UNIVERSIDAD AUSTRAL DE CHILE
FACULTAD DE CIENCIAS DE LA INGENIERÍA
ESCUELA DE ELECTRICIDAD Y ELECTRÓNICA

SISTEMA DE RECONOCIMIENTO DE VOZ PARA ACTIVACIÓN DE DISPOSITIVOS ELECTRÓNICOS, IMPLEMENTADO CON REDES NEURONALES

Trabajo de Titulación para optar al
Título de Ingeniero Electrónico

PROFESOR PATROCINANTE
Sr. Jorge Morales Villugrón

Sergio Eduardo Mansilla Vidal
Valdivia – Chile
2007

Comisión de Titulación

Profesor Patrocinante:

Sr. Jorge Morales Villugrón. Firma:_____

Profesores Informantes:

Sr. Pedro Rey Clericus. Firma:_____

Sr. Raúl Urra Ríos. Firma:_____

AGRADECIMIENTOS Y DEDICATORIAS

A Dios por ser mi fuente de fortaleza en todo momento.

A mis padres Crisólogo y Teresa, porque gracias a su esfuerzo, apoyo incondicional, amor y paciencia pude alcanzar esta meta. Los quiero y les dedico este trabajo.

A mis tíos Mely, Guille, Pedro y a toda mi familia por apoyarme siempre que los necesité.

A mis compañeros de carrera, Alfredo Arce, Carlos Burgos, José Campos y Moisés Vergara, quienes fueron un apoyo fundamental durante mis días de estudiante y que además de mis compañeros se convirtieron en grandes amigos.

Y todos aquellos que de alguna u otra forma me ayudaron a conseguir este logro.

CONTENIDO

RESUMEN

Este trabajo consiste en el desarrollo de un sistema capaz de reconocer palabras pronunciadas por un locutor y a partir de ello realizar una acción determinada. Para este propósito se recurrió al recurso tecnológico enmarcado en lo que es la inteligencia artificial denominado redes neuronales artificiales, caracterizadas por su capacidad de generalizar y reconocer patrones en informaciones cuyo conocimiento no está formalizado, y que en la voz humana es complicado de modelar debido a la aleatoriedad de su forma. En este sentido las redes neuronales artificiales surgen como una solución a este problema y muestran mejores prestaciones que otras tecnologías.

El sistema será capaz de reconocer palabras aisladas independientes del locutor, lo que significa que cualquier persona podrá utilizar el sistema sin modificarlo. Los fonemas identificados serán usados como comandos que ordenen realizar una acción de control sobre un dispositivo eléctrico. Para esto se desarrollará un software basado en redes neuronales, utilizando VISUAL BASIC e instalado en un PC siendo este último quien se comunique con el dispositivo a controlar por medio de sus puertos.

La implementación del sistema involucrará el diseño de la red neuronal, la creación de los set de entrenamiento y el entrenamiento mismo, presentando un detalle de los resultados obtenidos en este trabajo. Además, se desarrollará un sistema de comando que permita una fácil operación y visualización de los eventos. Asimismo se construirán algunos dispositivos electromecánicos que permitan su manipulación desde el PC a través de los comandos de voz reconocidos por el sistema.

Finalmente se mostrarán los resultados conseguidos realizando un análisis d estos en las conclusiones del trabajo, donde se tratarán los puntos altos y bajos de este, así como algunas propuestas para mejorar el sistema.

SUMMARY

This research involves the development of a system capable of recognizing words pronounced by a speaker and from that performing a determined action. For this purpose, within the framework of the artificial intelligence, a technological resource, named Artificial Neural Network was utilized. This technology is characterized by its ability of generalizing and recognizing patterns in pieces of information whose knowledge is not formalized, and in the human voice is difficult to model them due to their random shape. In this sense, Artificial Neural Networks emerge as a solution of this problem and they show better performance than other technologies.

The system will be able to recognize isolated words, independently from its speaker, which means that any person could use the system without modifying it. The identified phonemes will be used as commands that order to perform a control action over an electrical device. For this matter, a software based on neural networks installed in a PC will be developed, using VISUAL BASIC. The PC will be in charge of communicating with the controlling device through its ports.

The system implementation will involve the neural network design, the creation of the training sets and the training itself, presenting a detailed description of the results obtained in this work. Besides that, a control system that allows its easy display and operation of the events will be developed. Moreover, some electrical-mechanical devices will be built which will allow their operation from the PC through voice commands recognized by the system.

Finally, the obtained results will be shown in the conclusions of this work, carrying out a deeper analysis of them. Here, some advantages and disadvantages of the system, likewise, some answers to improve it will be explained.

INTRODUCCIÓN

Con el correr del tiempo, las computadoras se han convertido en máquinas sumamente veloces y eficientes a la hora de procesar información, realizan millones de cálculos por segundo sin equivocarse, no se cansan y sirven para innumerables tareas desde la edición de un texto hasta la simulación de la reacción del choque de dos partículas a gran velocidad dentro de un reactor. Sin embargo y a pesar de la gran evolución que estos aparatos han experimentado, aún no pueden realizar tareas tan sencillas para un humano como es reconocer un rostro o distinguir la voz de alguien, o entender lo que una persona dice. Así en todas estas tareas donde el objeto tiene características abstractas, las computadoras han demostrado estar todavía lejos de igualar la capacidad natural del hombre. Y es esta capacidad la que le permite subsistir y convivir con su entorno.

Así, investigadores tratando de imitar las maravillosas habilidades del cerebro humano como la de reconocer patrones y generalizar para solucionar problemas nunca antes vistos a partir de información conocida, crearon un modelo matemático de neurona basado en la neurona biológica que consistió en un nodo dotado de un determinado número de entradas (dendritas) y salidas (axón), donde cada entrada estaba afectada por un peso sináptico. De esta manera se buscaba obtener respuestas similares a las que entrega el cerebro humano en máquinas creadas con el fin de reemplazar al hombre en tareas en las que se requiere de la capacidad de manipular información que no pueden ser procesadas mediante algoritmos. De esta forma nacen las Redes Neuronales Artificiales como una rama de la Inteligencia Artificial.

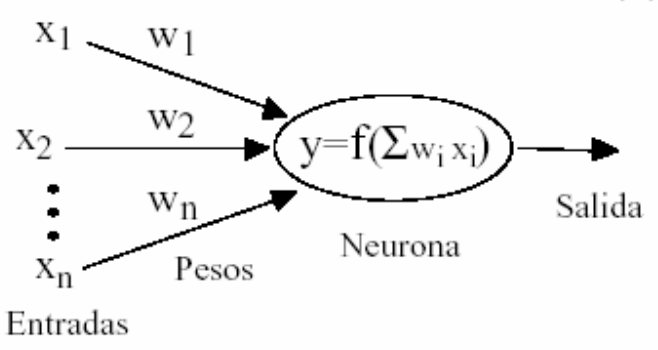


Fig.1 Modelo matemático de una neurona biológica simplificada.

Entre las distintas aplicaciones que ha encontrado esta tecnología destacan el reconocimiento de imágenes, de texto manuscrito, predicción de las condiciones meteorológicas, realizar predicciones en el mercado financiero, entregar diagnósticos médicos, detectar fallas en engranajes y reconocimiento de patrones de voz. Es en esta última donde se fusiona con otra rama de la IA denominada "Reconocimiento automático del habla". El reconocimiento automático del habla tiene como objetivo fundamental permitir al hombre comunicarse con las máquinas a través de su lenguaje natural, dejando de lado otros dispositivos de entrada, dando paso al diálogo.

Paralelamente existen otras técnicas de clasificación de patrones como los ***Modelos Ocultos de Markov*** que corresponden a estructuras probabilísticas basadas en estados, donde cada uno de estos depende del anterior. Así esta técnica presenta características apropiadas para el reconocimiento de voz, ya que cada estado depende tanto del estado futuro como del previo, por lo que posee entonces una memoria que le permite relacionar las señales que se le ingresan para, en el caso de reconocer palabras, formarlas. Sin embargo a diferencia de las Redes Neuronales Artificiales este sistema no es capaz de generalizar y aprender, y depende en gran medida de la técnica de extracción de parámetros su correcto funcionamiento. Han demostrado una gran eficiencia en la tarea de reconocer palabras.

Otra técnica utilizada en la clasificación de patrones es la denominada ***Alineación Dinámica Temporal*** (DTW, Dinamic Time Wrapping por sus siglas en inglés), que consiste en comparar el patrón a detectar con una base de datos que contiene un codebook o plantillas previamente almacenadas. Este sistema es especialmente útil para el procesamiento de información que no siempre se presenta con la misma duración, ya que permite comparar señales con longitudes temporales distintas y calcula la distancia euclídea del vector desconocido con los vectores almacenados. De esta forma al patrón que se acerque más en distancia será el que será reconocido como correcto.

Se puede ver que todo sistema clasificador de patrones posee 3 etapas: cuantización de la señal, extracción de parámetros y clasificación de patrones. Como se mencionó existen distintas técnicas de clasificación de patrones, pero también existen variadas técnicas de extracción de

parámetros, cuya eficiencia es crucial en el rendimiento del clasificador de patrones, ya que de este depende cuánta información útil de una señal se le proporcione al clasificador durante su entrenamiento y como éste se ajusta a esos datos y es capaz de discriminar acertadamente entre los distintos modelos que se están tratando de diferenciar.

Entre las técnicas de extracción de parámetros están la ***Transformada de Fourier,*** en el tiempo discreto para este caso, que entrega información de la constitución del espectro de una señal en un determinado instante de tiempo, ***La Predicción De Coeficientes Lineales*** que entregan información temporal de parámetros característicos de una señal o el ***Cepstrum Complejo*** que corresponde a la "Transformada inversa del logaritmo de la Transformada de Fourier" que entrega información del comportamiento tanto temporal como espectral de una señal.

Sin embargo lograr la habilidad en el reconocimiento de voz que poseemos los humanos, está lejos de ser alcanzado eficientemente por un aparato electrónico, ya que en él es necesario combinar una serie de factores tales como acústica, fonética, sintáctica, semántica entre otros, además de incertidumbres y errores inevitables que se producen cuando hablamos, para lograr un reconocimiento aceptable.

Por otra parte el problema que presenta el reconocimiento de patrones de voz es que necesita de un sistema que sea capaz de procesar datos tanto en el espacio como en el tiempo, y en este sentido las redes neuronales por sí solas son capaces de procesar información solo en el dominio espacial, sin embargo se verá que con ayuda de la programación se soluciona este inconveniente de manera que la red nos permitan procesar información variante en el tiempo y el espacio.

Se verá entonces cómo se implementó un sistema capaz de reconocer un conjunto limitado de palabras aisladas de 4 letras independientes del locutor sobre un computador y se construyeron 2 dispositivos para su operación mediante la utilización de estas palabras como comandos de voz a través del puerto paralelo del PC. Como sistema de reconocimiento de patrones este consistió en 3 procesos: como sensor se utilizó la tarjeta de audio del PC para

digitalizar la voz de un locutor, posteriormente se extrajeron parámetros característicos de la señal por medio de la Transformada Rápida de Fourier y se eligió una red neuronal *Perceptrón Multicapa* con la regla de aprendizaje *Backpropagation,* como técnica de clasificación de patrones. La red fue escrita en Visual Basic 6.0 así como el programa para controlar los dispositivos y la FFT.

El 1er. dispositivo implementado fue un display de 7 segmentos, para el cual se entrenó la red neuronal con las primeras 8 letras del abecedario, estás son presentadas en el display a través del puerto paralelo y en el software cada vez que se pronuncia una de las letras en el micrófono.

Fig.2 Display e Interfaz.

El 2º dispositivo construido fue un brazo mecánico. Este está construido a partir de 3 servomotores que realizan los movimientos horizontal, vertical y de agarre. Para la comunicación entre los motores y el computador fue necesario implementar un interfaz que permitiese controlar los movimientos de los motores de acuerdo a las órdenes recibidas desde el PC. Este estuvo basado en un microcontrolador PIC 16F84a que se encargaba de recibir las ordenes desde el computador y activar los motores. Para este caso se entrenó la red con un conjunto de 5 palabras que definían todos los movimientos posibles del brazo mecánico. Así al pronunciar uno de los 5 comandos el brazo reaccionaba de acuerdo al comando pronunciado en el micrófono y aparecía reflejado en el software de control.

Fig.3 Brazo mecánico

A continuación se abordarán en profundidad las distintas etapas del trabajo, explicando en forma detallada la investigación, desarrollo e implementación del sistema de reconocimiento de palabras y los dispositivos electrónicos construidos.

Hipótesis

- Utilizando las capacidades de las redes neuronales artificiales se conseguirá reconocer un vocabulario de palabras.

- El sistema reconocerá el vocabulario de palabras independientemente del locutor.

- Mediante el uso del vocabulario de palabras se comandará por medio de software implementado en un computador distintos dispositivos electrónicos.

- Se crearán distintos vocabularios para manejar diferentes dispositivos.

Objetivo General

Desarrollar un sistema informático (software) que permita reconocer la voz humana en tiempo real y que a través de palabras definidas permita activar y controlar dispositivos electrónicos. Para ello se utilizarán Redes Neuronales Artificiales.

Objetivos Específicos

- Capacidad del sistema de reconocer vocabulario limitado de palabras.
- Reconocimiento de palabras independientes del locutor.
- Vocabulario modificable.
- Implementación de la red neuronal en software para el reconocimiento de palabras.
- Utilización de la red neuronal para el aprendizaje de un vocabulario definido de acuerdo a la tarea que se quiere realizar.
- Conseguir relacionar la palabra reconocida con una acción.
- Control de dispositivo eléctrico o electrónico a través de los puertos de un PC.

Metodología

Durante la primera parte el trabajo está orientado a la investigación y análisis del material existente respecto a lo que son las Redes Neuronales Artificiales y su utilización en el reconocimiento de voz. Posteriormente se realizará un breve análisis de la información a manipular que es la voz humana y las técnicas disponibles para la extracción de características. Seguidamente viene una etapa de desarrollo que consistirá en el diseño de la Red Neuronal y su construcción, a lo que se agrega todo lo que es el entrenamiento de la misma y el período de pruebas y de funcionamiento.

El trabajo consistirá en el desarrollo de una aplicación real por lo que este proyecto de tesis se enmarca en lo que es el desarrollo de un proyecto profesional.

CAPÍTULO I. PLANTEAMIENTO DEL PROBLEMA.

Antes de plantear el problema como tal y ver las formas de abordarlo se verá cuáles son los factores que lo condicionan y como estos influyen en su funcionamiento. Para ello ser hará una revisión de lo que son las redes neuronales artificiales como sistema de reconocimiento y clasificación de patrones, y posteriormente en los que son las técnicas de reconocimiento de voz.

I.1 Redes Neuronales Artificiales

Las redes neuronales artificiales tienen sus orígenes allá por el año 1943 cuando el neurofisiólogo Warren McCulloch y el matemático Walter Pitts dieron a luz el primer modelo matemático que describía el comportamiento de una neurona biológica simplificada. Estos estudios dieron asidero a otros investigadores como Hebb, Minsky, Widrow, Rosenblatt y Kohonen entre muchos otros.

I.1.1 Neuronas Biológicas

Una neurona es una célula biológica especial que procesa información. Está compuesta de un cuerpo celular, o *soma*, y dos tipos de ramas que llegan al exterior: el *axón* y las *dendritas*. Una neurona recibe señales (impulsos) de otras neuronas a través de sus dendritas (receptores) y transmite señales generadas por su cuerpo celular a lo largo del axón (transmisor), el cual eventualmente se divide en ramales y sub-ramales. En los extremos de estos ramales están las *sinapsis*. Una sinapsis es una estructura elemental y una unidad funcional entre dos neuronas (un axón de una neurona y una dendrita de otra).

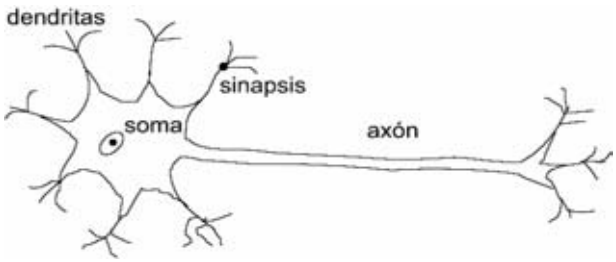


Fig.4 Neurona biológica

Cuando el impulso alcanza el extremo de la sinapsis, ciertos químicos llamados neurotransmisores son liberados. Los neurotransmisores se difunden a través del espacio sináptico, para aumentar o inhibir, dependiendo del tipo de la sinapsis, la tendencia de la neurona receptora a emitir impulsos eléctricos. La efectividad de la sinapsis puede ser ajustada por las señales que pasan a través de ella de tal manera que las sinapsis pueden *aprender* de las actividades en las que participan. Esta dependencia de la historia actúa como una memoria.

I.1.2 Neuronas Artificiales y Redes Neuronales

Existen varias definiciones de lo que es una Red Neuronal Artificial, pero en términos generales se puede decir que una *red neuronal artificial* es un sistema de procesamiento de información en paralelo, basado en las neuronas biológicas y diseñado para modelar la forma en que el cerebro realiza una tarea particular. Así este sistema puede ser implementado por medio de circuitos electrónicos o simulada por medio de software en un computador. Estas redes están constituidas por elementos simples de procesamiento de información llamadas neuronas. Las señales son transmitidas entre neuronas por medio de conexiones. Cada conexión tiene un peso asociado, que en una red neuronal normal multiplica la señal transmitida. De esta forma las neuronas aplican una función de activación, generalmente no lineal, a su entrada, que corresponde a la suma de cada salida de otras neuronas multiplicadas por su correspondiente peso, para calcular su salida.

Una red neuronal se caracteriza por:

- Su patrón de conexión entre neuronas, también llamado *arquitectura*,
- Su método para ajustar los pesos de las conexiones, llamado también *tipo de aprendizaje*, y,
- Su *función de activación.*

Una red neuronal consiste en un gran número de elementos simples de procesamiento llamados neuronas. Cada neurona está conectada con otras neuronas por medio de enlaces dirigidos, cada uno con un peso asociado. Los pesos representan la información utilizada por la red para resolver una tarea.

Cada neurona tiene un estado interno llamado nivel de actividad o de activación, que es una función de sus entradas. Comúnmente una neurona envía su estado de activación como una señal a varias neuronas. Es importante notar que una neurona puede enviar solo una señal a la vez, aunque esa señal sea transmitida a varias otras neuronas.

Una *neurona artificial* es una unidad de procesamiento de información que es fundamental para la operación de una red neuronal. En el modelo de neurona artificial se distinguen 3 elementos básicos:

1. Un conjunto de sinapsis o enlaces de conexión, cada uno de los cuales está caracterizado por un peso propio. Específicamente, una señal x_j en la entrada de la sinapsis j conectada a la neurona k es multiplicada por el peso sináptico w_{kj}.
2. La suma de las señales de entrada, ponderada por las respectivas sinapsis de la neurona; estas operaciones constituyen una *combinación lineal*.
3. Una *función de activación* para limitar la amplitud de la salida de una neurona en un rango permisible. Típicamente, el rango de amplitud normalizado de la salida de una neurona está dado por el intervalo unitario cerrado [0, 1] o alternativamente [-1, 1].

