos para todas sus conexiones, que la capacite para la resolución eficiente de un problema.

El proceso general de aprendizaje consiste en ir introduciendo paulatinamente todos los ejemplos del conjunto de aprendizaje, y modificar los pesos de las conexiones siguiendo un determinado esquema de aprendizaje. Una vez introducidos todos los ejemplos se comprueba si se ha cumplido cierto criterio de convergencia; de no ser así se repite el proceso y todos los ejemplos del conjunto vuelven a ser introducidos. La modificación de los pesos puede hacerse después de la introducción de cada ejemplo del conjunto, o una vez introducidos todos ellos. [1]

La finalización del periodo de aprendizaje se puede determinar:

- *Mediante un número fijo de ciclos*. Se decide con antelación cuántas veces será introducido todo el conjunto, y una vez superado dicho número se detiene el proceso y se da por aceptada la red resultante.

- *Cuando el error descienda por debajo de una cantidad preestablecida*. En este caso habrá que definir en primer lugar una función de error. Se decide un valor

aceptable para dicho error, y sólo se detiene el proceso de aprendizaje cuando la red produzca un valor de error por debajo del prefijado.

- *Cuando la modificación de los pesos sea irrelevante*. Cuando en el proceso de aprendizaje se llegue a un punto en que ya no se producen variaciones de los valores de los pesos de ninguna conexión, se detiene el proceso de aprendizaje.

Dependiendo del esquema de aprendizaje y del problema a resolver, se pueden distinguir tres tipos de esquemas de aprendizaje:

- **Aprendizaje supervisado**. En el esquema de aprendizaje supervisado, cada vez que un ejemplo es introducido y se procesa para obtener una salida, dicha salida se compara con la salida que debería haber producido, y de la que se dispone al estar incluida dicha información en el conjunto de aprendizaje. La diferencia entre ambas influirá en cómo se modificarán los pesos. Si los dos datos son muy diferentes, se modificarán mucho los pesos si son parecidos la modificación será menor. [1]

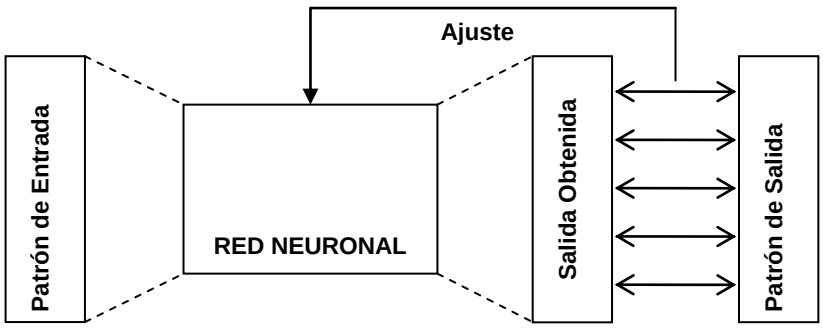


Figura 1.4. Aprendizaje supervisado

- **Aprendizaje no supervisado**. En este aprendizaje los datos del conjunto de aprendizaje sólo tienen información de los ejemplos, y no hay nada que permita guiar en el proceso de aprendizaje. La red modificará los valores de los pesos a partir de información interna. Cuando se utiliza aprendizaje no supervisado, la red trata de determinar características de los datos del conjunto de entrenamiento: rasgos significativos, regularidades o redundancias. [1]

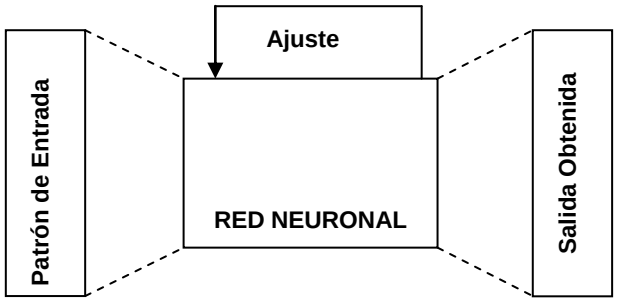


Figura 1.5. Aprendizaje no supervisado

- **Aprendizaje por refuerzo**. Es una variante del aprendizaje supervisado en el que no se dispone de información concreta del error cometido por la red para cada ejemplo de aprendizaje, sino que simplemente se determina si la salida producida para dicho patrón es o no adecuada. [1]

1.3.2.4 Funciones de Activación

La función de activación produce la señal de salida de la neurona. Para la función de activación de la neurona es característico que tanto los valores de entrada y los pesos tengan una influencia en su valor de salida. Por otra parte, para la mayoría de las aplicaciones, la activación de la neurona es linealmente independiente de la suma ponderada de las entradas y puede ser súbita (Figura 1.6 a) o gradual (b, c, d). [2]

Las funciones de activación más comunes son las siguientes:

(a) Función *switch* (o *Hardlim*):

Acerca la salida de la red a 0, si el argumento de la función (*u*) es menor que cero y la lleva a 1 si este argumento es mayor que 1. Esta función crea neuronas que clasifican las entradas en dos categorías diferentes, característica que le permite ser empleada en la red tipo perceptrón[1].

[1] Fue uno de los primeros modelos de redes neuronales. Imita una neurona tomando la suma ponderada de sus entradas y enviando a la salida un 1 si la suma es más grande que algún valor umbral ajustable, si ocurre de otro modo, devuelve 0.

$$y = \begin{Bmatrix} 1, & \text{si} u \geq 0 \\ 0, & \text{si} u < 0 \end{Bmatrix}$$

(b) Función sigmoidal:

Esta función es comúnmente utilizada en redes multicapa. Toma los valores de entrada, los cuales pueden oscilar entre más y menos infinito, y restringe la salida a valores dentro del intervalo [0 ,1], de acuerdo a la siguiente expresión:

$$y = \frac{1}{1 + e^{-u}}$$

(c) Función tangente hiperbólica:

Es similar a la función sigmoidal, a diferencia que restringe el espacio de salida a valores dentro del intervalo [-1, 1], y viene dada por la siguiente expresión:

$$y = \tanh(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}}$$

(d) Función lineal

La salida de una función de activación lineal es igual a su entrada. Las neuronas que emplean esta función de transferencia son utilizadas en la red tipo *Adaline*[2].

$$y = u$$

Generalmente, la función de activación es elegida por el diseñador la cual puede ser una función lineal o no lineal, elección que se realiza por las características de cada función basándose en los valores de activación que se desee que alcancen las neuronas y las especificaciones que la neurona tenga que resolver.

Los diagramas presentados a continuación muestran las curvas de cada función.

[2] *ADAptive LInear NEuron*, uno de los primeros modelos de redes neuronales, su función de activación es del tipo lineal y su aprendizaje incluye la diferencia entre el valor real producido y su salida esperada, para un patrón de entrada específico

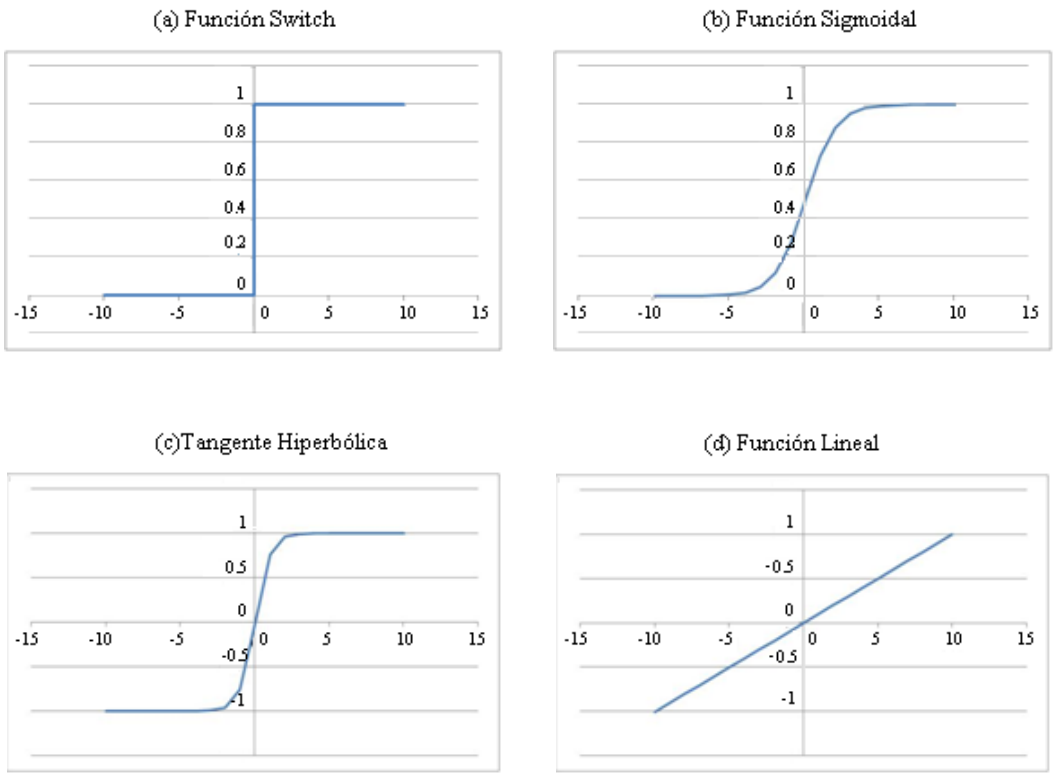


Figura 1.6. Funciones de activación más comunes

1.3.3 Características de las Redes Neuronales Artificiales

Los computadores tradicionales operan secuencialmente porque sólo tienen una unidad de proceso. Construir computadores con varias unidades de proceso es muy costoso y complejo. Sin embargo, el ámbito de aplicación de los computadores es mucho más general, y este tipo de problemas no se presentan con frecuencia.

Las Redes de Neuronas Artificiales no son programadas, aprenden a partir de ejemplos. Normalmente, a una Red de Neuronas Artificial le presentamos una serie de patrones ejemplo de los cuales ella debe aprender. Estos ejemplos, o patrones de entrenamiento, vienen representados por vectores, y pueden ser obtenidos por fuentes de imágenes, sensores, movimientos del brazo de un robot, etc.

Las Redes de Neuronas Artificiales codifican la información de manera distribuida. Normalmente la información que es almacenada en una Red de Neuronas Artificial es compartida por varias de sus unidades de proceso. Este tipo de almacenamiento está en

evidente contraste con los esquemas tradicionales de memoria, donde las piezas de información son almacenadas en lugares específicos de memoria.

Este tipo de esquemas de almacenamiento distribuido tiene muchas ventajas; la más importante radica en que la información es almacenada de forma redundante, lo que permite un correcto funcionamiento de la red, aun cuando el sistema pueda haber sufrido una parcial destrucción. Aunque la redundancia puede ser implementada en otros tipos de sistemas, las Redes de Neuronas Artificiales disponen de una vía natural de organizar e implementar esta redundancia; el resultado es un sistema que resulta tolerante a fallos, por sí mismo.

Esto está en concordancia con la tolerancia a fallos de nuestros propios sistemas neuronales. Las células nerviosas son las únicas células del organismo que no se regeneran; a partir de los 25 años, los humanos empiezan a perder neuronas que jamás son recuperadas. Sin embargo, esta pérdida no merma en absoluto ninguna de las capacidades de nuestros sistemas neuronales. En los sistemas artificiales, en general, esta característica se conserva, y si la red es suficientemente grande, la eliminación de alguna de sus unidades no afectará a su funcionamiento.

Las redes pueden ser entrenadas para resolver problemas de forma genérica, y no sólo para memorizar los patrones de entrenamiento, siempre que estos patrones representen adecuadamente al problema. Esta característica es importante y hay que tenerla en cuenta, ya que existen algunos problemas que no pueden ser descritos de forma exhaustiva mediante ejemplos, y no es posible programar procedimientos para su resolución.

1.3.4 Aplicaciones de las Redes Neuronales Artificiales

Las redes neuronales se pueden utilizar para problemas, cuya estructura y solución no se conocen o son sólo parcialmente conocidas. Utilizando un ejemplo, la red neuronal aprende la solución a un problema. Este tipo de procesamiento de información es fundamentalmente diferente de un sistema convencional de programación.

Las redes neuronales se utilizan normalmente en las siguientes tareas:

- Reconocedores de patrones y memorias asociativas
- Identificación de características o procesos
- Evaluación de datos (por ejemplo, para sistemas de diagnostico)
- Control de procesos
- Optimización de tareas
- Pronósticos

Existen otras tareas que se han resuelto con éxito mediante redes neuronales, entre las que se encuentran: aprender a jugar el backgammon (Tesauro y Sejnowski, 1989), clasificar señales de sonar (Gorman y Sejnowski, 1988), comprimir imágenes (Cottrell et al., 1987), y conducir un vehículo por una carretera (Pormerleau, 1989). Aunque existen otras técnicas para poder abordar todos estos problemas, los sistemas basados en el aprendizaje se pueden construir frecuentemente con más rapidez y con menos conocimientos de sus homólogos tradicionales. [3]

1.3.5 Historia de las redes neuronales

Las primeras redes neuronales fueron desarrolladas por los biólogos con la esperanza de poder simular redes neuronales naturales. El término "neurona" se utiliza en la medicina y la biología para designar la célula nerviosa. Los matemáticos e ingenieros han adoptado este modelo para su uso en la mecánica de procesamiento de la información. Estas redes neuronales artificiales "*son un intento para imitar la función de las células nerviosas que se encuentran en la parte baja del orden de los organismos*". Cuando hablamos de *red neuronal* nos referimos a *red neuronal artificial*. Al igual que las estructuras biológicas del cerebro las redes neurales tienen la capacidad de aprender. [2]

Los orígenes de las redes neuronales artificiales vienen a principios de los años 40. En 1943, Warren McCulloch y Walter Pitts [4] desarrollaron un modelo "*neurona MP*" que imitaba la función de las células nerviosas humanas. Este modelo de neurona simplificado ya tiene las siguientes propiedades funcionales de una célula nerviosa:

- Estructura: Muchas entradas (sinapsis) y una salida (axón).
- Estados: dos posibles estados (activo o inactivo).
- Conexión: Las neuronas están relacionados entre sí (en red).
- Independencia: El estado de una neurona sólo depende de su entrada de activación.
- Condición de Activación: Una neurona es estimulada si suficientes entradas son activadas.

Donald Hebb [5] formuló el primer método de aprendizaje en 1949 "*Regla de aprendizaje Hebbian*". Redes neuronales capaces de aprender se han creado sobre esta base y, por último, alrededor de 1955, se llevaron a cabo las primeras simulaciones por computadora. P.J. Werbos [6] y D.E. Rumelhart [7] desarrollaron el algoritmo de aprendizaje *BACKPROPAGATION*[3], que es el método de aprendizaje más frecuentemente utilizado.

En base a este trabajo la teoría de redes neuronales artificiales se desarrollaron aún más. Después de 1970, han sido creadas y desarrolladas las redes neuronales con la función de memoria asociativa. Teuvo Kohonen [8] desarrolló mapas con características de auto-organización. Después de 1985, el mayor desarrollo y aplicación de redes neuronales realmente despegó.

Otro enfoque se basa en las redes de Funciones de Base Radial (RBF). Se origina de la teoría aproximación de matemática y se utilizó por primera vez en 1988 por D.S. Broomhead y D. Lowe [9].

1.3.6 Clasificación de las Redes Neuronales Artificiales

La disposición de los elementos de proceso en una red neuronal artificial y la estructura de las conexiones sinápticas constituyen los parámetros principales para la clasificación de diversos tipos de arquitecturas neuronales.

[3] El algoritmo *BACKPROPAGATION* o *RetroPropagación* consiste en un aprendizaje supervisado; es decir la modificación de los parámetros se realiza para que la salida de la red sea lo más próxima posible a la salida deseada.

1.3.6.1 Según su Topología

Las redes neuronales se organizan en capas, con un determinado número de unidades de procesamiento por bloque, con un comportamiento similar. La cantidad de niveles determinan la clasificación topológica en:

Redes Monocapa

Poseen una sola capa de neuronas en su estructura, destinadas a recibir señales del exterior y emitir la respuesta de la red simultáneamente. En este tipo de red, las conexiones entre las diferentes neuronas son laterales, cruzadas o autorrecurrentes y son utilizadas principalmente para regenerar información de entrada incompleta o con señales de ruido.

Redes Multicapa

La red se encuentra conformada por varios niveles de redes monocapa dispuestas en cascada, donde la entrada de una capa es la salida de otra; en este tipo de redes existe la presencia de por lo menos una capa oculta entre los niveles de entrada y salida de la misma. [22]

1.3.6.2 Según el Patrón de Conexión

Redes Unidireccionales

El flujo de los datos se realiza en una sola dirección sin posibilidad de retroalimentación, la salida de una neurona es la entrada de la capa siguiente por tanto no existen conexiones entre la salida de una capa y la entrada de la misma o de niveles previos.

Redes Recurrentes

Contienen lazos de realimentación, con lo cual adquieren un aspecto dinámico que les permiten evolucionar hacia nuevos estados de equilibrio. [22]

1.3.7 Perceptrón multicapa

El Perceptrón Multicapa es una generalización del perceptrón simple, y surgió como consecuencia de las limitaciones de dicha arquitectura en lo referente al problema de la separabilidad no lineal. Si añadimos capas intermedias (ocultas) a un perceptrón simple, obtendremos un Perceptrón Multicapa o *MLP*. Esta arquitectura suele entrenarse mediante el algoritmo denominado *RetroPropagación*. [11]

Minsky y Papert [12] mostraron en 1969 que la combinación de varios perceptrones simples -inclusión de neuronas ocultas- podía resultar en una solución adecuada para tratar ciertos problemas no lineales. Sin embargo, los autores no presentaron una solución al problema de cómo adaptar los pesos de la capa de entrada a la capa oculta, pues la regla de aprendizaje del perceptrón simple no puede aplicarse en este escenario. No obstante, la idea de combinar varios perceptrones sirvió de base para estudios posteriores realizados por Rumelhart, Hinton y Willians en 1986 [13]. Estos autores presentaron una manera de retropropagar los errores medidos en la salida de la red hacia las neuronas ocultas, dando lugar a la llamada regla delta generalizada, que no es más que una generalización de la regla delta, para funciones de activación no lineales y redes multicapa.

La habilidad del Perceptrón Multicapa para aprender a partir de un conjunto de ejemplos, aproximar relaciones no lineales, filtrar ruido en los datos, etc. hace que sea un modelo adecuado para abordar problemas reales, sin que esto indique que sean los mejores aproximadores universales. Cada una de las clases de aproximadores tiene sus propias características; se conocen ciertas condiciones bajo las cuales un método es preferible a otro, pero en ningún caso se puede decir que un método sea absolutamente el mejor. Serán las consideraciones prácticas de cada problema las que determinen la elección de un aproximador u otro. [1]

1.3.7.1 Arquitectura del Perceptrón Multicapa

La arquitectura del Perceptrón Multicapa se caracteriza porque tiene sus neuronas agrupadas en capas de diferentes niveles. Cada una de las capas está formada por un

conjunto de neuronas y se distinguen tres tipos de capas diferentes: la capa de entrada, las capas ocultas y la capa de salida, como se observa en la Figura 1.7.

Las neuronas de la capa de entrada no actúan como neuronas propiamente dichas, sino que se encargan únicamente de recibir las señales o patrones que proceden del exterior y propagar dichas señales a todas las neuronas de la siguiente capa. La última capa actúa como salida de la red, proporcionando al exterior la respuesta de la red para cada uno de los patrones de entrada. Las neuronas de las capas ocultas realizan un procesamiento no lineal de los patrones recibidos.

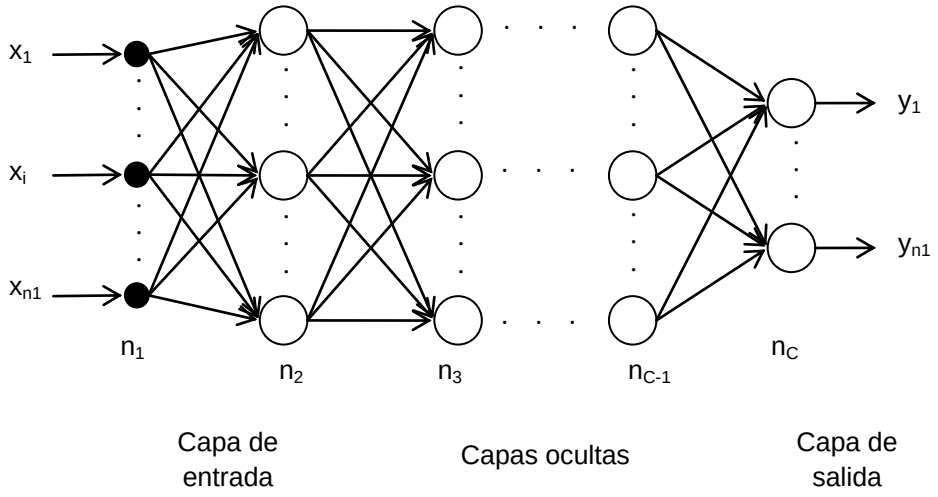


Figura 1.7. Arquitectura del Perceptrón Multicapa

Como se observa en la Figura 1.7, las conexiones del Perceptrón Multicapa siempre están dirigidas hacia adelante, es decir, las neuronas de una capa se conectan con las neuronas de la siguiente capa, de ahí que reciban también el nombre de redes unidireccionales propagadas hacia adelante.

Diseño de la arquitectura del Perceptrón Multicapa

Cuando se aborda un problema utilizando el Perceptrón Multicapa, uno de los primeros pasos a realizar es el diseño de la arquitectura de la red. Este diseño implica la determinación de la función de activación a emplear, el número de neuronas y el número de capas en la red.

Como se ha comentado anteriormente, la elección de la función de activación se suele hacer basándose en el recorrido deseado, y el hecho de elegir una u otra, generalmente, no influye en la capacidad de la red para resolver el problema.

En lo que respecta al número de neuronas y capas, algunos de estos parámetros vienen dados por el problema y otros deben ser elegidos por el diseñador. Así, por ejemplo, tanto el número de neuronas en la capa de entrada, como el número de neuronas en la capa de salida, vienen dados por las variables que definen el problema.

El número de capas ocultas y el número de neuronas en estas capas deben ser elegidos por el diseñador. No existe un método o regla que determine el número óptimo de neuronas ocultas para resolver un problema dado. En la mayor parte de las aplicaciones prácticas, estos parámetros se determinan por prueba y error. Partiendo de una arquitectura ya entrenada, se realizan cambios aumentando y disminuyendo el número de neuronas ocultas y el número de capas hasta conseguir una arquitectura adecuada para el problema a resolver, que pudiera no ser la óptima, pero que proporciona una solución. [1]

1.3.7.2 Capacidad de generalización

A la hora de evaluar el comportamiento de una red de neuronas, y en particular del Perceptrón Multicapa, no sólo es importante saber si la red ha aprendido con éxito los patrones utilizados durante el aprendizaje, sino que es imprescindible, también, conocer el comportamiento de la red ante patrones que no se han utilizado durante el entrenamiento. Es decir, de nada sirve disponer de una red que haya aprendido correctamente los patrones de entrenamiento y que no responda adecuadamente ante patrones nuevos. Es necesario que durante el proceso de aprendizaje la red extraiga las características de las muestras, para poder así responder correctamente a patrones diferentes. Esto se conoce como la capacidad de la red para generalizar las características presentes en el conjunto de muestras o capacidad de generalización de la red.

La Figura 1.8 muestra cómo la generalización podría ser llevada a cabo por una red. Los puntos etiquetados con cruces representan los patrones de entrenamiento, la línea

punteada representa la función a aproximar y la línea sólida representa la salida que proporciona el Perceptrón Multicapa una vez entrenada. En la Figura 1.8(a), la red ha alcanzado un buen nivel de generalización, mientras que en la Figura 1.8(b) se observa que la red ha memorizado los patrones de entrenamiento y no ha conseguido una buena generalización. [1]

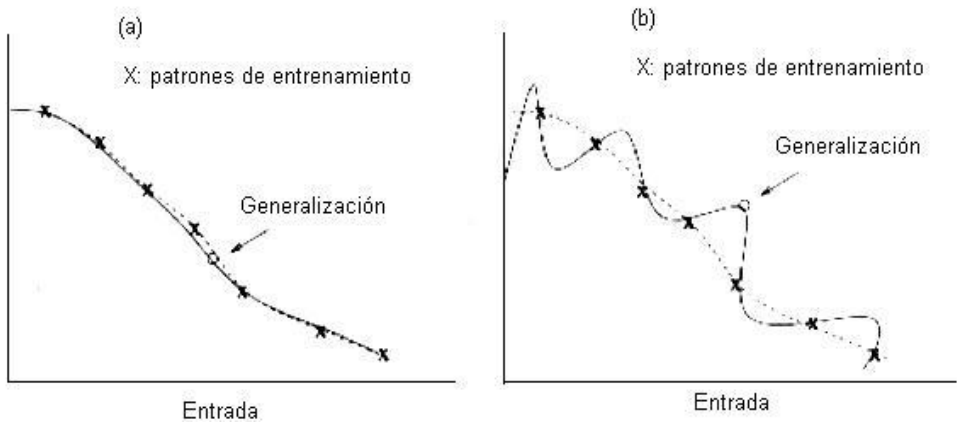


Figura 1.8. Generalización: (a) Buenas propiedades; (b) Escasas propiedades

Por tanto, cuando se realiza el proceso de aprendizaje de la red es muy importante, e incluso imprescindible, evaluar la capacidad de generalización. Para ello, es necesario disponer de dos conjuntos de muestras o patrones; uno para entrenar la red y modificar sus pesos y umbrales -*conjunto de entrenamiento*-, y otro para medir la capacidad de la red para responder correctamente ante patrones que no han sido utilizados durante el entrenamiento -*conjunto de validación o test*-. Estos conjuntos se obtienen de las muestras disponibles sobre el problema y es conveniente que la separación sea aleatoria, con el fin de tener conjuntos lo más representativos posible, tanto de entrenamiento como de validación.

A veces, y dependiendo de las características de los conjuntos de entrenamiento y validación, un entrenamiento riguroso podría anular la capacidad de generalización de la red. Por tanto, en ocasiones puede ser conveniente exigir un menor aprendizaje de la red sobre los patrones de entrenamiento, con el objetivo de obtener mejores propiedades de generalización.

El sobreaprendizaje en un Perceptrón Multicapa ocurre, por tanto, cuando la red ha aproximado correctamente los patrones de aprendizaje, pero no es capaz de responder adecuadamente ante los patrones de validación. Este hecho puede producirse, como se ha comentado anteriormente, debido a un número elevado de ciclos de aprendizaje.

Sin embargo, no es la única causa de sobreaprendizaje. En ocasiones, es producido por la utilización de demasiadas neuronas ocultas en la red. Un número excesivo de neuronas ocultas puede conducir a una escasa capacidad de generalización de la red. En estos casos, la red tiende a ajustar con mucha exactitud los patrones de entrenamiento, lo cual evita que la red extraiga la tendencia o las características del conjunto de entrenamiento. Particularmente, en problemas en los que las muestras poseen ruido, la utilización de muchas neuronas ocultas hace que la red se ajuste al ruido de los patrones, impidiendo así la generalización. [1]

1.3.8 Redes de Neuronas de Base Radial

El modelo de Funciones de Base Radial [15, 16] o *RBF*, aunque de reciente introducción, cada vez cuenta con más aplicaciones prácticas gracias a su simplicidad, generalidad y rapidez de aprendizaje. Se trata de un modelo que a menudo se estudia junto al *MLP* por ser una red unidireccional para aproximación funcional, pero que puede considerarse de tipo híbrido por incorporar aprendizaje supervisado y no supervisado.

Como sucede en el caso del *MLP*, las *RBF* permiten modelar con relativa facilidad sistemas no lineales arbitrarios, con la particularidad de que el tiempo requerido para su entrenamiento suele ser mucho más reducido.

1.3.8.1 Arquitectura de las Redes de Base Radial

La arquitectura de una red RBF cuenta con tres capas de neuronas, de entradas, oculta y capa de salida (similar a un MLP de una sola capa oculta). Las neuronas de entrada, como suele ser habitual, simplemente envían la información del exterior hacia las neuronas de la capa oculta. Las neuronas de la capa de salida son lineales,

esencialmente calculan la suma ponderada de las salidas que proporciona la capa oculta. [11]

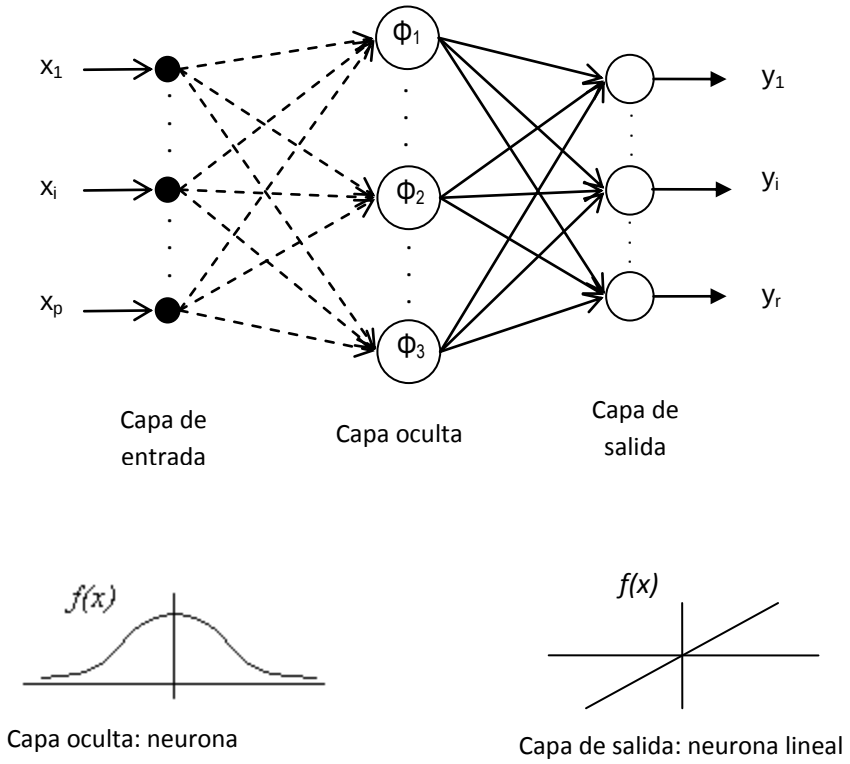


Figura 1.9. Arquitectura del *RBF* y funciones de activación

La diferencia fundamental entre la arquitectura de este modelo y la del MLP se encuentra en la operación de las neuronas ocultas. Éstas, en vez de computar la suma ponderada de las entradas y aplicarle una función de tipo sigmoideo, operan en base a la distancia que separa el vector de entradas respecto del vector sináptico que cada una almacena (centroide), cantidad a la que aplican una función radial con forma gaussiana (Figura 1.9). Es decir, así como en el MLP las neuronas ocultas poseen una respuesta de rango infinito (cualquier vector de entrada, con independencia del lugar del espacio de entrada de donde proceda puede causar que la neurona se active), en el RBF las neuronas son de **respuesta localizada**, pues sólo responden con una intensidad apreciable cuando el vector de entradas presentado y el centroide de la neurona pertenecen a una zona próxima en el espacio de las entradas. [11]

Carácter local de las Redes de Base Radial

Las funciones de base radial ϕ (ver Figura 1.10) se caracterizan porque poseen un nivel máximo de activación para valores de entrada cercanos a cero y dicho nivel decrece a medida que la variable de entrada se aleja de dicho punto.

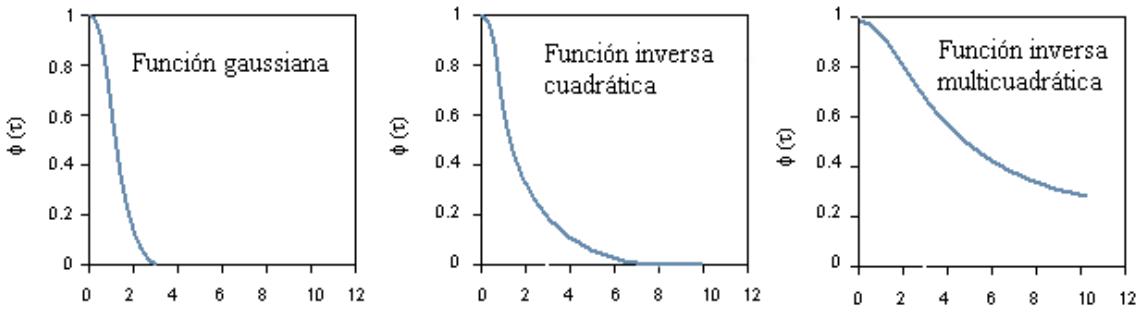


Figura 1.10. Funciones de base radial

En la Figura (1.9) se observa que la función de activación de la neurona oculta viene dada por una función gaussiana. Por tanto, si el patrón de entrada a la red está en la vecindad del centro, la neurona oculta alcanzará un valor alto de activación. A medida que el patrón de entrada se aleja del centro -dependiendo de la desviación- la activación de la neurona disminuye, y puede activarse otra neurona oculta de la red. [1]

Por esta razón, se dice que las redes de neuronas de base radial son redes con carácter local, ya que, dado un patrón de entrada a la red, sólo aquellas neuronas ocultas cuyos centros estén en la vecindad de dicho patrón se van a activar; el resto de las neuronas ocultas permanecerán inactivas o con un menor nivel de activación. Esto no sucede cuando se utilizan funciones de activaciones sigmoidales, como en el caso del Perceptrón Multicapa, pues éstas se activan en todo un rango de valores. Esto hace que, aunque ambas arquitecturas sean redes multicapa con conexiones hacia adelante, cada una de ellas posea sus propias características, así como sus ventajas e inconvenientes.

Diseño de la arquitectura de las Redes de Base Radial

El número de entradas y salidas en una *RBF* viene dado por el número de variables que definen el problema. Como ocurría cuando se utilizaba un *MLP*, en algunas aplicaciones no hay lugar a duda sobre dichas variables. Sin embargo, existen aplicaciones en las que

pudiera ser necesario llevar a cabo un análisis de las variables más relevantes y significativas que definen el problema.

En lo que respecta al número de neuronas ocultas en la red, generalmente, se determina por prueba y error, variando el número de neuronas hasta conseguir una red capaz de resolver el problema, como en el caso del *MLP* para determinar el número de capas ocultas y el número de neuronas en cada capa. No obstante, y a diferencia del MLP, para las redes de base radial añadir o eliminar unas pocas neuronas ocultas podría influir significativamente en los resultados obtenidos por la red. Esto es debido a que cada neurona oculta representa una determinada región del espacio de entrada, pudiendo no estar adecuadamente representado dicho espacio, bien por la presencia de pocas neuronas, o bien por la presencia de demasiadas neuronas ocultas en una misma zona, con su posterior consecuencia negativa en la aproximación de la red. [1]

1.3.8.2 Redes de Base Radial frente a Perceptrón Multicapa

El uso de diferentes funciones de activación con las propiedades anteriormente mencionadas, hace que cada una de estas arquitecturas, Perceptrón Multicapa y Redes de Base Radial, tenga sus propias características, las cuales se analizan a continuación.

- **El Perceptrón Multicapa construye aproximaciones globales**

  Debido al uso de funciones de activación sigmoidal, el Perceptrón Multicapa construye relaciones globales entre los datos de entrada y salida disponibles. Esto hace que el aprendizaje de la red sea lento, pues el cambio en un solo peso de la red provoca cambios en la salida para todos los patrones de entrada presentados anteriormente, reduciéndose así el efecto de previos ciclos de aprendizaje y retrasando la convergencia del algoritmo de aprendizaje.

- **Las Redes de Base Radial construyen aproximaciones locales**

  Cada neurona oculta de la red de base radial se especializa en una determinada región del espacio de entrada y construyen una aproximación local en dicha región. Por tanto, la relación que definen las redes de base radial entre los datos

> de entrada y salida es una suma de funciones no lineales y locales para diferentes regiones del espacio de entrada. A diferencia de cuando se construyen aproximaciones globales, la construcción de aproximaciones locales permite que el aprendizaje sea más rápido, ya que el cambio en un solo peso de red afecta únicamente a la neurona oculta asociada a dicho peso y, por tanto, a un determinado grupo de patrones de entrada, los pertenecientes a la clase que representa la neurona oculta en cuestión. [1]

En muchos casos, sin embargo, ocurre que para poder construir una aproximación mediante la suma de aproximaciones locales se requiere un alto número de neuronas ocultas, lo cual podría influir negativamente en la capacidad de generalización de las redes de base radial.

CAPÍTULO II

HARDWARE

2.1. SISTEMA DE AUTOMATIZACIÓN S7-300

El S7-300 es un sistema de automatización que vigila las entradas y cambia el estado de las salidas conforme al programa de usuario, que puede incluir operaciones de lógica booleana, operaciones con contadores y temporizadores, operaciones aritméticas complejas, así como comunicación con otros aparatos inteligentes. Gracias a su diseño compacto, su configuración flexible y su amplio juego de operaciones, el S7-300 es especialmente apropiado para solucionar numerosas tareas de automatización.

2.1.1 Interfaces de comunicación

2.1.1.1 Multi Point Interface (MPI)

La MPI es la interfaz de la CPU con una PG/OP[4], o bien para la comunicación en una subred MPI. La velocidad de transferencia predeterminada es de 187,5 kbits/s en todas las CPUs. Para la comunicación con un S7-200, la velocidad de transferencia se puede ajustar a 19,2 Kbit/s. [14]

Aparatos conectables vía MPI

• PG/PC
• OP/TP
• S7-300/S7-400 con interfaz MPI
• S7-200 (solo a 19,2 Kbits/s)

[4] La comunicación PG permite intercambiar datos entre los equipos de ingeniería (por ejemplo PG, PC) y los módulos aptos para comunicación SIMATIC. La comunicación OP permite intercambiar datos entre los equipos de operador (por ejemplo OP, TP) y los módulos aptos para comunicación SIMATIC.

2.1.1.2 PROFIBUS (DP)

La interfaz PROFIBUS DP sirve principalmente para conectar aparatos de la periferia descentralizada. Por ejemplo, con PROFIBUS DP se pueden configurar subredes de gran tamaño.

La interfaz PROFIBUS DP se puede configurar como maestro o como esclavo, permitiendo utilizar una velocidad de transferencia máxima de 12 Mbit/s. [14]

Aparatos conectables vía PROFIBUS

- PG/PC
- OP/TP
- Esclavos DP
- Maestro DP
- Actuadores/sensores
- S7-300/S7-400 con interfaz PROFIBUS DP

2.1.1.3 PROFINET (PN)

Si se desea establecer un enlace con Industrial Ethernet, se puede hacer a través de la interfaz PROFINET integrada en la CPU. La interfaz PROFINET integrada en la CPU se puede configurar tanto a través de MPI como a través de la interfaz PROFINET. [14]

Dispositivos conectables vía PROFINET

- Dispositivos PROFINET IO (por ejemplo el módulo interfaz IM 151-3 PN en un ET 200S)
- Componentes PROFINET CBA
- S7-300/S7-400 con interfaz PROFINET (CPU 317-2 PN/DP o CP 343-1)
- Componentes de red activos (por ejemplo un *switch*)

2.1.2 Datos técnicos generales

A continuación se presenta un resumen de los datos técnicos generales del sistema de automatización S7-300. Las características técnicas en detalle, para consulta adicional, están disponibles en la referencia bibliográfica [15].

Normas y homologaciones

En la siguiente tabla se resumen las homologaciones internacionales en las que se basan las características técnicas y las pruebas realizadas con los productos de la gama S7-300.

Underwriters Laboratories Inc.	UL 508 (Industrial Control Equipment)
Canadian Standards Association	C22.2 No. 142 (Process Control Equipment)
Underwriter Laboratory Canada & United States	UL 508 (Industrial Control Equipment) CSA C22.2 No. 142 (Process Control Equipment)
HAZ. LOC. **Underwriters Laboratories Inc.**	UL 508 (Industrial Control Equipment) CSA C22.2 No. 142 (Process Control Equipment) UL 1604 (Hazardous Location) CSA-213 (Hazardous Location) ▪ Class I, Division 2, Group A, B, C, D Tx; ▪ Class I, Zone 2, Group IIC Tx
APPROVED **Factory Mutual Approved**	Approval Standard Class Number 3611, 3600, 3810 ▪ Class I, Division 2, Group A, B, C, D Tx; ▪ Class I, Zone 2, Group IIC Tx
	según EN 60079-15:2003 (Electrical apparatus for potentially explosive atmospheres; Type of protection "n") ▪ II 3 G EEx nA II T4..T6

C-Tick for Australia & New Zeeland	El sistema de automatización S7-300 cumple las exigencias de la norma AS/NZS 2064 (Class A).
IEC 61131	El sistema de automatización S7-300 cumple los requisitos y criterios especificados en la norma CEI 61131-2 (autómatas programables, Parte 2: requisitos y verificaciones del material).

Tabla 2.1. Homologaciones Internacionales

2.1.3 Datos técnicos generales de la CPU 315-2 PN/DP

Datos técnicos

Datos técnicos

Memoria	
Memoria de trabajo	
▪ Memoria de trabajo	256 KB
▪ Ampliable	No
▪ Tamaño máximo de la memoria remanente para bloques de datos remanentes	128 KB

Tiempos de ejecución	
Tiempos de ejecución para	
▪ Operaciones de bits	Mín. 0,1 µs
▪ Operaciones de palabras	Mín. 0,2 µs
▪ Aritmética en coma fija	Mín. 2,0 µs
▪ Aritmética en coma flotante	Mín. 3 µs

Temporizadores/contadores y su remanencia	
Contadores S7	256
Contadores IEC	Sí
Temporizadores S7	256
Temporizadores IEC	Sí

Áreas de datos y su remanencia	
Marcas	2048 bytes
Bloques de datos	
▪ Cantidad	1023 (en el rango numérico de 1 a 1023)
▪ Tamaño	16 KB

Tabla 2.2. Datos técnicos de la CPU 315-2-PN/DP

Las características técnicas en detalle de la CPU 315-2-PN/DP, para consulta adicional, están disponibles en la referencia bibliográfica [14].

2.2 MÓDULOS DE ENTRADA Y SALIDA

A continuación se presenta un resumen con las principales características del módulo de entrada y salida utilizado en la elaboración del presente proyecto. Los detalles técnicos de todos los módulos de entrada/salida, digital y analógicos disponibles para el PLC S7-300, están disponibles en la referencia bibliográfica [15].

2.2.1 Módulo de entrada/salida analógica

En la siguiente tabla se resumen las propiedades principales del módulo SM334 AI4/AO2x8/8 bit de entradas/salidas analógicas.

Propiedades	**Módulos**
	SM 334; AI 4/AO 2 x 8/8 Bit (-0CE01-)
Cantidad de entradas	4 entradas formando 1 grupo de canales
Cantidad de salidas	2 salidas formando 1 grupo de canales
Resolución	8 bits
Tipo de medición	Ajustable por cada grupo de canales: • Tensión • Intensidad
Tipo de salida	En cada canal: • Tensión • Intensidad
Soporta modo isócrono	No
Diagnóstico parametrizable	No
Alarma de diagnóstico	No
Supervisión de valores límite	No
Alarma de proceso al rebasarse el valor límite	No
Alarma de proceso al finalizar el ciclo	No
Emisión de valores sustitutivos	No
Relaciones de potencial	• Sin separación galvánica frente a la interfaz con el bus posterior • Con separación galvánica frente a la tensión de carga
Particularidades	no parametrizable; ajuste del tipo de medición y de salida mediante cableado

Tabla 2.3. Módulos de entradas/salidas analógicas: Resumen de las propiedades

CAPÍTULO III

SOFTWARE

3.1 SIMATIC STEP 7

STEP 7 es el software estándar para programar y configurar los sistemas de automatización *SIMATIC*. Dicho software se compone de una serie de aplicaciones o herramientas que permiten implementar soluciones parciales, como por ejemplo:

- Configurar y parametrizar el hardware
- Crear y comprobar los programas de usuario
- Configurar segmentos y enlaces

El paquete de programación *STEP* 7 constituye un entorno de fácil manejo para desarrollar, editar y observar el programa necesario con objeto de controlar la aplicación. Comprende tres editores que permiten desarrollar de forma cómoda y eficiente el programa de control.

A todas las aplicaciones se accede desde el Administrador *SIMATIC*, reuniendo en un proyecto todos los datos y ajustes necesarios para su tarea de automatización. Dentro de dicho proyecto, los datos se estructuran por temas y se representan en forma de objetos. [16]

3.1.1 Herramientas auxiliares

El software estándar *STEP* 7 ofrece toda una serie de herramientas necesarias para la elaboración de cualquier tarea de automatización:

Administrador SIMATIC

El Administrador *SIMATIC* gestiona todos los datos pertenecientes al proyecto de automatización, independientemente del sistema de destino (S7/M7/C7) donde se encuentren.

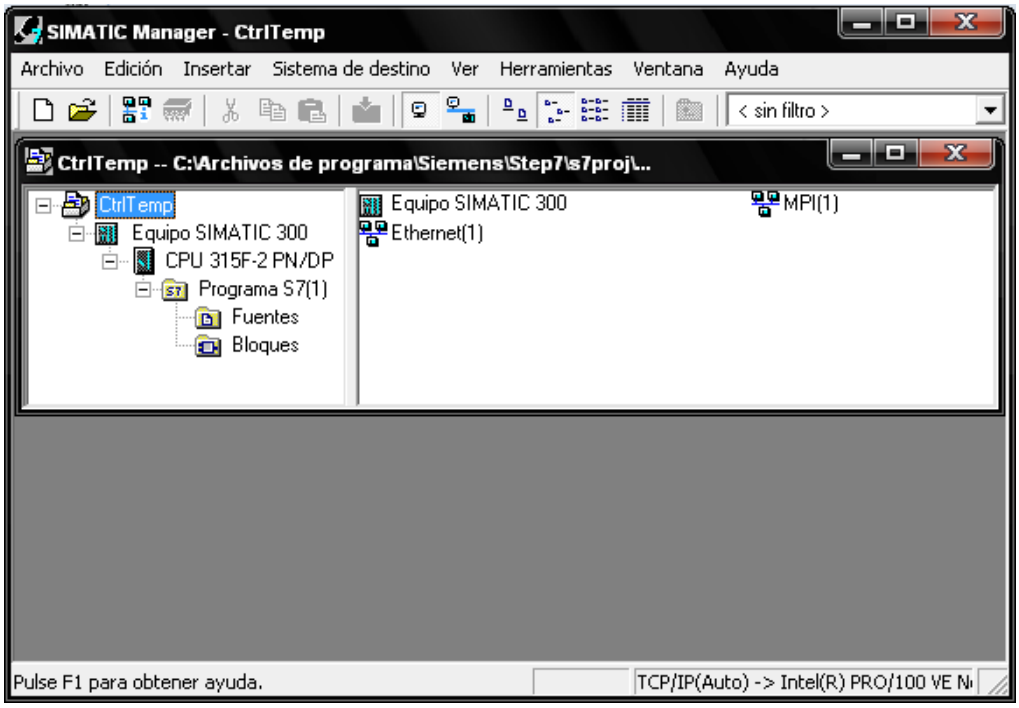


Figura 3.1. Administrador SIMATIC

KOP/AWL/FUP: Programar bloques

Permite crear y probar bloques para las CPUs *SIMATIC* S7 en los lenguajes de programación KOP, FUP y AWL.

Además de poder crear programas, es decir, de crear y editar bloques lógicos, bloques de datos y tipos de datos de usuario, dispone de funciones adicionales para programar, comprobar y poner en marcha el programa:

- Programar con símbolos
- Declarar variables

- Leer informaciones de estado y datos de operación de la CPU mediante el comando de menú Información del módulo (Menú Sistema de destino)
- Visualizar y cambiar el estado operativo de la CPU mediante el comando de menú Estado operativo (Menú Sistema de destino)
- Borrado total de CPUs
- Visualizar y ajustar la fecha y la hora de la CPU mediante el comando de menú Ajustar la hora (Menú Sistema de destino)
- Crear o visualizar los datos de referencia
- Comparar bloques
- Comprobar la consistencia de los bloques
- Declarar multi-instancias:
- Generar fuentes AWL.
- Definir definiciones de errores para el diagnóstico del proceso. [17]

HW-Config: Configuración del hardware

Esta herramienta se utiliza para configurar y parametrizar el hardware de un proyecto de automatización. Se dispone de las siguientes funciones:

- Parametrización de módulos centrales (CPU) de los sistemas de automatización S7 y M7, así como de algunos módulos de función especiales (FM) del sistema de automatización M7.
- Configuración de la periferia descentralizada (PROFIBUS DP).
- Configuración de los módulos digitales y analógicos para los sistemas de automatización S7-300 y M7-300, así como de los módulos especiales, por ejemplo los módulos de ampliación para M7-300. [18]

NetPro

Para poder establecer comunicaciones hay que configurar en cualquier caso una red, independientemente de si desea establecerla mediante comunicación de datos globales o mediante bloques de comunicación en el programa de usuario.

El paquete de configuración *NetPro* permite:

- Crear una representación gráfica de la red (compuesta por una o varias subredes).
- Definir las propiedades y los parámetros de cada subred.
- Definir las propiedades de estación de los módulos que están conectados a la red.
- Documentar la configuración de la red. [18]

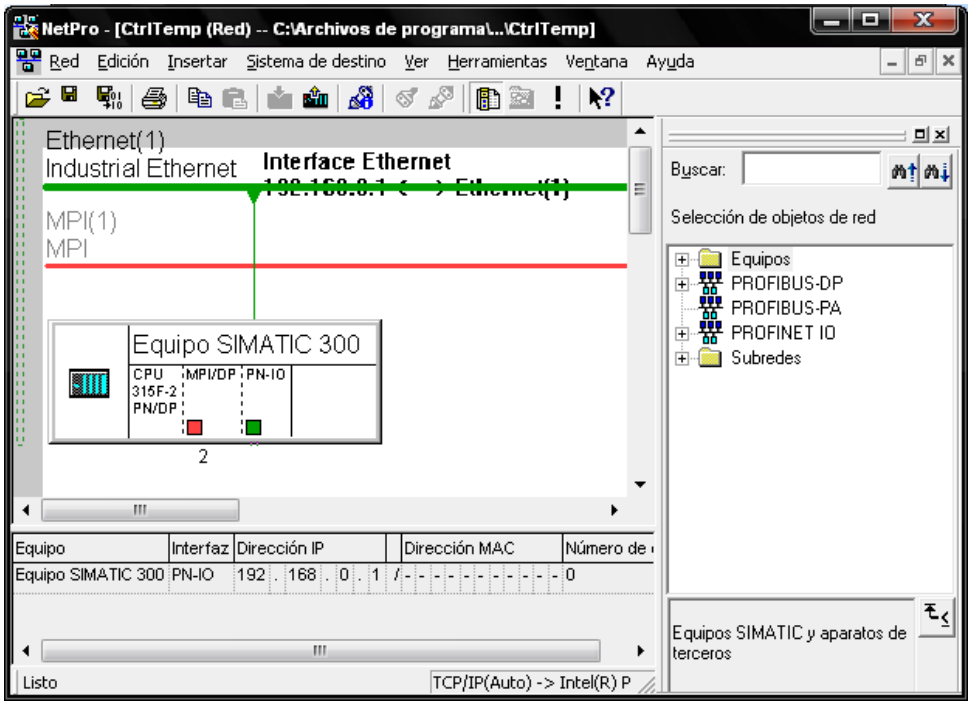


Figura 3.2. NetPro

3.2 SIMATIC *WinCC*

SIMATIC WinCC es un sistema *HMI* (Interfaz Hombre-Máquina) de supervisión sobre PC ejecutable bajo Microsoft Windows 2000 y Windows XP. Está concebido para la visualización y manejo de procesos, líneas de fabricación, máquinas e instalaciones.

El control sobre el proceso en sí, lo tiene el autómata programable PLC (Controlador Lógico Programable). Es decir, por un lado hay una comunicación entre *WinCC* y el operador, y por otro lado entre *WinCC* y los autómatas programables. [19]

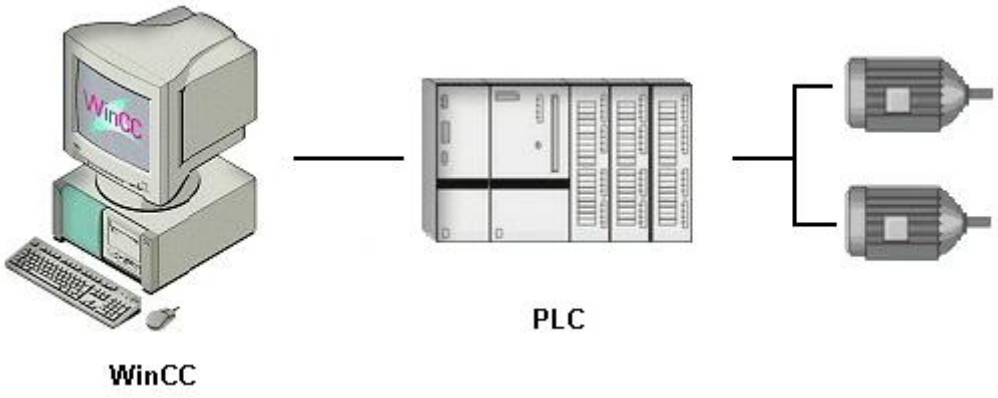


Figura 3.3. Comunicación entre *WinCC*, el PLC y los actuadores

3.2.1 Sistema base *WinCC*

El sistema básico *WinCC* se compone de los siguientes subsistemas:

- Sistema de gráficos
- Sistema de avisos
- Sistema de archivo
- Sistema de informes
- Comunicación

3.2.1.1 Sistema de gráficos

Con el sistema de gráficos se confeccionan las imágenes que reproducen el proceso en tiempo de ejecución (*runtime)*. Las tareas del sistema de gráficos son:

- Representar todos los elementos de imagen estáticos y manejables, tales como textos, gráficos o botones de comando
- Actualizar elementos de imagen dinámicos, por ejemplo modificar la longitud de una barra en función de un valor del proceso
- Reaccionar a las entradas operativas, por ejemplo la pulsación de un botón o la entrada de un texto en un campo de entrada

Componentes del sistema de gráficos.

- *Graphics Designer* es el editor con el que se confeccionan las imágenes.

- *Graphics Runtime* muestra las imágenes en la pantalla en tiempo real y administra todas las entradas y salidas.

3.2.1.2 Sistema de avisos

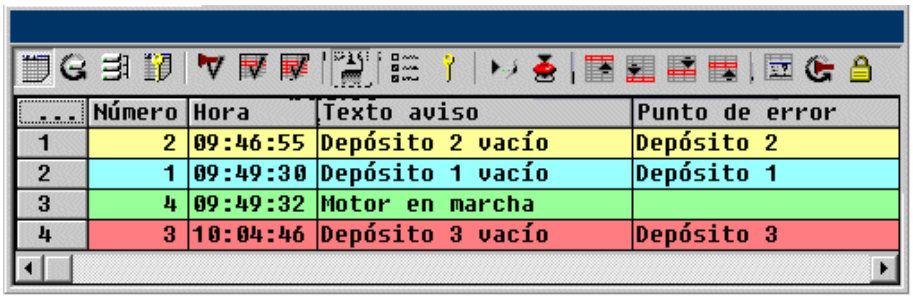
Los avisos informan al operador acerca de los estados de funcionamiento y de fallo que se producen durante el proceso. Sirven para poder detectar con antelación situaciones críticas y para evitar periodos de inactividad.

Componentes del sistema de avisos.

- *Alarm Logging* se emplea para la configuración del sistema de avisos. Para visualizar los avisos se dispone en *Graphics Designer*, además, de un objeto de visualización especial, *WinCC Alarm Control*.

- *Alarm Logging Runtime* se encarga de controlar, durante el tiempo de ejecución, la emisión de los avisos y de administrar los acuses de los mismos.

Los avisos se visualizan en tablas de control de alarmas *WinCC Alarm Control*.



...	Número	Hora	Texto aviso	Punto de error
1	2	09:46:55	Depósito 2 vacío	Depósito 2
2	1	09:49:30	Depósito 1 vacío	Depósito 1
3	4	09:49:32	Motor en marcha	
4	3	10:04:46	Depósito 3 vacío	Depósito 3

Figura 3.4. Tabla de control de alarmas

3.2.1.3 Sistema de archivos

Los valores reales del proceso se pueden visualizar en todo momento. Pero si se quiere representar la evolución cronológica que experimenta un valor de proceso, por ejemplo en un diagrama o en una tabla, se tendrá que acceder a valores de proceso del pasado. Esos valores están memorizados en archivos de valores del proceso.

Componentes del sistema de archivos.

- *Tag Logging* sirve para configurar archivos de valores de proceso y archivos comprimidos, define los ciclos registrados y archivados y elige los valores de proceso archivados.

- *Tag Logging Runtime* se encarga de escribir en el fichero de valores del proceso, aquellos valores que deben ser archivados en *runtime*, así como de leer los valores, en el fichero de valores de proceso. Por ejemplo cuando se necesitan esos valores para representarlos en uno de los controles, o para evaluarlos con mayor minuciosidad.

3.2.1.4 Sistema de informes

Los informes para documentar el proyecto contienen sinopsis de los datos de configuración, por ejemplo una tabla con todas las variables, funciones o gráficos utilizados en el proyecto.

Componentes del sistema de informes.

- *Report Designer*. Este componente se utiliza para adaptar diseños estándar preconfigurados a los requerimientos concretos del proyecto, así como para confeccionar nuevos diseños. Con *Report Designer* también se crean las órdenes para iniciar la impresión.

- *Report Runtime* toma de los ficheros o controles los datos a imprimir y dirige la salida por impresora.

3.2.1.5 Comunicación

La comunicación entre *WinCC* y los autómatas programables se realiza a través del respectivo bus de proceso, por ejemplo Ethernet o PROFIBUS. Controladores de comunicación especializados, denominados canales, se encargan de gestionar la comunicación. *WinCC* dispone de canales para los sistemas de automatización *SIMATIC* S5/S7/505 como de canales independientes del productor, como PROFIBUS DP y OPC. [19]

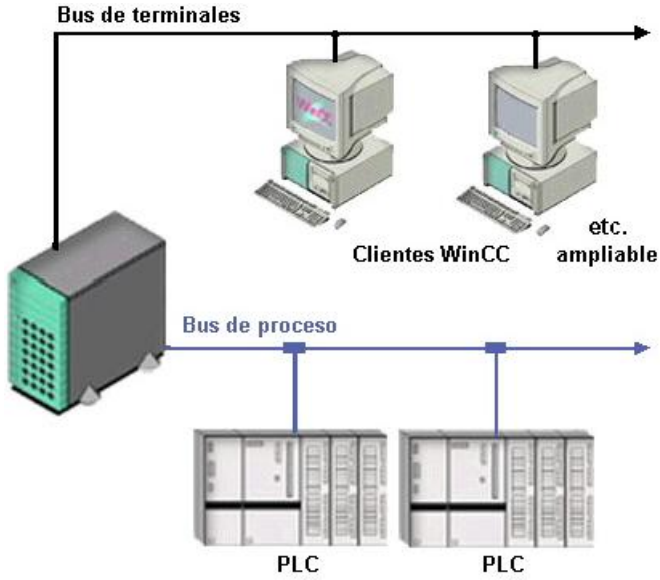


Figura 3.5. Esquema del sistema de comunicación entre *WinCC* y los autómatas programables

Comunicación con los autómatas programables.