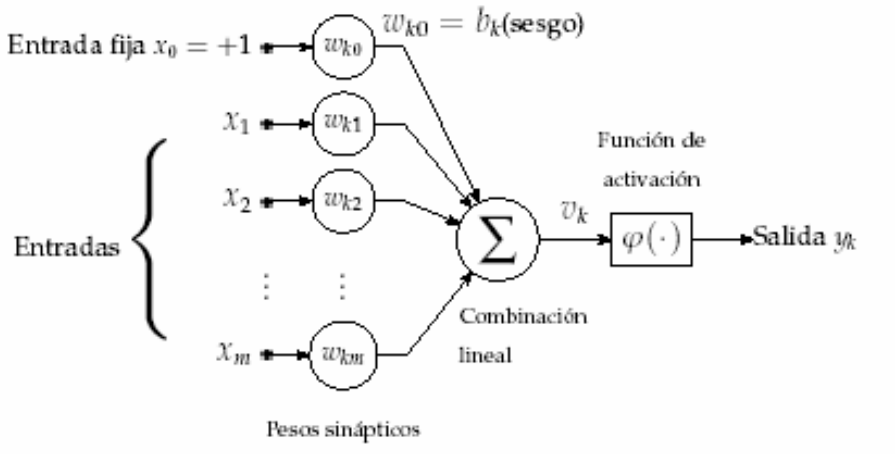


Fig.5 Neurona Artificial

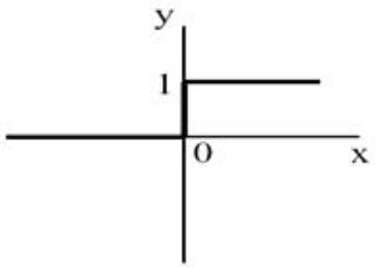
El modelo de neurona incluye un valor umbral que representa el umbral natural de una neurona biológica para que esta se active o se inhiba. Este valor umbral actúa como una entrada más de la neurona que tiene por valor siempre +1. De esta forma en términos matemáticos podemos expresar la neurona k de acuerdo a las siguientes ecuaciones:

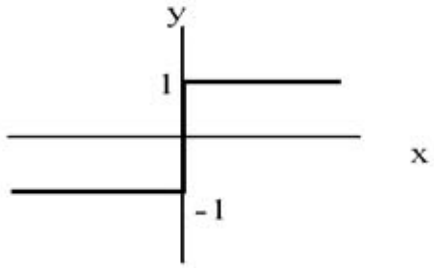
$$Net_k = \sum_{j=1}^{m} w_{kj} x_j + \theta_{k0}$$

$$y_k = \phi(Net_k)$$

donde $x_1, x_2, \ldots, x_m$ son las señales de entrada, $w_{k1}, w_{k2}, \ldots, w_{km}$ son los pesos sinápticos de la neurona *k*, Net_k es la sumatoria de las entradas de la neurona *k* multiplicada por su respectivo peso sináptico, llamada función base e y_k es la salida de la neurona *k* donde $\varphi()$ corresponde a la función de activación de la neurona, de estas se identifican 3 tipos:

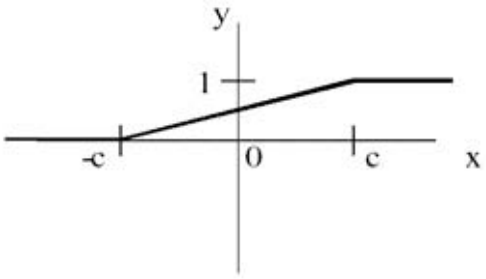
- Función escalón definida por:

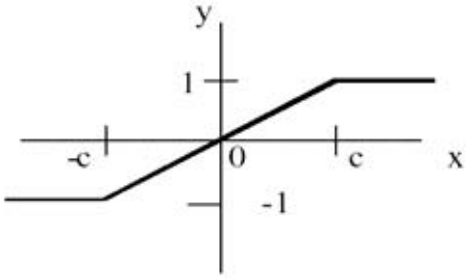




$$\varphi(Net_k) = \begin{cases} 1, & v_k \geq 0 \\ 0, & v_k < 0 \end{cases} \qquad \varphi(Net_k) = \begin{cases} 1, & v_k \geq 0 \\ -1, & v_k < 0 \end{cases}$$

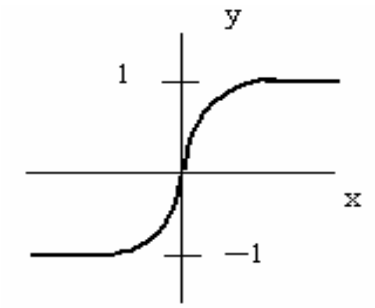
- Función lineal por partes definida por:

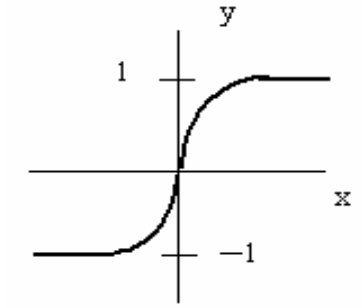




$$\varphi(Net_k) = \begin{cases} 1, v_k \geq c \\ v_k, -c < v_k < c \\ 0, v_k < c \end{cases} \qquad \varphi(Net_k) = \begin{cases} 1, v_k \geq c \\ v_k, -c < v_k < c \\ -1, v_k < c \end{cases}$$

- Función sigmoide: es la función más utilizada en la construcción de redes neuronales, permite el procesamiento de información tanto binaria como analógica. Es la función más completa ya que exhibe las bondades tanto de las funciones escalón como parcialmente lineal, además de ser derivable. Está definida por:





$$\varphi(Net_k) = \frac{1}{1 + e^{-\alpha Net_k}} \qquad \varphi(Net_k) = \tanh(Net_k)$$

I.1.3 Arquitectura De Redes Neuronales Artificiales

Al arreglo de las neuronas en capas y los patrones de conexión al interior y entre capas se denomina *arquitectura de red*.

Existen 2 tipos:

I.1.3.1 feedforward o de propagación hacia delante*:*

Cuando ninguna salida de las neuronas de una capa es entrada de neuronas del mismo nivel o de niveles precedentes y cada salida de las neuronas de un nivel son entradas de neuronas ubicadas en niveles posteriores, se dice que la red tiene propagación hacia delante.

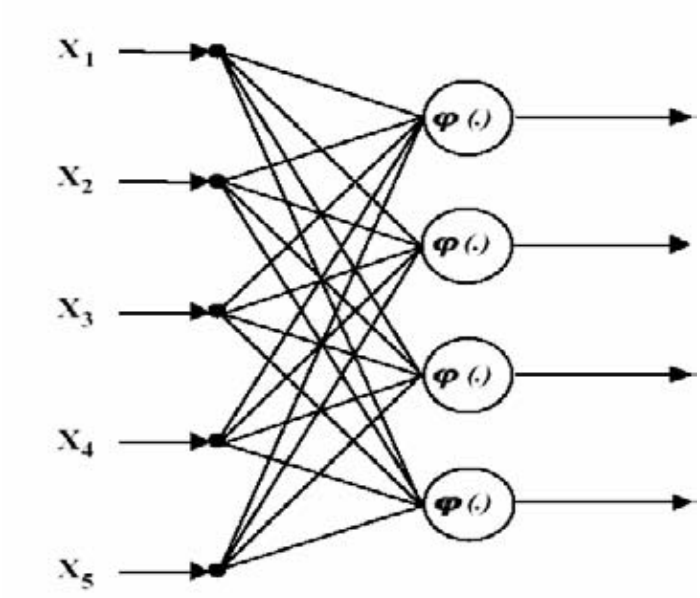


Fig.6 Red Neuronal de una capa de pesos

I.1.3.2 feedback o recurrentes*:*

Cuando las salidas de neuronas de una capa son entradas tanto de neuronas del mismo nivel (incluso de sí mismas) o de otros niveles precedentes y posteriores se dice que la red es recurrente. Este tipo de red se dice que posee memoria ya que la respuesta en un momento depende de un estado anterior.

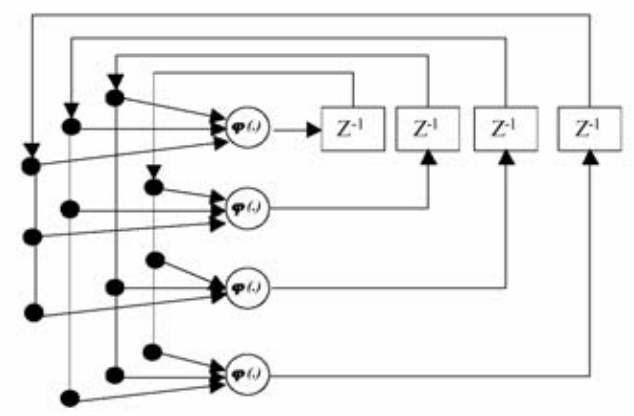


Fig.7 Red monocapa recurrente o realimentada.

Por otro lado existe lo que se llama topología de red donde se diferencian las redes *monocapa* y *multicapa*. Las redes *monocapa* poseen un solo nivel de pesos y pueden ser de propagación hacia adelante o recurrentes. Las redes multicapa poseen más de una capa de pesos y por lo menos una capa de neuronas ocultas. Estas pueden ser feedforward o feedback al igual que las monocapa. Las primeras se utilizan en la reconstrucción de datos parcialmente destruidos y las segundas en clasificación de patrones.

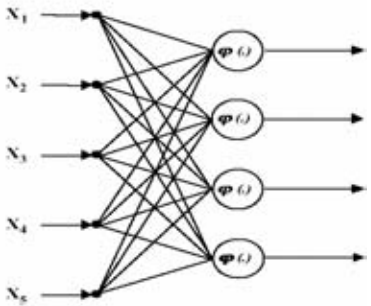


Fig.8 Red monocapa con propagación hacia adelante

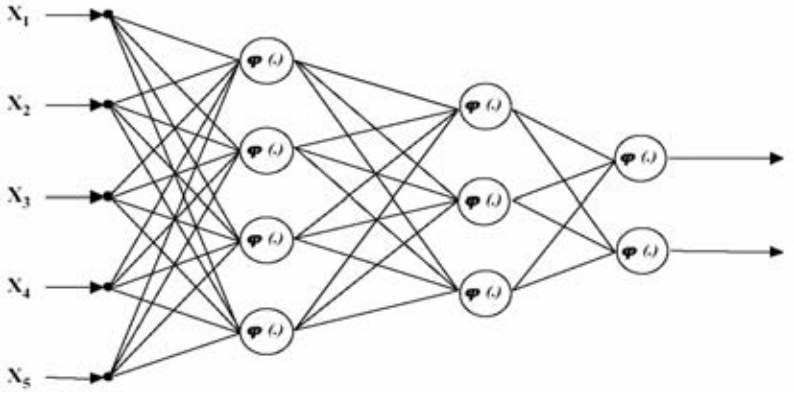


Fig.9 Red multicapa con propagación hacia delante

I.1.4 Aprendizaje de la RNA

La manera en que una red neuronal artificial ajusta los valores de sus pesos sinápticos se llama entrenamiento. Este puede ser *supervisado* o *no supervisado*.

En el *aprendizaje supervisado* el entrenamiento es realizado presentando una secuencia de vectores o patrones como ejemplo, cada uno asociado a un vector de salida correspondiente a la respuesta que se desea entregue la red. Así los pesos son ajustados de acuerdo al error generado y a los datos entregados en la entrada de la red mediante un algoritmo de aprendizaje. Los algoritmos de entrenamiento generalmente buscan minimizar una función de error. Una vez conseguido este objetivo se puede poner la red en funcionamiento o por un periodo de pruebas para verificar si efectivamente la red aprendió. En este tipo de aprendizaje se diferencian varias técnicas, entre ellas:

a. *Aprendizaje por corrección de error*: consiste en presentar al sistema un conjunto de pares de datos representando la entrada y la salida deseada para dicha entrada. Este conjunto recibe el nombre de conjunto de entrenamiento. En este grupo tenemos la regla del *Perceptrón,* la regla *Delta o de Mínimos Cuadrados Medios* y el algoritmo *Backpropagation*.

b. *Aprendizaje por Refuerzo*: este aprendizaje es más lento que el anterior, y en él no se dispone de un ejemplo completo del comportamiento deseado, es decir no se conoce la salida deseada exacta para cada entrada, pero si se sabe como debería de ser el comportamiento de manera general ante diferentes entradas. Es un aprendizaje ON LINE y ajusta sus pesos a través de un proceso de éxito o fracaso, por medio de la relación de entrada-salida, produciendo una señal de refuerzo que mide el funcionamiento del sistema. Entre los algoritmos de aprendizaje más importantes están *Linear Reward Penalty, Associative Reward Penalty* y el *Adaptative Heuristic Critic.*

c. *Aprendizaje Estocástico*: este tipo de aprendizaje consiste básicamente en realizar cambios aleatorios en los valores de los pesos y evaluar su efecto a partir del objetivo deseado y de distribuciones de probabilidad. Para esto se utiliza una función de energía de la cual el objetivo es encontrar el estado de equilibrio. Entre los más conocidos están el método de *Boltzmann* y el de *Cauchy*".

El *aprendizaje no supervisado* no busca aprender una respuesta correcta de acuerdo a un vector ejemplo ingresado en la red. Lo que hace es buscar relaciones entre los datos de entrada y de acuerdo a esto crear categorías, detectar características, regularidades, correlaciones o codificar los patrones ingresados una vez que estos se han estabilizado. En algunos casos estas realizan un mapeo de características, obteniéndose en las neuronas de salida una disposición geométrica que representa un mapa topográfico de las características de los datos de entrada. No necesitan influencia externa para ajustar los pesos de las conexiones. La red no recibe ninguna información por parte del entorno que le indique si la salida generada en respuesta a una determinada entrada es o no correcta; por ello se dice que estas redes tienen la capacidad de autoorganizarse.

De este tipo de aprendizaje se distinguen varios tipos de aprendizaje, entre los principales están:

a. Aprendizaje Asociativo: el objetivo de este aprendizaje es extraer características los datos que se introducen en la red o el grado de familiaridad de estos, de esta forma las informaciones similares son clasificadas formando parte de la misma categoría y por tanto deben activar la misma neurona de salida. Las clases o categorías deben ser creadas por la propia red, puesto que se trata de un aprendizaje no supervisado a través de las correlaciones entre los datos de entrada. De esta forma la red puede reconstruir datos incompletos o distorsionados.

b. Aprendizaje Competitivo: estas redes deben encontrar las características, regularidades, correlaciones o categorías que se puedan establecer entre los datos que se presenten en su entrada; puesto que no hay supervisor que indique a la red la respuesta que debe generar ante una entrada concreta, cabría preguntarse precisamente por lo que la red genera en estos casos, existen varias posibilidades en cuanto a la interpretación de la salida de estas redes que dependen de su estructura y del algoritmo de aprendizaje empleado. Entre los tipos de aprendizaje más importantes están el *"Winner take all" y el "Leaky Learning", "Learning Vector Quantizer"*.

I.1.5 Tipos de Redes Neuronales

I.1.5.1 Perceptrón Simple.

El Perceptrón de Rosenblatt o simplemente Perceptrón, se construye en base a una neurona con funcionamiento no lineal. Este modelo neuronal consiste de una combinación lineal seguida por una función de activación no lineal. Sean los pesos del perceptrón denotados por w_1, w_2,. . ., w_m, y las entradas aplicadas al perceptrón por x_1, x_2,. . ., x_m. El Perceptrón es un modelo unidireccional, compuesto por dos capas de neuronas, una sensorial o de entradas y otra de salida, y está diseñado para trabajar con entradas discretas. Las neuronas de entrada no realizan ningún cómputo, únicamente envían información. La función de activación de las neuronas de la capa de salida es de tipo escalón. Así la operación del Perceptrón con *n* neuronas en la capa de entrada y *m* en la capa de salida puede escribirse como:

$$y_i(t) = signo\left(\sum_j^n w_{ij} \cdot x_j - \theta_i\right), \forall i, 1 <= i <= m$$

$$signo(x) = \begin{cases} 1 & si\, x >= 0 \\ -1 & si\, x < 0 \end{cases}$$

El umbral externamente aplicado se denota por ***b***. A partir de este modelo se tiene que el potencial de activación de la neurona es

$$v = \sum_{i=1}^{m} w_i x_i + b.$$

El objetivo del perceptrón es clasificar correctamente el conjunto de estímulos externamente aplicados x_1, x_2,. . ., x_m en una de dos clases, C1 o C2. La regla de decisión para la clasificación es asignar el punto representado por las entradas *x*1, *x*2,. . ., *xm* a la clase C1 si la salida del perceptrón *y* es +1 y a la clase C2 si ésta es -1. Para visualizar el comportamiento del perceptrón como un clasificador de patrones, se puede realizar un mapa de las regiones de decisión en el espacio *m*-dimensional de la señal correspondiente a las *m* variables de entrada *x*1, *x*2,. . ., *xm*. En la forma más simple del perceptrón, existen dos regiones de decisión separadas por un *hiperplano* definido por:

$$\sum_{i=1}^{m} w_i x_i + b = 0.$$

Esto se ilustra en la figura 10 para el caso de dos variables de entrada *x*1 y *x*2, para las cuales la frontera de decisión toma la forma de una línea recta. Un punto (*x*1, *x*2) que cae sobre la línea de frontera se asigna a la clase C1, mientras que un punto (*x*1, *x*2) que cae debajo de esta línea se asigna a la clase C2. El efecto del umbral es el de trasladar la frontera de decisión fuera del origen

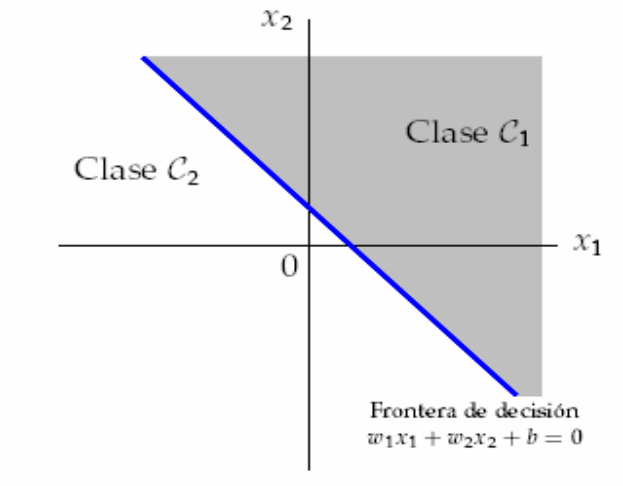


Fig.10 Hiperplano Perceptrón

Los pesos sinápticos *w*1, *w*2,. . ., *wm* del perceptrón pueden ser adaptados iterativamente. Para esta adaptación se usa una regla de corrección de error llamada el algoritmo de convergencia del perceptrón (Back Coupled Error Correction).