Las variables de proceso constituyen el eslabón de enlace para intercambiar datos entre *WinCC* y los autómatas programables. A cada variable de proceso de *WinCC* le corresponde un determinado valor de proceso en la memoria de uno de los autómatas programables conectados. En *Runtime*, *WinCC* lee en la memoria el autómata programable el área de datos donde está guardado ese valor de proceso y determina así cuál es el valor de la variable de proceso.

Viceversa, *WinCC* también puede volver a escribir datos en el autómata programable. El usuario maneja el proceso con *WinCC* en la medida en que el autómata programable procesa estos datos. [19]

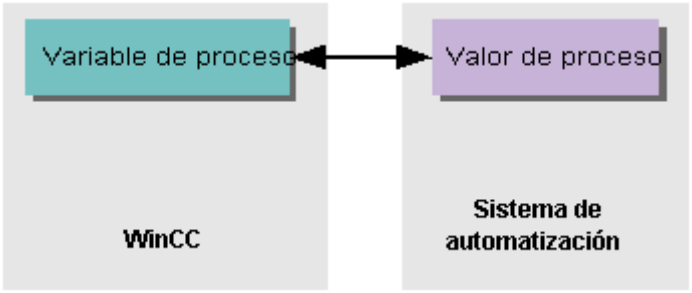


Figura 3.6. Intercambio de datos entre *WinCC* y los autómatas programables

3.3 ACTIVEX

ActiveX es una arquitectura definida como parte de la tecnología *Win32* para la comunicación entre aplicaciones de software que corren simultáneamente, ya sea dentro o fuera de proceso. Es un estándar de Windows para módulos de programas con su propia interfaz de usuario. A estos módulos de programas se les denomina controles *ActiveX*. Un control *ActiveX* puede contener un botón de comando específico, o un elemento indicador gráfico, por ejemplo.

Los controles *ActiveX* son similares a los de los lenguajes de programación como Microsoft Visual Basic para Aplicaciones. Estos controles pueden ser barras de desplazamiento, botones de comando, botones de opción, casillas de verificación u otros controles, y se utilizan para crear programas, cuadros de diálogo o formularios personalizados, ofrece opciones a los usuarios o ejecuta macros o secuencias de comandos que automatizan una tarea.

3.3.1 Controles *ActiveX* en *WinCC*

Con *WinCC* se suministran numerosos controles *ActiveX*. *Siemens* proporciona el control *NeuroSystems* para enlazar a *WinCC* con la red neuronal configurada en el software *NeuroSystems*.

3.3.1.1 Propiedades

El control *NeuroSystems* cuenta con 113 propiedades que se pueden dividir en cuatro grupos:

- Ruta de archivo (1)
- Valores de entrada (100)
- Valores de salida (10)
- Disparador (*Trigger*) (2)

Las propiedades se enumeran en la siguiente lista:

Propiedades	**Descripción**
NeuroFilePath	Ruta a un archivo *NEUROSYSTEMS* valido (.snl)
NeuroIn1	Primera entrada (del tipo: float)
…	…
NeuroIn100	100th entrada (del tipo: float)
NeuroOut1	Primera salida (del tipo: float)
…	…
NeuroOut10	10th output (del tipo: float)
Trigger	Calcula la nueva salida cuando el control cambia de 0 a 1 (del tipo: boleana)
TriggerSetup	*=0, el control espera la propiedad Trigger* *=1, el control calcula la salida cada vez que un valor de entrada cambia (del tipo: entero)*

Tabla 3.1. Propiedades del control ActiveX

Hay 100 valores de entrada, cada uno con el nombre *NeuroIn* y el número de la entrada (1 a 100) al final del nombre, por ejemplo, *NeuroIn1*, *NeuroIn2*, *NeuroIn3*, etc. La propiedad es del tipo 'float'.

Hay 10 valores de salida, cada uno con el nombre *NeuroOut* y el número de la salida (1 a 10) al final del nombre, por ejemplo, *NeuroOut1*, *NeuroOut2*, *NeuroOut3*, etc. La propiedad es del tipo 'flotante'.

El *NeuroFilePath* es normalmente una entrada que especifica la ruta del archivo a un archivo válido *NeuroSystems* (*.snl).* Tan pronto como la ruta de acceso a un archivo *NeuroSystems* (*.snl*) sea válida, los valores de las entradas podrán ser usadas para calcular las salidas y los productos serán sobrescritos.

El valor de la propiedad *TriggerSetup* puede ser fijado a 0 o 1. Si se configura a 0, el control espera que la propiedad cambie de 0 a 1 para calcular la nueva salida. Sin embargo, si se establece en 1, el control calcula los valores de salida cada vez que un valor de entrada sea cambiado. La propiedad *TriggerSetup* es dato del tipo 'entero', el disparador (*Trigger*) de la propiedad es un 'booleano'. [2]

3.3.1.2 Eventos

Los eventos se enumeran en la tabla a continuación:

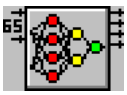
Evento	**Descripción**
NeuroOut1 cambia	El valor de la primera salida ha cambiado
NeuroOut2 cambia	El valor de la segunda salida ha cambiado
...	...
NeuroOut10 cambia	El valor de la décima salida ha cambiado

Tabla 3.2. Eventos del control ActiveX

Estos eventos se disparan (se activan) cuando los valores de las salidas cambian. Esto ocurre cuando las entradas cambian mientras un archivo válido *NeuroSystems* (*.snl*) ha sido cargado. [2]

3.3.1.3 Interfaz gráfica

Modo Normal



Este se muestra cuando la licencia a sido instalada, un archivo válido *NeuroSystems* (*.*snl*) es cargado y todos los archivos *DLL* están disponibles y funcionando

correctamente. Se puede apreciar el icono normal en el centro y los pequeños iconos que indican el número de entradas y salidas definidas por el *NeuroSystems*. Las entradas se muestran como flechas a la izquierda del icono principal cuando son inferiores o iguales a cuatro. Cuando hay más de cuatro entradas, un número entre dos flechas muestra la cantidad de entrada.

Modo de Error

Cuando se muestra este cuadro el archivo *NeuroSystems* (*.*snl*) no se ha cargado, o no se encontró la licencia, o la licencia no puede ser autentificada porque es un *DLL* que falta o está dañado.

Modo No NeuroDLL

Cuando se muestra este cuadro, el *NEUROSYSTEMS* DLL (*NeuroDLL.dll*) no ha sido encontrado o no se pudo cargar.

La solución en detalle para los problemas de Modo de Error o No NeuroDLL está disponible en la referencia bibliográfica [2].

3.4 NEUROSYSTEMS

La herramienta de configuración *NeuroSystems* se ha creado para definir y configurar redes neuronales especialmente para PLCs de la familia *SIMATIC* S7 y para el sistema de visualización de procesos *SIMATIC WinCC*. *NeuroSystems* se puede utilizar, por ejemplo, para identificar un patrón, crear modelos, optimizar controladores en lazo cerrado.

3.4.1 Características Principales [20]

- **Escritorio de operador intuitivo de NeuroSystems.**

  Se presenta un escritorio de operador intuitivo para asegurar que se puede empezar a trabajar sin tener conocimientos de especialista sobre redes neuronales. *NeuroSystems* no sólo permite el proceso de aprendizaje, sino también la configuración y análisis de redes neuronales.

- **Entrenamiento sencillo de las redes**

  Se pueden entrenar las redes neuronales de manera muy sencilla y sin necesidad de tener conocimientos especializados, a través de procedimientos de aprendizaje estacionarios con ajuste automático de los parámetros de aprendizaje. De esta forma se pueden optimizar los Perceptrones Multicapa, las redes de Función Base Radial y las redes *Neuro-Fuzzy*.

- **Análisis en línea a través de diagramas**

  Se pueden utilizar diagramas gráficos de dos, tres y cuatro dimensiones para ofrecer un análisis en línea de las redes neuronales. En el modo de funcionamiento *offline*, se dispone de un generador gráfico programable para la simulación de procesos.

- **Redes *Neuro-Fuzzy***

  NeuroSystems es el complemento a *FuzzyControl*++, la herramienta de configuración para lógicas difusas (*fuzzy*). Si se están utilizando redes *Neuro-Fuzzy*, se pueden optimizar sistemas *Fuzzy* ya configurados a través de *NeuroSystems* con relación a la posición y forma de las funciones asociadas.

- **Intercambio de datos**

  NeuroSystems permite un intercambio de datos cruzados entre sistemas a través de ficheros FPL (*Fuzzy Programming Language*) y SNL (*Siemens Neuro Language*).

3.4.2 Sistemas de Destino (*Targetsystems*)

Los módulos de ejecución para los sistemas de destino S7, *ActiveX*, *OPC* y *SIMATIC WinCC* calculan los valores de salida para los valores de entrada en línea, utilizando el algoritmo de parámetros de la red neuronal. Si usamos el S7, la red de información se transmitirá al sistema de destino por escritura en un bloque de datos (DB) S7. Si se utiliza *SIMATIC WinCC* y *ActiveXControl* la información de la red serán entregados a los módulos de ejecución *Neuro.OLL* a través de un archivo **.snl*.

El número máximo de entradas y salidas, neuronas y pesos que pueden definirse depende del sistema de destino elegido:

Targetsystem	Entradas	Salidas	Neuronas	Pesos
S7-4K	4	4	99	199
S7-20K	100	10	160	1660
WinCC	10	5	Ilimitada	Ilimitada
ActiveX	100	10	Ilimitada	Ilimitada
OPC	100	10	Ilimitada	Ilimitada
CFC-4K	4	4	?	?
CFC-20K	100	10	?	?

Tabla 3.3. Parámetros de los Sistemas de Destino.

Los S7-4K y S7-20K pueden ejecutarse en *SIMATIC* S7-300 y *SIMATIC* S7-400.
El sistema de destino debe seleccionarse al crear un nuevo proyecto, y no cambiarlo más tarde, porque algunas operaciones en *NeuroSystems* dependen de esta configuración. Si se modifica la configuración, el rendimiento y la exactitud de la red neuronal finalmente no podrá alcanzar el máximo nivel posible. [2]

3.4.3 Descripción General de *NeuroSystems*

3.4.3.1 Aprendizaje

Inicio del Proceso de Aprendizaje (*Start Learning Process*)

Contiene:

• El nombre y el directorio de los archivos de datos de aprendizaje asociados,
• Opción de selección para la validación de datos,
• Opciones para dar por terminado el aprendizaje.

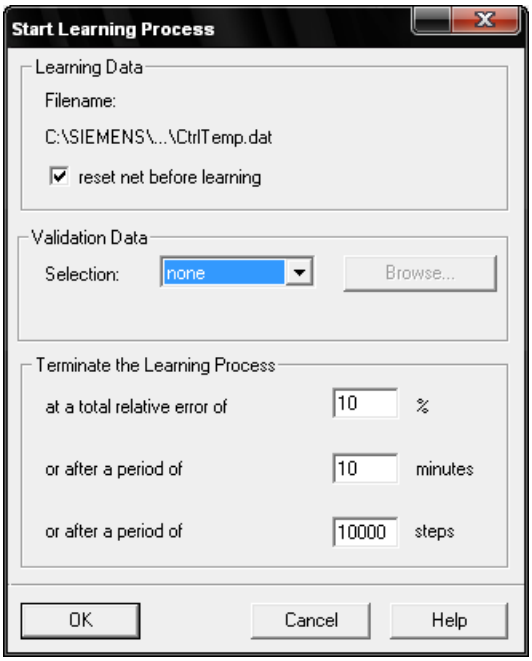


Figura 3.7. Start Learning Process.

Selección de Datos de Validación (*Validation Data*)

La red aprende mediante los datos de aprendizaje disponibles y puede comprobar el objetivo de aprendizaje utilizando un archivo de validación de datos o un conjunto de datos para validación seleccionados del archivo de datos de aprendizaje.

Para la selección adecuada de los datos para la validación se dispone de las siguientes opciones:

- *No usar datos para validación (*none*)*

  El archivo de datos de aprendizaje es utilizado completamente para el entrenamiento.

- *Selección aleatoria de los datos para validación del archivo de datos de aprendizaje (*random*)*

  40% de los conjuntos de datos que deben aprenderse son seleccionados al azar para la validación de datos. La red será entrenada entonces con el 60% restante del conjunto de datos.

- *Usar un archivo que contiene los datos de validación (*file*)*

  Es necesario seleccionar un nuevo archivo *.dat, en la que el que se encuentran los datos para la validación. En este caso el archivo de datos para el aprendizaje se utiliza completamente.

Terminación del proceso de aprendizaje (*Terminate the Learning Process*)

En esta parte se puede establecer la terminación de aprendizaje en función de dos condiciones:

- *Un % de error relativo*

  Es posible introducir un número para el límite de porcentaje de error. El error se calcula de la siguiente manera: La suma de errores cuadráticos entre los valores de salida deseados y los reales, divididos por el número de salidas y el número de conjuntos de datos aprendizaje y validación. La raíz de esta expresión es el error relativo total.

- *Después de un período de tiempo en minutos*

  Se puede fijar un límite en el tiempo de aprendizaje, mediante la introducción de un entero positivo para el máximo tiempo de aprendizaje, en cuestión de minutos.

- *Después de un número de pasos de aprendizaje*

  Ingresando un número entero positivo, entre 1 y 1.000.000 se puede determinar el número de pasos de aprendizaje.

Durante el proceso de aprendizaje, en la ventana *Error Development* se visualiza el error relativo total como un porcentaje respecto al número de pasos de aprendizaje.

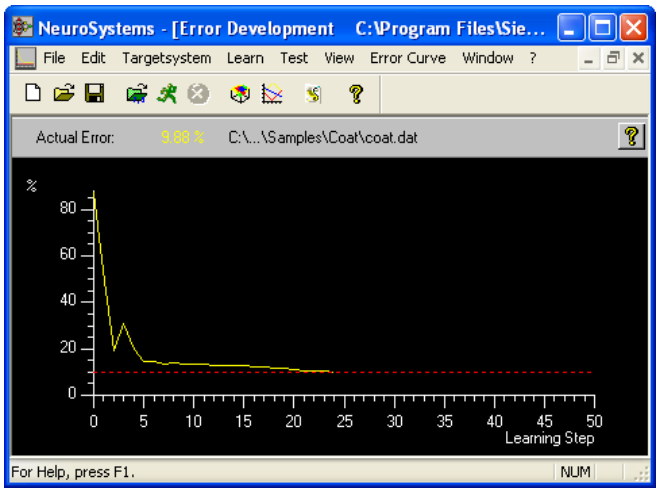


Figura 3.8. Error Development.

3.4.3.2 Prueba (*Test*)

Diagrama de Error (*Error Diagram*)

La ventana *Error Diagram* se utiliza para proporcionar un examen visual del éxito del aprendizaje. Para cada conjunto de datos de aprendizaje y validación, el error relativo se muestra como una columna.

El cálculo del error de cada conjunto de aprendizaje y validación se calcula como sigue: la suma de los errores cuadráticos entre el valor de salida deseado y el real, se divide por el número de salidas. La raíz de esta expresión es el error de un conjunto de datos de aprendizaje o validación en particular que se muestra en el diagrama de error. [2]

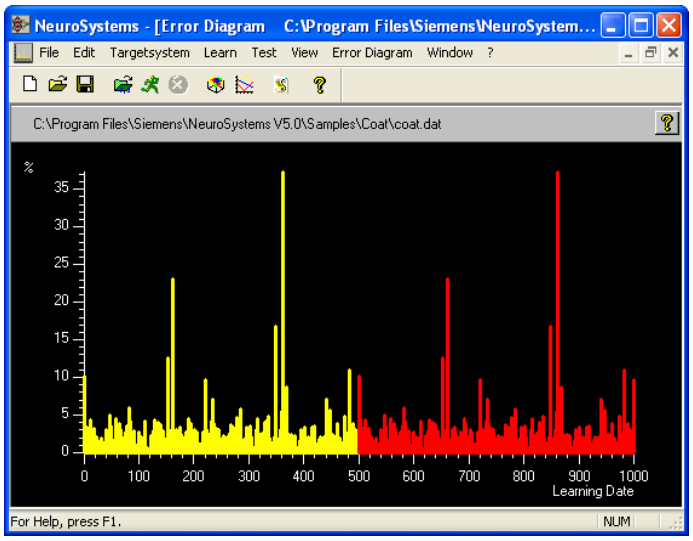


Figura 3.9. Error Diagram.

Representación de un Vector (*Vector Representation*)

En la ventana *Vector Representation*, se muestran los vectores de entrada o salida del archivo de prueba. Si se muestran los vectores de salida del archivo de prueba, estos serán comparados con los vectores de salida calculados por la red. Los vectores de salida del archivo de prueba son de color amarillo y los vectores de salida generados por la red neuronal como una respuesta a los vectores de entrada de prueba se muestra en rojo. Esto nos muestra de que manera responde la red a una entrada/salida característica, si utilizamos distintos datos del archivo de aprendizaje.

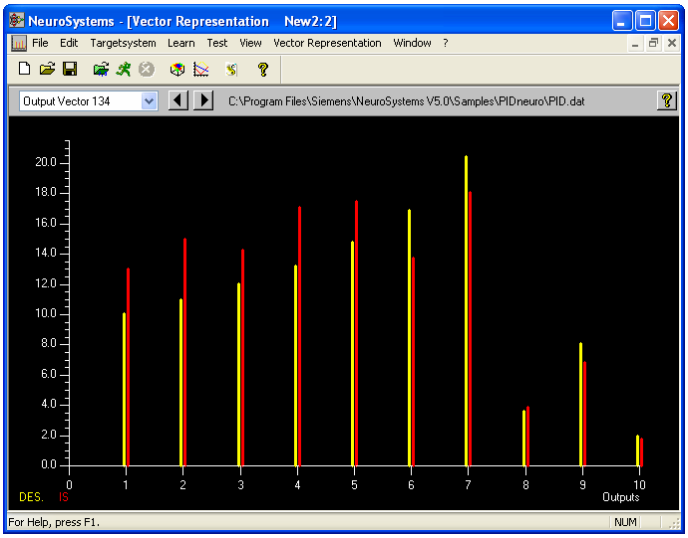


Figura 3.10. Vector Representation.

Representación de Señal (*Signal Representation*)

En la ventana *Signal Representation*, se muestra gráficamente la secuencia cronológica de cada señal de entrada o salida, de acuerdo al archivo de datos de prueba. La curva de la señal es interpolada linealmente entre los valores de los puntos en el tiempo (valores de muestreo). Las señales de salida del archivo de prueba se muestran junto con las señales de salida calculadas por la red.

La ventana de representación de la señal se puede abrir varias veces, por tanto, ofrece una visión general de las relaciones de todas las señales de entrada a una señal de salida. [2]

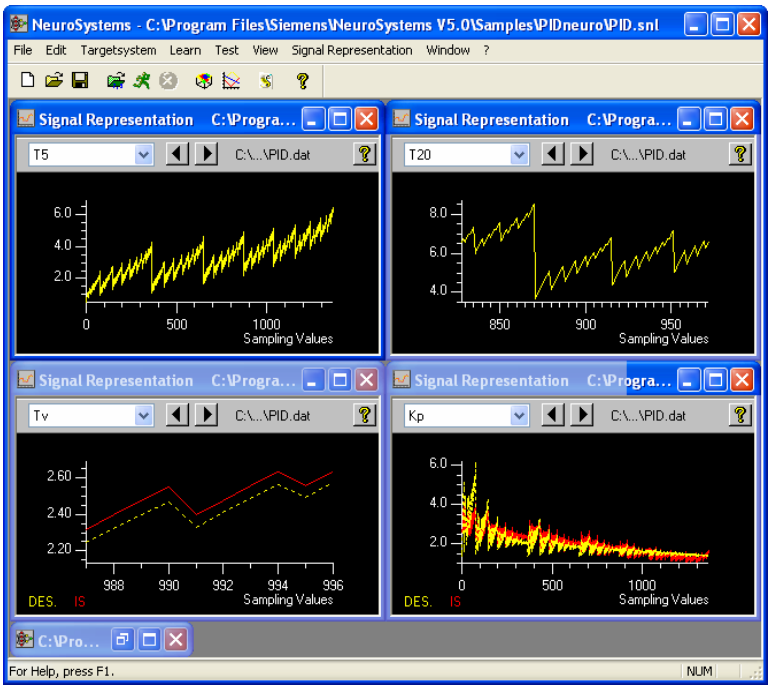


Figura 3.11. Visualización de múltiples ventanas de *Signal Representation.*

Resumen de Error (*Error Summary*)

En la ventana de *Error Summary* se muestra gráficamente la señal de salida Deseada - Real - Desviación. En este sentido los valores "deseados" de los archivo de prueba se comparan con los que se han calculado por la red "reales". La desviación se muestra gráficamente. La diagonal que se traza en el diagrama corresponde al comportamiento ideal de la red.

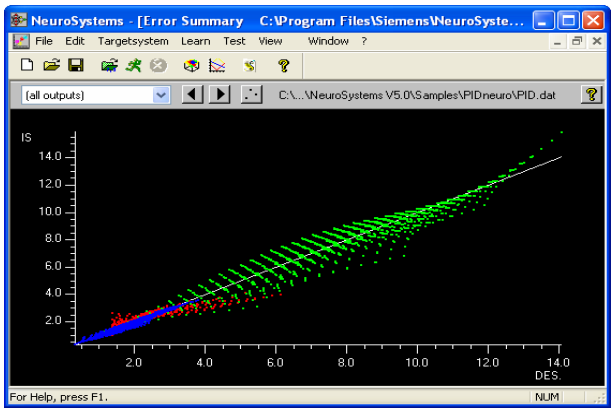


Figura 3.12. Error Summary.

Todos los puntos deben estar en o al lado de esta línea. Las señales de salida que se han creado por la red neuronal como una respuesta a las señales de entrada de prueba se muestran como una nube de puntos.

Este diagrama permite una rápida estimación de qué tan bueno es el entrenamiento de la red y en qué áreas la red se desvía del comportamiento ideal. También se puede utilizar para hacer una rápida comparación con otras redes entrenadas. [2]

Trazador Grafico de Curva (*Curve Plotter*)

El C*urve Plotter* permite desplegar gráficamente un valor de entrada o salida arbitrario. La representación es en tiempo real. Es posible archivar la curva de datos en el disco duro para leerlo de nuevo cuando sea necesario. Los valores que se muestran por encima del diagrama corresponden a los valores que se han adoptado desde la línea de lectura amarilla. [2]

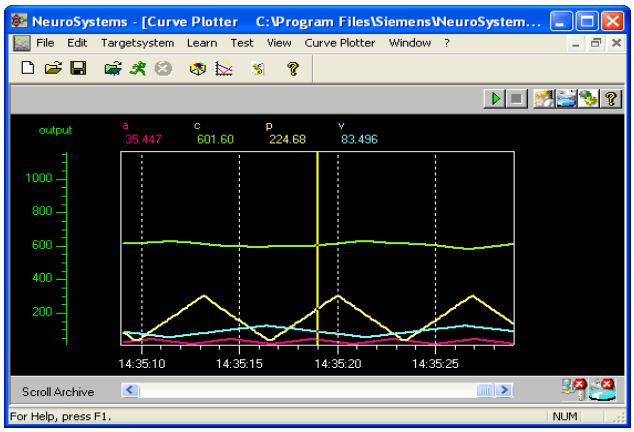


Figura 3.13. Curve Plotter

CAPÍTULO IV

SISTEMA DE ENTRENAMIENTO

Un proceso o sistema dinámico puede definirse como una combinación de elementos o componentes relacionados entre sí y que actúan para alcanzar una determinada meta. El estudio de un proceso dinámico es el estudio de la variación a lo largo del tiempo de las variables de salida de dicho proceso, a partir de las variables de entrada que afectan al comportamiento dinámico del proceso.

En toda la literatura se hace una distinción entre procesos dinámicos lineales y procesos dinámicos no lineales. Se dice que un proceso es lineal cuando la relación entre las variables que intervienen en el comportamiento de dicho proceso es lineal. En caso contrario, se dice que el proceso es no lineal. Propiedades como la capacidad de las redes para aprender a partir de ejemplos o capturar relaciones no lineales, han hecho que se hayan utilizado ampliamente en el estudio y análisis de procesos dinámicos.

El control de un proceso dinámico consiste en regular el comportamiento del proceso, regulación que se lleva a cabo manipulado la variable de entrada al proceso, también referida en este contexto como señal o acción de control. De este modo, un sistema de control se encarga de proporcionar la acción que hay que aplicar al proceso para que la salida de dicho proceso tienda al valor deseado, también llamado objetivo de control (en este caso se ha asumido, por simplicidad, que el proceso dinámico tiene una única variable de entrada. En el caso de poseer varias variables de entrada, el control se realiza a través de aquellas entradas que puedan ser manipuladas).

Las redes de neuronas artificiales ocupan un lugar importante en el desarrollo de técnicas de control para procesos dinámicos no lineales. Cuando se habla de control de procesos utilizando redes de neuronas, generalmente, se entiende que es una red la encargada de calcular la acción de control que hay que aplicar al proceso para que se alcance el objetivo de control deseado. La diversidad de los métodos o estrategias de

control radica en el modo de realizar el entrenamiento de la red, así como en el tipo de red de neuronas a utilizar. [1]

4.1 DISEÑO DEL PROBLEMA

Como caso práctico se plantea un problema para realizar un sistema controlador de temperatura mediante la utilización de una red neuronal diseñada en el software *NeuroSystems*. No es de mayor interés asignar características específicas al controlador, y sólo se hará mención a aquellas que se necesiten para la construcción del sistema de control.

Mediante una interfaz gráfica, realizada en *WinCC,* se realizará un monitoreo continuo del estado de la planta. Dicha interfaz permitirá el ingreso de una temperatura deseada (*set point*) en un rango de 20ºC a 70ºC. Por último se presentaran las graficas de tendencias y las tablas de valores de la temperatura deseada y la temperatura real.

El primer paso en la construcción de sistemas para controlar la temperatura utilizando redes de neuronas es conocer las variables que intervienen en la dinámica de dicho proceso. En este caso, nuestro sistema dispone de una variable de entrada *"u"* y una de salida *"y"*. Por tanto, el control de la temperatura se realizará mediante esta única señal de entrada (*señal de control*), que será la salida de la red de neuronas que actuará como controlador.

Figura 4.1. En el sistema controlador se debe establecer cuál va ser nuestra variable de control y cuál es nuestra salida de proceso

La variable *"u"* (*variable manipulada*), es una señal digital que actúa para subir o bajar la temperatura del proceso (*variable controlada*), de acuerdo a la temperatura deseada (*set point*).

$$y = f(u)$$

En primer lugar, y como en cualquier aplicación de redes de neuronas, es necesario disponer de un conjunto de muestras o variables numéricas que representen el problema. Para que el conjunto de datos sea representativo de la dinámica del proceso, la señal de control *"u"* debe cubrir todo su rango de operación.

Con estos datos la red de neuronas se entrena para que aprenda la dinámica inversa del proceso. En la figura a continuación se muestra el esquema general del aprendizaje. Se observa que la entrada al proceso actúa como salida deseada para la red, mientras que la salida del proceso es la entrada a la red.

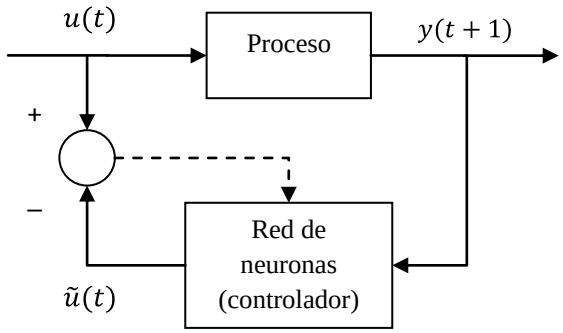


Figura 4.2. Esquema de control inverso con aprendizaje generalizado

Una vez realizado el entrenamiento de la red, ésta puede utilizarse para el control; basta sustituir la entrada a la red $y(t+1)$ por el objetivo de control y^{des}, como se muestra en la figura a continuación. [1]

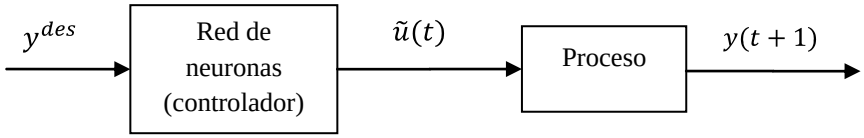


Figura 4.3. Red de neuronas actuando como controlador

4.2 PROGRAMACIÓN

4.2.1 Configuración y entrenamiento de la red neuronal en *NeuroSystems*

La configuración de la red incluye la selección del *targetsystem* (sistema de destino), la estructura interna y del número de entradas y salidas de la red.

4.2.1.1 Definir el número de entradas, salidas y el targetsystem

Para nuestro proyecto, la función se ha modelado para representar la temperatura como una función de la variable *"u"*. Para ello ingresamos 3 entradas y 1 salida. Como sistema de destino elegimos *ActiveX*. En la figura se observa la estructura de externa de la red.