I.1.5.2 Perceptrones Multicapa.

La clase más popular de redes multicapa con alimentación hacia adelante son los perceptrones multicapa (en la figura 11 se muestra un perceptrón de dos capas ocultas y una de salida). En general, una red estándar de *L*-capas consiste de una capa de entrada, (*L* - 1) capas ocultas, y una capa de salida de neuronas todas sucesivamente conectadas (completamente o parcialmente) con conexiones de alimentación hacia adelante, pero sin enlaces entre unidades de la misma capa o enlaces retroalimentados entre capas.

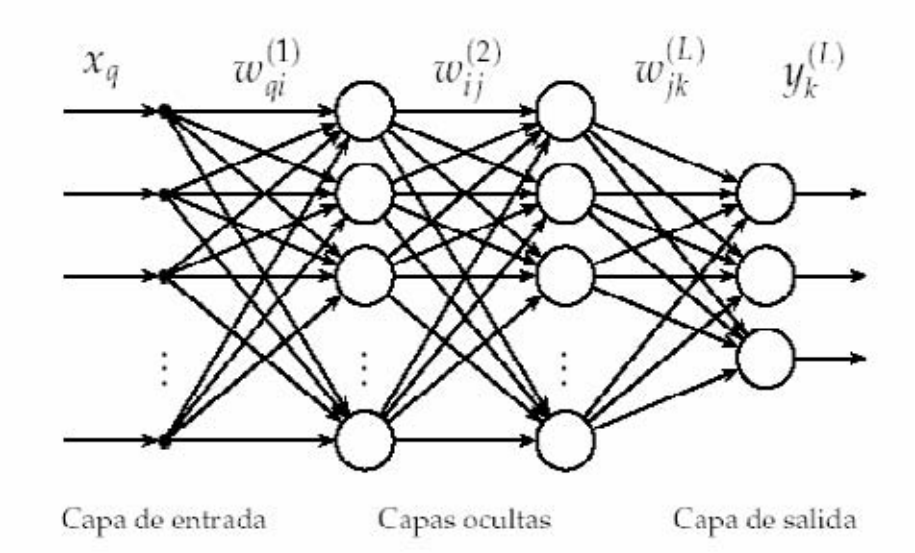


Fig.11 Perceptrón multicapa

En los perceptrones multicapa cada neurona emplea una función de activación la cual es generalmente una función sigmoide. Este tipo de redes puede formar arbitrariamente fronteras de decisión complejas y representar cualquier función boleana. Este algoritmo se basa en el principio de corrección de error.

I.1.5.3 El ADALINE

El ADALINE es otro modelo de red basado en un elemento de procesamiento similar al perceptrón. El término ADALINE es un acrónimo, inicialmente denominado ADAptive Linear Neuron que cambió a ADAptive LINear Element. Su estructura es similar al perceptrón, acepta entradas continuas y discretas, y tiene una salida lineal, sin embargo hay 2 modificaciones básicas con respecto al anterior. La primera modificación es la adición de una conexión con un peso ω_0, el cual se refiere a un término denominado *bias* con el fin de darle mayor grado de libertad al modelo. La segunda corresponde al algoritmo de entrenamiento. El ADALINE utiliza la regla de Mínimos Cuadrados, también conocida como la Regla Delta. Esta regla realiza una actualización de los pesos sinápticos de acuerdo a la sumatoria de errores cometido por todas las neuronas de la red.

I.1.5.4 El MADALINE

La Adaline tiene limitaciones semejantes al Perceptrón en cuanto a los tipos de funciones que pueden procesar, como es el caso de la funcion boleana OR exclusiva. Este problema se solucionó combinando varias Adalines, dando origen al MADALINE término que viene de Multiple Adaline.

I.1.5.5 La Máquina de Boltzmann

La máquina de Boltzmann es una máquina estocástica constituida por unidades de proceso (neuronas) estocásticas con conexiones sinápticas simétricas entre las mismas. Dichas unidades de proceso son de dos tipos funcionales: visibles u ocultas. Las unidades visibles sirven de interfaz entre la red y el entorno en el que opera, mientras que las unidades ocultas siempre operan libremente y se utilizan para aplicar restricciones subyacentes contenidas en los patrones de entrada.

La máquina de Boltzmann está basada en el principio físico del Templado. Este es un proceso por el cual un material es calentado y luego enfriado muy lentamente hasta su punto de congelación. En esta red este sistema es llamado Temple Simulado (Simulated Annealing).

I.1.5.6 Redes ART (Teoría de Resonancia Adaptativa)

Esta teoría se aplica a redes de aprendizaje competitivo donde una neurona de salida de la red se activa para alcanzar su valor de respuesta máximo después de competir con las otras. Esa neurona es llamada vencedora. El modelo ART posee un mecanismo de realimentación entre las neuronas competitivas de la capa de salida de la red y la capa de entrada. Este mecanismo facilita el aprendizaje de nueva información sin destruir la ya almacenada. Las redes ART1 pueden trabajar con valores de entrada binarios, mientras que ART2 es capaz de procesar información analógica.

La estructura de una red ART consta básicamente de dos capas entre las que se establecen conexiones hacia delante y hacia atrás. En el aprendizaje de las redes ART es de tipo ON LINE. Lo que se pretende es categorizar los datos que se introducen en la red, donde las clases o

categorías deben ser creadas por la propia red, puesto que se trata de un aprendizaje no supervisado, a través de las correlaciones entre los datos de entrada.

I.1.5.7 Modelo Hopfield

Funcionalmente son redes categorizadas como memorias autoasociativas, es decir que aprenden a reconstruir los patrones de entrada que memorizan durante el entrenamiento. Son monocapa con interconexión total, donde su primer modelo trabajaba solo con entradas y salidas binarias el cual extendió posteriormente para trabajar con entradas y salidas continuas y siguen una regla de aprendizaje no supervisado. Están formadas por N neuronas interconectadas que actualizan sus valores de activación en forma independiente. Todas son a la vez de entrada y salida.

La red de Hopfield no tiene una ley de aprendizaje asociada, esto significa que la red no es entrenada ni realiza un proceso de aprendizaje, sin embargo es posible determinar la matriz de pesos por medio de un procedimiento basado en la función de alta ganancia de Lyapunov.

El procedimiento consiste en escoger la matriz de pesos $\boldsymbol{W}$ y el vector de ganancias $\boldsymbol{b}$ tal que $\boldsymbol{V}$ toma la forma de la función que se quiere minimizar, convirtiendo el problema que se quiere resolver, en un problema de minimización cuadrática, puesto que la red de Hopfield minimizará a $\boldsymbol{V}$**.**

I.1.5.8 Modelo Kohonen

Esta red está basada en la capacidad que tiene el cerebro de formar mapas característicos de la información recibida del exterior. Esta red contiene solamente una capa de neuronas y una capa de entrada, que se ramifica para todos los nodos. Pertenece a la categoría de redes competitivas o mapas de autoorganización, es decir tiene un aprendizaje no supervisado. Tiene también funciones de activación lineales y flujo de información unidireccional (red en cascada). La red cuenta con N neuronas de entrada y M de salida, cada una de las neuronas de entrada está conectada a todas las de salida.

Las unidades de entrada reciben datos continuos normalizados. La red clasifica los patrones de entrada en grupos de características simulares, de tal manera que cada grupo activa siempre las mismas salidas. El aprendizaje en el modelo de *Kohonen* es de tipo Off-line, por lo que se distingue una etapa de aprendizaje y otra de funcionamiento.

I.1.5.9 La Red Backpropagation

Ahora veremos una de las redes más populares e importantes en la historia de las redes neuronales, la red *Backpropagation*. Es común referirse a la *Red Backpropagation*, pero en realidad a lo que se esta refiriendo es al algoritmo de *aprendizaje supervisado* que lleva este nombre para una determinada arquitectura de red. Al hablar de Redes *Backpropagation* entonces se está refiriendo implícitamente a una red con arquitectura feedforward, multicapa con algoritmo de aprendizaje *Backpropagation* basado en la corrección de error. Su funcionamiento será analizado con más detalle en las siguientes páginas ya que el *Sistema De Reconocimiento De Palabras* está basado en una **Red Backpropagation.**

I.1.5.9.1 Reseña

El desarrollo del algoritmo ***Backpropagation*** hizo retomar el interés por las redes neuronales en el mundo científico, pues la falta de métodos de entrenamiento apropiados para los perceptrones multicapa hizo que declinara el interés en las redes neuronales en los años 60 y 70.

Con este algoritmo se puso fin al pesimismo que sobre el campo de las redes neuronales se había puesto en 1969 con la aparición del libro de Minsky y Papert (Perceptrons) que demostraba las limitaciones del Perceptrón para separar datos no separables linealmente, como el clásico problema de la función boleana OR Exclusiva.

De forma simplificada, el funcionamiento de la red *Backpropagation* consiste en el aprendizaje de un conjunto predefinido de pares de entradas-salidas dados como ejemplo, empleando un ciclo propagación – adaptación de 2 etapas: durante la primera se aplica un patrón de entrada como estímulo para la primera capa de neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida, se compara el resultado obtenido en las

neuronas de salida con la salida que se desea obtener, y se calcula un valor del error para cada neurona de salida. A continuación, estos errores se transmiten hacia atrás, partiendo de la capa de salida, hacia todas las neuronas de la capa intermedia que contribuyeron directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia en la salida original.

Este proceso se repite capa por capa, hasta que todas las neuronas de la red hayan recibido un error que describa su aportación relativa al error total. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cerca de la deseada; es decir, el error disminuya.

La importancia de la red *Backpropagation* consiste en su capacidad de autoadaptar los pesos de las neuronas de las capas intermedias para *aprender* la relación que existe entre un conjunto de patrones dados como ejemplo y sus salidas correspondientes. Después del entrenamiento, puede aplicar esta misma relación a nuevos vectores de entrada con ruido o incompletas, dando una salida activa si la nueva entrada es parecida a las presentadas durante el aprendizaje.

Esta característica importante, que se exige a los sistemas de aprendizaje, es la capacidad de *generalización,* entendida como la facilidad de dar respuestas satisfactorias para entradas que el sistema no ha visto nunca en su fase de entrenamiento. La red debe encontrar una *representación interna* que permita generar las salidas deseadas cuando se le dan las entradas de entrenamiento, y que pueda aplicar, además, a entradas no presentadas durante la etapa de aprendizaje para clasificarlas según las características que compartan con los ejemplos de entrenamiento.

I.1.5.9.2 Funcionamiento del Algoritmo

El algoritmo ***Backpropagation*** es conocido también como la Regla Delta Generalizada, debido a su parecido con la Regla Delta utilizada para el entrenamiento de la Red Adaline. La

condición que deben cumplir las funciones de activación de las neuronas es que deben ser funciones continuas, derivables y no decrecientes, a diferencia de la función escalón utilizada por el perceptrón que no es derivable en su punto de discontinuidad. Este algoritmo utiliza también una función o superficie de error asociada a la red, buscando el estado estable de mínimo error a través del camino descendente de la superficie del error. Por ello, realimenta el error del sistema para realizar la modificación de los pesos en un valor proporcional al gradiente decreciente de dicha función de error.

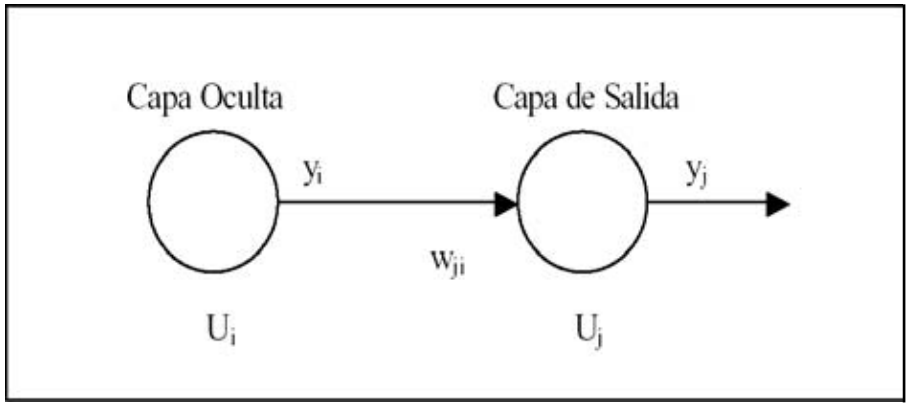


Fig.12 Conexión entre una neurona de una capa oculta con una neurona de salida

El método que sigue el algoritmo *Backpropagation* para ajustar los pesos es actualizarlos de forma proporcional a la delta o diferencia entre la salida deseada y la obtenida. Dada una neurona U_i y la salida que produce, y_i, el cambio que se produce en el peso de la conexión que une la salida de dicha neurona con la unidad $U_j(w_{ji})$ para un patrón de aprendizaje p determinado es:

$$\Delta w_{ji}(t+1) = \alpha \ \delta_{p\,j}\, y_{pi}$$

donde el subíndice p se refiere al patrón de aprendizaje concreto y $\boldsymbol{\alpha}$ es la constante o tasa de aprendizaje. El punto en que difieren la regla delta generalizada de la regla delta es en el valor concreto de δ_{pj}. Por otro lado, en las redes multinivel a diferencia de las redes sin neuronas ocultas, en principio no se puede conocer la salida deseada de las neuronas de las capas ocultas para poder determinar los pesos en función del error cometido. Sin embargo, inicialmente sí podemos conocer la salida deseada de las neuronas de salida. Según esto, si consideramos la unidad U_j de salida, entonces definimos la siguiente ecuación,

$$\delta_{pj} = (d_{pj} - y_{pj}) \cdot f'(net_j)$$

donde d_{pj} es la salida deseada de la neurona j para el patrón p, y_{pj} es su salida real y net_j es la entrada neta que recibe la neurona j.

Esta fórmula es similar a la regla delta, excepto en lo que se refiere a la derivada de la función de transferencia. Este término representa la modificación que hay que realizar en la entrada que recibe la neurona *j*. En el caso en que dicha neurona no sea de salida, el error que se produce estará en función del error que se cometa en las neuronas que reciban como entrada la salida de dicha neurona. Esto es lo que se denomina el procedimiento de propagación del error hacia atrás.

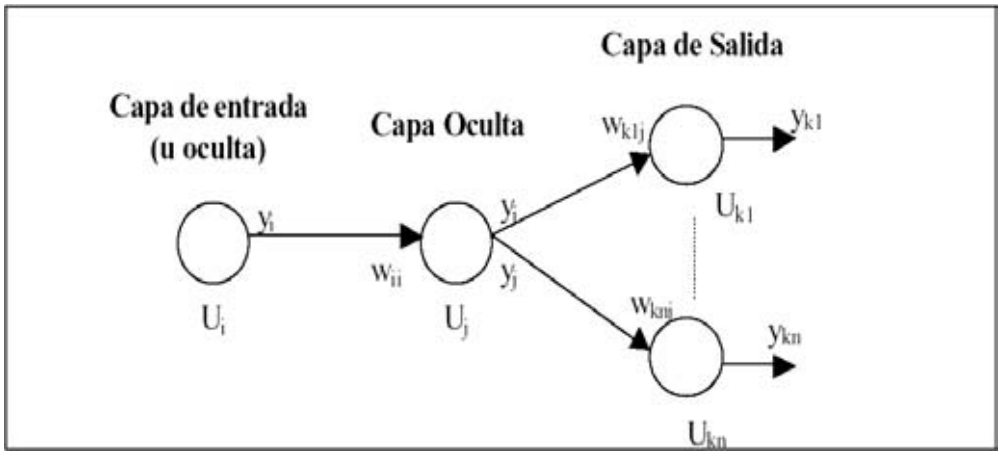


Fig.13 Conexiones entre neuronas de capa oculta y de salida

Según esto, en el caso de que U_j *no sea una neurona de salida*, el error que se produce está en función del error que se comete en las neuronas que reciben como entrada la salida de U_j de acuerdo a la siguiente ecuación,

$$\delta_{pj} = (\sum_k \delta_{pk} \, w_{kj}) \bullet f'(net_j)$$

donde el rango de k cubre todas aquellas neuronas a las que está conectada la salida de U_j. De esta forma, el error que se produce en una neurona oculta es la suma de los errores que se producen en las neuronas a las que está conectada la salida de ésta, multiplicando cada uno de

ellos por el peso de la conexión. Con esto el nuevo valor de los pesos en las conexiones entre neuronas es:

$$w_{ji}(t+1) = w_{ji}(t) + \alpha\, \delta_{pj}\, y_{pi}$$

El problema que presenta este algoritmo es que su entrenamiento resulta demasiado lento por lo que se han propuesto modificaciones para hacerlo más rápido. Este algoritmo provee una "aproximación" a la trayectoria en el espacio de pesos calculada por el método del gradiente descendiente. Así mientras más pequeño hacemos el parámetro α (tasa de aprendizaje), más pequeños serán los cambios a los pesos en la red de una iteración a la siguiente, y más suave será la trayectoria en el espacio de pesos. Esta mejora, sin embargo, se alcanza a costa de un aprendizaje más lento. Si, por el contrario, hacemos el parámetro α demasiado grande de manera de acelerar el aprendizaje, los grandes cambios resultantes en los pesos asumirán una forma en la cual la red puede volverse inestable y no ser capaz de encontrar un mínimo oscilando indefinidamente alrededor de uno. El la figura 14 se ilustra este hecho.

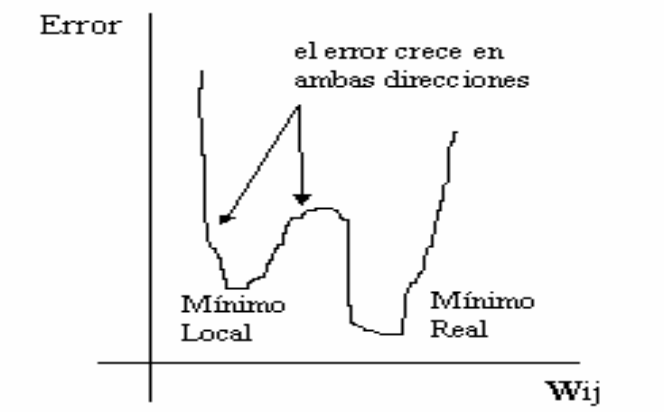


Fig.14 Función de Error de una Red Neuronal.

Un método sencillo de incrementar la tasa de aprendizaje y al mismo tiempo evitar el riesgo de inestabilidad es modificar a la Regla Delta Generalizada incluyendo el término *Momento* β de manera que la expresión de actualización de los pesos quede:

$$w_{ji}(t+1) = w_{ji}(t) + \alpha\, \delta_{pj}\, y_{pi} + \beta\,(w_{ji}(t) - w_i(t-1))$$

donde β es una constante (momento) que determina el efecto en *t+1* del cambio de los pesos mediante el instante *t.*

Con este *momento* se consigue la convergencia de la red en menor número de iteraciones, ya que si en *t* el incremento de un peso era positivo y en *t+1* también, entonces el descenso por la superficie de error en *t+1* es mayor. Sin embargo, si en *t* el incremento era positivo y en *t+1* es negativo, el paso que se da en *t+1* es más pequeño, lo cual es adecuado, ya que esto significa que ha pasado por un mínimo, y que los pasos deben ser menores para poder alcanzarlo.

I.2 El problema de reconocer palabras

Para atacar el problema del reconocimiento de palabras, se va a enfocar éste desde el punto de vista acústico, esto significa que sólo se analizará el habla como una señal analógica que produce un flujo continuo de ondas sonoras, dejando de lado la articulación y la percepción auditiva. De esta forma se tiene una señal variante en el tiempo, tanto en amplitud como en frecuencia. Esta señal contiene información imprecisa y distorsionada, debido a que es imposible pronunciar exactamente igual una misma palabra, sin embargo se conservan rasgos característicos de ésta que la distinguen de otras lo que se denomina *patrón*. Es aquí donde nos encontramos con el problema, ya que la arquitectura de los computadores actuales (Von Neumann) es eficiente en procesar información precisa y secuencial mediante algoritmos, pero para la interpretación de una señal de voz que no puede ser modelada por la matemática convencional e imprevisible en su forma, se muestran ineficientes o inoperantes. Así el procesamiento secuencial es incapaz de interpretar información que es imprecisa y que debe ser tratada en su conjunto, es decir se necesita de un sistema que procese información de una vez como un todo, esto es procesamiento en paralelo.

Considerando que no se puede procesar con un mismo algoritmo una información que es distinta cada vez que se produce, que posee imprecisiones, pero que guarda similitudes que la hacen entendible si se analiza en su totalidad. Por ejemplo si se quisiera escribir un programa que detecte la palabra “hola”. Una vez digitalizada la señal de voz de ésta palabra se convierte en

un conjunto de valores, entonces el programa debería ser escrito en función de esos valores para utilizarlos de referencia para detectar cuando nuevamente se pronuncie la palabra *hola* y nos entregue el resultado.

Como se pudo comprobar en el programa Cool Edit 2 Pro, aún pronunciando repetidamente la misma palabra, en similares condiciones ambientales, el resultado de su digitalización generó ***siempre*** valores distintos para cada vez que se pronunció la palabra "hola" manteniendo la similitud en su forma. Entonces si se vuelve a pronunciar la palabra hola, generándose nuevos valores en el proceso de digitalización, el programa, al comparar éstos valores con los utilizados de referencia será incapaz de decidir si la palabra pronunciada es o no la que se busca reconocer, ya que los parámetros de comparación utilizados son ***fijos*** y no pueden ser modificados y los que se presenten posteriormente serán inevitablemente *distintos*. De esta forma no se puede pensar en reconocer 2 o más palabras con este sistema que se muestra inapropiado para esta labor. Considerando que la naturaleza de la voz es imprecisa y con distorsiones, sin pensar en los errores que incorpora el hardware durante el proceso de digitalización, la computación tradicional es incapaz de ofrecer una solución a un problema de reconocimiento de patrones. Con todo esto las Redes Neuronales Artificiales surgen como una solución al reconocimiento de patrones, ya que presentan características tales como, procesamiento de información imprecisa y en paralelo, capacidad de generalización, y adaptación.

I.3 Elección del tipo de Red

La red *Backpropagation* surge como un tipo de red apropiado para el desarrollo del sistema de reconocimiento de palabras, ya que ésta posee la capacidad de clasificación y reconocimiento de patrones. Su entrenamiento es relativamente rápido, así como su puesta en marcha. Posee la capacidad de generalización que es proveer salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento. Y es justamente esta virtud la que es imprescindible, ya que durante la etapa de funcionamiento el sistema solo procesará información imposible de reproducir en forma idéntica entre una repetición y otra como ya se vio que es la voz humana.

CAPÍTULO II. IMPLEMENTACIÓN DE LA RED NEURONAL

La implementación de la red neuronal *Backpropagation* se realizó en **Visual Basic 6.0**, esto por las siguientes razones:

- el algoritmo *Backpropagation* es relativamente sencillo, constando este de funciones conocidas y sumatorias que pueden ser fácilmente implementadas en Visual Basic 6.0.
- para el desarrollo del entorno gráfico, Visual Basic 6.0 es especialmente apropiado, ya que es un lenguaje de programación visual en el que no es necesario escribir código para generar una interfaz gráfica, y por esto su diseño se hace más fácil, liberando de la tarea de escribir complejo código para conseguir un interfaz amigable y sencilla. Este lenguaje deriva del Basic y por tal su sencillez tanto de su aprendizaje como su utilización.

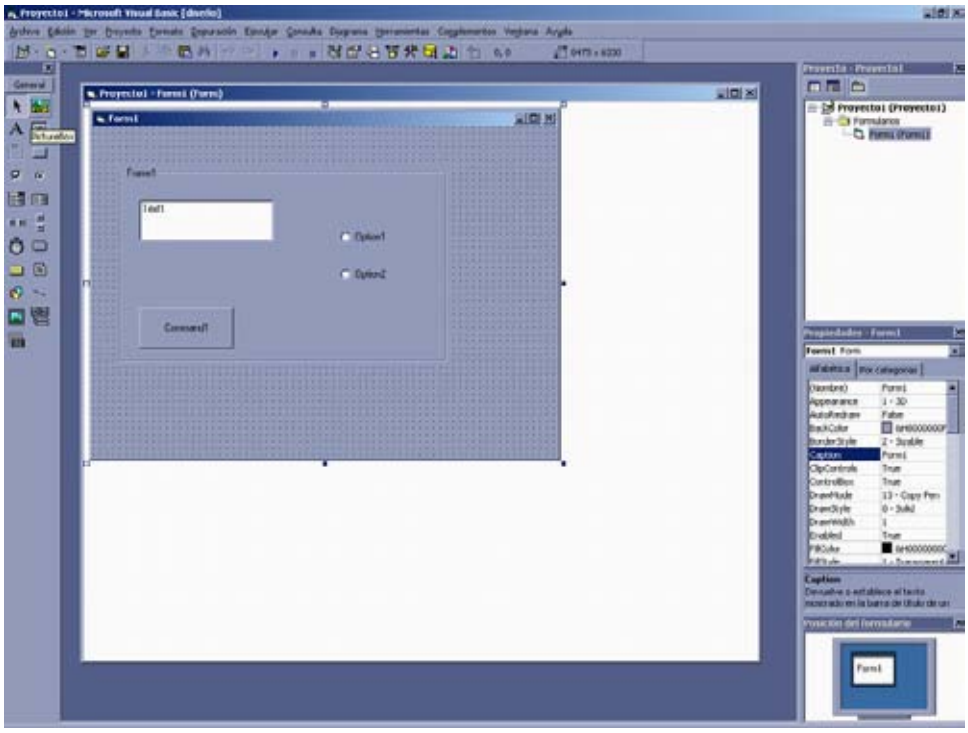


Fig.15 Entorno de Visual Basic 6.0

II.1 Escribiendo el algoritmo Backpropagation

Como se vio anteriormente el algoritmo *BP* es relativamente sencillo de escribir en un lenguaje de programación ya que este consta de sumatorias y funciones como la exponencial que están incluidas en la mayoría de los lenguajes modernos. Visual Basic 6.0 cuenta con funciones matemáticas que permiten su implementación fácilmente. A continuación se verá la forma en que se implementó el algoritmo BP.

El algoritmo consta de 2 etapas, una hacia delante donde se genera la salida de la red y la otra hacia atrás, donde, a partir de la salida generada y el error cometido se produce la retropropagación del error medio cuadrado, modificando el valor de los pesos sinápticos.

En la primera etapa tenemos las siguientes funciones a implementar:

$$Net_k = \sum_{j=1}^{m} w_{kj} x_j + \theta_{k0}$$

Esta función corresponde a la sumatoria de los pesos que recibe la neurona *k* que puede pertenecer a cualquier capa, incluyendo el valor umbral. De acuerdo a esto el código escrito fue el siguiente:

```
x1(0) = -1       se inicializa el umbral con -1 fijo
For i = 1 To n°nX    por cada una de las neuronas de la capa de entrada
For j = 0 To n°nX   por cada uno de los pesos desde el umbral hasta el nºnX
netE(i) = netE(i) + x1(j) * we(i, j) se calcula la suma de las entradas de la neurona
Next j
Next i
```

En este código tenemos ciclos *for* anidados. El primer ciclo *for* se repite para cada una de las neuronas de la capa *n* y el segundo ciclo *for* calcula la suma de las entradas multiplicadas por

los pesos sinápticos para cada neurona de la capa *n*. Esto se realiza hasta *nºnX* que corresponde al número de neuronas de una capa. En otras palabras se calcula la suma de las entradas por los pesos sinápticos de cada neurona y esto se repite para todas las neuronas de una misma capa obteniéndose el valor de Net_k. La variable *netE(i)* contiene todos los valores de las sumatorias de las entradas de la capa multiplicadas por sus respectivos pesos y la variable *we(i,j)* es la matriz de pesos la primera capa. Esto es válido para todas las capas de la red.

$$\varphi(Net_k) = \frac{1}{1 + e^{-\alpha Net_k}}$$

La segunda función a implementar es la función de activación de la neurona (arriba) denominada sigmoide. Como se aprecia, es bastante simple, y el código escrito para ella fue el siguiente.

```
For i = 1 To n°nX  por cada salida de la capa de entrada
   ye(i) = 1 / (1 + Exp(-netE(i))) se calcula el valor de activación de la neurona
Next i
```

Aquí se calcula el valor de la función de activación a partir del cálculo del valor de la función *Net()* vista previamente, obteniéndose una función sigmoide que genera una salida entre 0 y 1. El ciclo *for* repite este cálculo para cada una de las neuronas de la capa *n*, hasta nºnX, generándose así las salidas de esta capa que alimentarán a la capa siguiente. Se tiene entonces que la variable *ye(i)* contiene todos los valores de activación de las neuronas correspondientes a una capa. Esto es válido para todas las capas.

Una vez que se llega a la salida de la red, de acuerdo al algoritmo se debe calcular el error de la red, aquí empieza la segunda etapa donde se tiene:

$$e_j(n) = d_j(n) - y_j(n)$$

donde e_j corresponde al error en una neurona de salida e y_j es la salida real y d_j la salida deseada y por lo tanto el error a la salida.

El algoritmo BP se basa en hacer una repartición del error que se genera en la salida de la red hacia todos los pesos de las capas previas, con lo que el valor de la actualización de los pesos se calcula de acuerdo a la siguiente ecuación:

$$w_{ji}(t+1) = w_{ji}(t) + \alpha\, \delta_{pj}\, y_{pi} + \beta\,(w_{ji}(t) - w_i(t-1))$$

donde α y β son los coeficientes de aprendizaje y momento respectivamente y el gradiente δ se divide en dos funciones, una cuando la neurona es de salida y otra para las demás capas, ya sean ocultas o de entrada, de acuerdo a esto los δ quedan así:

$$\delta_{pj} = (d_{pj} - y_{pj})\ .\ f'(net_j)$$

cuando es una capa de salida. Notar que en el caso de la función sigmoide, su derivada puede ser expresada en términos de la función original. De acuerdo a esto se tiene lo siguiente:

$$f(Net) = \frac{1}{1+e^{-Net}}$$

$$f'(Net) = \frac{e^{-Net}}{(1+e^{-Net})^2}$$

arreglando la derivada se obtiene:

$$f'(Net) = \frac{1+e^{-Net}-1}{(1+e^{-Net})^2}$$

$$f'(Net) = \frac{1}{1+e^{-Net}}\left(1 - \frac{1}{1+e^{-Net}}\right)$$

$$f'(Net) = f(Net)(1 - f(Net))$$

En este caso se tiene el siguiente código:

For i = 1 To n°n3
*sigs(i) = (des(i) – ys(i)) * ys(i) * (1 – ys(i)) se obtienen los valores del gradiente de cada neurona de salida*
Next i

En el código se calcula el valor del gradiente para una neurona de la capa de salida de acuerdo al error entre la salida deseada *des(i)* y la salida real *ys(i)* multiplicado por la derivada de la función de activación que está expresada en términos de su función primitiva y se guarda en la variable *sigs(i)*. De acuerdo al ciclo *for*, este cálculo se repite desde la neurona 1 hasta *nºn3*, es decir para todas las neuronas de salida.

Cuando es una neurona de cualquier otra capa que no sea una de salida se tiene la siguiente ecuación para calcular el gradiente:

$$\delta_{pj} = (\sum_{k} \delta_{pk}\, w_{kj}) \bullet f'(net_j)$$

El código para esto es el que sigue:

For i = 1 To n°n2 una vez por cada neurona de la capa oculta
For j = 1 To n°n3
*sig1(i) = sig1(i) + sigs(j) * ws(j, i) suma de los sigma de la salida por cada peso*
Next j
Next i

Nuevamente se tiene dos ciclos *for* anidados. El interno tiene por finalidad realizar la multiplicación de los gradientes de la capa anterior por el peso que los conecta, sumarlos y guardarlos en la variable *sig1(i)*. El segundo repite el cálculo anterior para todas las neuronas de una capa que no es de salida. Con estas funciones calculadas se pueden obtener los valores de las actualizaciones de los pesos de toda la red, representados en la siguiente ecuación:

$$w_{ji}(t+1) = w_{ji}(t) + \alpha\, \delta_{pj}\, y_{pi} + \beta\,(w_{ji}(t) - w_i(t-1))$$

Para esta ecuación se implementó el siguiente código:

```
For i = 1 To n°n1          se realiza el calculo para cada una de las neuronas de entrada
   For j = 0 To n°n1       se calculan los pesos por las 35 entradas + el umbral
   we_f(i, j) = we(i, j) + alfa * sige(i) * x1(j) + Beta * (we(i, j) – we_p(i, j))
Next j,i
```

Otra vez dos ciclos *for* anidados, el interno es para calcular el nuevo valor de los *nºnX* pesos de una neurona en una capa cualquiera. Se tiene el parámetro α, el gradiente δ, y el momento β involucrados en el valor futuro del peso *we(i,j)*. El ciclo *for* externo se encarga de ejecutar el mismo cálculo para todas las neuronas de una capa cualquiera. Es en esta parte donde se produce la *retropropagación del error* hacia las capas previas a la salida y se finaliza con la implementación del algoritmo.

II.2 Pre-procesamiento de la Señal de Voz

Una señal de voz contiene mucha información, que aún al digitalizarla es excesiva para el procesamiento directo por la red neuronal. Por ejemplo muestreando a 44.1 khz se tienen 44100 muestras para un segundo de voz. Eso significaría que la red neuronal debería tener 44100 neuronas de entrada, lo que provocaría una reducción en la velocidad de reconocimiento y eficiencia del sistema.

Todo sistema de reconocimiento de patrones consta de 3 etapas:

- digitalización de la señal,
- extracción de características y,
- reconocimiento de patrones.

Para digitalizar la voz se utilizó una tarjeta de sonido Intel, con una velocidad de muestreo de 44.1 khz, 16 bits, monofónico.

Cómo método de extracción de características se utilizó ***La Transformada De Fourier***, ya que ésta entrega rasgos espectrales característicos de una señal, eliminando datos redundantes y reduciendo la cantidad de información útil necesaria, lo que se traduce en una importante disminución en las dimensiones de la Red Neuronal, optimando la cantidad de cálculos necesarios y aumentando la velocidad de respuesta del sistema. Para entrenar la red se utilizaron 128 componentes de frecuencia desde 43 Hz hasta los 5,5 kHz. Esto se debe a que la voz humana puede limitarse hasta 4 khz y ser aún reconocible su contenido.

Los valores de frecuencia de las demás componentes pueden ser calculadas de acuerdo a la siguiente fórmula:

$$frecuencia(Hz) = \frac{TasaDeMuestreo * i}{n}$$

Donde:

Tasa de muestreo = 44100 Hz

i = subíndice (i=1,2,3,…n/2)

n = número de puntos para el cálculo de la fft.

CAPÍTULO III. ENTRENAMIENTO DE LA RED NEURONAL ARTIFICIAL

Una vez digitalizada la voz y extraídas sus características espectrales se generaron los ejemplos de entrenamiento. El conjunto de palabras que se busca reconocer es: TOMA, DEJA, PARA, DERÉ (para referirse a la derecha) e IZKÉ (para referirse a la izquierda). El objetivo es poder controlar con estas 5 palabras los movimientos de un brazo mecánico y las 8 1as. letras para la visualización en el display.

Para generar los ejemplos se utilizó la tarjeta de sonido Intel montada en un computador con procesador Celeron a 800 Mhz, frecuencia de muestreo de 44.1 khz, 16 bits, monofónico, y un micrófono unidireccional marca Panasonic de 600Ω de impedancia y ganancia de XX dB.

El proceso de generación de los ejemplos se realizó sobre un pequeño programa implementado en Visual Basic 6.0 que calculaba la *Transformada Rápida De Fourier* de la señal digitalizada por la tarjeta de sonido. Se calculó la FFT con 1024 muestras, lo que se traduce en coger trozos de 23 milisegundos de señal donde se la puede considerar estacionaria. Este programa entregaba un archivo de texto con 128 componentes de frecuencia. Así se pronunció 100 veces cada palabra, de las cuales solo se utilizaron 20 ejemplos que tenían mayor similitud en sus respectivas formas de onda. Este análisis se realizó a mano mediante el programa Excel donde se graficaron las formas de onda digitalizadas por la tarjeta de sonido y se eligieron las que tenían mayor relación para cada palabra. Esto se explica porque en algunos casos los espectros obtenidos para una misma palabra eran "muy" distintos, por lo que para la red se volvía difícil vincular los patrones entre ellos generando inestabilidad y un error durante el entrenamiento e impidiendo una discriminación adecuada de la red durante la etapa de funcionamiento.

Una vez listos los 20 ejemplos de cada palabra estos se mezclaron y se aumentaron duplicándolos hasta conseguir una cantidad suficiente que permitiese iterar a la red hasta su convergencia.

III.1 Inicialización de los Pesos Sinápticos

Este ítem es de gran importancia puesto que una elección errónea del valor de los pesos sinápticos de la red puede provocar que ésta no logre aprender. Para evitar esto se utilizó la fórmula de inicialización de pesos dada por: $\pm\frac{1}{\sqrt{Fi}}$, donde Fi es el fan in o número de neuronas de entrada de la red. De esta forma se asegura que los pesos no estén saturados y la red pueda aprender.