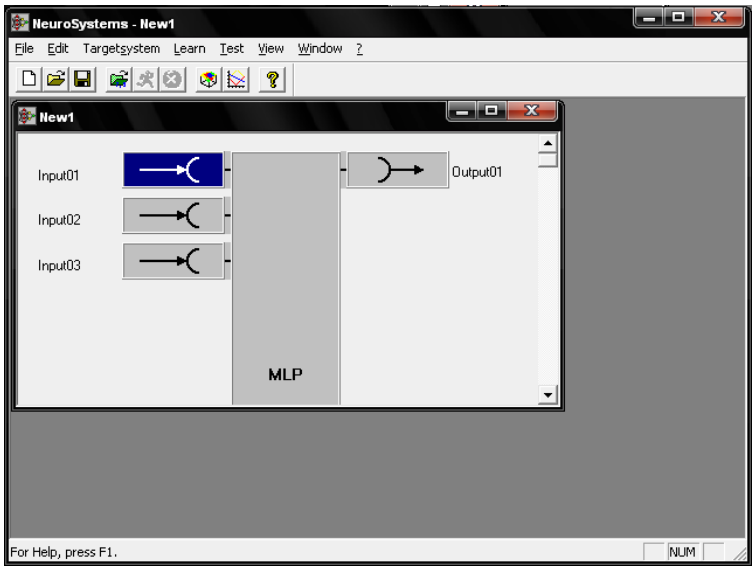


Figura 4.4. Estructura final de la red externa (bloque de red, entradas y salida)

4.2.1.2 Definición de la estructura de la red

Los siguientes tipos de red son proporcionados por *NeuroSystems*:

- *MLP (Red Perceptrón Multicapa):* Este tipo de red se recomienda si se dispone de un archivo de aprendizaje relativamente pequeño.
- *RBF (Red de Función de Base Radial):* Este tipo de red debe usarse si disponemos de un archivo de aprendizaje grande.
- *NFN (Red NeuroFuzzy):* Las redes de este tipo procesan las funciones miembro en base a un sistema difuso.

La estructura de la red se define por el número de capas y el número de neuronas en las capas (para *MLP* y *RBF*), o el número de funciones miembro de las entradas y salidas (para *NFN*).

Debido a que las capas de entrada y salida conectan la red con el mundo exterior, el número de neuronas en estas capas es necesariamente el mismo que el número de entradas y salidas definidas en el proyecto.

Hay que tener en cuenta que la capa de entrada también cuenta para el número de capas de red en el programa *NeuroSystems*. En la literatura, esta a menudo no cuenta como una capa de neuronas, ya que su principal función es distribuir las señales de las entradas. Así que generalmente trabaja de una manera diferente a las neuronas de las otras capas. [2]

En este caso usamos una red tipo *MLP*. En una primera aproximación, se fijan dos capas ocultas con 10 neuronas cada una, debido a que nuestro proyecto, tiene muy pocas entradas y salidas, y el archivo de datos de aprendizaje incluye sólo un pequeño número de conjuntos de datos. Posteriormente, este parámetro será modificado con el objetivo de conseguir la red más adecuada. La figura a continuación muestra la estructura interna de la red.

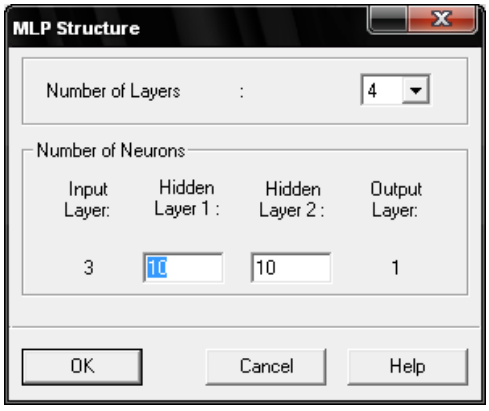


Figura 4.5. Estructura interna de la red *MLP*, 2 capas ocultas con 10 neuronas cada una

4.2.1.3 Proceso de Aprendizaje

El proceso de aprendizaje consiste en entrenar la red mediante la ejecución de pasos de aprendizaje hasta que los patrones de salida reales de la red coinciden con los patrones de salida del conjunto de datos de aprendizaje, tan estrechamente como sea posible. Un conjunto de patrones de entradas y salidas asociadas se llaman un *conjunto de datos de aprendizaje*. [1]

Para el entrenamiento de la red, deberá ser utilizado un archivo "*CtrlTemp.dat*". Este fichero contiene el comportamiento de la entrada/salida que hay que aprender, el cual se obtiene manipulando la acción de control "u" en sus diferentes rangos de operación y observando o midiendo la salida del proceso "y" para dicha señal. Esta información está representada por los conjuntos de datos que muestra la relación $y = f(u)$.

Iniciar y detener el proceso de aprendizaje

Una vez definida la arquitectura de la red, se procede a realizar el proceso de aprendizaje o entrenamiento de la red. Para ello, es necesario fijar los parámetros que intervienen en dicho proceso: los datos de validación, el límite de error, el período máximo de tiempo de entrenamiento y el número de ciclos de aprendizaje.

Estos parámetros dependen siempre del problema a resolver y será necesario realizar varias simulaciones previas para fijarlos de acuerdo con el problema. En la ventana "*Start Learning Process*" se realizan los preparativos para el aprendizaje.

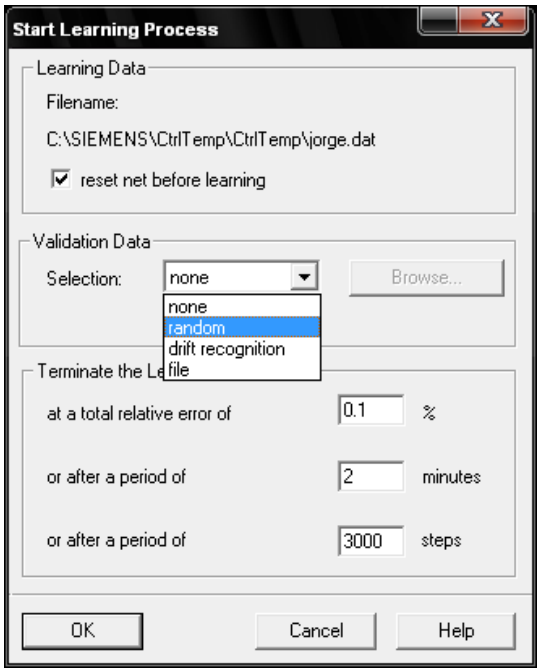
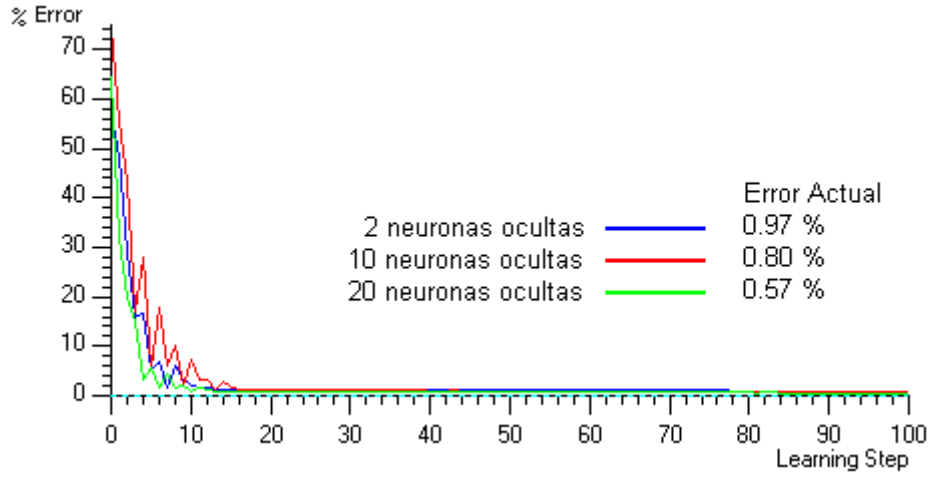
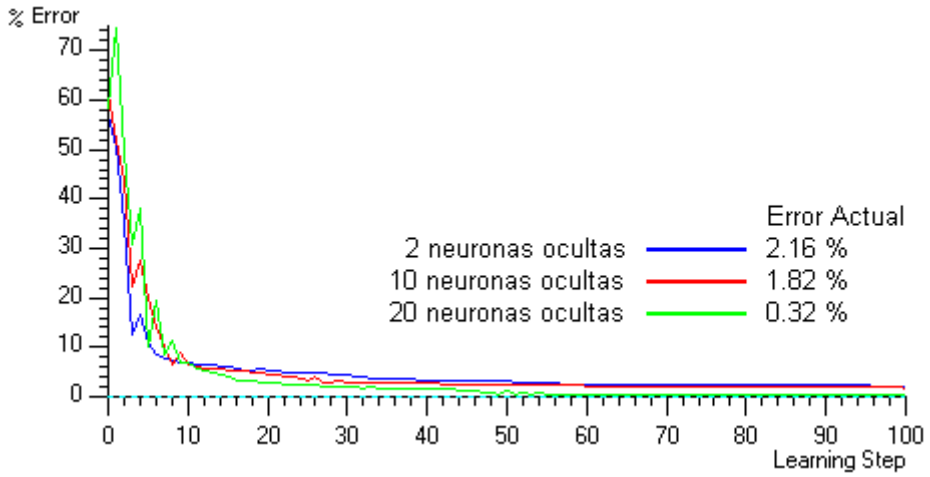


Figura 4.6. *Start Leraning Process*, permite fijar los parámetros que intervienen en el entrenamiento: datos de validación, límite de error, tiempo de entrenamiento y el número de ciclos de aprendizaje.

Una vez iniciado el aprendizaje una ventana nos muestra una curva de error del número de pasos de aprendizaje realizado por *NeuroSystems*. Si el error actual cae por debajo de los puntos rojos (0.1%), el proceso de aprendizaje es detenido ya que ha alcanzado el límite de error y el aprendizaje termina con éxito.



(a)

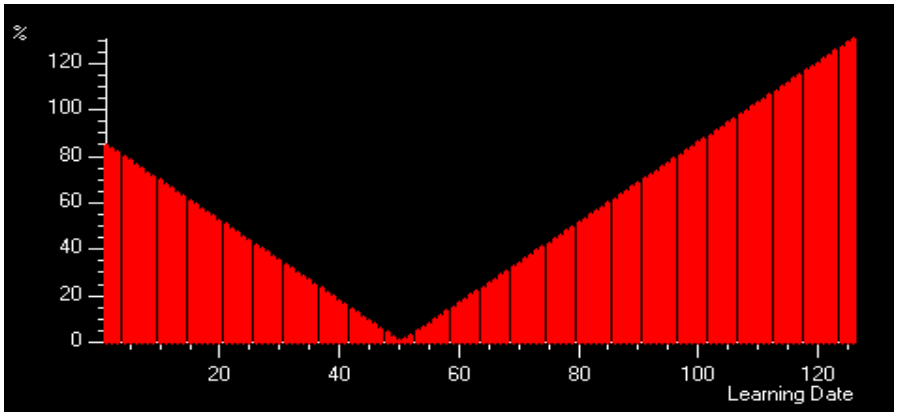


(b)

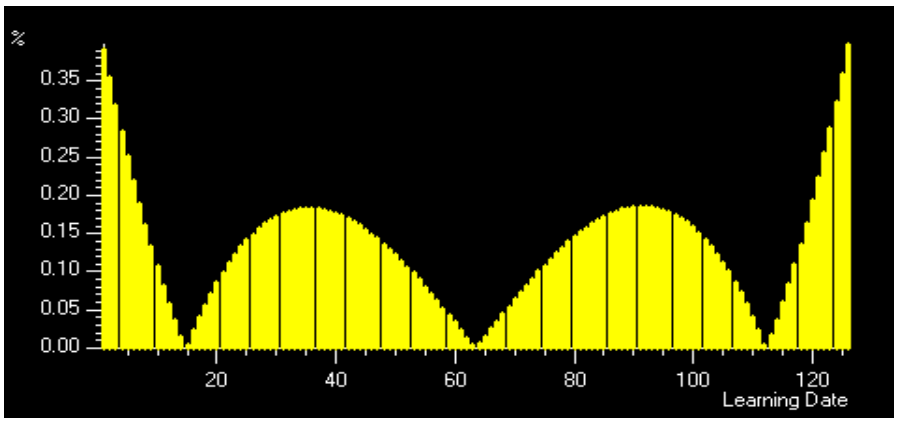
Figura 4.7. Evolución de los errores de entrenamiento y validación, para redes de (a) una capa, (b) dos capas.

En la Figura 4.7 se muestra la evolución de los errores cuadráticos para el conjunto de entrenamiento y validación a lo largo de 100 ciclos de aprendizaje. Se observa que dichos errores van decreciendo y llega un momento en el que se estabilizan, alrededor

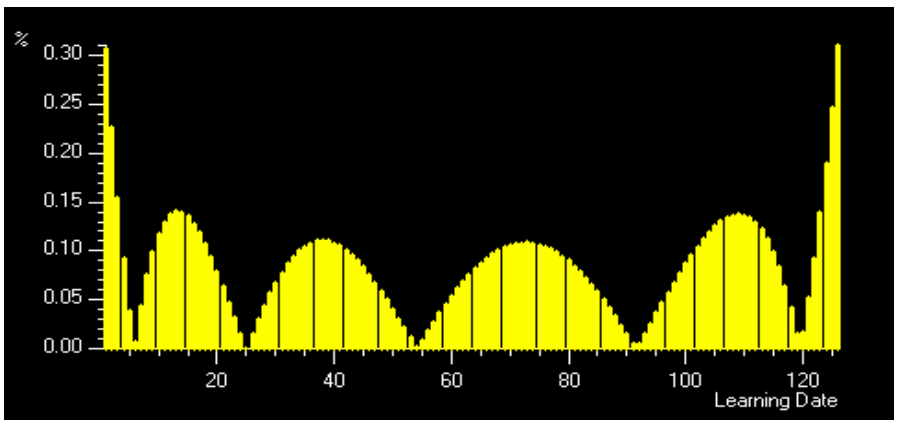
de los 20 ciclos de aprendizaje para la figura (a), y de los 60 ciclos de aprendizaje para la figura (b).



(a)



(b)



(c)

Figura 4.8. Diagrama de error de las redes de 20 neuronas ocultas con, (b) una capa oculta, (c) dos capas ocultas, luego de los 3000 ciclos de aprendizaje, y (a) antes de iniciar el entrenamiento.

A este punto, surgen una serie de interrogantes acerca de la optimización de la red y del proceso de aprendizaje, sobre cómo afecta el número de capas y de neuronas ocultas de la red a la resolución de este problema, es decir si puede conseguirse el mismo nivel de aproximación disminuyendo el número de neuronas ocultas, o, si puede mejorarse el nivel de aproximación incorporando un mayor número de neuronas ocultas en la red.

Para responder a estas interrogantes, realizamos nuevos procesos de aprendizaje de la red, utilizando 2, 10, 20 y 50 neuronas con 1 y 2 capas ocultas. En la Figura 4.8 se muestra un diagrama de error, al inicio y luego de los 10000 ciclos de aprendizaje, de los errores de entrenamiento para las redes de 20 neuronas ocultas con una y dos capas.

Se observa como al final del entrenamiento, la red ha aproximado las relaciones entre la salida y la entrada al porcentaje de error establecido. En ambos casos, la red *MLP* encuentra dificultades para aproximar correctamente los extremos de la función.

En la Tabla 4.1 se muestra un resumen de los errores sobre el conjunto de entrenamiento para las simulaciones realizadas, variando el número de neuronas y capas ocultas por el número de pasos de aprendizaje.

Capas	**Neuronas**	**% de error por ciclos de Aprendizaje**		
		1000	**2000**	**3000**
1	2	1.00	0.74	0.62
	10	0.43	0.29	0.20
	20	0.28	0.10*	-
	50	0.22	0.20	0.10
2	2	0.83	0.65	0.57
	10	0.21	0.16	0.11
	20	0.12	0.10**	-
	50	0.16	0.10***	-

* La red estabiliza el error en 1300 ciclos de aprendizaje

** La red estabiliza el error en 1650 ciclos de aprendizaje

*** La red estabiliza el error en 1900 ciclos de aprendizaje

Tabla 4.1. Errores cuadráticos obtenidos de las diferentes simulaciones

Se observa que, para el caso de una o dos capas ocultas, 2 y 10 neuronas no son suficientes para aproximar la función, mientras que utilizar 20 o 50 no afecta en gran medida los resultados, salvo que con 50 neuronas ocultas, la red necesita más ciclos de aprendizaje para conseguir la estabilización del error, lo cual es debido a que el número de parámetros de la red ha aumentado. Ante esta situación, escogemos la red de dos capas con 20 neuronas, como la más adecuada para aproximar nuestra función.

Debido a que no existe un método automático general para determinar el número óptimo de neuronas y capas ocultas más adecuadas, una posible metodología a seguir es la que se ha utilizado en esta sección: partir de ciertos valores y variarlos para analizar el comportamiento de los errores de entrenamiento y validación con vistas a obtener la red más adecuada. [10]

4.2.2 Programación de la interfaz gráfica en *WinCC*

Para nuestro proyecto en particular se realizará el control de temperatura de un módulo de temperatura PCT - 2. Se enviará una señal de control desde *WinCC* hacia el módulo a través del PLC, y se detectará mediante un sensor la temperatura del módulo, la misma que se guardará en el PLC. Los canales de comunicación transferirán el valor de temperatura a *WinCC* vía una conexión ethernet.

4.2.2.1 Creación de tags

Un *tag* o variable es el elemento de enlace entre la base de datos del *WinCC*, las variables del PLC y los objetos de nuestra aplicación. Se dispone de dos tipos de *tags*:

- **Tags internos**: son asignaciones de memoria dentro de *WinCC* que cumplen la misma funcionalidad que un PLC real.
- **Tags de proceso:** son asignaciones de memoria dentro del PLC conectado a nuestro proceso (o de un dispositivo similar).

En la Tabla 4.2 presentamos un resumen con los tags internos y tags de procesos creados para nuestro proyecto.

	NOMBRE	LONGITUD	DIRECCIÓN	
			Área de datos	Direccionamiento
INTERNOS	NeuroInput	32 bits IEEE 754	-	-
PROCESO	NeuroOutput	32 bits IEEE 754	Salida	AD 0
	TempReal	32 bits IEEE 754	Marca	MD 0

Tabla 4.2. *Tags* internos y de procesos utilizados en el proyecto *CtrlTemp*

La variable "*TempReal*" deberá almacenar la temperatura que medimos del módulo, para ello utilizamos una marca de memoria, en este caso una Palabra Doble de Marcas MD con dirección 0.

La variable "*NeuroOutput*" es la variable de salida del controlador (red de neuronas), que enviamos al módulo para controlar la temperatura, en este caso la direccionamos como una Palabra Doble de Salida AD con dirección 0.

4.2.2.2 Edición de imágenes de proceso

El editor *Graphics designer* se encarga de la confección de las pantallas del *WinCC*. Básicamente es un entorno de dibujo con la característica de que los objetos poseen la capacidad de asociar sus propiedades a variables de comunicaciones que son proporcionadas por el *Tag Management*. Creamos tres imágenes y las nombramos:

- Inicio.pdl,
- Graficas.pdl, y
- Neuro.pdl

4.2.2.2.1 Imagen Inicio.pdl

Inicio es la ventana principal del proyecto. En esta creamos una imagen que contenga una caldera que represente el módulo de temperatura. Todos los objetos gráficos necesarios para nuestra imagen de proceso se encuentran en la librería de *WinCC* y en la gama de objetos. Además, necesitamos campos de entrada y salida, barras, texto estático y el objeto *ActiveX* de *NeuroSystems*. Una vez dibujada deberá tener el aspecto de la figura 4.9.

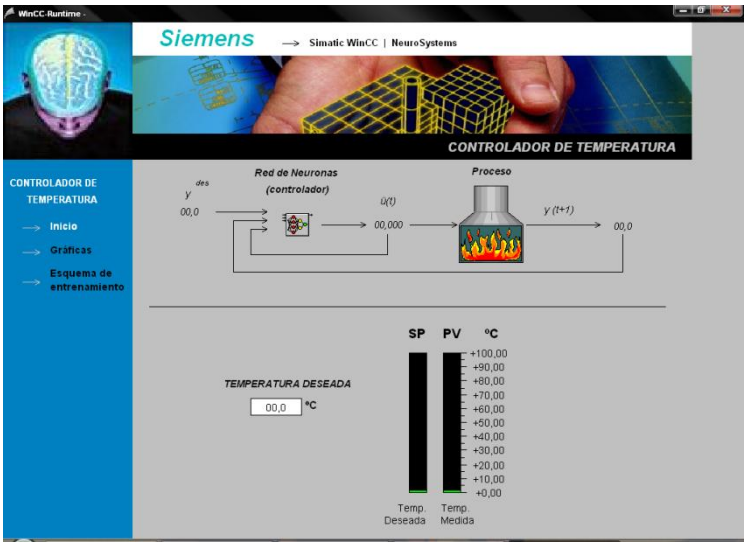


Figura 4.9. Formato final de la imagen *Inicio.pdl*

Dinamizar un campo de entrada/salida

Un campo de entrada/salida sirve para visualizar el valor de una variable y también para indicar sus cambios del valor. Procedemos la dinamización de estos campos, con las características detalladas en la tabla a continuación.

Campo E/S	Variable	Tipo de campo	Límite inferior	Límite Superior
SetPoint	NeuroInput	Entrada	20	70
NeuroInput	NeuroInput	Salida	-	-
NeuroOutput	NeuroOutput	Salida	-	-
TempReal	TempReal	Salida	-	-

Tabla 4.3. Características de los campos de E/S disponibles en *Inicio.pdl*

Dinamizar barra

La barra sirve para mostrar valores en forma de sinopsis puramente gráfica o como representación combinada de valores en una mezcla de gráfico y escala numérica disponible arbitrariamente.

En el cuadro de dialogo "*Configuración de barra*" establecemos los parámetros como se muestra en la fiugra 4.10.

Figura 4.10. Cuadro de dialogo *Configuración de barra*

La barra "*BarTempMedida*" se configura de igual manera a diferencia que la variable asignada será "*TempReal*".

Dinamizar el control *NeuroSystems*

Asignamos la ruta del archivo "*CtrlTemp.snl*" de *NeuroSystems* al control *ActiveX*. Seleccionamos "1" en *TriggerSetup* para que el control calcule la salida cada vez que un valor de la entrada del control cambia.

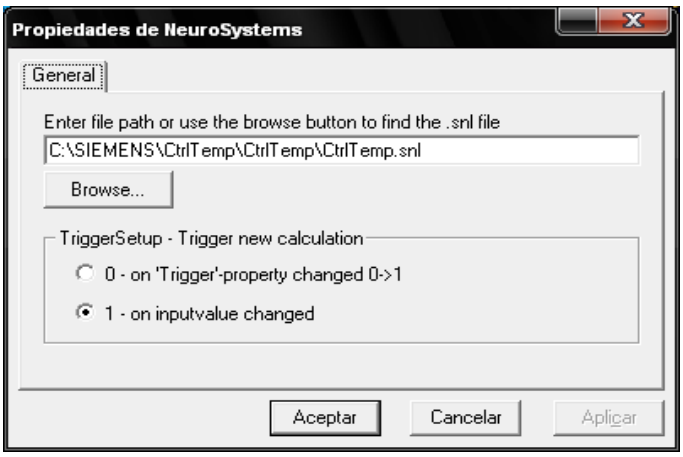


Figura 4.11. Ventana Propiedades de NeuroSystems.

La siguiente tabla muestra un resumen de las variables asignadas a las entradas del control *NeuroSystems.*

Atributo	Dinámica	Actualización
NeuroIn1	NeuroInput	Al cambiar
NeuroIn2	TempReal	Al cambiar
NeuroIn3	NeuroOutput	Al cambiar

Tabla 4.4. Características de las entradas del control *NeuroSystems* utilizados en el proyecto *CtrlTemp*

En la Figura 4.12 se muestra la configuración final de las entradas.

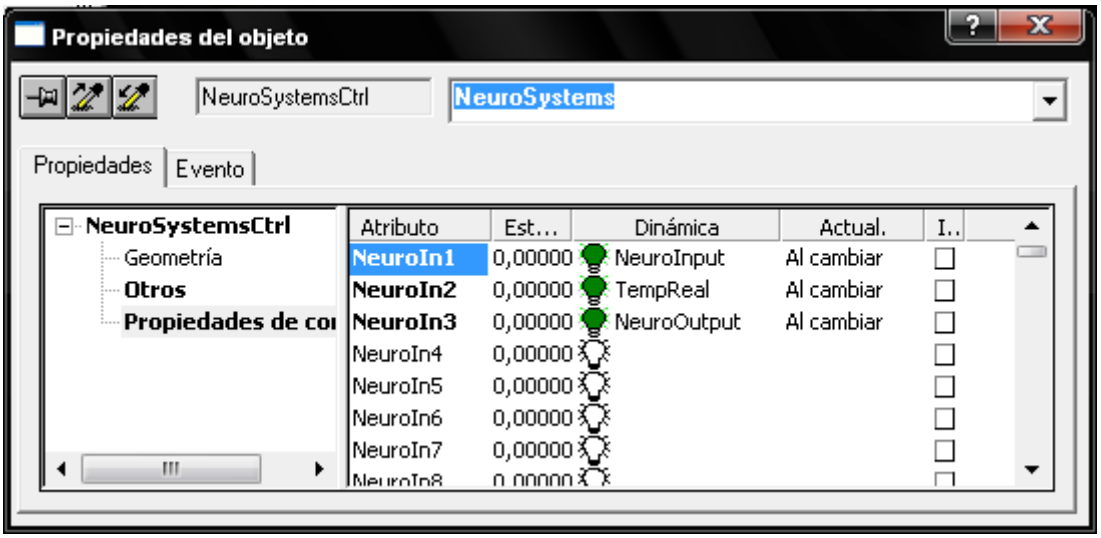


Figura 4.12. Ventana Propiedades del objeto – *NeuroSystems*. Configuración de las entradas

La salida de la red se asigna mediante eventos, es decir que la variable asignada tomara un valor cada vez que el controlador genere una nueva respuesta a las variables de entrada. En la ficha *Eventos-NeuroOut1Changed-Conexión directa* asignamos la variable *"NeuroOutput"* a la propiedad *NeuroOut1*, tal como se muestra en la siguiente figura.

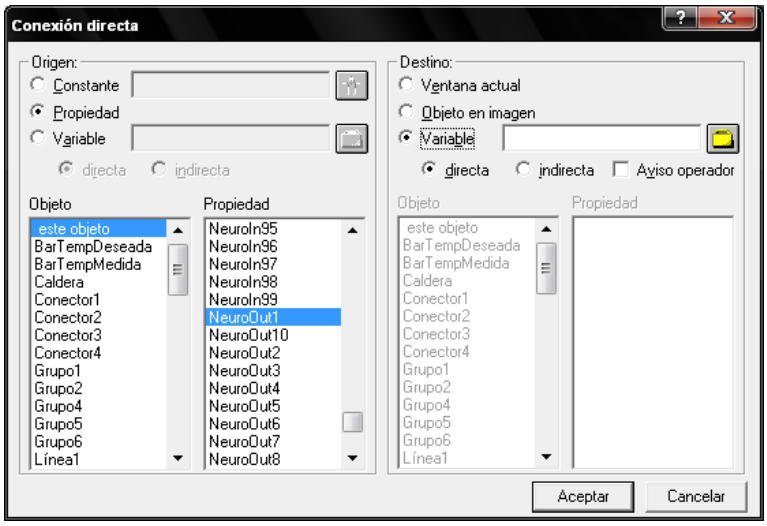


Figura 4.13. En la ventana *Conexión Directa*, se asigna una acción a un objeto seleccionado

El bloque de control *NeuroSystems* requiere que el atributo *TriggerSetup* siempre esté en 1. Para ello creamos una variable binaria interna adicional llamada "*trigger*" y la asignamos al *TriggerSetup*.

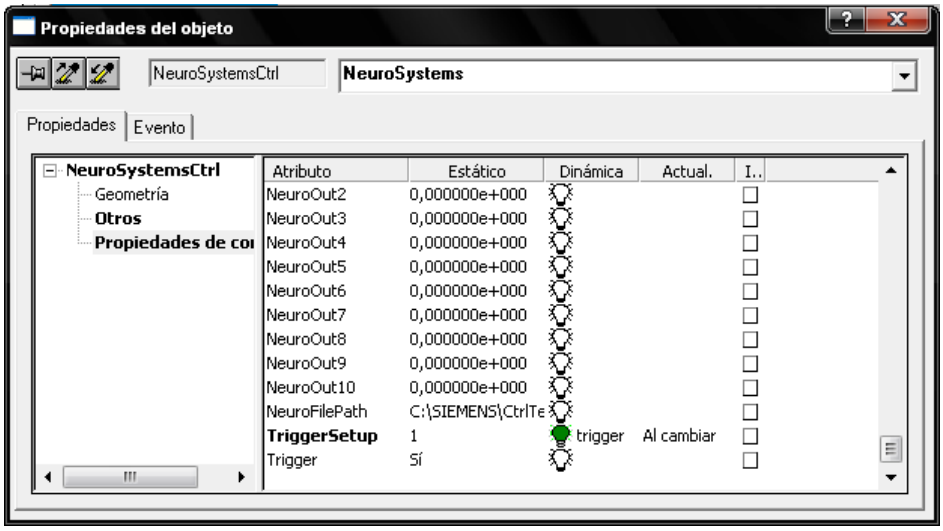


Figura 4.14. Ventana Propiedades del objeto – *NeuroSystems*. Configuración del *TriggerSetup*

En el atributo "*Visualización"* asignamos una "*Acción VBS"* como se indica en la figura 4.15.

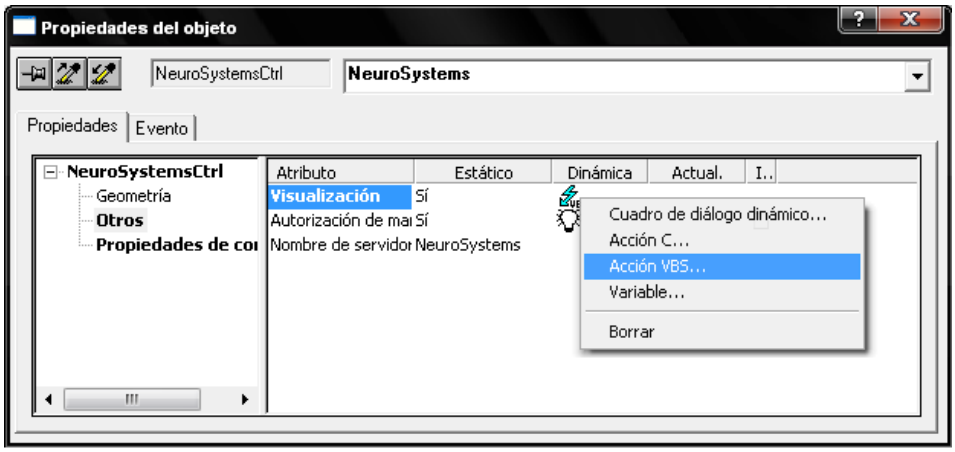


Figura 4.15. Ventana Propiedades del objeto – *NeuroSystems*. Configuración del *Trigger*

Y escribimos el siguiente código:

```
Function Visible_Trigger(ByVal Item)
HMIRuntime.Tags("trigger").Write 1
End Function
```

Con esto logramos asignar 1 a la variable "*trigger"* cada vez que arranque el programa.

4.2.2.2.2 Imagen Gráficas.pdl

Graficas contiene las curvas de tendencia y las tablas donde se visualizan valores de variables actuales o archivados. Las variables que se van a representar son las de temperatura deseada (*NeuroInput*) y temperatura real (*TempReal*). De la paleta de objetos insertamos los controles "*WinCC Online Trend Control*" y "*WinCC Online Table Control*", la imagen *Graficas* deberá como se muestra en la figura 4.16.

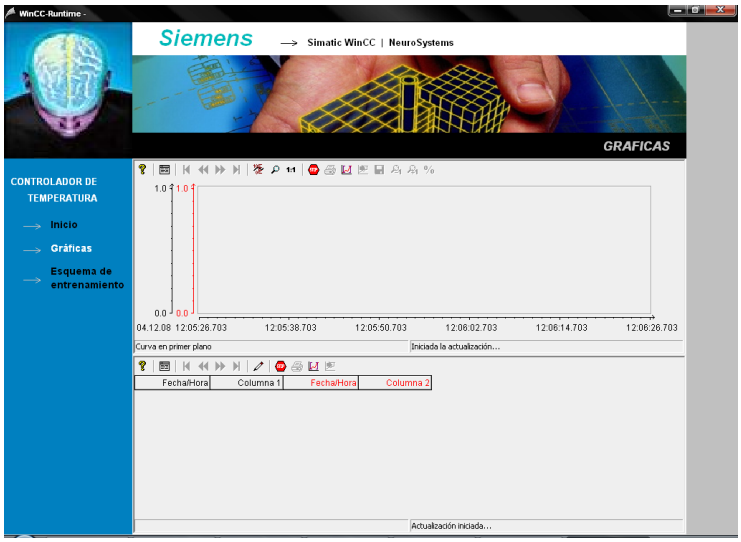


Figura 4.16. Formato final de la imagen *Graficas.pdl*

Las variables para estas tablas y curvas de tendencia, deben proceder de un archivo de valores de proceso (*ficheros*), deben tener el mismo ciclo de actualización y deben ser registradas de manera continua por ciclos.

Creación de un fichero

Tag Logging es una herramienta de configuración de los ficheros, los valores de proceso a archivar y los tiempos de los ciclos de registro y archivo. El asistente de ficheros ofrece un método automático y sencillo para crear un archivo.

Para este proyecto creamos dos ficheros de valores de proceso, y asignamos las variables como se muestra en la siguiente tabla:

Nombre	Variable de proceso	Variable de fichero
Fichero_SetPoint	NeuroInput	Fich_NeuroInput
Fichero_TempReal	TempReal	Fich_TempReal

Tabla 4.5. Características y variables de los ficheros utilizados en el proyecto *CtrlTemp*

Para las características de ciclo, en la pestaña "*Archivar"* de la ventada "*Propiedades de las variables de proceso*" establecemos las propiedades mostradas en la Figura 4.17.

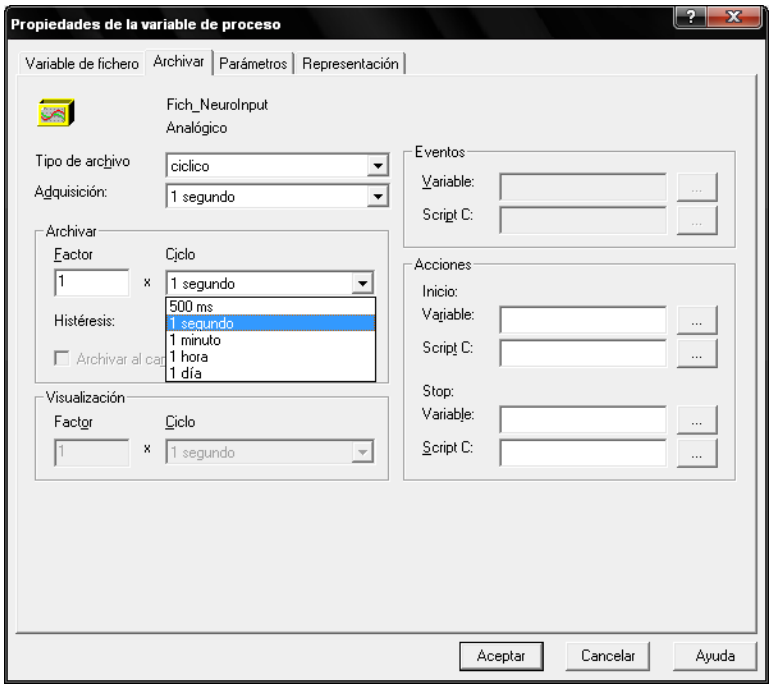


Figura 4.17. Proyecto *CtrlTemp* de WinCC. Selección de parámetros

En el control *WinCC Online Trend Control* y *Table Control* agregamos dos curvas y dos columnas, respectivamente, y les asignamos las variables de acuerdo a la siguiente tabla:

Curva/Columna	Fichero/Variable
Temp. Deseada	Fichero_SetPoint
Temp. Medida	Fichero_TempReal

Tabla 4.6. Configuración de los controles *WinCC Online Trend Control* y *Table Control* utilizados en el proyecto *CtrlTemp*

En ambos casos editamos el color de las curvas y los valores de las tablas, a fin de poder identificarlas posteriormente.

4.2.2.2.3 Imagen Neuro.pdl

Neuro es solo una imagen que muestra el esquema general de aprendizaje. La red de neuronas se entrena para que aprenda la dinámica inversa del proceso. Se observa que la

entrada al proceso actúa como salida deseada para la red, mientras que la salida del proceso es la entrada a la red.

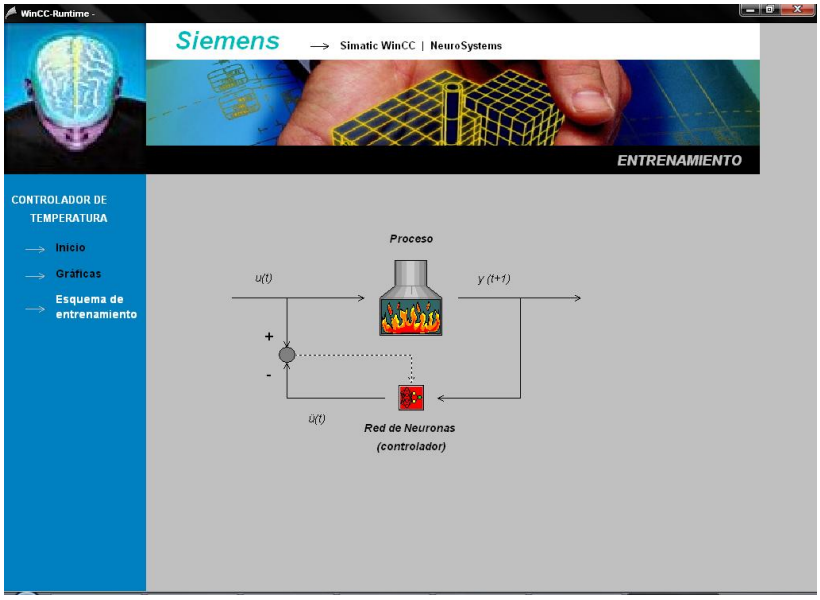


Figura 4.18. Formato final de la imagen *Neuro*

4.2.3 Código del programa STEP7

Módulos de organización (OB)

Los módulos de organización OB constituyen la forma de comunicación entre el sistema operativo de la CPU y el programa de usuario. Existen 3 tipos de OB, los cuales están accesibles o no según el tipo de CPU:

- OB 1 (ciclo libre): es el módulo principal, el que se ejecuta cíclicamente y del que parten todos los saltos a otros módulos.
- OB de error y alarma: son los que contienen la secuencia de acciones a realizar en caso de que se produzca una alarma o error programado.
- OB de arranque: en este módulo podemos introducir valores por defecto que permiten el arranque definido a la instalación, bien en un arranque inicial o tras un fallo en la alimentación. [21]

Tipos de datos

Los operandos de las instrucciones se componen de un dato que puede ser de distintos tipos. Los tipos de datos posibles son:

E entrada
A salida
M marca
P periferia (acceso directo)
L datos locales
T temporizador
Z contador
DB módulo de datos

Entradas y salidas

- Entradas analógicas: Se leen directamente de la periferia PEW.
- Salidas analógicas: Se escriben directamente en la periferia PAW.

Direccionamiento de las señales analógicas

Si el primer módulo analógico está enchufado en el slot 4, tiene la dirección inicial prefijada 256. La dirección inicial de cada módulo analógico añadido se incrementa en 16 por cada *slot*. Un módulo de entrada/salida analógica tiene las mismas direcciones iniciales para los canales de entrada y salida analógicos. [21]

Usando el editor de programa *KOP/AWL/FUP*, editamos el bloque OB1 con el siguiente código, que permite el ingreso y escalamiento de un valor analógico que viene del sensor en un rango entre 0 y 13824 o 0 a 5V.

La primera parte corresponde al ingreso de la temperatura actual del módulo en el área de memoria MD 0 correspondiente a la variable "*TempReal"*. Se toma el valor analógico que viene del sensor y se lo escala a un rango de 20 a 70

```
L    PEW  272
ITD
DTR
L    1.382400e+004
/R
L    5.000000e+001
*R
L    2.000000e+001
+R
T    MD    0
```

La segunda parte corresponde al envió de la señal de control hacia el módulo, a través de la salida AD 0 correspondiente a la variable "*NeuroOutput*". Se toma el valor de entre 0 y 5 que viene de *WinCC* y se lo escala a un valor analógico entre 0 y 13824

```
L    AD    0
L    5.000000e+000
/R
L    1.382400e+004
*R
RND
T    PAW  272
```

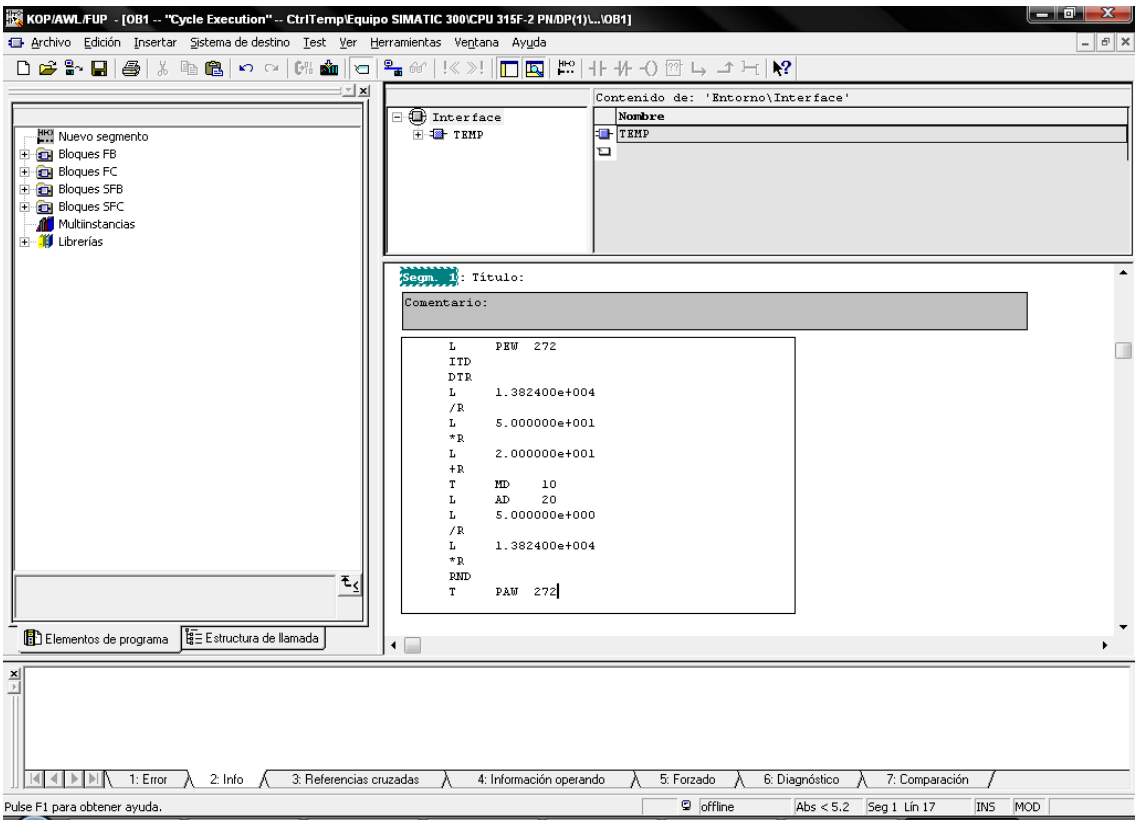


Figura 4.19. En *KOP/AWL/FUP* se realiza del código del programa en el lenguaje de programación seleccionado

4.3 SIMULACIÓN

Para realizar la simulación del funcionamiento de la red, *NeuroSystems* dispone de la herramienta "*Curve Plotter*" la cual permite ilustrar gráficamente un valor de entrada o salida arbitrario de la red.

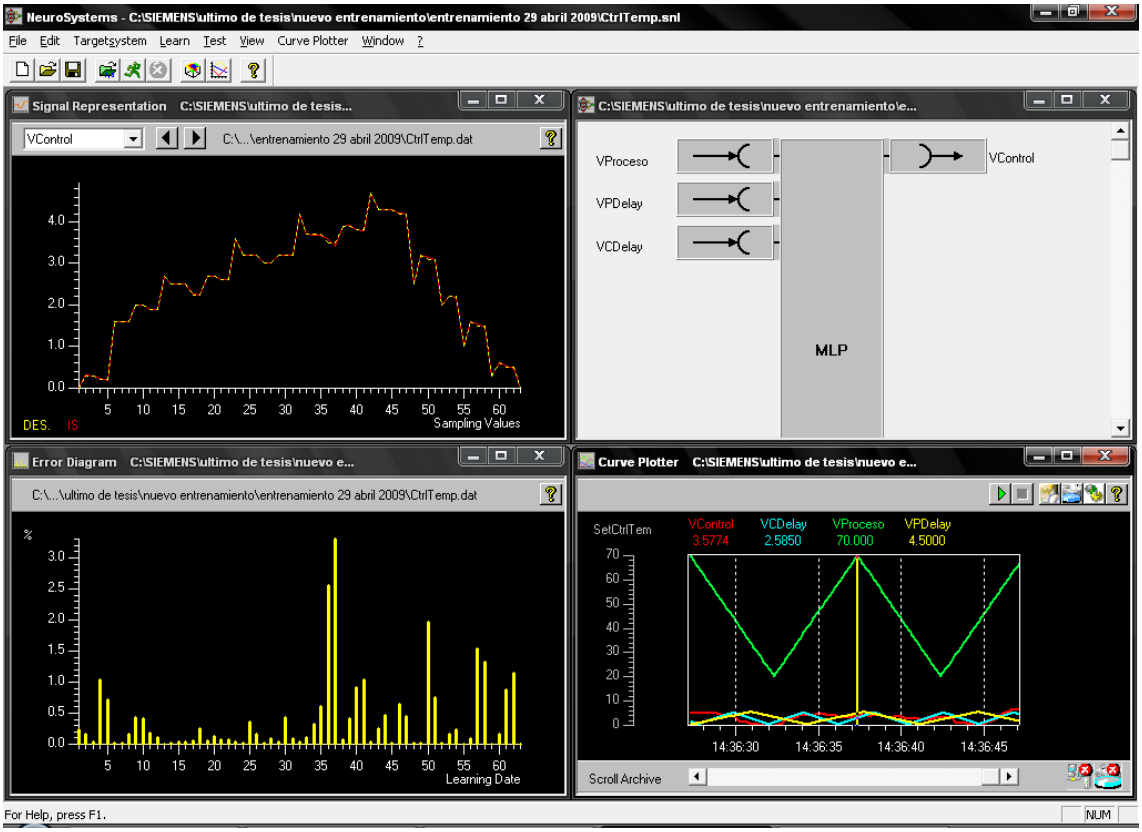


Figura 4.20. Visualización simultanea de las ventanas *Curve Plotter, Signal Representation, Error Diagram* y el bloque de red

El *Curve Plotter* nos muestra una respuesta simulada para la totalidad de posibles combinaciones de entrada para comprobar que no exista o se cumpla el error establecido en el aprendizaje de los valores de entrenamiento y para observar su capacidad de generalización en los casos restantes.

Verificamos el estado de la salida de la red como respuesta a las entradas propuestas en la tabla 4.7. Ubicamos la línea de lectura vertical del *Curve Plotter* en dichos puntos y observamos la salida real de la red.

Temperatura	Valor Deseado	Valor Real
20	0	0.0079
30	1	0.9980
40	2	2.0023
50	3	2.9978
60	4	4.0003
70	5	4.9931

Tabla 4.7. Respuesta de la red a seis patrones de entrenamiento del archivo "Ctrltemp.dat" mostrados en la tabla

La respuesta de la red a todos los patrones de entrenamiento fue exitoso, como se puede observar en la Tabla 4.7, ya que el porcentaje de error en todos los casos está por debajo del límite establecido en el entrenamiento.

De la misma forma es posible visualizar mediante el *curve plotter* la respuesta de la red a los patrones de entrada distintos a los del entrenamiento, obteniéndose los siguientes resultados (para un conjunto de datos propuestos):

Temperatura	Valor Deseado	Valor Real
25	0.5	0.4963
35	1.5	1.5014
45	2.5	2.5004
55	3.5	3.4972
65	4.5	4.5033

Tabla 4.8. Respuesta de la red a un conjunto de patrones propuestos para verificación, distintos a los de entrenamiento

Las combinaciones que no forman parte del set de entrenamiento, al ser presentadas a la red fueron aproximadas al patrón del conjunto de entrenamiento aprendido con menor distancia euclidiana.

4.4 IMPLEMENTACIÓN

4.4.1 Dispositivos requeridos

Para la implementación del sistema se requiere:

- PLC Siemens S7-300 (CPU 315F-2 PN/DP, módulo SM334 AI4/AO2x8/8 Bits)
- Módulo de temperatura (*AIR FLOW TEMPERATURE CONTROL SYSTEM PCT-2*)
- Cable Ethernet
- PC Pentium IV o superior con tarjeta de red

El módulo de temperatura PCT-2 usa un sensor de circuito integrado IC para medir la temperatura. Este módulo simulará la planta para realizar el control de temperatura.

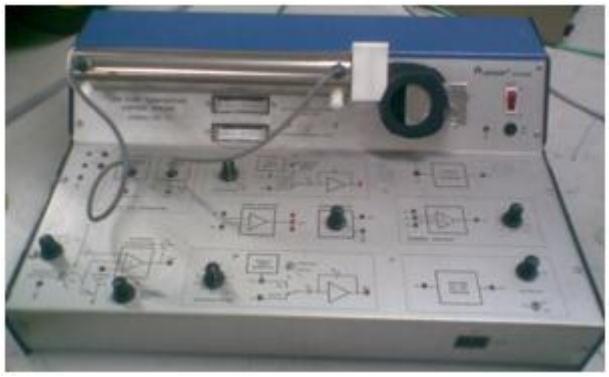


Figura 4.21. Sistema de control de la temperatura de flujo de aire PCT-2

4.4.2 Diagramas de Conexión del PLC y Módulo PCT-2

Los primero que haremos será conectar el módulo de entradas y salidas. Este deberá ser alimentado con 24V de acuerdo al esquema de conexión que viene descrito en la parte superior del mismo, como se ilustra en la Figura 4.22.

El módulo de E/S que usaremos es el SM334 AI4/AO2x8/8Bits, este un módulo analógico con 4 entradas y 2 salidas. Vamos a usar una entrada que será para tomar los datos del sensor de temperatura, la misma que se guardará en el PLC y se transferirá el valor a *WinCC* y una salida para enviar una señal de control desde *WinCC* hacia el módulo de temperatura *PCT-2* a través del PLC.

La conexión se realizará de acuerdo al esquema mostrado a continuación.

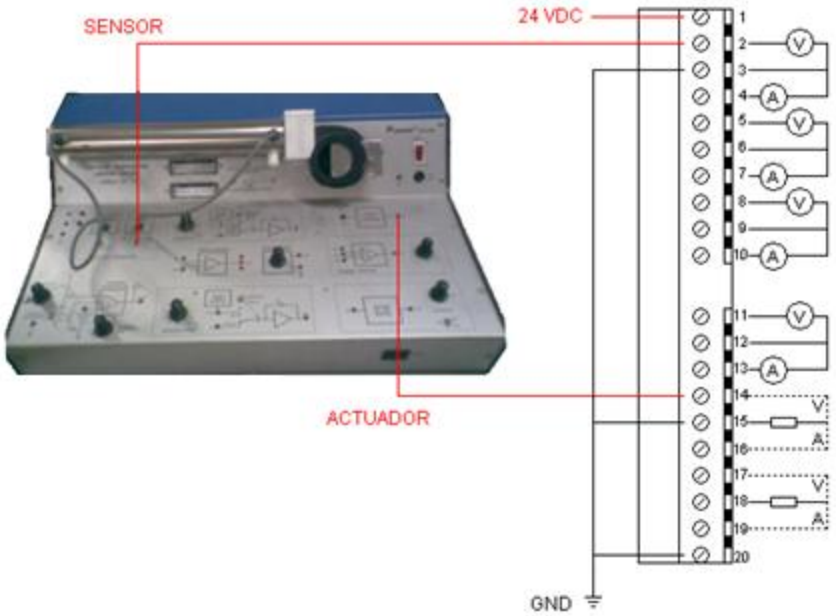


Figura 4.22. Conexión de la entrada y salida del módulo SM334 AI4/AO2x8/8Bits hacia la salida del sensor y entrada del actuador del módulo de temperatura PCT-2

De igual manera, conectamos el cable Ethernet a la CPU del PLC y a la tarjeta de red del computador.

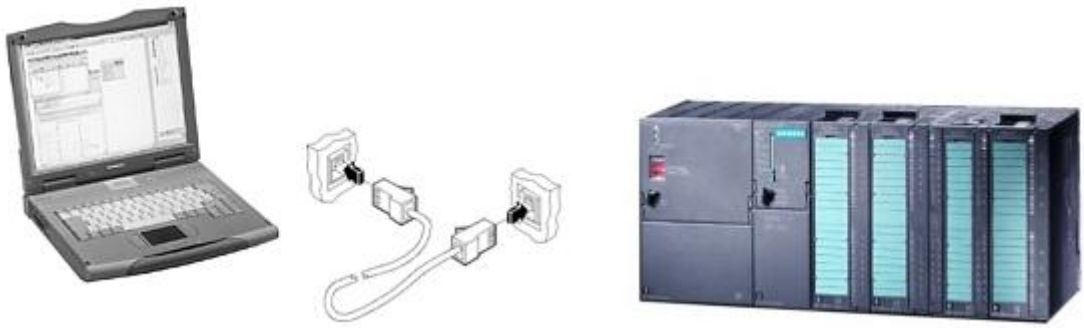


Figura 4.23. Comunicación PLC-PC vía cable Ethernet

Aplicamos tensión al PLC conectando la alimentación con el interruptor ON/OFF. Se encenderá el diodo "DC 5V" de la CPU. Así mismo, encendemos el módulo de temperatura PCT-2 con su correspondiente interruptor.

CAPÍTULO V

PRUEBAS Y RESULTADOS

En este capítulo se muestran las pruebas y resultados obtenidos para el sistema de entrenamiento en redes neuronales. Se analizará el comportamiento de este sistema controlador para el módulo de temperatura PCT-2, ante condiciones preestablecidas como: tiempos de respuesta, error en estado estable, porcentaje de precisión.

Es necesario aclarar que el objetivo principal de este capítulo no es obtener controladores óptimos como sugería la presencia de los índices antes mencionados, debido a que esto conllevaría un estudio mucho más profundo sobre el diseño, construcción y entrenamiento de los diferentes esquemas de control basados en redes neuronales.

El objetivo se centra en presentar las respuestas del sistema a una serie de pruebas básicas que tienen por objeto mostrar el comportamiento de los controladores basados en redes neuronales ante cambios en la entrada y perturbaciones en la salida del sistema.

Para el módulo de control de temperatura (módulo PCT-2) se llevaron a cabo cinco pruebas. Las tres primeras pruebas corresponden a cambios en el *Set Point*, es decir muestran el comportamiento en estado transitorio. Una cuarta prueba muestra el comportamiento en estado estable a un nivel de voltaje de entrada fijo, una quinta prueba muestra cómo responde el sistema ante una perturbación.

PRUEBA 1: Cambio en el Set Point de 40ºC a 60ºC

Esta prueba corresponde a un cambio escalón en la temperatura del módulo de 40ºC a 60ºC. A continuación se muestra la gráfica obtenida de esta prueba:

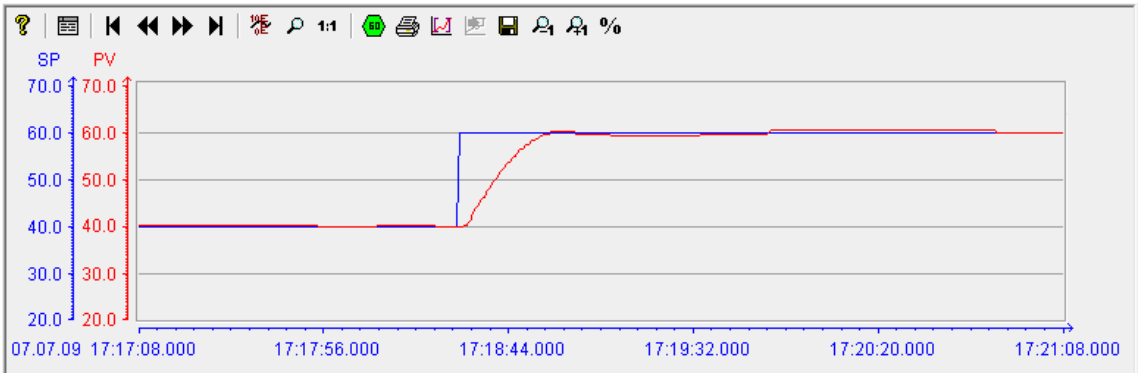


Figura 5.1. Cambio en el Set Point de 40ºC a 60ºC

De acuerdo a la gráfica, se puede decir que el controlador con redes neuronales presenta una respuesta bastante apropiada. Su tiempo de respuesta es de aproximadamente 70s, teniendo como tiempo de subida 31s y un tiempo de establecimiento de 39s desde el momento en que alcanza su máximo valor, luego de haber sobrepasado el Set Point. Presenta un rango de precisión de 0.9ºC, ya que se estabiliza en los 60.9ºC.

PRUEBA 2: Cambio en el Set Point de 60ºC a 43ºC

Esta prueba corresponde a una disminución en la temperatura del módulo de 60ºC a 43ºC. A continuación se muestra la gráfica correspondiente a esta prueba:

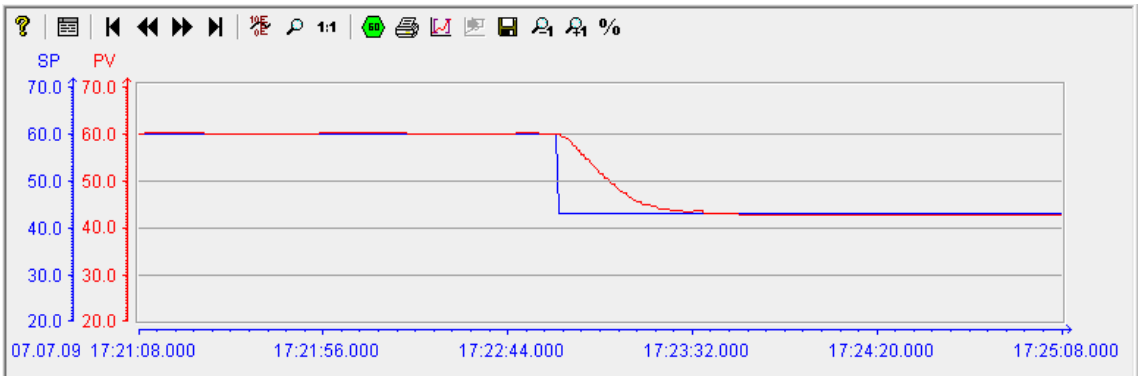


Figura 5.2. Cambio en el Set Point de 60ºC a 43ºC

De acuerdo a la gráfica, observamos que su tiempo de respuesta es de aproximadamente 50s, teniendo como tiempo de descenso 38s y un tiempo de establecimiento de 12s desde el momento en que alcanza su mínimo valor, luego de haber sobrepasado el Set Point. Presenta un rango de precisión de 0.8ºC, ya que se estabiliza en los 42.2ºC.