La implementación de esta fórmula en Visual Basic fue la siguiente:

```
For i = 1 To n°n1          de la 1 neurona hasta la nºn1
   For j = 0 To n°n1       se asigna un valor aleatorio a todos los pesos partiendo de
   Randomize          'we0 y hasta we128
   a = Rnd
   we(i, j) = (Rnd * 1 / Sqr(n°n1)) * (-1) ^ (Int((2 – 1 + 1) * Rnd + 1))
   we_f(i, j) = 0
   we_p(i, j) = 0
   Next j
Next i
```

Primero se genera un valor aleatorio que es guardado en la variable "a", luego se aplica la fórmula de inicialización de los pesos, y se multiplica por un valor que oscila al azar entre -1 y 1 para poder generar un cambio de signo aleatorio y conseguir una distribución uniforme de los valores de los pesos a través de su superficie. Esto se realiza en una neurona para todos los pesos a los que está conectada por medio del ciclo *for* interno, y para todas las neuronas de una capa gracias al ciclo *for* externo.

Con la función de inicialización implementada se asegura el correcto funcionamiento del algoritmo de aprendizaje de la red, pero no se garantiza su convergencia.

III.2 Afinando detalles

Antes de ingresar los ejemplos a la red fue necesario considerar algunos detalles con respecto al funcionamiento de la misma. Como la naturaleza del habla humana es aleatoria, aún pronunciando la misma palabra en condiciones similares, su amplitud y forma siempre presenta variaciones que podrían provocar un comportamiento inesperado durante la fase de aprendizaje de la red, se decidió normalizar los valores de las componentes de frecuencia ingresadas a la red con el fin de presentar amplitudes homogéneas ante las mismas palabras, evitando las diferencias que pudieran generarse al momento de digitalizar los comandos de voz. Esto se realizó implementando el siguiente código:

```
kk = 2
 a = aux(ii)
      Do While kk <= 128
      b = aux(kk)
        If a > b Then          pregunta si a es mayor que b, si es asi pasa al siguiente
             kk = kk + 1            valor del arreglo, conservando el valor de a
        Else                      incrementa en 1 el contador para pasar al siguiente
     a = b                  elemento del arreglo
    kk = kk + 1
       End If
   Loop                 si no es mayor guarda el valor de b en a y aumenta en uno el
                          contador para ir a la siguiente posición del arreglo
For k = 1 To 128
   x_(k) = aux(k) / a
Next k
```

El bloque *Do While* tiene por finalidad determinar el mayor de los 128 componentes de frecuencia almacenados en la variable *aux()*. Los valores normalizados son guardados en *x_(k)*.

III.3 Conformación de los ejemplos de entrenamiento

Una vez realizados estos ajustes se ejecutó el entrenamiento de la red. Recordando el set de palabras que se desea aprenda la red para el control del brazo mecánico, este consta de 5 comandos y estos son: TOMA, DEJA, PARA, DERÉ e IZKÉ (las últimas dos abreviaturas de Derecha e Izquierda respectivamente).

Como se puede apreciar se eligió un conjunto de palabras de 4 letras cada una. Se dividieron las palabras de a 4 letras con el fin de reconocerlas por cada letra. De este modo se formaron conjuntos de entrenamiento con cada primera letra de las 5 palabras, cada segunda letra y así sucesivamente hasta completar el conjunto. Esto se explica así: se creó un conjunto de aprendizaje con las primeras letras de los comandos T, D, P e I, en este caso sólo 4 letras ya que la D está repetida. Se hizo lo mismo con las segundas letras, O, E, A y Z, donde también está repetida la letra E. Para el caso de las terceras letras se formó un conjunto con M, J, R y K, donde nuevamente se repite una letra, la R y finalmente se formó un conjunto con solo 2 letras A y E, ya que estás están presentes en 3 y 2 comandos respectivamente:

1ºgrupo: T, D, P, I
2ºgrupo: O, E, A, Z
3ºgrupo: M, J, R, K
4ºgrupo: A, E

La FFT se calculó con 1024 puntos de la señal PCM con lo que se tomaron trozos de 23 ms, tiempo en el cual se puede considerar la señal estacionaria.

Para la operación del display de 7 segmentos se utilizó un set de 8 palabras con un máximo de 3 letras. En este caso el objetivo fue visualizar en el display las letras pronunciadas. El set está conformado por las letras A, Be, Ce, De, E, eFe, Ge y acHe, tal como se pronuncian. Los grupos se conformaron así:

1º grupo: A_1, B_1, C_1, D_1, E_1, y G_1

2º grupo: A_2,B_2,C_2,E_2, F_2,G_2, CH_2

3º grupo: A_3,E_3

Con los ejemplos listos se generaron 4 archivos de entrenamiento para cada grupo de letras. En el caso de los comandos de control del brazo mecánico estos fueron denominados: DITDP1dES.txt, DITDP2dES.txt, DITDP3dES.txt, DITDP4dES.txt y 3 archivos para los 3 grupos de letras en el caso del display llamados abc1dES, abc2dES y abc3dES. Se realizaron distintas pruebas para ajustar el aprendizaje de la red debido a que no existen reglas que aseguren un resultado óptimo. Se modificó el Nº neuronas de la capa oculta, y se observó que aproximadamente desde 40 neuronas hasta 128 neuronas el resultado del entrenamiento no variaba en cuanto al número de ejemplos que utilizaba la red en empezar a estabilizarse, pero si se percató de una mayor demora en las iteraciones en el caso de 128 neuronas en la capa oculta, por lo que se fijó el Nº de neuronas en esta oculta en 40 para optimizar la velocidad de entrenamiento. Luego con este valor fijo se variaron α y β. En primer lugar se realizaron las pruebas para los comandos del brazo y luego las letras del display en ambos casos para cada grupo de letras por separado. Los resultados de las pruebas se pueden ver a continuación:

Se escogieron al azar los valores de α y β en 0.5 y 0.7 respectivamente:

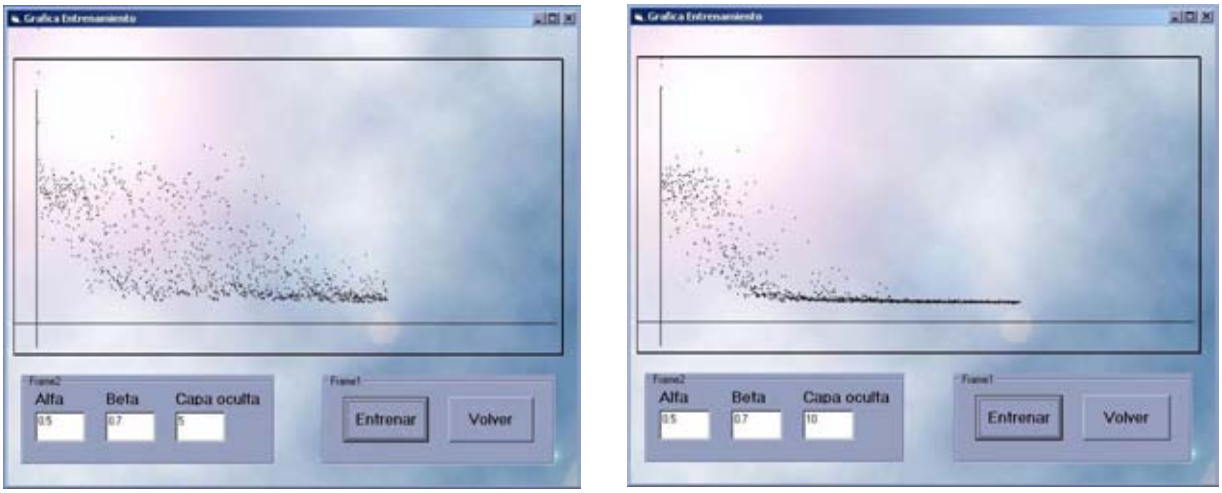
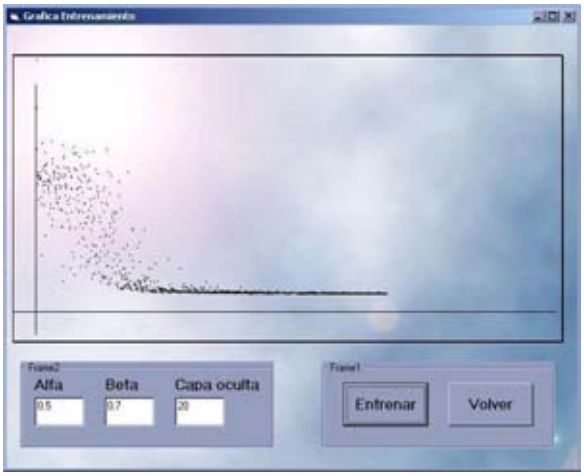


Fig.16 y 17



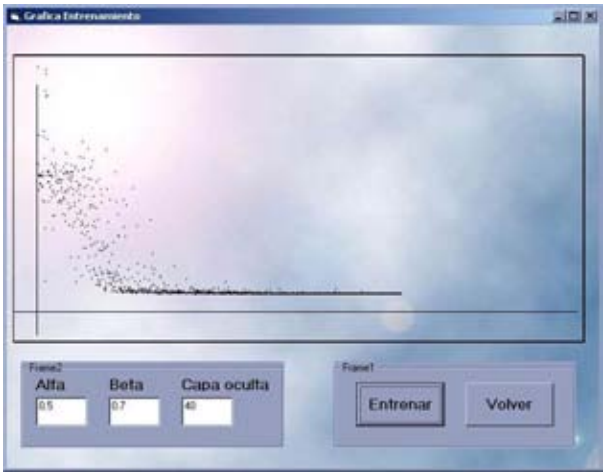
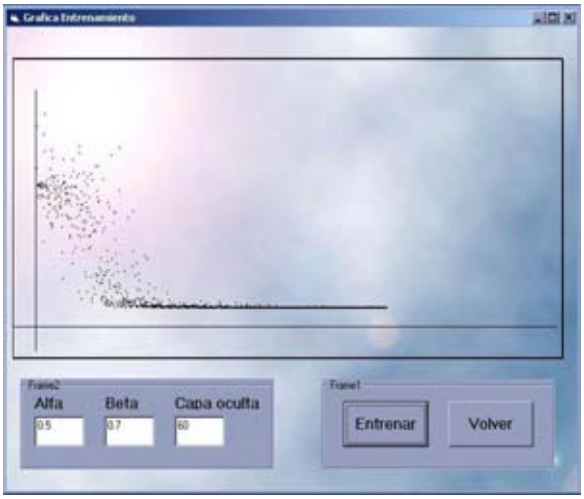


Fig. 18 y 19



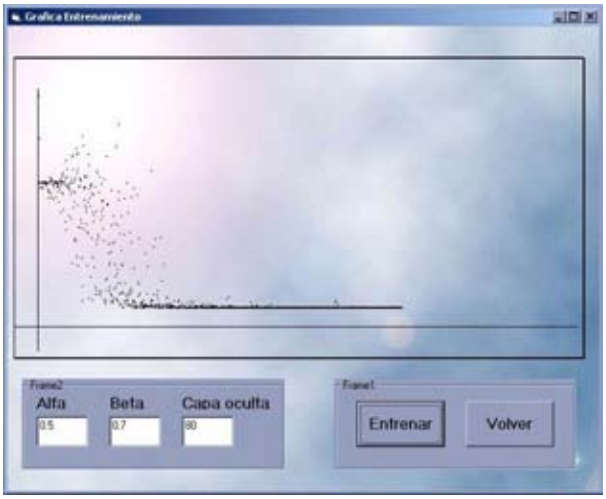
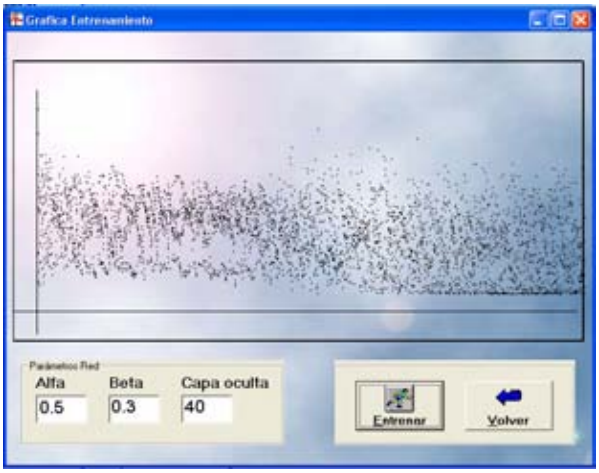


Fig. 20 y 21

Fig. 22

Se puede apreciar que se obtiene el mismo resultado tanto con 40 neuronas en la capa oculta como con otra cantidad mayor. Con el fin de minimizar las operaciones a calcular por el programa se decidió trabajar con 40 neuronas en la capa oculta. Con este valor constante se modificaron los demás parámetros de la red para todos los set de entrenamiento.

1er. Grupo: T,D,P,I.



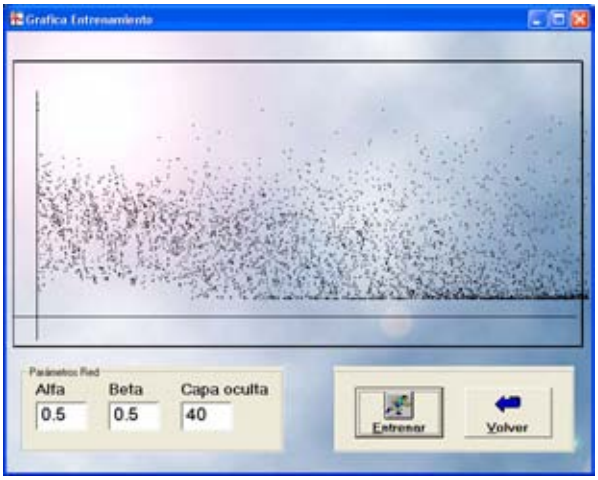
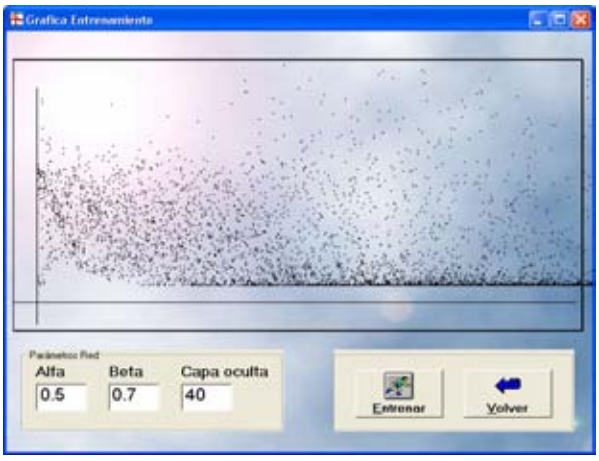


Fig.23 y 24



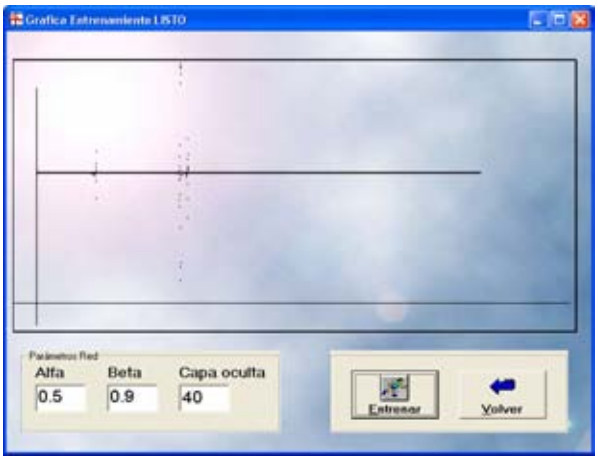
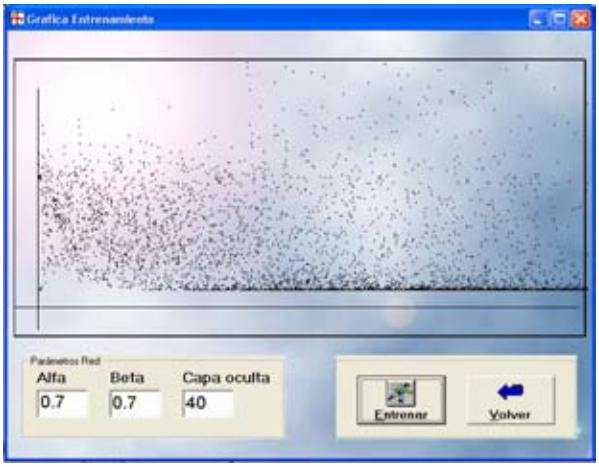


Fig.25 y 26



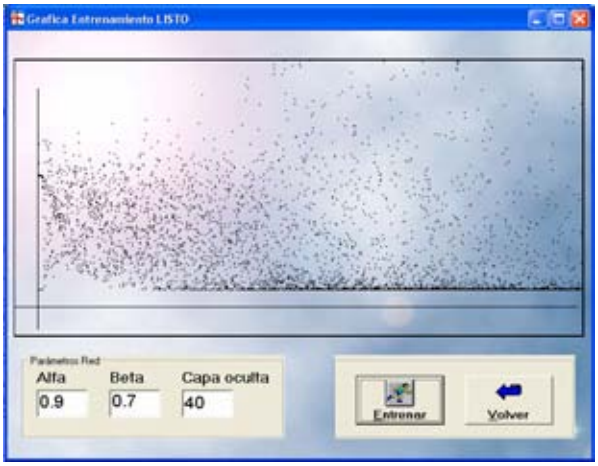


Fig.27 y 28

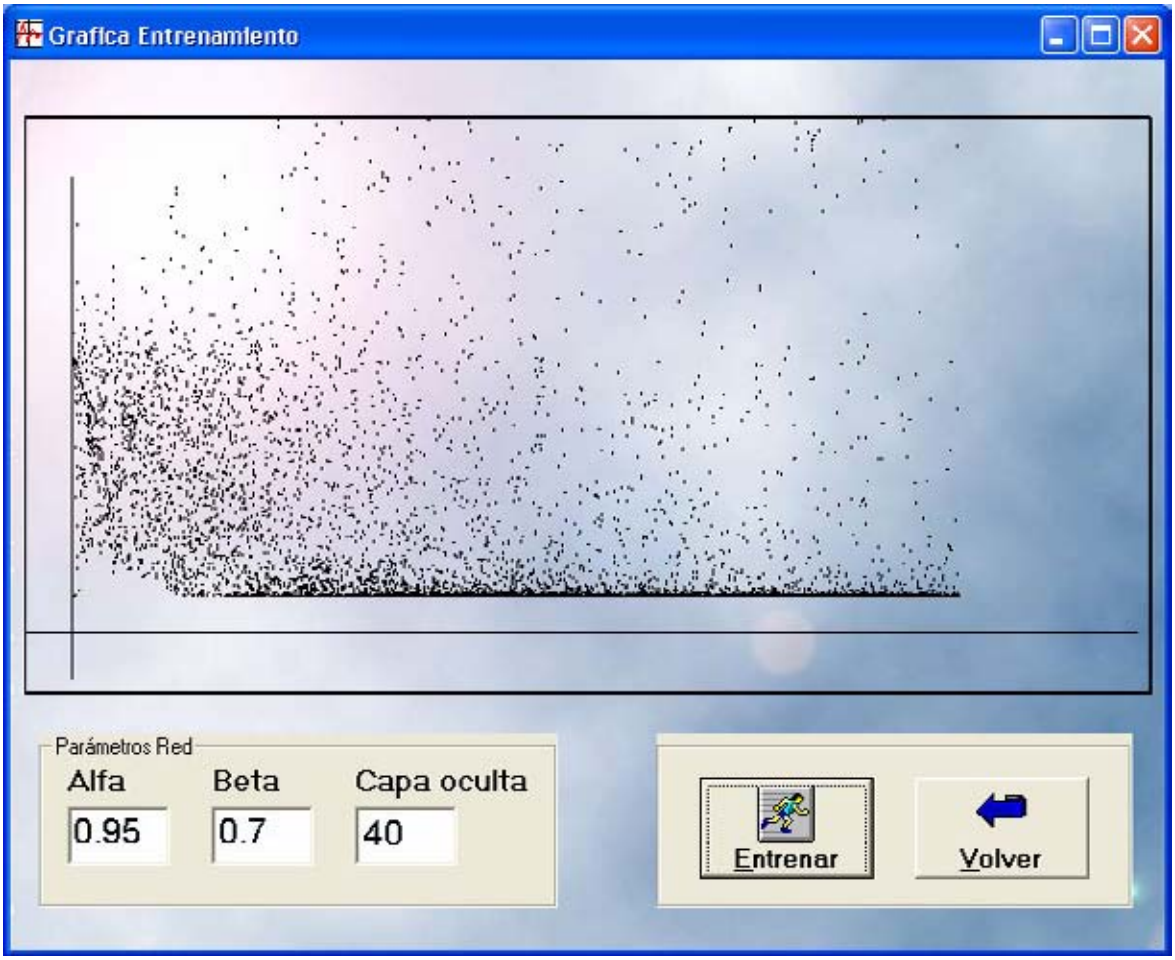


Fig.29

El entrenamiento no logró conseguir que la red encontrara un punto de estabilidad para el primer grupo de letras, pero si se observa que a pesar de las oscilaciones, el error tiende a disminuir, con lo que se presume que la red puede haber aprendido algunos de los patrones y en otros no consigue separar las zonas de discriminación, ya que los patrones son similares, se equivoca en la clasificación cada cierto número de iteraciones, pero acierta en un número mayor de veces, lo que explica que en la gráfica el error tienda a dibujar una línea recta. Así el mejor resultado se obtuvo con un valor de α=0.95, β=0.7 y 40 neuronas en la capa oculta.

Ahora se verán los resultados del entrenamiento para el segundo grupo de letras:

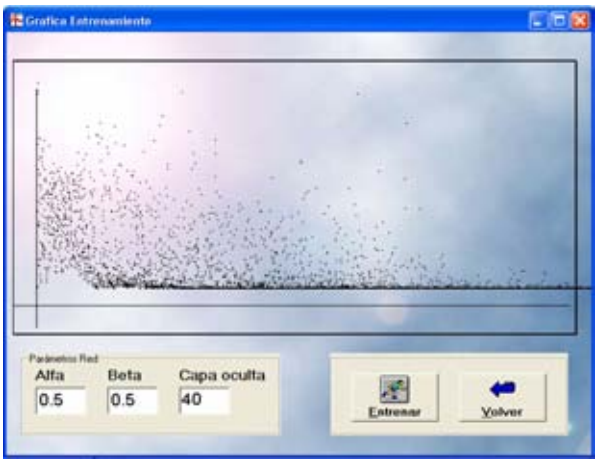
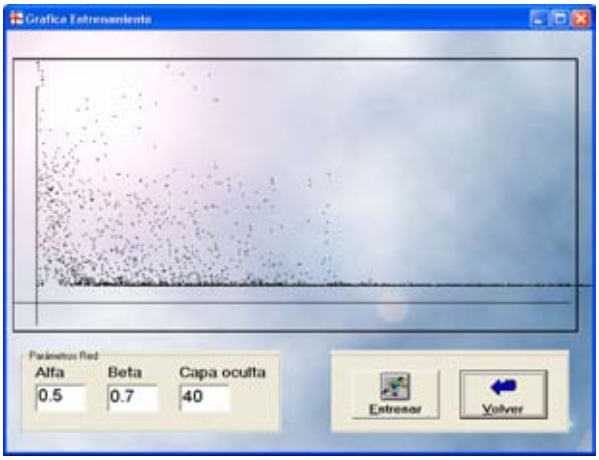


Fig.30 y 31



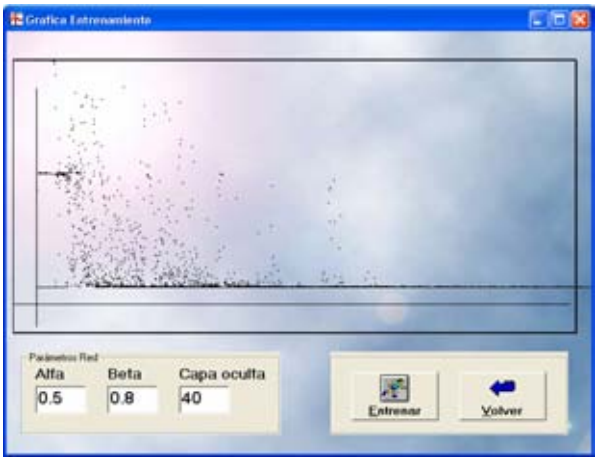


Fig.32 y 33

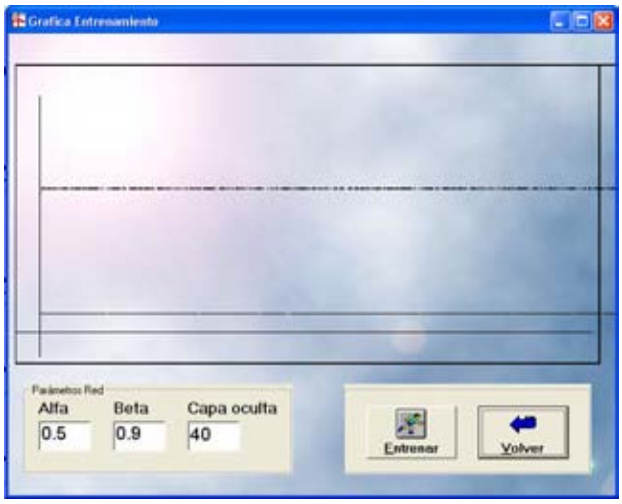

Grafica Entrenamiento
Parámetros Red
Alfa
Beta
Capa oculta
0.5
0.9
40
Entrenar
Volver

Fig.34

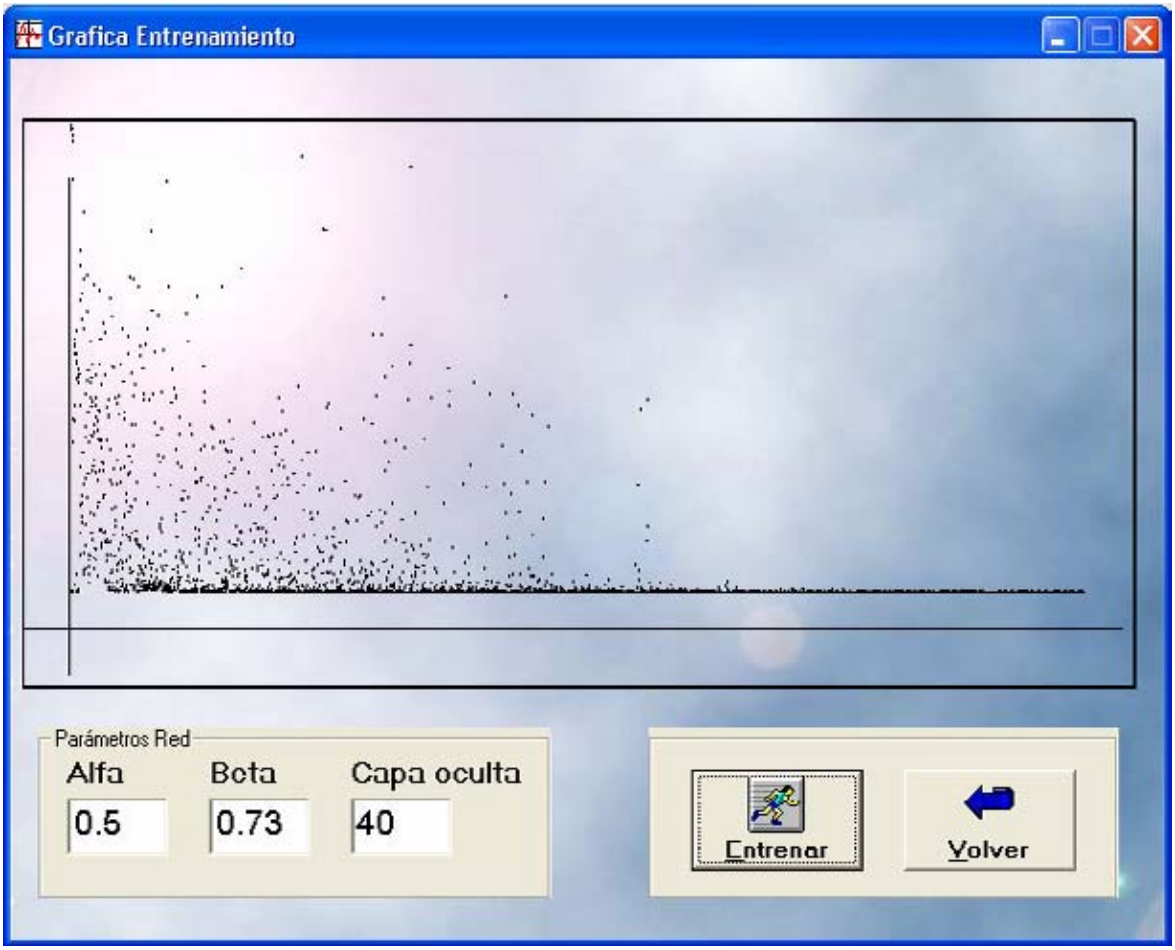
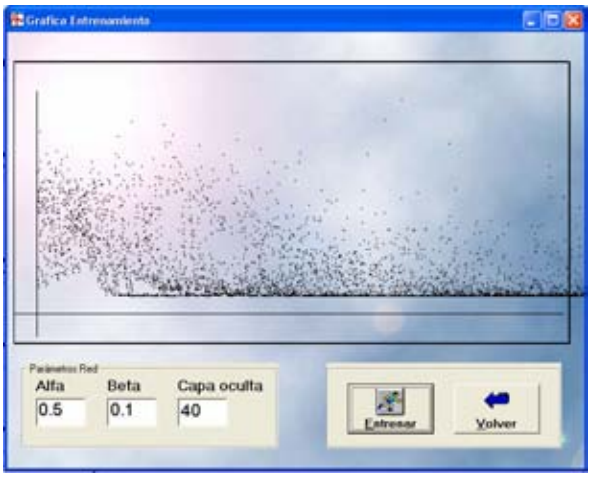

Grafica Entrenamiento
Parámetros Red
Alfa
Beta
Capa oculta
0.5
0.73
40
Entrenar
Volver

Fig.35

En el caso del 2º grupo de letras se consiguió que la red aprendiera, visualizándose claramente en los gráficos donde el error se minimiza y se mantiene constante, luego de oscilar durante la mitad del set de entrenamiento. Se observa que el mejor entrenamiento se consiguió con α=0.5, β=0.73 y 40 neuronas en la capa oculta. A continuación los resultados del entrenamiento del 3er. Grupo de letras:



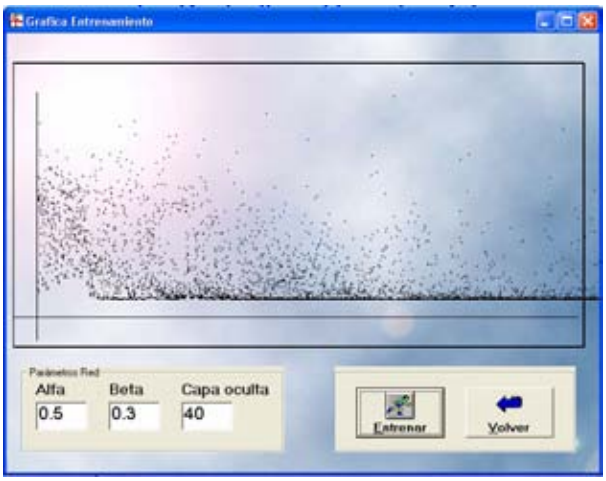
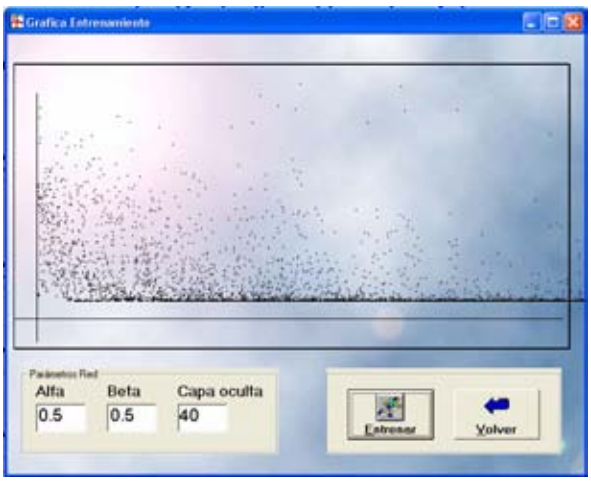


Fig.36 y 37



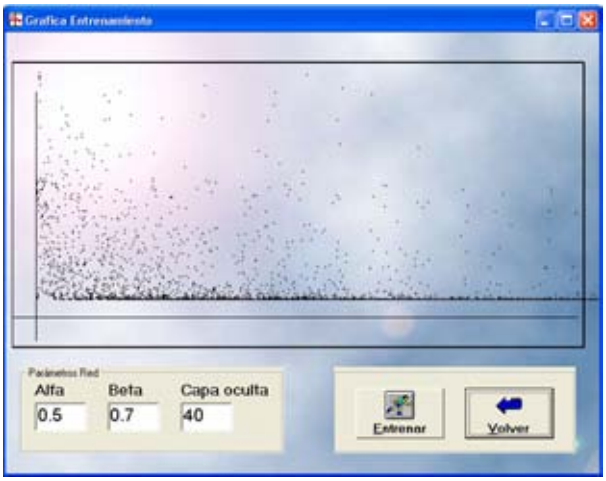
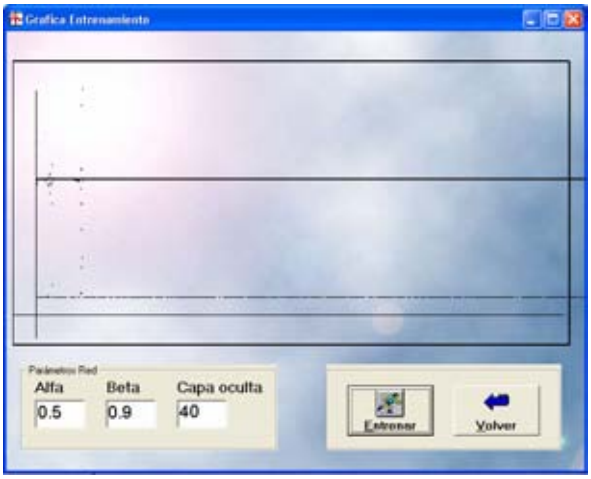


Fig.38 y 39



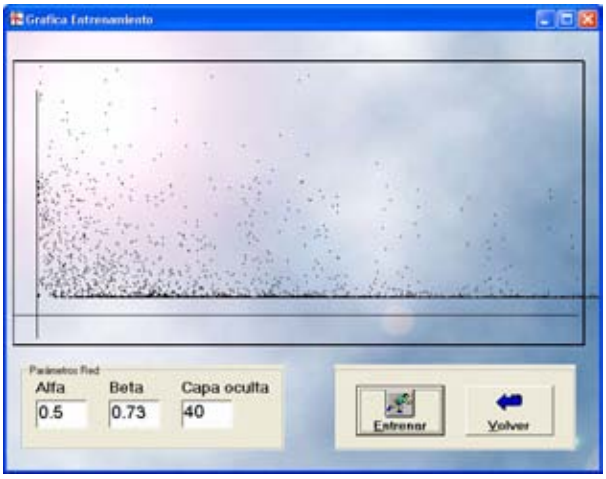


Fig.40 y 41

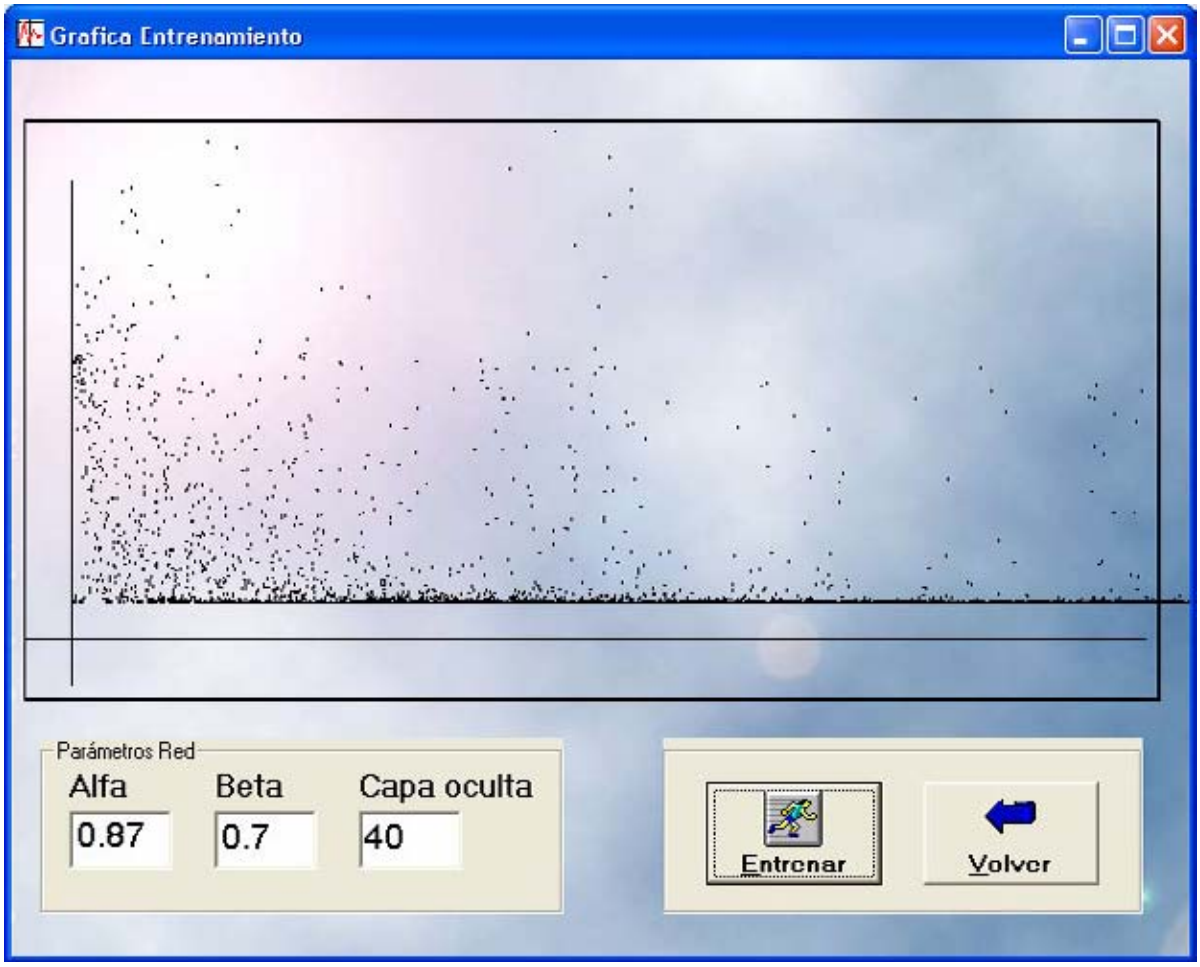
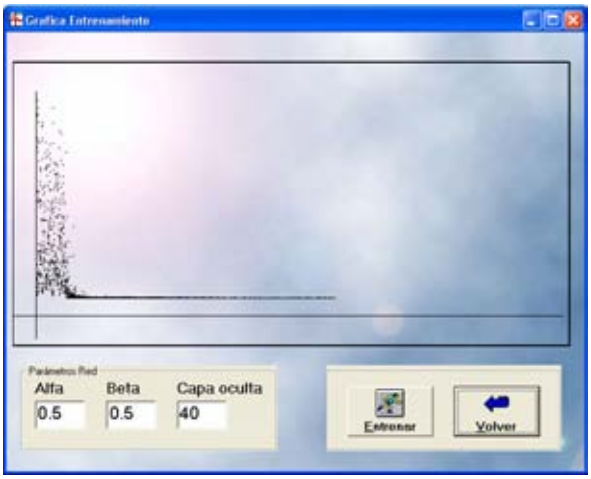


Fig.42

Se encontraron problemas similares a los presentados en el caso del 1er. set de entrenamiento. En este caso el error de la red presentó oscilaciones menores y la tendencia a la estabilización fue mayor, de lo que se deduce que la red consigue hacer una representación correcta de algunos patrones y presenta pequeñas dificultades para diferenciar otros. Se obtienen los mejores resultados con α=0.87 y β=0.7 .Y finalizando con el último set de letras.