PRUEBA 3: Cambio en el Set Point de 35ºC a 52ºC

Esta prueba corresponde a un cambio en la temperatura del módulo de 35ºC a 52ºC. La gráfica obtenida para esta prueba se muestra a continuación:

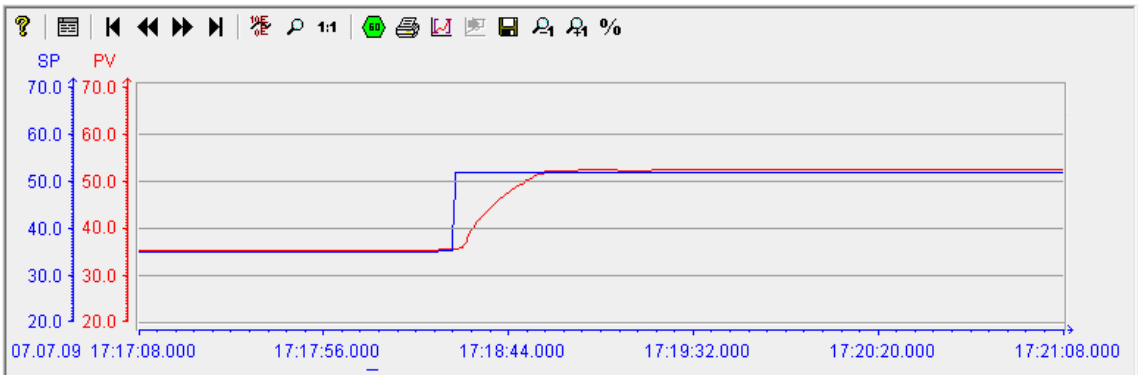


Figura 5.3. Cambio en el Set Point de 35ºC a 52ºC

De acuerdo a la gráfica, observamos que su tiempo de respuesta es de aproximadamente 55s, teniendo como tiempo de subida 35s y un tiempo de establecimiento de 20s desde el momento en que alcanza su máximo valor, luego de haber sobrepasado el Set Point. Presenta un rango de precisión de 1.1ºC, ya que se estabiliza en los 53.1ºC.

PRUEBA 4. Temperatura en estado estable para 3V (50ºC) en el Set Point

En esta prueba se mostrará de forma gráfica como varía la variable controlada (temperatura) en estado estable, para el controlador con redes neuronales. A continuación se muestra la gráfica obtenida para un Set Point de 3V equivalentes a 50ºC.

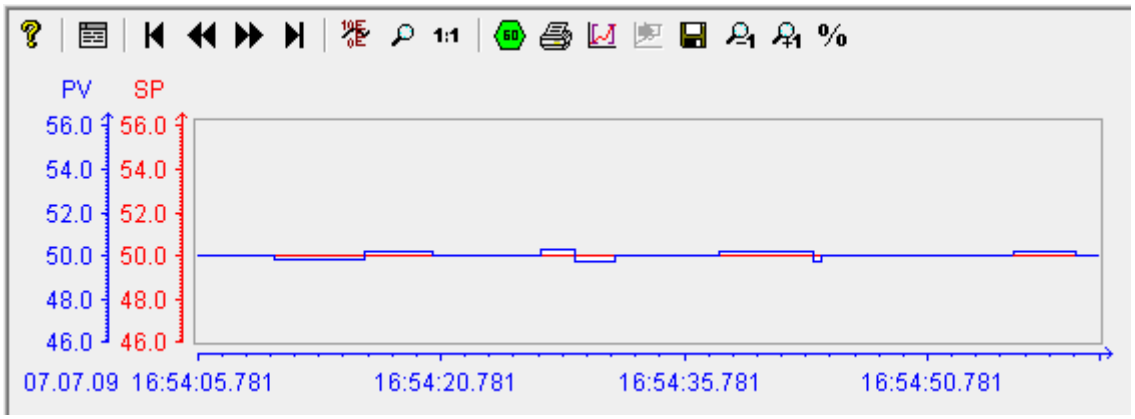


Figura 5.4. Temperatura en estado estable para 3V (50ºC) en el Set Point

Como se puede observar en la figura, la variación de la variable en estado estable para el controlador se encuentra dentro de un pequeño rango, alrededor del 0.2% del valor final, reflejando así su estabilidad en régimen permanente.

A través de este análisis del estado estable se trató de probar otra cualidad de las redes neuronales, la cual es su excelente respuesta en régimen permanente, minimizando tanto como sea posible las desviaciones con el punto de consigna (Set Point).

PRUEBA 5. Respuesta del controlador ante una perturbación en la salida

En esta prueba se mostrará de forma gráfica como responde el controlador ante una perturbación en la salida, en este caso para una temperatura de 60ºC. Esta perturbación consiste en obstruir la salida del flujo de aire caliente del módulo de temperatura por un pequeño lapso de tiempo. A continuación se muestra la gráfica obtenida para esta prueba:

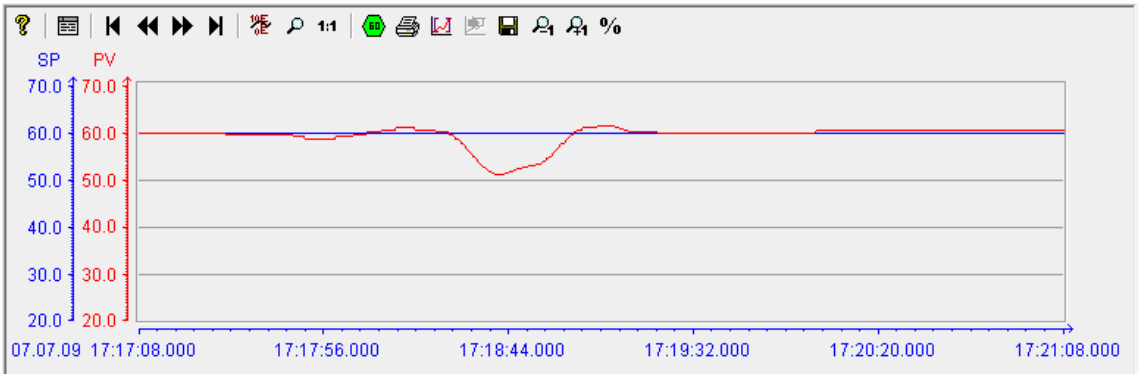


Figura 5.5. Respuesta del controlador ante una perturbación en la salida

Tal como muestra la figura, una vez aplicada la perturbación al sistema, este se desestabiliza hasta alcanzar un pico negativo máximo de aproximadamente 51ºC. Su tiempo de reestablecimiento es de aproximadamente 150s desde el momento en que el sistema se desestabiliza hasta alcanzar una temperatura estable de 61.2ºC.

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

Una vez recopilados, analizados y puestos en práctica los fundamentos sobre redes neuronales artificiales mediante la utilización del software *NeuroSystems*, *Step7 Professional* y *WinCC*, se logró diseñar e implementar un sistema de entrenamiento en redes neuronales, cumpliendo satisfactoriamente con todos los objetivos propuestos.

NeuroSystems es una herramienta de configuración que permite definir y configurar redes neuronales para aplicaciones en la resolución de problemas relacionados con el reconocimiento de formas o patrones, predicción, codificación y en nuestro caso con el control de procesos.

En cuanto al entrenamiento de la red, se observó que, para el caso de una o dos capas ocultas, 2 y 10 neuronas no son suficientes para aproximar la función, mientras que utilizar 20 o 50 no afecta en gran medida los resultados, salvo que con 50 neuronas ocultas, la red necesita más ciclos de aprendizaje para conseguir la estabilización del error, lo cual es debido a que el número de parámetros de la red ha aumentado.

El conjunto de aprendizaje debe ser significativo. Debe haber un número suficiente de ejemplos. Si el conjunto de aprendizaje es reducido, la red no será capaz de adaptar sus pesos de forma eficaz.

Utilizando un conjunto de datos de aprendizaje con aproximadamente 100 pares de datos, se ha alcanzado una red que responde adecuadamente a los parámetros establecidos para el control de temperatura. Un número mayor de datos de aprendizaje mejora en cierta medida la respuesta de la red. Mejora el porcentaje de precisión, pero

se incrementa el tiempo de respuesta. Por otro lado, un número mayor de datos de entrenamiento implica un incremento en el tiempo y ciclos de aprendizaje para estabilizar el error de la red.

Con 50 pares de datos, la red presenta problemas para alcanzar los parámetros establecidos para el control de temperatura. Si bien responde adecuadamente a los patrones de entrenamiento utilizados, sus tiempos de respuesta son mayores, y su respuesta ante valores distintos a los del entrenamiento no es eficaz.

De acuerdo a los resultados obtenidos en el Capítulo 5 concernientes a las pruebas realizadas para el módulo de temperatura PCT-2 se puede concluir que el controlador con Redes Neuronales presenta una respuesta bastante apropiada para el control de temperatura realizado, manteniéndose dentro de los parámetros establecidos para el control.

Sus tiempos de respuesta son adecuados, debido a que no presentan un sobreimpulso apreciable y dependiendo del cambio de temperatura involucrado en la prueba considerada.

Respecto al estado estable, los controladores con redes neuronales presentan una excelente respuesta en régimen permanente, minimizando tanto como sea posible las desviaciones con el punto de consigna (*Set Point*).

6.2 RECOMENDACIONES

Debido a que no existe un método automático general para determinar el número óptimo de neuronas ocultas o la razón de aprendizaje más adecuada, una posible metodología a seguir es partir de ciertos valores y variarlos para analizar el comportamiento de los errores de entrenamiento y validación con vistas a obtener la red más adecuada.

Para una selección adecuada de la estructura de la red es necesario tener en cuenta lo siguiente:

- Si el conjunto de datos de aprendizaje es pequeño, hay que seleccionar una menor cantidad de neuronas.
- Se debe comenzar con muy pocas neuronas y luego aumentar el número hasta obtener un resultado satisfactorio.
- Cuando aumenta la complejidad de una función a ser modelada por la red neuronal, el número de capas de neuronas también debe aumentar.

Se recomienda revisar en detalle la configuración de comunicación de WinCC y Step7, pues de ello depende el correcto funcionamiento del bloque de control *NeuroSystems*, así como el desarrollo de cualquier aplicación.

Revisar que las conexiones de alimentación y comunicación de todos los dispositivos se hayan realizado correctamente antes de energizar el sistema.

REFERENCIAS BIBLIOGRÁFICAS

[1] Isasi Viñuela, P., & Galván León, I. M. (2004). *Redes de Neuronas Artificiales.* Madrid: PEARSON PRENTICE HALL.

[2] SIEMENS. (2006). *Manual NeuroSystems V 5.0.* Siemens AG, I&S.

[3] Rich, E., & Knigh, K. (1994). *Inteligencia Artifical* (Segunda edición ed.). Madrid, España: McGrawHill.

[4] McCulloch, W., & Pitts, W. (1943). *A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics.*

[5] Hebb, D. O. (1949). *Orgazination of Behaviour.* New York: John Wiley & Sons.

[6] Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences.* Werbos, P.J. (). . Diss. Harvard University. Cambridge.

[7] Rumelhart, D., Hinton, G., & Willians, R. (1986). *Parallel Distributed Processing, chapter Learning representarios by back-propagating error.* MIT Press.

[8] Kohonen, T. (1982). *Self-organized formation of topologically correct feature maps. Biological Cubernetics.*

[9] Broomhead, D., & Lowe, D. (1988). *Multivariable functional interpolation and adaptive networks. Complex Systems.*

[10] Acosta Buitrago, M. I., & Zuluaga Muñoz, C. A. (2000). *Tutorial sobre redes neuronales aplicadas en ingenieria eléctrica.* Pereira: Universidad Tegnológica de Pereira.

[11] del Brío, M., & Sanz Molina, A. (2002). *Redes Neuronales y Sistemas difusos* (Segunda edición ed.). AlfaOmega.

[12] Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry.* MIT Press.

[13] Rummelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). *Learning internal representations by error propagation.* MIT Press.

[14] SIEMENS. (2006). *SIMATIC S7-300, Datos Técnicos CPU 31xC y CPU 31x.* Nürnberg: Siemens AG, I&S.

[15] SIEMENS. (2007). *SIMATIC S7-300: Datos de los módulos.* Nürnberg: Siemens AG, I&S.

[16] Ayuda del Administrador SIMATIC

[17] SIEMENS. (2006b). *SIMATIC: STEP 7 Introducción y ejercicios prácticos.* Nürnberg: Siemens AG, I&S.

[18] SIEMENS. (2006c). *SIMATIC: Configurar el hardware y la comunicación con STEP7.* Nürnberg: Siemens AG, I&S.

[19] SIEMENS. (2003). *SIMATIC HMI: WinCC V6.0 Documentación estándar.* Nürnberg: Siemens AG, I&S.

[20] *NeuroSystems*, SIEMENS, http://www.industry.siemens.de/IT4Industry/EN/solution_services/simatic_addons/neuro.htm?PIdent=1001&SIdent=1018&TIdent=1032, 20/11/2008

[21] Tutorial de programación en Simatic S7, IESPANA, http://electronicapic.iespana.es/manual/programacion_simatic_s7.pdf, 15/10/2008

[22] Pérez Rivas, L. (2005). *Implementación de Esquemas de Control Neuronal bajo Control - Logix.* Mérida, Venezuela: Universidad de Los Andes.

ANEXOS

A1 MANUAL DE USUARIO

CONTENIDO

1. DESCRIPCIÓN DEL CONTROLADOR

El controlador realizado se basa en el uso de redes neuronales. Cuando se habla de control de procesos utilizando redes de neuronas, se entiende que es una red la encargada de calcular la acción de control que hay que aplicar al proceso para que se alcance el objetivo de control deseado.

Para nuestro proyecto en particular se realizará el control de temperatura de un módulo de temperatura PCT - 2. Se enviará una señal de control desde *WinCC* hacia el módulo a través del PLC, y se detectará mediante un sensor la temperatura del módulo, la misma que se guardará en el PLC. Los canales de comunicación transferirán el valor de temperatura a *WinCC* vía una conexión ethernet.

Una interfaz gráfica realizará un monitoreo continuo del estado de la planta. Dicha interfaz permitirá el ingreso de una temperatura deseada (*set point*) en un rango de 20ºC a 70ºC. Por último se presentaran las graficas de tendencias y las tablas de valores de la temperatura deseada y la temperatura real.

Características generales

- Animación simple en *WinCC*
- Monitoreo continuo del módulo de temperatura
- Representación de los datos de proceso en forma gráfica y tablas de datos
- Control de temperatura en el rango de 20ºC a 70ºC
- Margen de error de ± 1ºC

2. CONFIGURACIÓN DE DRIVERS Y COMUNICACIONES

2.1 WinCC

2.1.1 Agregar un driver de PLC

Un driver de comunicaciones es una *dll*, con la extensión **.CHN* que posibilita al *WinCC* comunicar un determinado protocolo con un tipo determinado de PLC o

aplicación software. En este paso definimos qué dispositivo vamos a utilizar como interfaz de comunicación con la planta.

Para agregar un driver de PLC, hacemos clic mediante el botón derecho del ratón sobre "*Administración de variables*" en la subventana izquierda del explorador de *WinCC*. Hacemos clic en la función "*Agregar nuevo driver* " del menú contextual.

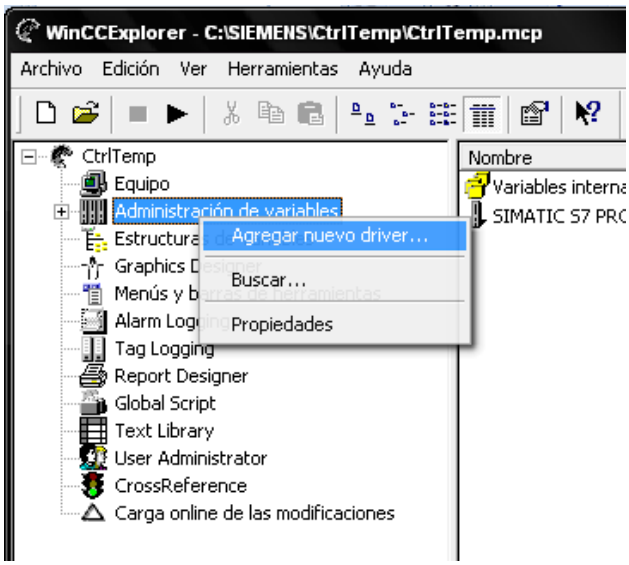


Figura 2.1. Proyecto *CtrlTemp* de *WinCC*. Agregar conexión de driver.

Seleccionamos "*SIMATIC S7 Protocol Suite*" y pulsamos el botón "Abrir".

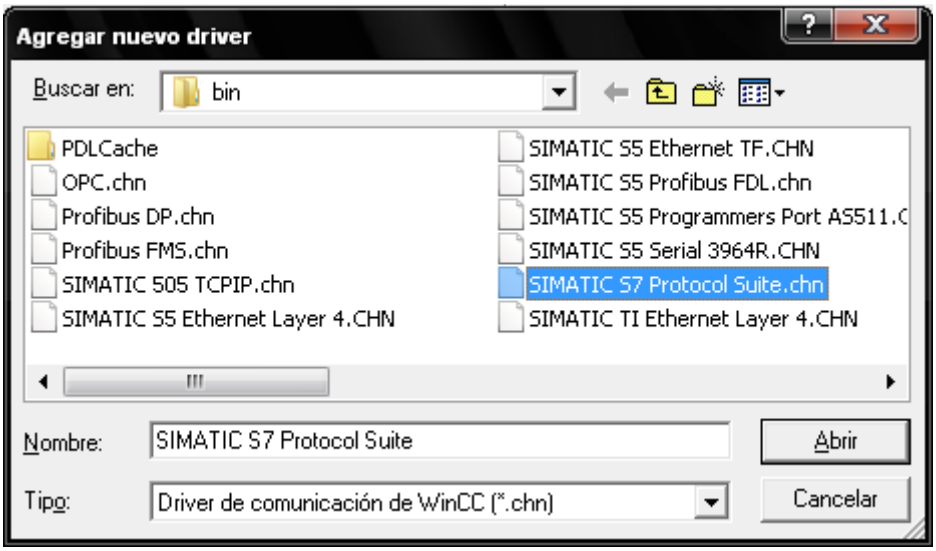


Figura 2.2. *Agregar nuevo driver* permite seleccionar un driver de un listado disponible

Ahora el driver seleccionado aparece bajo la línea "*Administrador de variables*". Seleccionamos la unidad de canal *TCP/IP* del *SIMATIC S7 Protocol Suite* y hacemos clic en ella con el botón derecho del ratón. En el menú contextual que aparece seleccionamos "*Nueva conexión*".

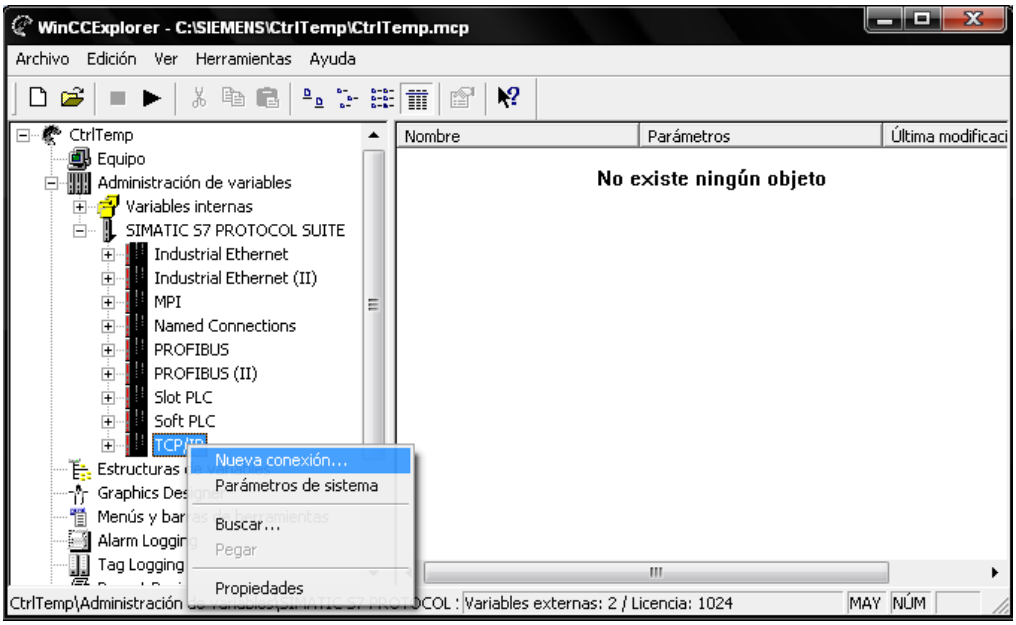


Figura 2.3. Proyecto *CtrlTemp* de *WinCC*. Añadiendo un nuevo driver.

En el cuadro de diálogo "*Propiedades del enlace*" que aparece, escribimos "*PLC1*" como nombre y pulsamos "*Propiedades*". Establecemos los parámetros como indica la figura 2.4.

Figura 2.4. Parámetros de enlace *TCP/IP*

Nuevamente seleccionamos la unidad de canal *TCP/IP* del *SIMATIC S7 Protocol Suite* y hacemos clic en ella con el botón derecho del ratón. En el menú contextual que aparece seleccionamos "*Parámetros de sistema*".

En el cuadro de dialogo que aparece ubicamos la pestaña "*Unidad*" y seleccionamos como nombre del dispositivo lógico: *TCP/IP*(*Auto*) -> *Intel*(R) *PRO/100 VE Ne...*

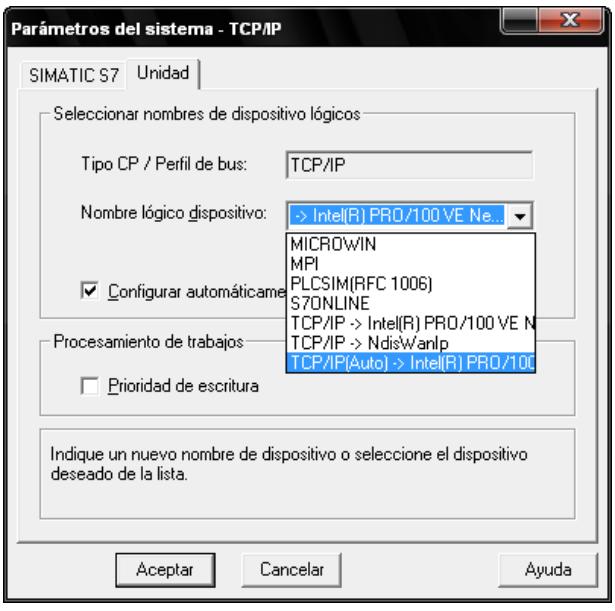


Figura 2.5. Parámetros del sistema - *TCP/IP*

2.1.2 Registrar el Control *ActiveX* con *Simatic WinCC*

Para poder usar el control *ActiveX NeuroSystems*, primero debemos registrarlo entre los controles *ActiveX* disponibles en *WinCC* y luego insertarlo en la paletas de objetos para tenerlo a disposición.

Una vez abierto el *Graphics Designer* hacemos clic en la pestaña "*Controles*" en la "*Paleta de objetos*", luego hacemos clic con el botón derecho del ratón en cualquier lugar del control y escogemos "*Agregar/Quitar*" del menú que aparece

En el cuadro de dialogo que aparece hacemos clic en "*Registrar OCX…*". Ubicamos el archivo "*NeuroSystems.ocx*" y hacemos clic en "*Abrir*". De esta manera queda registrado el control ActiveX en el sistema.

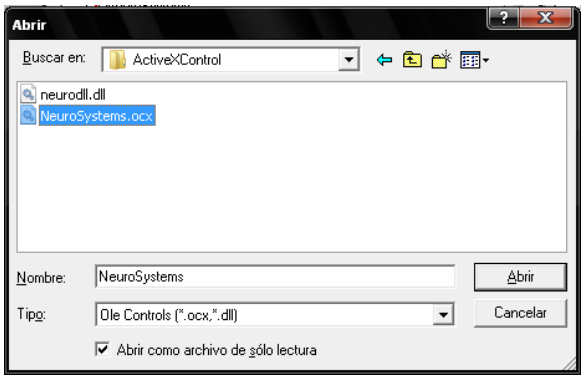


Figura 2.6. En la ventana *Abrir* escogemos *NeuroSystems.ocx* para registrar el control *ActiveX* en el sistema

2.1.3 Insertar el Control *ActiveX* en el *SIMATIC WinCC*

Siguiendo los pasos anteriores damos clic con el botón derecho del ratón en cualquier lugar de la subventana "*Control*" y seleccionamos "*Agregar/Quitar*". En el cuadro que aparece a continuación buscamos en la lista *NeuroSystems*, colocamos un visto en el cuadro de selección y presionamos "*Ok*".

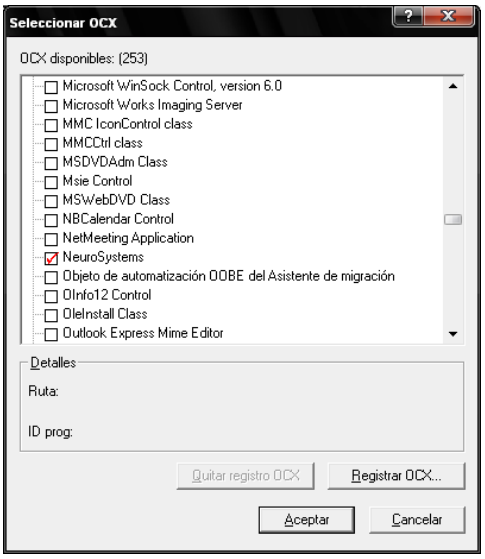


Figura 2.7. En la ventana *Seleccionar OCX* escogemos *NeuroSystems,* con ello estará disponible en la *Paleta de Objetos*.

2.2 STETP7

2.2.1 Configurar los módulos centrales

Para la creación de un proyecto, *STEP7* requiere que se describa la configuración del hardware que vamos a utilizar (CPU, fuente de alimentación, módulos de entrada y salida).

2.2.1.1 Configurar el hardware

Para configurar el hardware, el proyecto "*CtrlTemp*" tiene que estar abierto en el Administrador SIMATIC.

Abrimos la carpeta "*Equipo SIMATIC 300*" y hacemos doble clic en el icono Hardware.

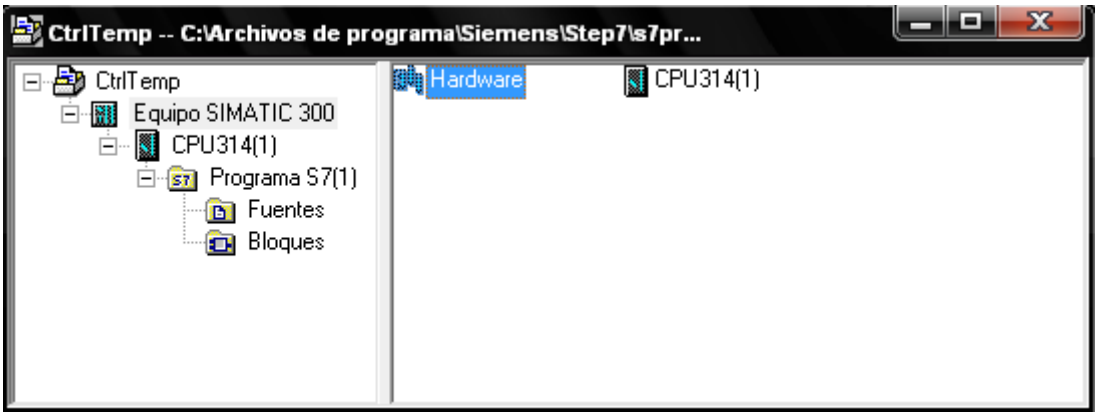


Figura 2.8. Ventana de la estructura del proyecto "*CtrlTemp*"

Entonces se abrirá la ventana "*HW Config*" y se visualizará la CPU seleccionada al crear el proyecto. En "*CtrlTemp*", se trata de la CPU 315F-2 PN/DP.

Lo primero que se necesita es una fuente de alimentación. Navegamos por el catálogo hasta la PS307 2A y la insertamos en el slot 1 mediante arrastrar y soltar.

Navegamos por los módulos de entrada (DI, Digital Input) hasta el SM323 DI16/DO16x24V/0.5A y lo insertamos en el slot 4. El slot 3 queda vacío[5].

Del mismo modo insertamos el módulo entrada y salida analógicas (AI/AO-300) SM334 AI4/AO2x8/8 bit en el slot 5.

[5] El módulo SM323 DI16/DO16x24V/0.5A no fue utilizado para nuestra aplicación, pero al estar instalado en el PLC debe ser igualmente configurado

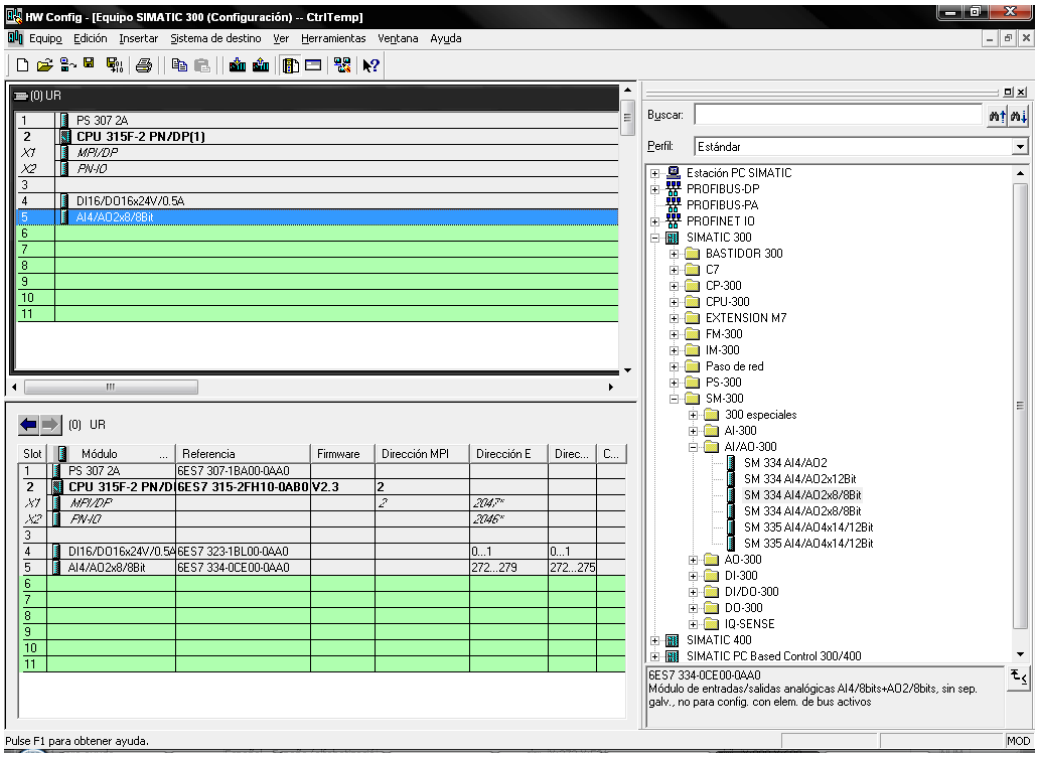


Figura 2.9. En *HW Config* se configura el hardware que utilizamos en nuestro proyecto: CPU, fuente de alimentación y los módulos de entrada y salida.

Configurar la comunicación

Hacemos doble clic sobre *PN-IO* para abrir las propiedades de este enlace. En la ventana que aparece a continuación escogemos la ficha "*General"* y pulsamos el botón "*Propiedades*".

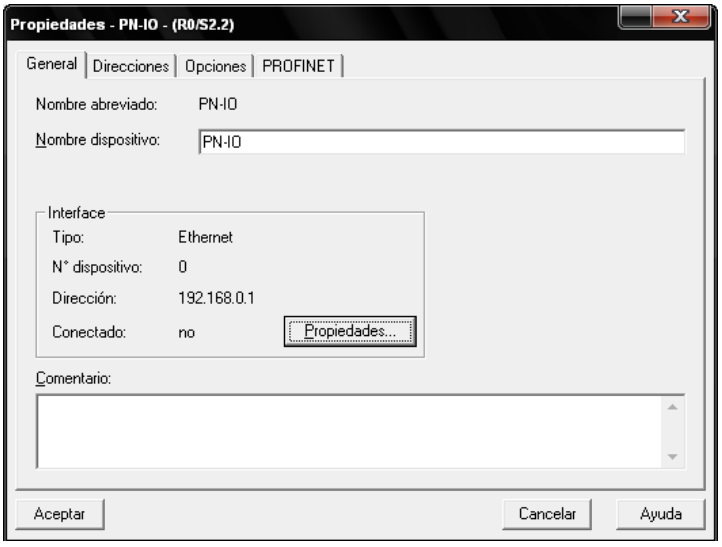


Figura 2.10. Ventana Propiedades – *PN-IO*

En la ficha "*Parámetros"* asignamos las siguientes propiedades:

- Dirección IP: 192.168.0.1
- Máscara de subred: 255.255.255.0
- Routing: Sin router,

Como se indica en la figura a continuación.

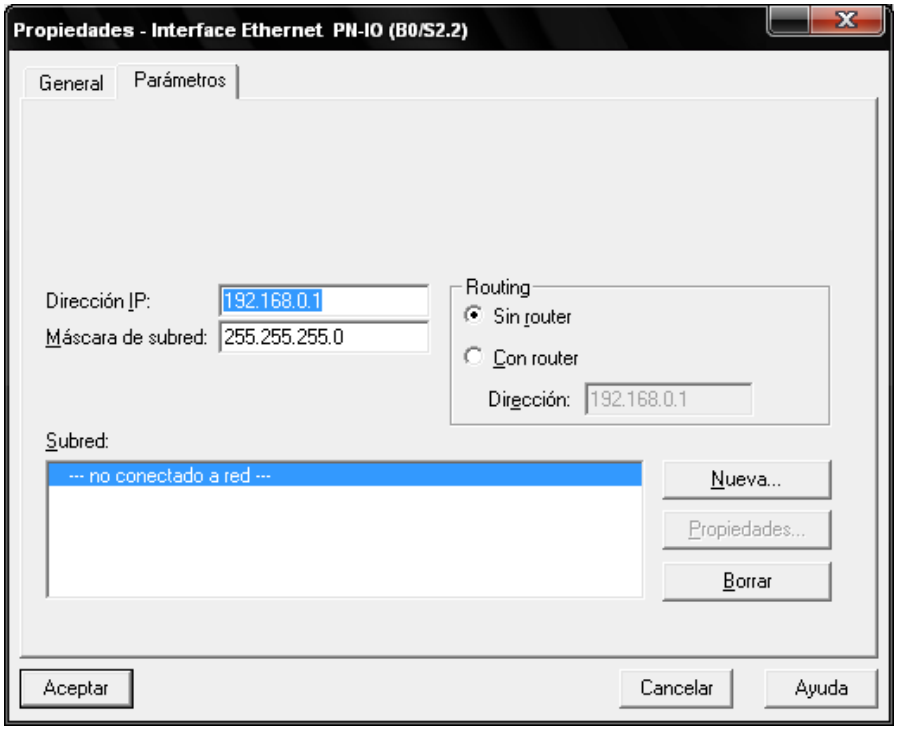


Figura 2.11. Propiedades – Interface Ethernet *PN-IO*

Pulsamos el botón "*Nueva*…" para crear una nueva subred. En la ventana "*Propiedades – Nueva subred Industrial Ethernet*" que aparece a continuación escogemos el nombre y la ID que aparecen por defecto en este caso *Ethernet(1)* y *005F-0008*. Confirmamos todos los cambios con el botón *Aceptar.*

Con el ícono *Guardar y compilar* se preparan los datos para transferirlos a la CPU. Mediante el ícono *Cargar en módulo* cargamos la configuración en la CPU. Escribimos la dirección IP 192.168.0.1, como se indica en la figura 2.12 y pulsamos *Aceptar*.

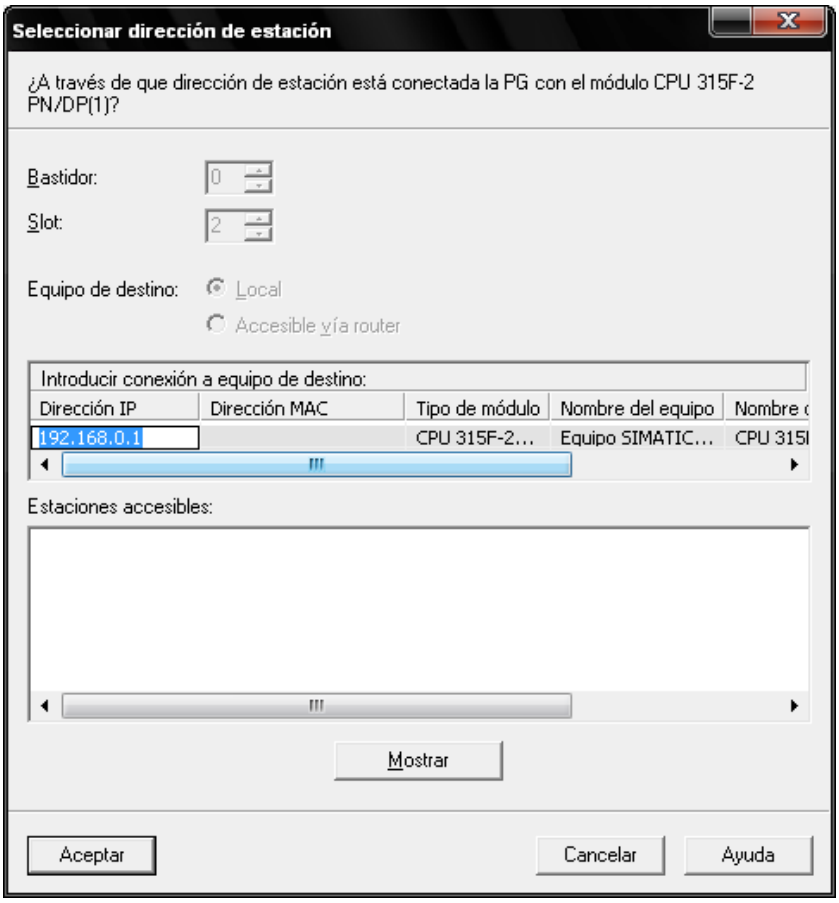


Figura 2.12. Selección de la dirección IP de la estación de destino

2.2.2 Introducir un nuevo enlace

Dado que para el presente proyecto no se dispone del cable de comunicación *MPI*, la comunicación PLC-PC se realiza con un cable *Ethernet*. Para ello debemos crear un enlace *TCP*. Abrimos la ventana de configuración de red "*NetPro"*.

Seleccionamos, en la representación gráfica de la red, el módulo para el cual deseamos crear el enlace. Hacemos clic con el botón derecho del ratón y escogemos "*Insertar nuevo enlace"*.

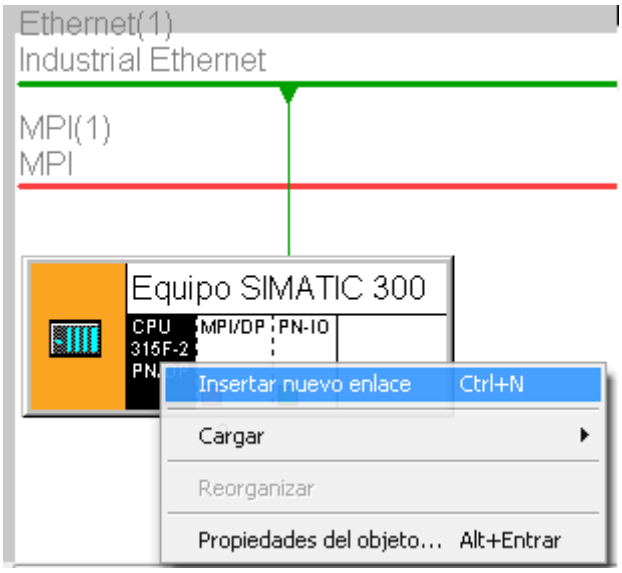


Figura 2.13. Representación de la red *NetPro*

En la ventana que se abre elegimos como interlocutor deseado "*no especificado*" y en tipo de enlace "*Enlace S7*".

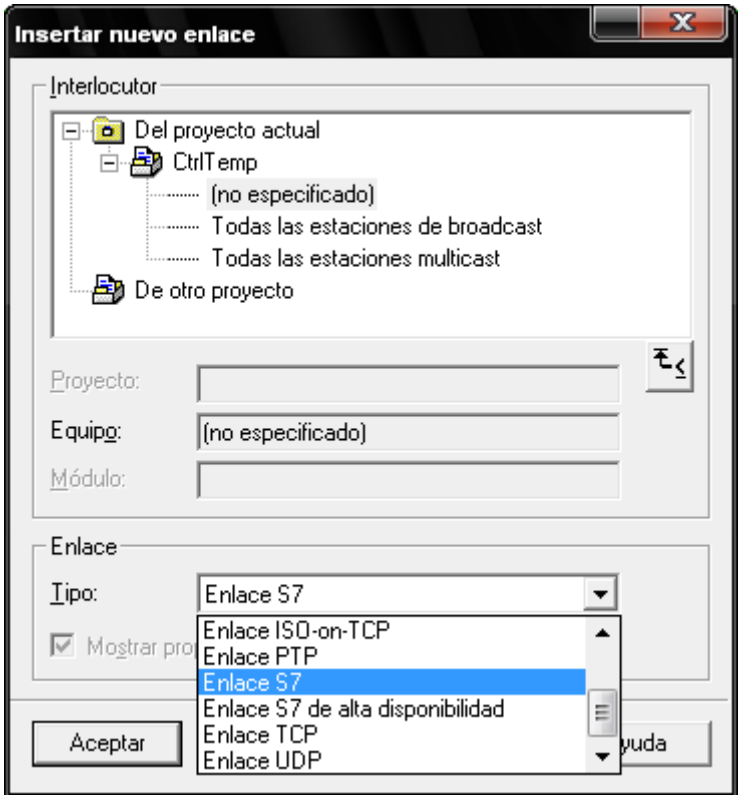


Figura 2.14. Insertar nuevo enlace

STEP 7 registra el enlace en la tabla de enlaces de la estación local (es decir, de la seleccionada) y asigna a este enlace el ID local y eventualmente el ID del interlocutor necesario para programar los bloques de función para la comunicación (valor para el parámetro de bloque "ID").

En la ventana "*Propiedades – Enlace S7"*, en la sección "*Vía de enlace*", escogemos la dirección del *Interlocutor* como 192.168.0.2, como se indica a continuación:

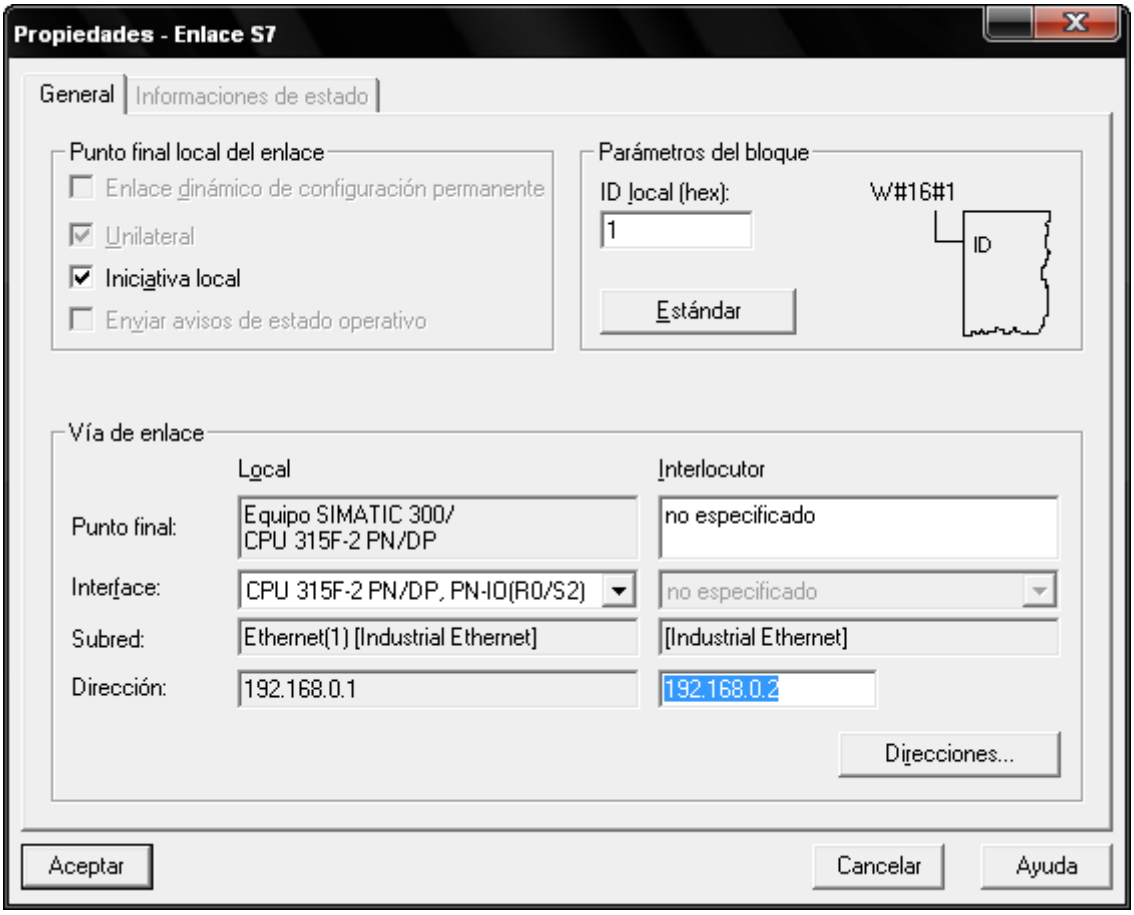


Figura 2.15. Propiedades – Enlace S7

Mediante los íconos y guardamos, compilamos y cargamos la configuración en la CPU.

Adicional a esto debemos configurar la IP de nuestro computador. En "*Propiedades de Conexión de área local*", seleccionamos las "*Propiedades de Protocolo de Internet (TCP/IP)*" y configuramos como se indica a continuación:

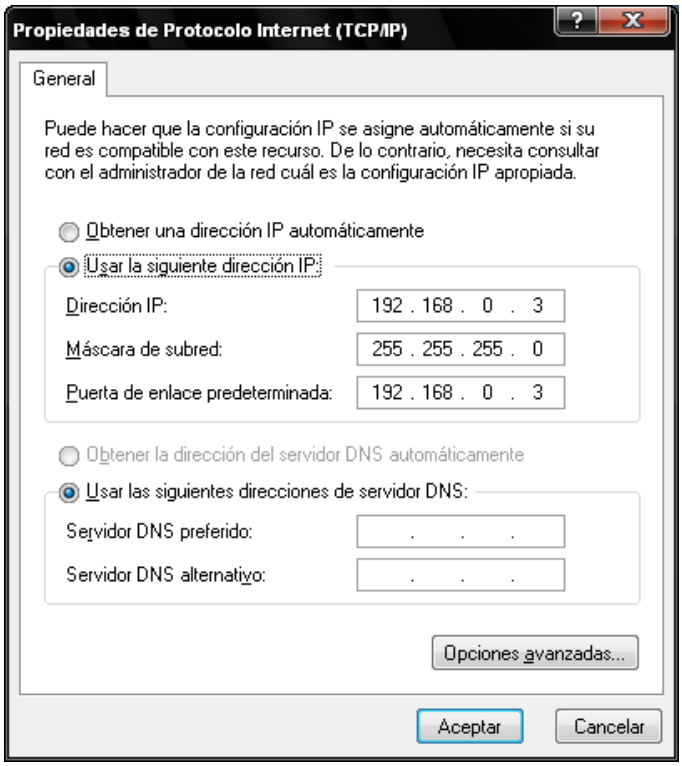


Figura 2.16. Propiedades de Protocolo Internet (TCP/IP)

Abrimos la ventana "*Ajustar interface PG/PC*" disponible en la carpeta *SIMATIC-STEP 7*, en la sección "*Parametrización utilizada*" escogemos "*TCP/IP(Auto)…"*, como se indica en la siguiente figura.

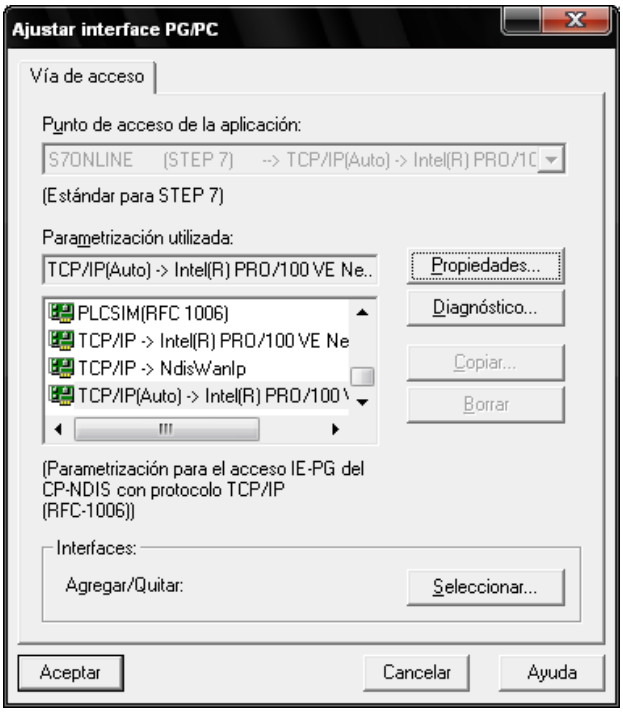


Figura 2.17. Ajustar interface *PG/PC*

3. EJECUCIÓN DEL PROYECTO

Para ejecutar nuestro proyecto, una vez acabada la configuración, abrimos el proyecto "*CtrlTemp*" de *STEP* 7 en la pantalla de programación *KOP/AWL/FUP* y lo cargamos mediante el ícono . Establecemos enlace con la CPU configurada mediante el ícono . Adicionalmente podemos observar el estado del programa mediante el ícono .

Giramos el selector de modo del CPU a la posición "*RUN*".

De la misma forma procedemos a abrir el proyecto "*CtrlTemp*" de *WinCC*, ahora definiremos las características de tiempo de ejecución para nuestro proyecto.

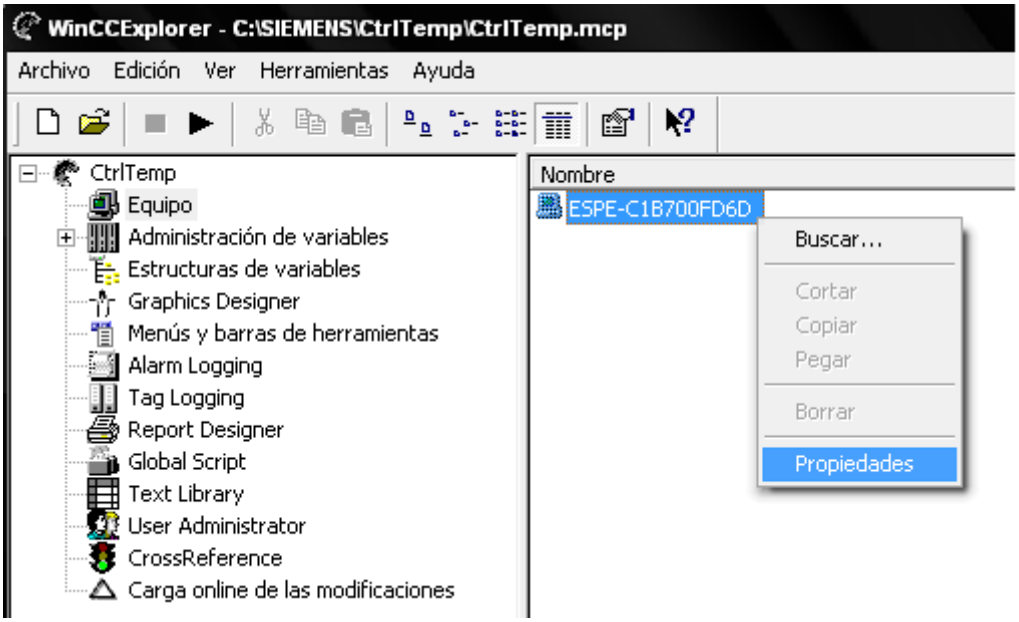


Figura 3.1. Proyecto "*CtrlTemp*" de *WinCC*. Propiedades de equipo

Hacemos clic en "*Equipo*" y con el botón derecho escogemos el nombre de nuestro ordenador. En el menú contextual, hacemos clic en "*Propiedades*", figura 3.1. En la ventana que aparece a continuación seleccionamos la pestaña "*Graphics Runtime*". En este panel definimos el aspecto de la pantalla en tiempo de ejecución y una imagen inicial.

Para seleccionar una imagen inicial, hacemos clic en "*Buscar*" y, seleccionamos la imagen "*Inicio.pdl*" en el cuadro de diálogo "*Imagen inicial*". Activamos las casillas de verificación "*Titulo*", "*Maximizar*", "*Minimizar*" y "*Ajustar imagen*" que aparecen en "*Atributos de ventana*". Figura 3.2.

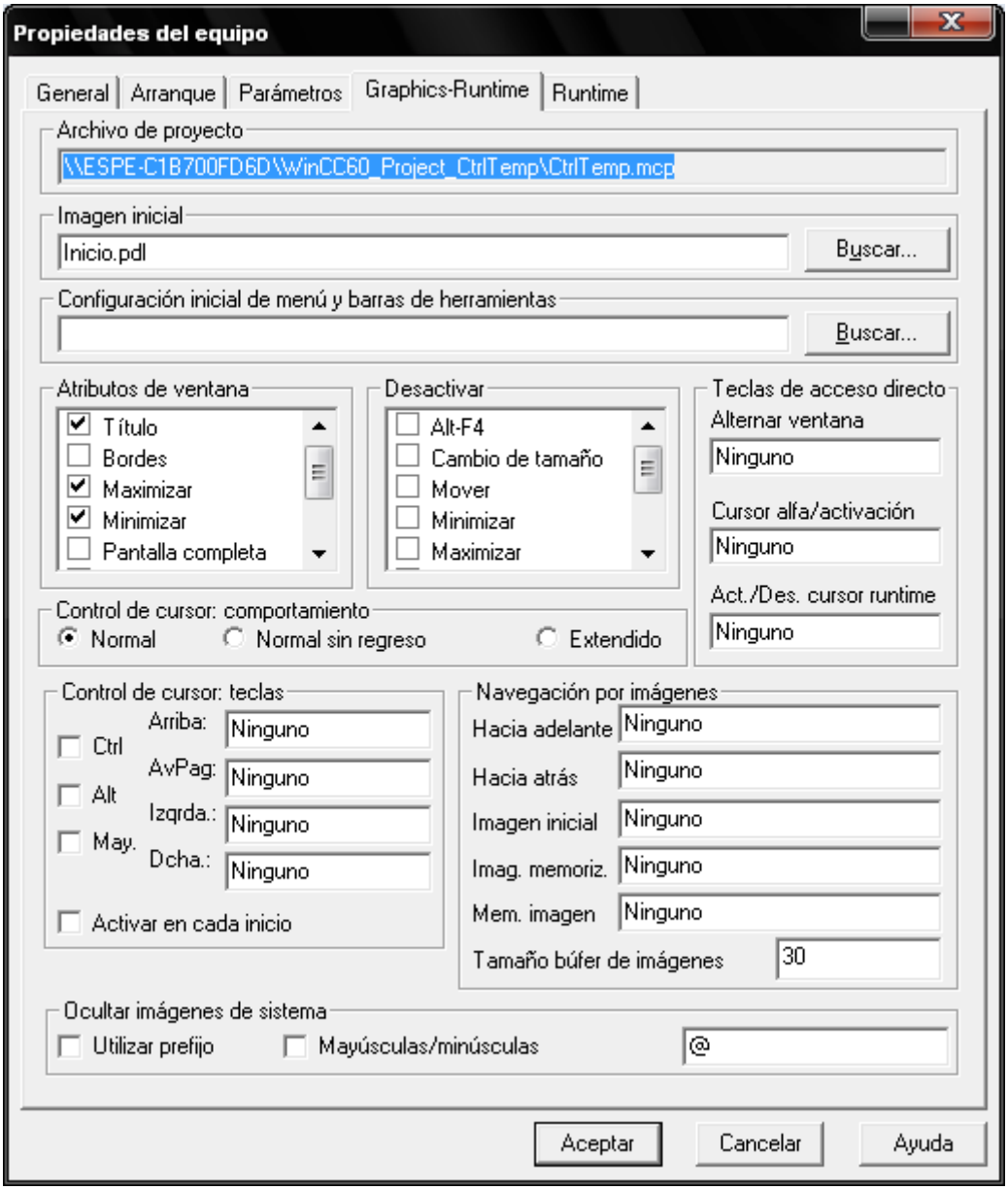


Figura 3.2. Proyecto "*CtrlTemp*" de *WinCC*. Definición de las características de tiempo de ejecución

Una vez definidas las características del tiempo de ejecución, activamos el proyecto mediante el ícono ▶ o mediante la opción de menú Archivo - Activar.

3.1 Esquema general de las pantallas

3.1.1 Inicio

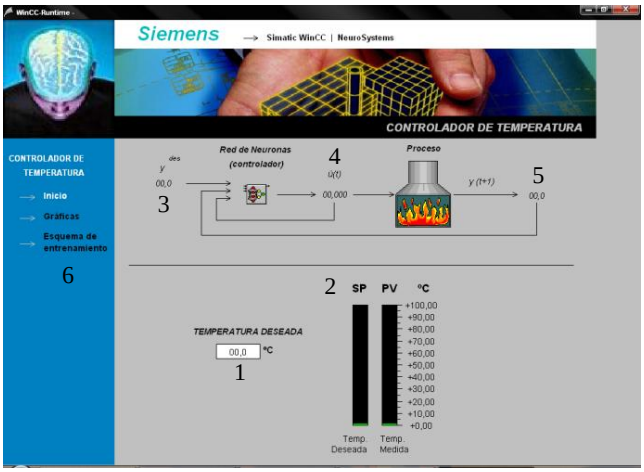


Figura 3.3. Pantalla Inicio

Componentes

1. Cuadro de ingreso del SETPOINT (Temperatura Deseada)
2. Barras de estado del SP y el PV (Temperatura Deseada y Temperatura Medida)
3. Entrada de la red neuronal (SP o Temperatura Deseada)
4. Salida de la red neuronal (Variable controladora del proceso)
5. Temperatura Medida del módulo
6. Barra de acceso a las diferentes ventanas

3.1.2 Graficas

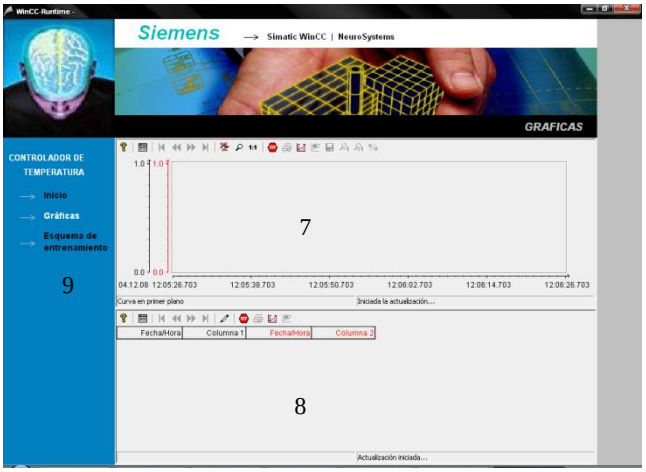


Figura 3.4. Pantalla Gráficas

Componentes

7. WinCC Online Trend Control
8. WinCC Online Table Control
9. Barra de acceso a las diferentes ventanas

3.1.3 Esquema de entrenamiento

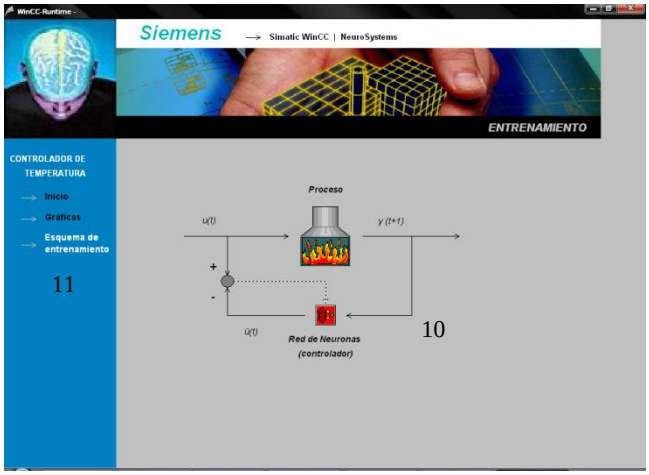


Figura 3.5. Pantalla Esquema de Entrenamiento

Componentes

10. Imagen general del esquema de entrenamiento
11. Barra de acceso a las diferentes ventanas

3.2 Modo de operación

El presente proyecto se planteó como un caso práctico para realizar un sistema controlador de temperatura usando Redes Neuronales. No es de mayor interés asignar características específicas al controlador, y sólo se hace mención a aquellas que se necesiten para la construcción del sistema de control. En este caso solo se necesita saber cómo responde el proceso al sistema controlador (Red de Neuronas).