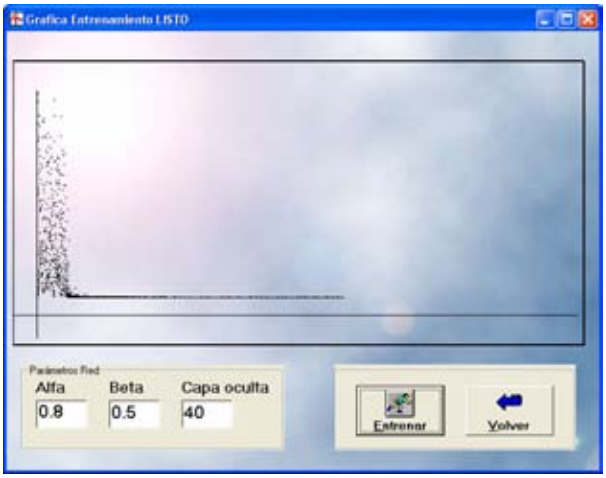


Fig.43 y 44

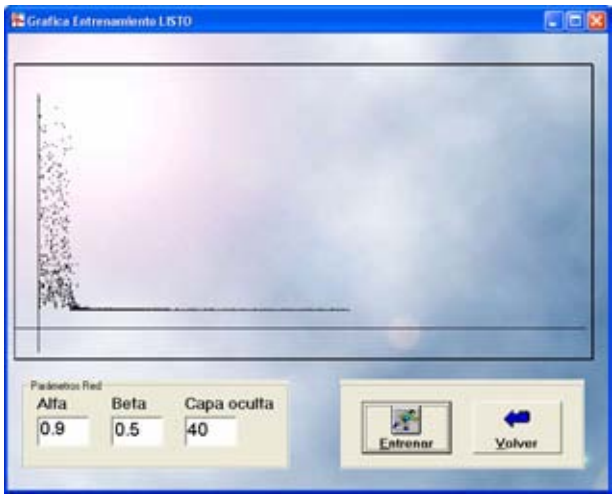


Fig.45

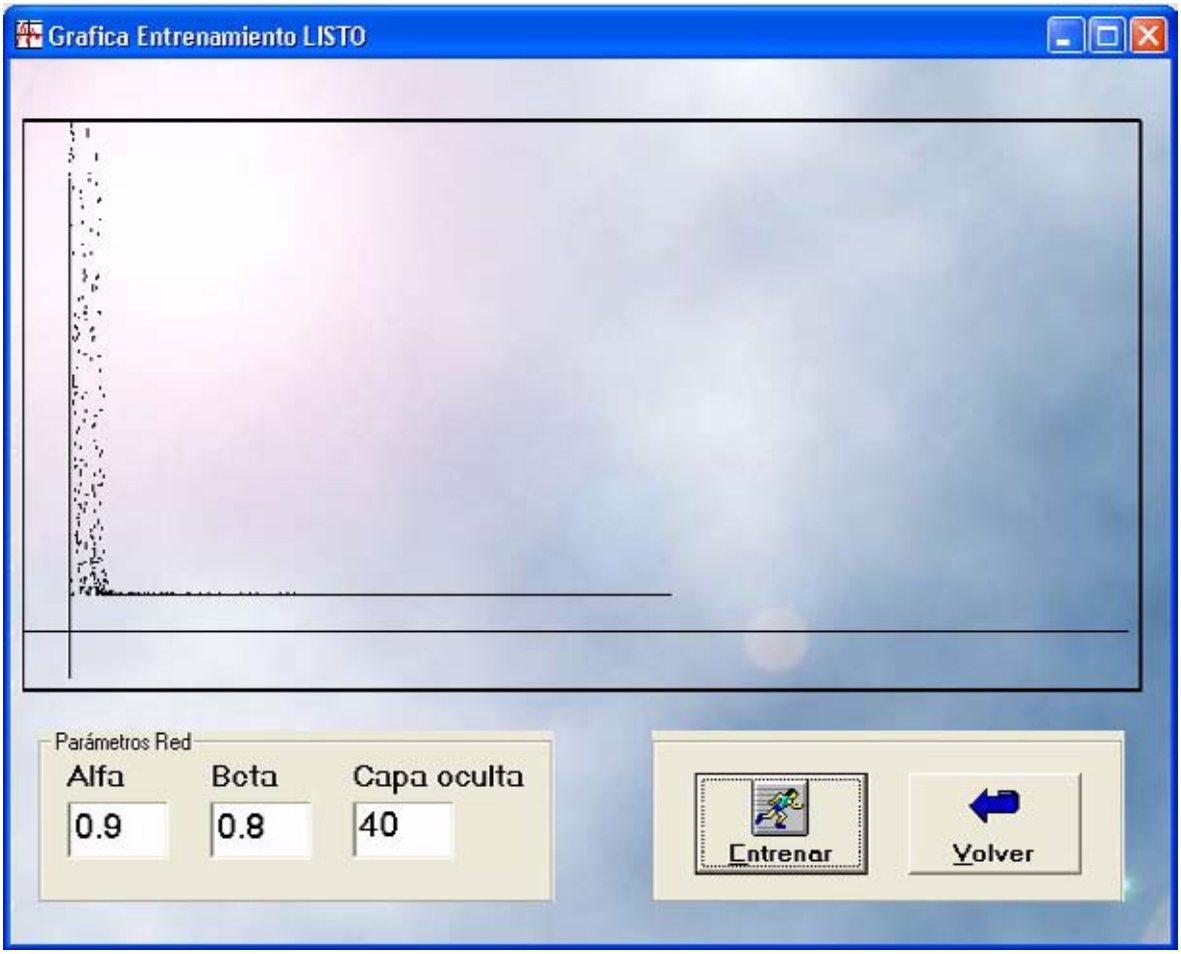
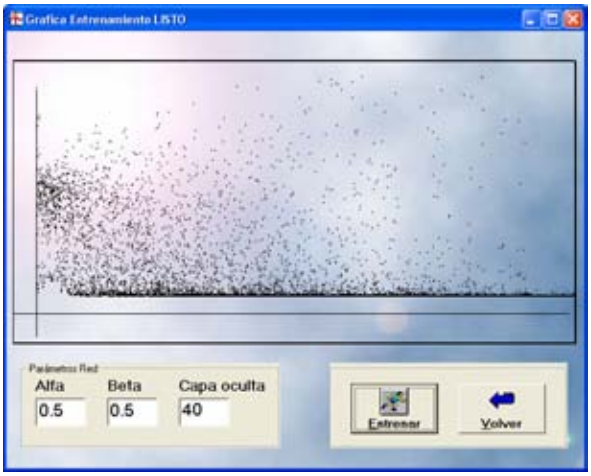


Fig.46

Aquí se aprecia la mayor rapidez en el aprendizaje, aun realizando varias pruebas para distintos valores de α y β se puede ver que el error se estabilizó con menos de un 20 % del set de entrenamiento. Los mejores resultados se consiguieron con α=0.9 y β=0.8.

En el caso del conjunto de letras para ser visualizadas en el display, los resultados para el primer set de entrenamiento fueron los siguientes:



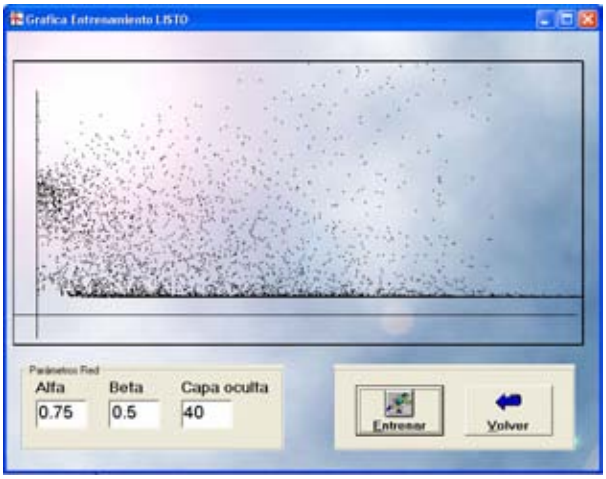


Fig.47 y 48

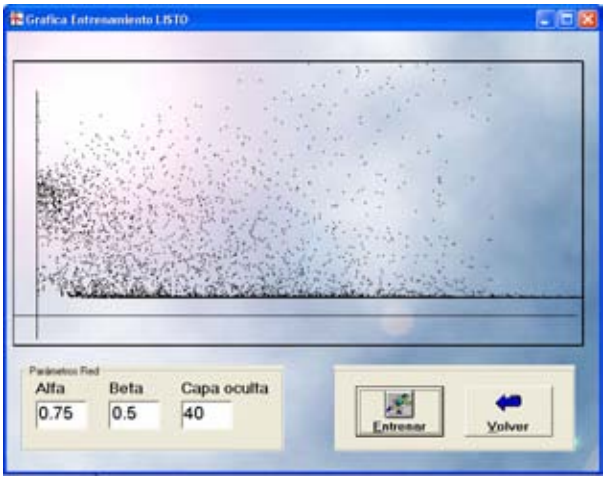


Fig.49 y 50

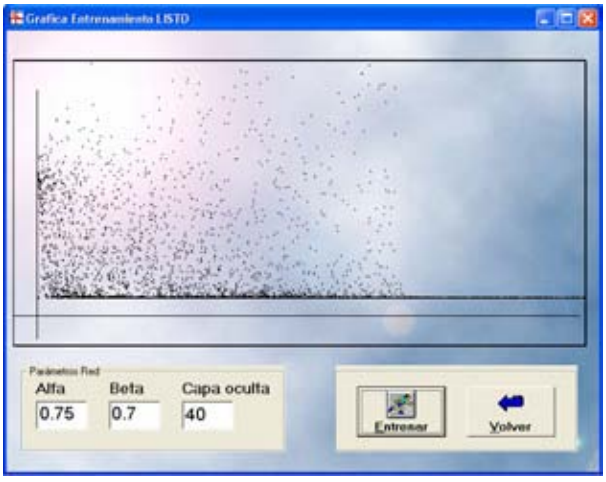


Fig.51 y 52

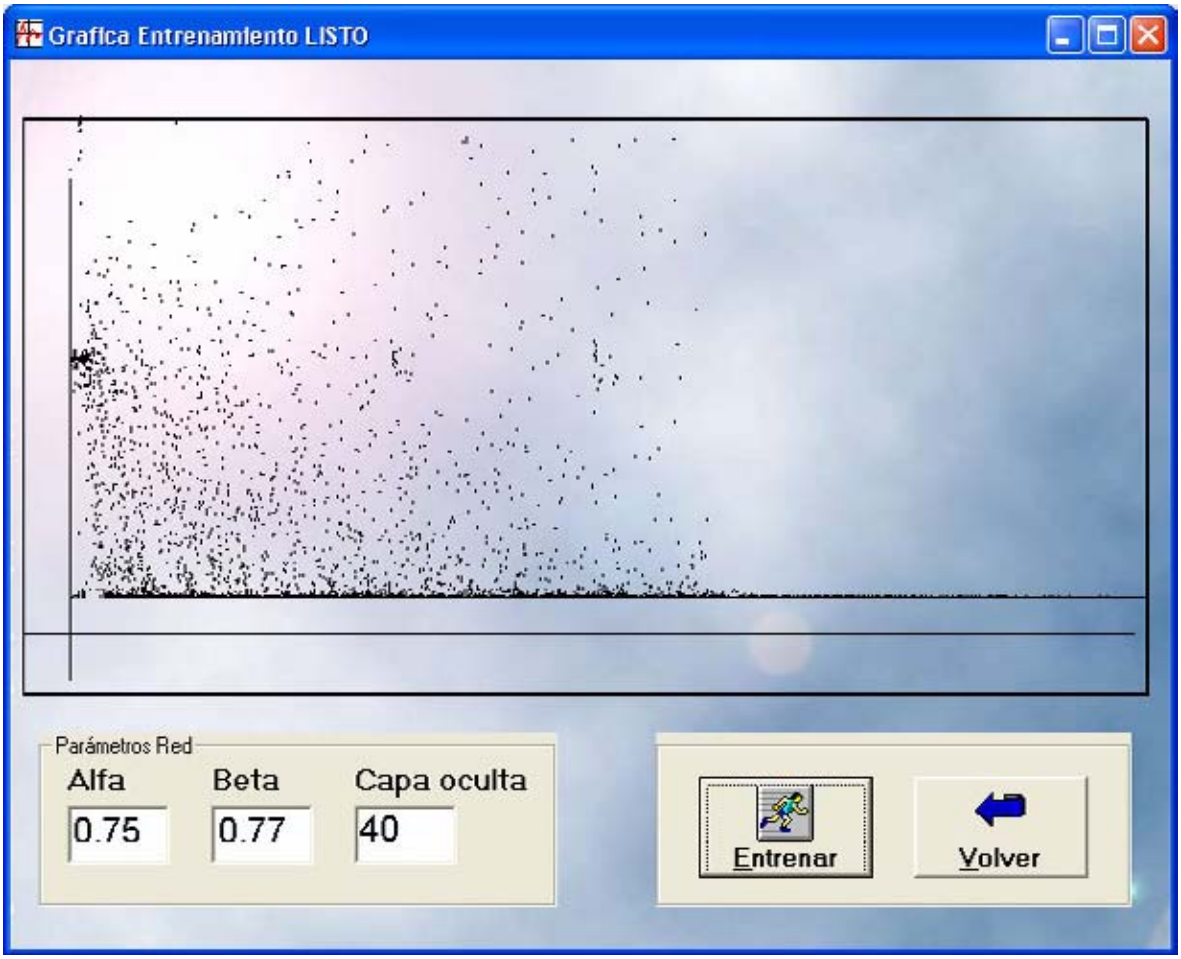


Fig.53

La red consiguió aprender correctamente, lo que se puede apreciar claramente en la gráfica donde el error disminuye gradualmente y se estabiliza en la parte inferior. Probando distintos valores de los parámetros se obtuvo el mejor resultado con α=0.75 y β=0.77.