Pantalla Inicio

Es la pantalla que hemos definido para que aparezca una vez activado el proyecto. En el campo de entrada "*TEMPERATURA DESEADA*" ingresamos el valor de temperatura deseado para el control. Las barras de estado *SP* y *PV*, mostraran el valor de temperatura deseado y la temperatura real del módulo de temperatura.

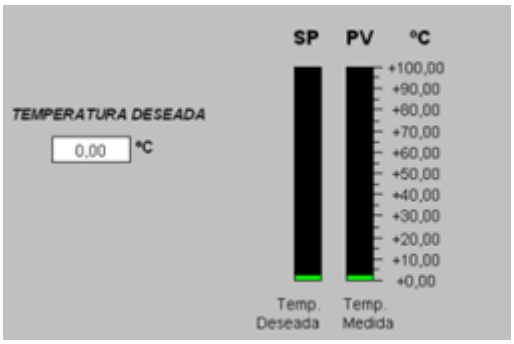


Figura 3.6. Barras de estado *SP-PV* y campo de entrada del *SetPoint*

Los campos de salida mostrarán los valores de entrada de la red neuronal (SP o Temperatura Deseada), de salida de la red neuronal (Variable controladora del proceso) y de la Temperatura Medida del módulo.

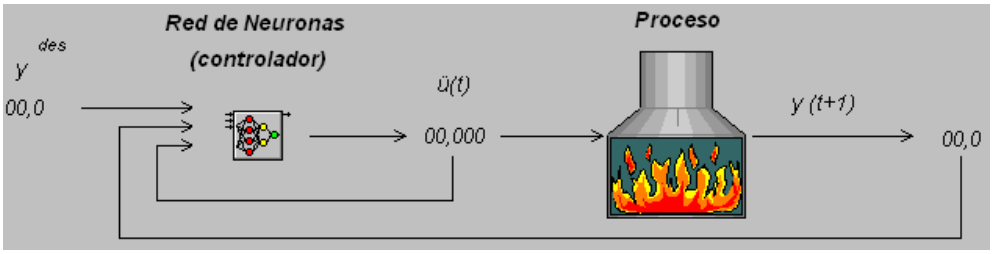


Figura 3.7. Los campos de salida mostrarán los valores de entrada de la red neuronal, de salida de la red neuronal, y de la temperatura medida del módulo.

A través de la barra de acceso ubicada a la derecha de la pantalla podemos acceder las diferentes ventanas, haciendo clic en los textos.

Pantalla Gráficas

Los controles *WinCC Online Trend Control* y *WinCC Online Table Contro*l nos darán una representación de los datos de proceso en forma gráfica (curvas) y como datos (tablas), respectivamente. Se tiene la posibilidad de visualizar valores actuales o archivados. De igual manera dispone de la barra de acceso a las demás pantallas

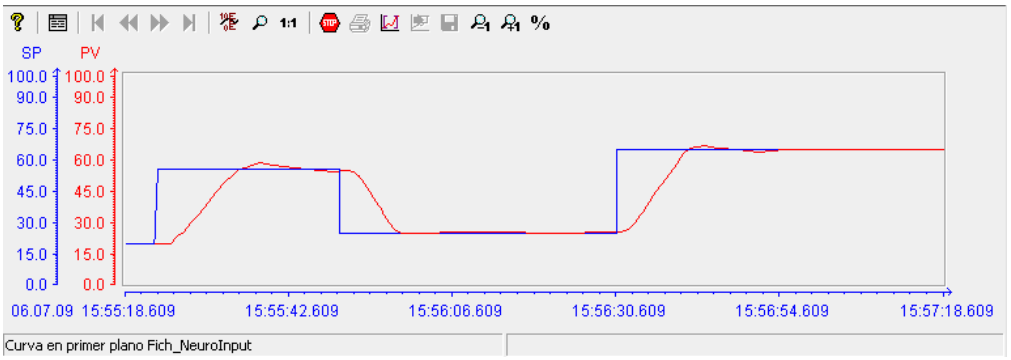
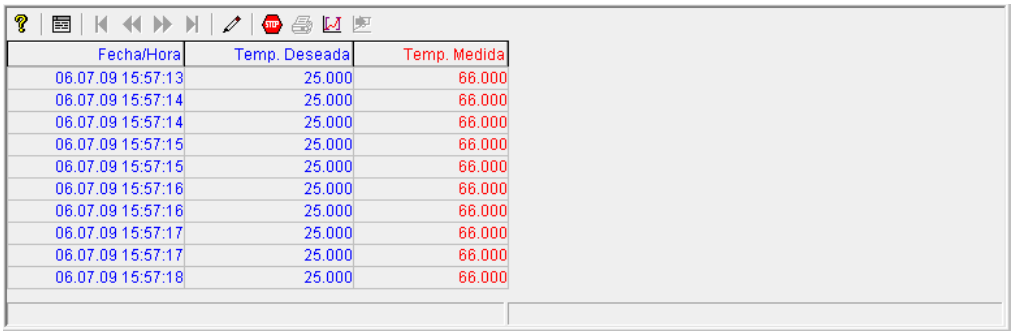


Figura 3.8. ***WinCC Online Trend Control***



Fecha/Hora	Temp. Deseada	Temp. Medida
06.07.09 15:57:13	25.000	66.000
06.07.09 15:57:14	25.000	66.000
06.07.09 15:57:14	25.000	66.000
06.07.09 15:57:15	25.000	66.000
06.07.09 15:57:15	25.000	66.000
06.07.09 15:57:16	25.000	66.000
06.07.09 15:57:16	25.000	66.000
06.07.09 15:57:17	25.000	66.000
06.07.09 15:57:17	25.000	66.000
06.07.09 15:57:18	25.000	66.000

Figura 3.9. ***WinCC Online Table Control***

Adicionalmente se dispone de una *PANTALLA DEL SISTEMA DE ENTRENAMIENTO*. Esta pantalla solo es una imagen que muestra el esquema general de aprendizaje de la red en forma gráfica.

ASISTENCIA TÉCNICA

CTRLTEMP Versión 1.0

Telf: 022475317 - 084314602

Jorge Salazar jesalazar@hotmail.com

A2 PRECONTROL - ARCHIVO DE ENTRENAMIENTO

En la fase de medición debemos registrar las diferentes señales de control y las señales de salida del proceso. Estimamos un tiempo constante de proceso *k* que corresponde a alrededor de un 20% de tiempo de retraso entre la causa (señal de control) y el efecto (señal de salida). Tomamos varias muestras de las señales medidas utilizando el tiempo de muestreo *k* proporcionando un cierto número de valores vinculados. Ejemplo:

Señal de control	Señal de salida
4.5	35
2.1	37
…	…
1.8	39
1.5	33

Luego organizamos estos valores en cuatro columnas de arriba hacia abajo utilizando la siguiente regla:

- Primera columna: los valores de muestreo de la señal de salida
- Segunda columna: los valores de muestreo de la señal de salida, a partir de la segunda fila
- Tercera columna: los valores de muestreo de la señal de control, a partir de la segunda fila
- Cuarta columna: los valores de muestreo de la señal de control

Señal de salida	Señal de salida, desplazada un valor	Señal de control, desplazada un valor	Señal de control
35	-	-	4.5
37	35	4.5	2.1
…	…	…	…
39	…	…	1.8
33	39	1.8	1.5
-	33	1.5	-

Salvo el caso de la primera y la última fila, guardamos todos los registros como un archivo ASCII de nombre "*CtrlTemp.dat*". Haciendo un precontrol de esta manera, logramos que el proceso se comporte casi linealmente y puede ser controlado más fácilmente.

A3 CONFIGURACIÓN DEL CURVE PLOTTER

Para iniciar el *Curve Plotter*, seleccionamos la opción de menú "Test/Curve Plotter".

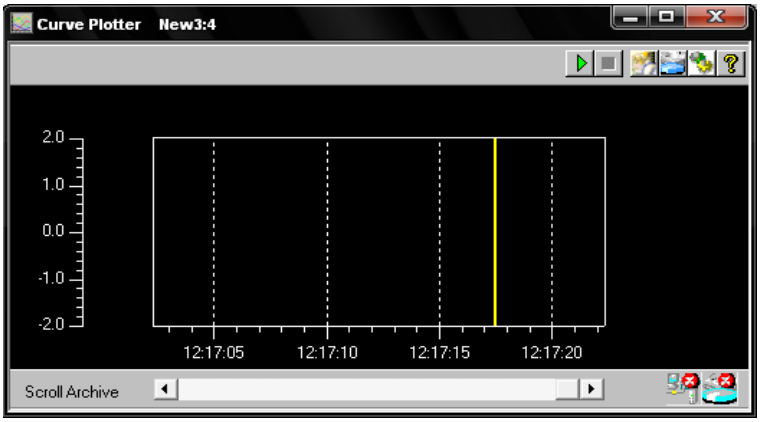


Figura A3.1. *Curve Plotter*

Abrimos la ventana "*Curve Settings*" haciendo clic en el botón correspondiente en la barra de herramientas del *Curve Plotter*.

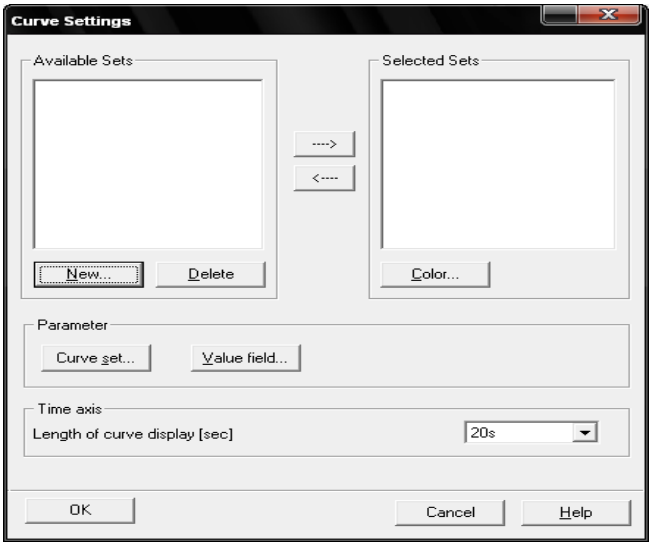


Figura A3.2. Ventana *Curve Settings* para configuración de las señales que vamos a visualizar en el *Curve Plotter*

Creamos un *Set* de señales para ser graficadas. Escogemos el botón *"New…"* de la sección "*Available Sets*".

Seleccionamos las señales *VCDelay*, *VControl*, *VPDelay* y *VProceso* para sean graficadas cambiándoles a cada una el color para poder distinguirlas. Adicionalmente pulsamos el botón "*AutoMinMax*" para establecer el valor mínimo y el máximo del eje Y, y cambiamos el nombre del *Set* por defecto por el de *SetCtrlTemp*, como se muestra en el cuadro a continuación.

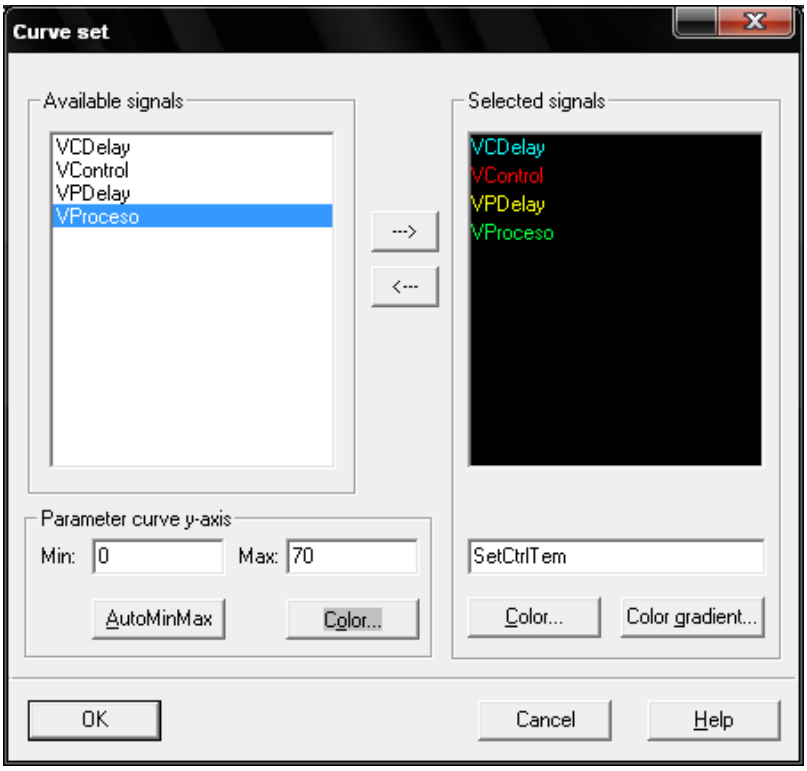


Figura A3.3. Ventana *Curve Set* para configuración del *Curve Plotter*

Damos doble clic sobre el *Set* creado para pasarlo a la sección de "*Selected Sets*", en esta sección se encuentran el set de variables que van a ser graficados en el *curve plotter*, y pulsamos el botón *"Value field…"* para seleccionar los valores que serán mostrados numéricamente sobre la pantalla del *curve plotter*.

Escogemos *CtrlTemp* para pasarlo a la sección "*Selected values*". Confirmamos en *Ok*.

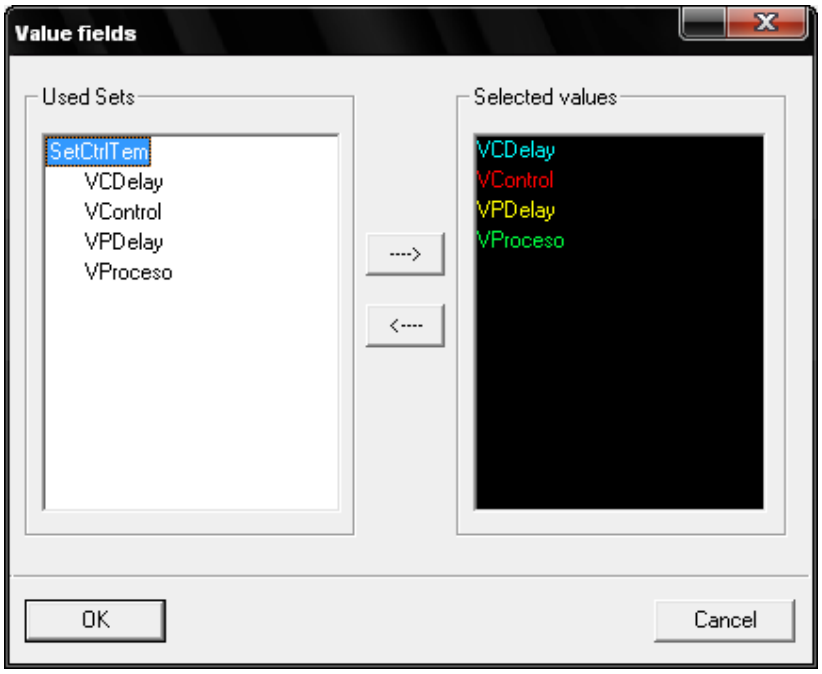


Figura A3.4. En la ventana *Value field,* seleccionamos los valores que serán mostrados numéricamente sobre la pantalla del *curve plotter.*

A continuación abrimos el cuadro "*Curve Generator*" haciendo clic en el botón correspondiente en la barra de herramientas del *Curve Plotter*, para establecer los parámetros de las curvas de las variables de entrada. Escogemos la opción *"curve"* como se muestra en el siguiente gráfico. Confirmamos en *Ok*.

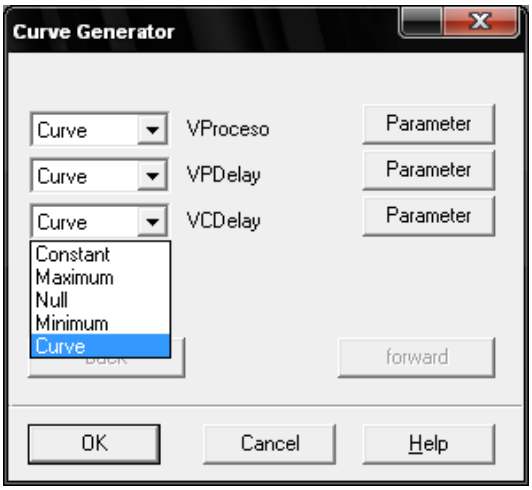


Figura A3.5. En el cuadro de dialogo *Curve Generator* se establecen los parámetros de la curva de la variable de entrada *VProceso.*

A4 HOJA TÉCNICA – DATOS TÉCNICOS GENERALES CPU 315-2 PN/DP

Datos técnicos	
CPU y versión de producto	
Referencia	6ES7315-2EH13-0AB0
• Versión de hardware	01
• Versión de firmware	V 2.5
• Paquete de programas correspondiente	STEP 7 a partir de V 5.4 + SP 1 + HSP
Memoria	
Memoria de trabajo	
• Memoria de trabajo	256 KB
• Ampliable	No
• Tamaño máximo de la memoria remanente para bloques de datos remanentes	128 KB
Datos técnicos	
Memoria de carga	Insertable mediante Micro Memory Card (máx. 8 MB)
Respaldo	Garantizado por la Micro Memory Card (libre de mantenimiento)
Conservación de datos en la Micro Memory Card (tras la última programación)	Mínimo 10 años
Tiempos de ejecución	
Tiempos de ejecución para	
• Operaciones de bits	0,1 µs
• Operaciones de palabras	0,2 µs
• Aritmética en coma fija	2 µs
• Aritmética en coma flotante	3 µs
Temporizadores/contadores y su remanencia	
Contadores S7	256
• Remanencia	Configurable
• Predeterminada	de Z 0 a Z 7
• Rango de contaje	0 a 999
Contadores IEC	Sí
• Clase	SFB
• Cantidad	Ilimitada (sólo por la memoria de trabajo)
Temporizadores S7	256
• Remanencia	Configurable
• Predeterminada	Sin remanencia
• Rango de tiempo	10 ms a 9990 s
Temporizadores IEC	Sí
• Clase	SFB
• Cantidad	Ilimitada (sólo por la memoria de trabajo)
Áreas de datos y su remanencia	
Marcas	2048 bytes
• Remanencia	Configurable
• Remanencia predeterminada	de MB0 a MB15
Marcas de ciclo	8 (1 byte de marcas)

Bloques de datos	
• Cantidad	1023 (en el rango numérico de 1 a 1023)
• Tamaño	16 KB
• Compatibilidad Non-Retain (remanencia ajustable)	Sí
Datos locales según prioridad	Máx. 1024 bytes por nivel de ejecución/ 510 bytes por bloque
Configuración	
Bastidores	máx. 4
Módulos por cada bastidor	8
Cantidad de maestros DP	
• Integrada	1
• a través de CP	4
Módulos de función y procesadores de comunicación compatibles	
• FM	máx. 8
• CP (punto a punto)	máx. 8
• CP (LAN)	máx. 10
Comunicación IE abierta	
Cantidad de conexiones / puntos de acceso, total	8
TCP/IP	Sí (a través de la interfaz PROFINET integrada y FBs cargables)
• Número de enlaces, máx.	8
• Longitud de datos con el tipo de conexión 01H, máx.	1460 bytes
• Longitud de datos con el tipo de conexión 11H, máx.	8192 bytes
ISO on TCP	Sí (a través de la interfaz PROFINET integrada y FBs cargables)
• Número de enlaces, máx.	8
• Longitud de los datos, máx.	8192 bytes
Comunicación básica S7	Sí
• Datos útiles por tarea - De ellos, coherentes	máx. 76 bytes 76 bytes
Comunicación S7	Sí
• Como servidor	Sí
• Como cliente	Sí (mediante la interfaz PN integrada y FBs cargables o también mediante CP y FBs cargables)
MPI	
Servicios	
• Comunicación PG/OP	Sí
• Routing	Sí
• Comunicación de datos globales	Sí
• Comunicación básica S7	Sí
• Comunicación S7	Sí
- Como servidor	Sí
- Como cliente	no (pero vía CP y FBs cargables)
• Velocidades de transferencia	Máx. 12 Mbit/s

A5 HOJA TÉCNICA - DATOS TÉCNICOS DEL MÓDULO SM334; AI 4/AO 2 x 8/8 Bit

Datos técnicos	
Dimensiones y peso	
Dimensiones A x A x P (mm)	40 x 125 x 117
Peso aprox.	285 g
Datos específicos del módulo	
Soporta modo isócrono	No
Cantidad de entradas	4
Cantidad de salidas	2
Longitud de cable - Apantallado	 máx. 200 m
Tensiones, intensidades, potenciales	
Tensión nominal de alimentación para electrónica y tensión nominal de carga L+	24 V c.c.
Separación galvánica • Entre los canales y el bus posterior • Entre los canales y la tensión de alimentación de la electrónica	 No Sí
Entre los canales	No
Diferencia de potencial admisible • Entre las entradas y MANA (UCM) • Entre las entradas (UCM)	 1 V c.c. 1 V c.c.
Aislamiento ensayado con	500 V c.c.
Consumo • Del bus posterior • De la tensión de alimentación y de carga L+ (sin carga)	 máx. 55 mA máx. 110 mA
Disipación del módulo	Típ. 3 W
Formación de valores analógicos para las entradas	
Principio de medida • Resolución (incl. margen de saturación) conversión de valores instantáneos	 8 bits
Período de integración/tiempo de conversión (por canal) • Parametrizable • Período integr. en µs	 No <500
Tiempo de ejecución básico de las entradas	máx. 5 ms
Constante de tiempo del filtro de entrada	0,8 ms
Formación de valores analógicos para las salidas	
• Resolución (incl. margen de saturación)	8 bits
Tiempo de conversión (por canal) • Parametrizable • Tiempo de conversión en µs	 No <500
Tiempo de ejecución básico de las salidas	máx. 5 ms
Tiempo de estabilización • Con carga óhmica • Con carga capacitiva • Con carga inductiva	 0,3 ms 3,0 ms 0,3 ms
Supresión de perturbaciones, límites de error para salidas	
Diafonía entre las salidas	> 40 dB
Límite de error práctico (en todo el rango de temperaturas, en referencia al rango de salida)	

• Salida de tensión • Salida de intensidad	 ± 0,6 % ± 1,0 %
Límite de error básico (límite de error práctico a 25 °C, en referencia al rango de salida) • Salida de tensión • Salida de intensidad	 ± 0,5 % ± 0,5 %
Error por temperatura (en referencia al rango de salida)	± 0,02 %/K
Error de linealidad (en referencia al rango de salida)	± 0,05 %
Exactitud de repetición (en estado estacionario a 25 °C, en referencia al rango de salida)	± 0,05 %
Ondulación de salida (ancho de banda en referencia al rango de salida)	± 0,05 %
Estados, alarmas, diagnóstico	
Alarmas	Ninguna
Funciones de diagnóstico	Ninguna
Datos para seleccionar un sensor	
Rango de entrada (valores nom.)/resistencia de entrada • Tensión • Intensidad	 0 a 10 V/100 k Ω 0 a 20 mA/50 Ω
Tensión de entrada admisible para las entradas de tensión (límite de destrucción)	máx. 20 V perman.;75 V durante máx. 1s (relación puls./pausa 1:20)
Intensidad de entrada admis. para las entradas de intensidad (límite de destrucción)	 40 mA
Conexión de los sensores • Para medir la tensión • Para medición de intensidad - Como transductor a 2 hilos - Como transductor a 4 hilos	Conector frontal de 20 pines Posible Posible, con alimentación externa Posible
Datos para seleccionar un actuador	
Rangos de salida (valores nominales) • Tensión • Intensidad	 0 a 10 V 0 a 20 mA
Resistencia de carga (en el rango nominal de la salida) • En salidas de tensión - Carga capacitiva • En salidas de intensidad - Carga inductiva	 mín. 5 kΩ máx. 1 µF máx. 300 Ω máx. 1 mH
Salida de tensión • Protección contra cortocircuitos • Corriente de cortocircuito	 Sí máx. 11 mA
Salida de intensidad • Tensión en vacío	 máx.15V
Límite de destrucción por tensiones/intensidades aplicadas desde el exterior • Tensión en las salidas respecto a MANA • Intensidad	 máx. 15 V perman. máx. 50 mA c.c.
Conexión de actuadores • Para salida de tensión - Conexión a 2 hilos - Conexión a 4 hilos (conductor de medida)	Conector frontal de 20 pines Posible Imposible

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE HOJAS TÉCNICAS

GLOSARIO

ADALINE

Las redes adaline (*ADAptative LInear NEuron*), fueron desarrolladas por Bernie Widrow en la Universidad de Stanford. Dicha red usa neuronas con función de transferencia escalón, y está limitada a una única neurona de salida. Esta red puede considerarse una generalización del perceptrón. Mientras que este último sólo trabaja con entradas y salidas binarias o bipolares, el Adaline trabaja con patrones de entrada y salida reales. Su aprendizaje incluye la diferencia entre el valor real producido y su salida esperada, para un patrón de entrada específico.

BACKPROPAGATION

La propagación hacia atrás de errores o retropropagación (*backpropagation*) es un algoritmo de aprendizaje supervisado que se usa para entrenar redes neuronales artificiales. El algoritmo consiste en minimizar un error (comúnmente cuadrático) por medio de gradiente descendiente, por lo que la parte esencial del algoritmo es el cálculo de las derivadas parciales de dicho error con respecto a los parámetros de la red neuronal.

PROFIBUS

Es el potente, abierto y robusto sistema de bus para la comunicación de proceso y de campo en redes de célula con pocas estaciones y para la comunicación de datos según IEC 61158/EN 50170. Los dispositivos de automatización tales como PLC, PC, HMI, sensores u actuadores pueden comunicarse a través de este sistema de bus.

PROFINET

Es el estándar Industrial Ethernet abierto y no propietario para la automatización. Con él es posible una comunicación sin discontinuidades desde el nivel de gestión hasta el nivel de campo.

KOP

Lenguaje KOP o Ladder, este lenguaje también llamado lenguaje de escalera permite crear programas con componentes similares a los elementos de un esquema de circuitos. Los programas se dividen en unidades lógicas pequeñas llamadas *networks*, y el programa se ejecuta segmento a segmento, secuencialmente, y también en un ciclo.

FUP

Lenguaje FUP, consiste en un diagrama de funciones que permite visualizar las operaciones en forma de cuadros lógicos similares de los de de las puertas lógicas. El estilo de representación en forma de puertas gráficas se adecua especialmente para observar el flujo del programa

AWL

Lenguaje AWL, este incluye una lista de instrucciones que se ejecutan secuencialmente dentro de un ciclo. Una de las principales ventajas que presenta es que cualquier programa creado en FUP o KOP puede ser editado por AWL, no así a la inversa.

MPI

Interface Multi Punto, es un protocolo de Siemens que permite la comunicación con los PLCs de esta marca con dispositivos externos

TCP/IP

Acrónimo inglés cuyo significado es *Transmission Control Protocol* / *Internet Protocol* (Protocolo de control de transmisión / Protocolo de Internet). Este sistema de protocolos de uso obligatorio en Internet y que permite la conexión de ordenadores y redes. El

protocolo TCP se encarga de dividir la información en paquetes de origen que luego es recompuesta en el punto de destino. El protocolo IP tiene como función dirigir la información adecuadamente a través de la red por las rutas correctas.

OCX

Es un acrónimo que significa "*OLE Control Extension*". OLE a su vez significa *Object Linking and Embedding*. OCX hace referencia a módulos que publican controles y funciones para ser utilizados en programas para Windows, incluyendo especialmente el navegador Internet Explorer. Típicamente, estas librerías se presentan en librerías de enlace dinámico (DLL) almacenadas con extensión .OCX.

DLL

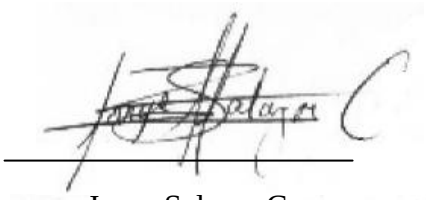
Dynamic Link Library (librería *de enlace dinámico*), es el término con el que se refiere a los archivos con código ejecutable que se cargan bajo demanda de un programa por parte del sistema operativo. Esta denominación se refiere a los sistemas operativos Windows siendo la extensión con la que se identifican los ficheros, aunque el concepto existe en prácticamente todos los sistemas operativos modernos.

FECHA DE ENTREGA

El presente proyecto de grado fue entregado en la fecha

Sangolquí, 23 julio 2009

Realizado por:



Jorge Salazar C.

Ing. Víctor Proaño

COORDINADOR DE CARRERA DE AUTOMATIZACIÓN Y CONTROL