Para el 2º set de letras se tienen los siguientes resultados:

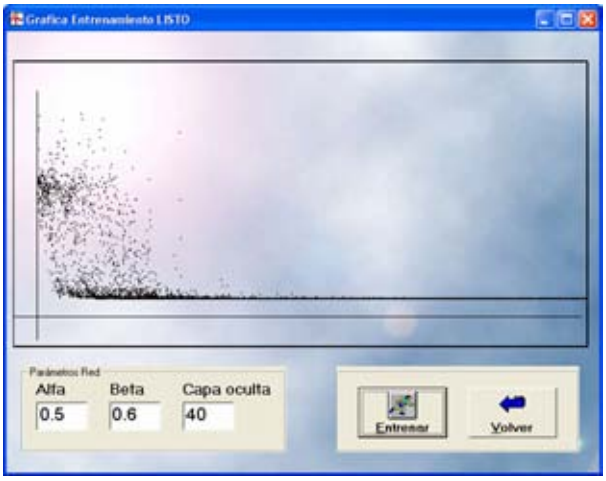
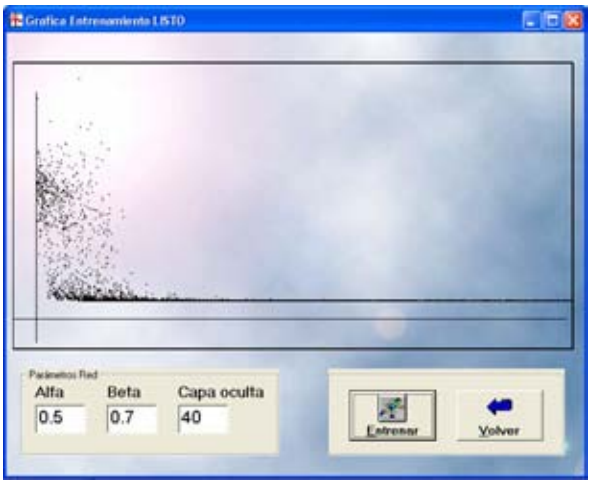


Fig.54 y 55



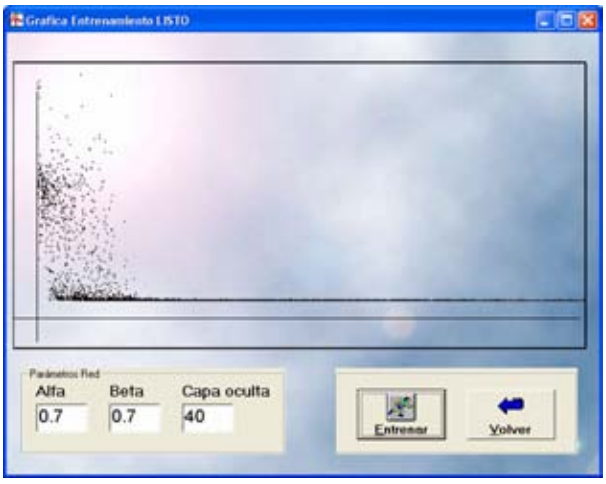
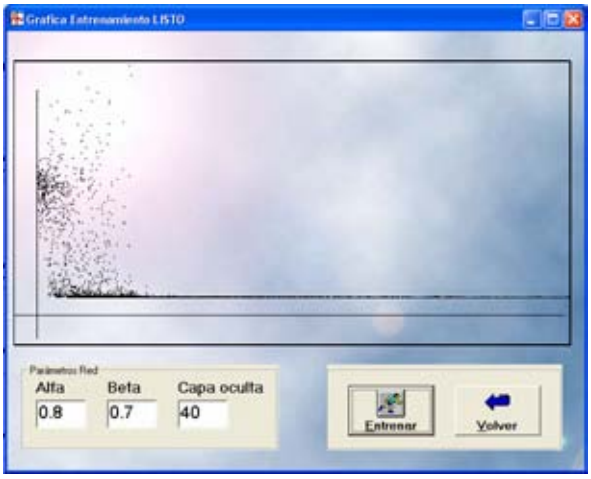


Fig.56 y 57



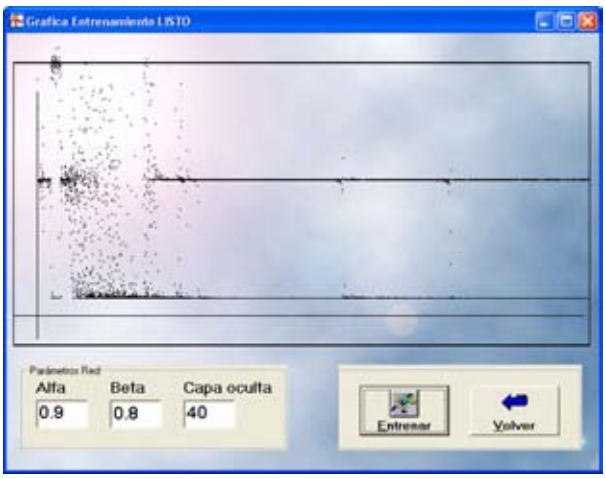


Fig.58 y 59

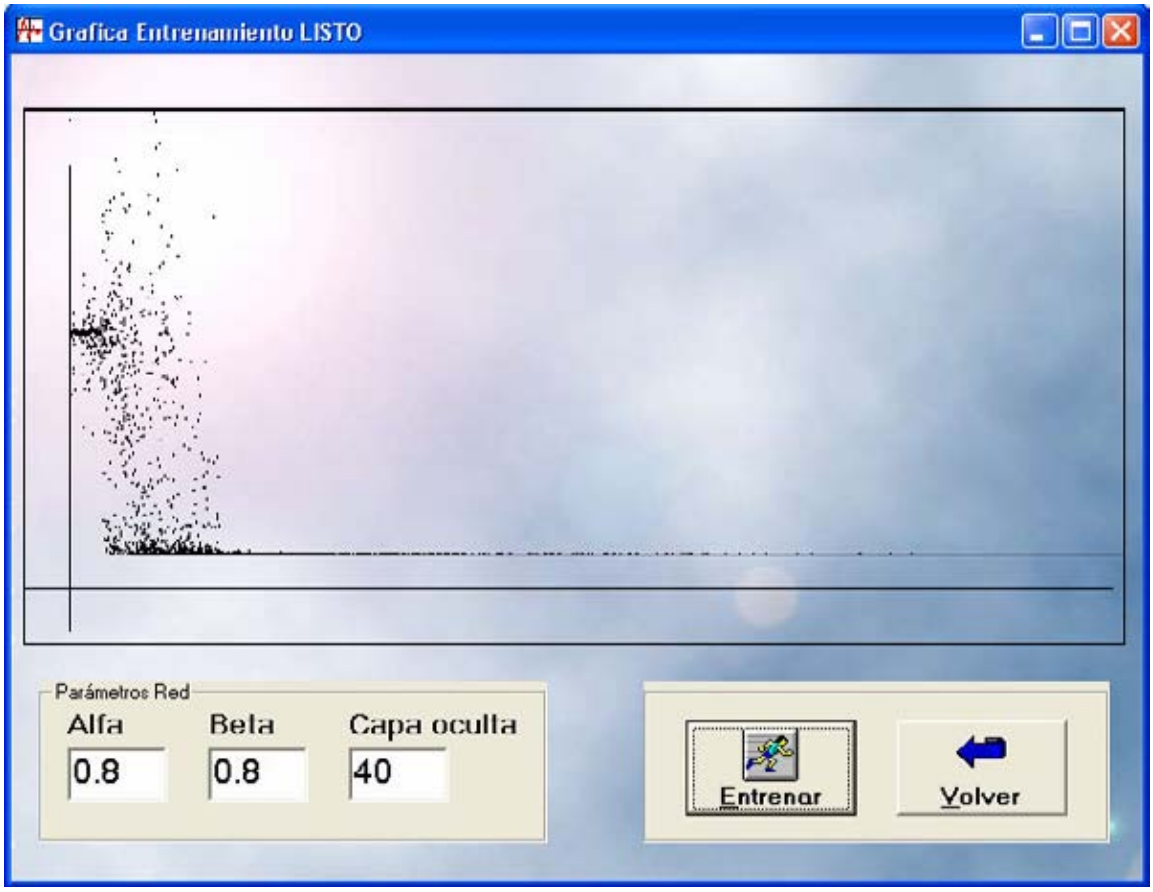
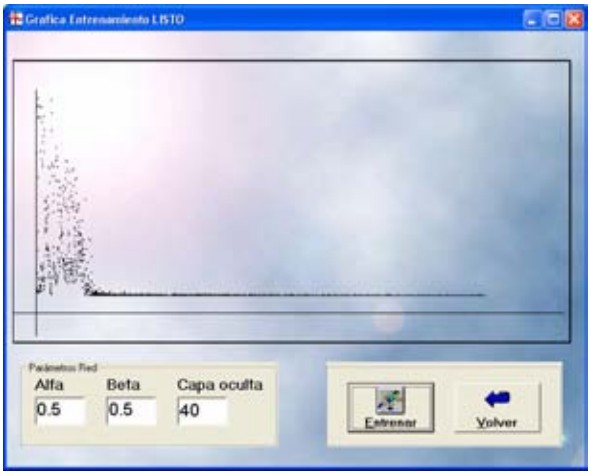


Fig.60

El aprendizaje del 2º set se realizó correctamente visualizándose la minimización del error y su estabilidad en todo el entrenamiento. Los mejores resultados se consiguieron con α=0.8 y β=0.8. Terminando con el 3er. Set de entrenamiento, los resultados a continuación:



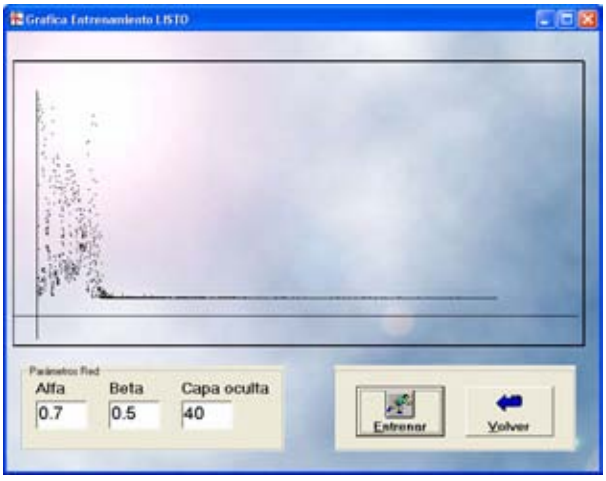


Fig.61 y 62


Grafica Entrenamiento LISTO
Parámetros Red
Alfa
Beta
Capa oculta
0.5
0.7
40
Entrenar
Volver

Fig.63

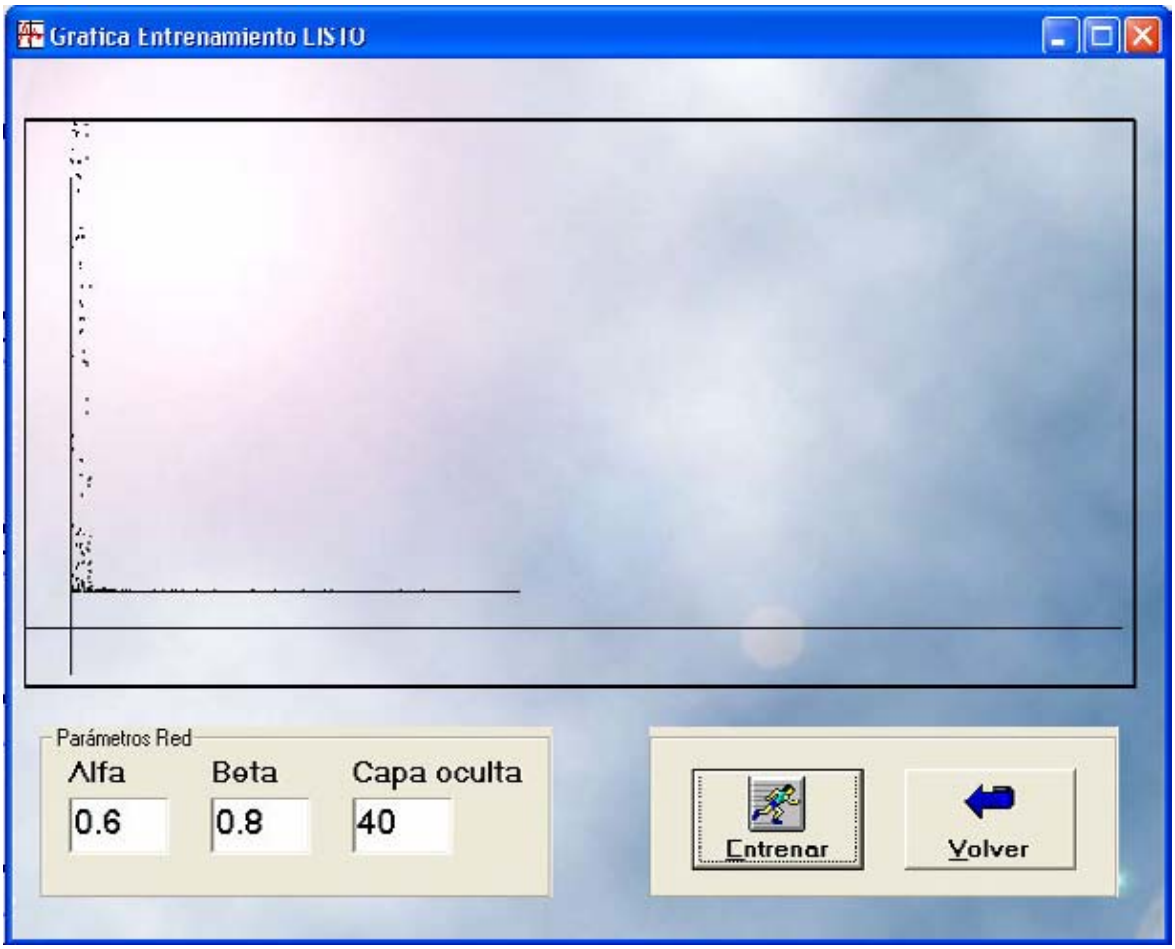

Grafica Entrenamiento LISTO
Parámetros Red
Alfa
Beta
Capa oculta
0.6
0.8
40
Entrenar
Volver

Fig.64

Sin problemas la red aprendió rápidamente los patrones de las últimas letras. Se ve que bastó aproximadamente un 15% del set para conseguir la minimización y estabilización del error.

Cada vez que se realizó un entrenamiento exitoso de un set de letras, se guardaron los pesos sinápticos resultantes en 3 archivos de texto, uno por cada capa. Estos archivos con los pesos obtenidos del entrenamiento son cargados posteriormente en las redes encargadas de reconocer las letras. Con estos resultados se procedió a la siguiente etapa y la más importante que es la de reconocimiento de las palabras.

CAPÍTULO IV. IMPLEMENTACIÓN DEL RECONOCEDOR DE PALABRAS

IV.1 Implementación

Una vez entrenadas las redes neuronales con todos los grupos de letras, ahora se busca reconocer palabras juntando las letras aprendidas separadamente. Para ello se utilizaron 4 redes neuronales donde cada una estaba encargada de reconocer una letra de la palabra pronunciada. Para el conjunto de 5 comandos cada una de las 4 redes utilizadas estaba entrenada con un grupo de letras. De esta forma al pronunciarse un comando, éste se dividía en 3 ó 4 partes y cada una de las partes era procesada separadamente por una red entrenada para el reconocimiento de una letra para una determinada posición. Luego cada vez que se reconocía una letra, ésta sumaba un punto a una variable que representaba la palabra a reconocer. Así a medida que se iban reconociendo las distintas letras del comando se sumaban puntos a la variable correspondiente de manera que al final la que obtuviese una mayor puntuación sería la palabra reconocida por el sistema. Las redes neuronales utilizadas en el reconocimiento de palabras hacen uso de los pesos sinápticos obtenidos en la etapa de entrenamiento, los que, previo al reconocimiento son cargados.

El diagrama que a continuación se presenta esquematiza el funcionamiento del reconocedor para los comandos de control del brazo. Es importante tener presente que estos comandos están formados por palabras de 4 letras. En el caso de las palabras utilizadas para controlar la visualización en el display, el esquema es similar, con la salvedad de que las letras pronunciadas son divididas en 3 partes.

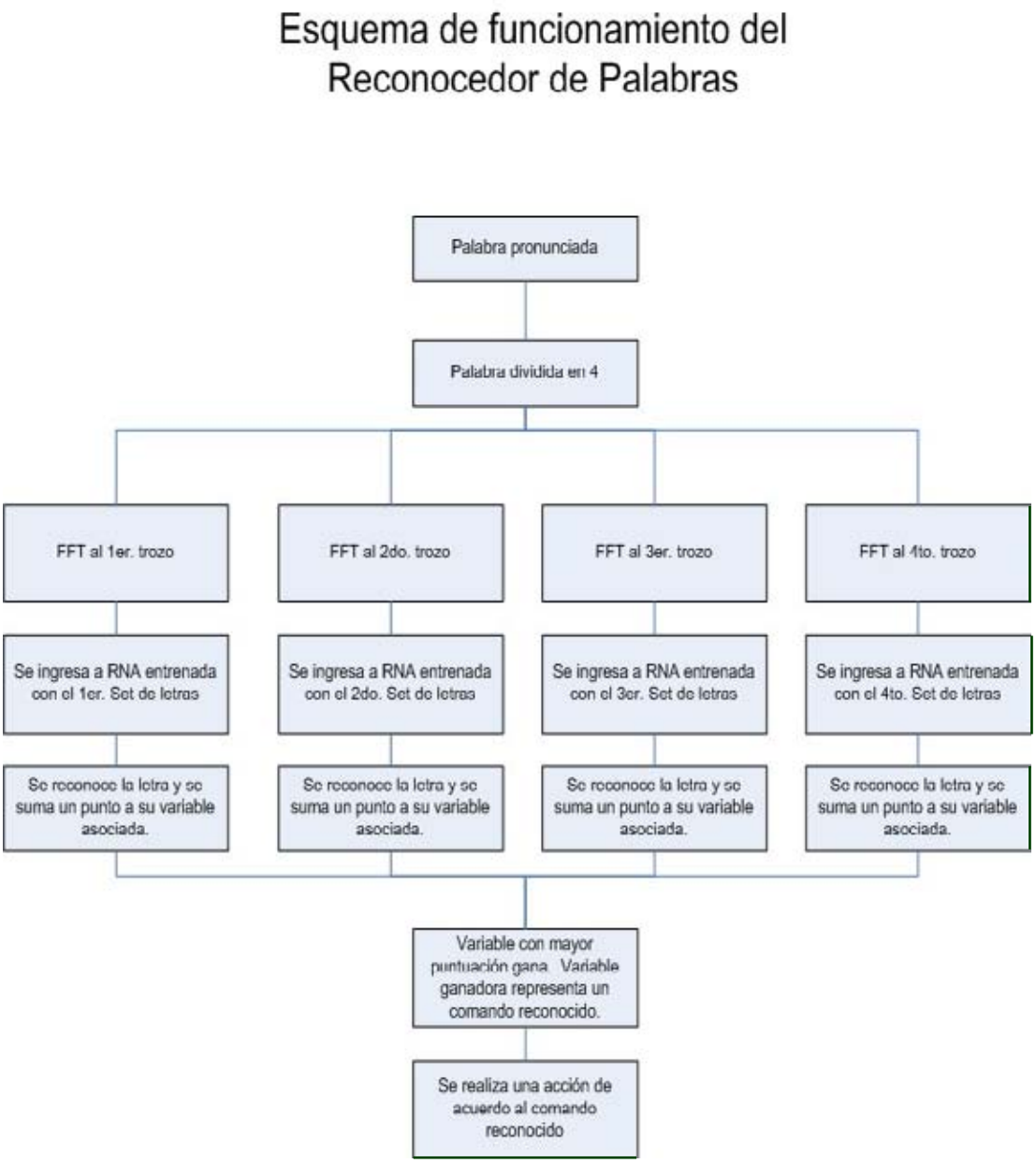


Fig.65

Con esta forma de reconocer palabras se crea un sistema flexible, pues permite que el reconocedor acepte equivocaciones por parte de alguna red en la tarea de reconocimiento, de esta forma no se afecta al resultado global, ya que el reconocimiento de las demás letras está a cargo de redes independientes y es menos probable que todas las redes se equivoquen a la vez, disminuyendo la posibilidad de error.

Antes de comenzar el proceso de reconocimiento de palabras es necesario utilizar el conocimiento adquirido o experiencia guardado en los pesos sinápticos. Se verá como es usado el entrenamiento previo para el caso de los comandos de control del brazo mecánico. En el caso de las letras usadas para controlar la visualización del display es aplicable el mismo ejemplo cambiando los archivos con los pesos guardados y considerando que se utilizaron 3 redes neuronales distintas a diferencia de los comandos del brazo donde fueron 4 redes las ocupadas. Para ello se escribió el siguiente código:

```
Open "PesosDITDP1_1.txt" For Input As #1
For i = 1 To n°n1          'de la 1 neurona hasta la 128
   For j = 0 To n°n1       'se asigna un valor aleatorio a todos los pesos partiendo de
                           we0 y hasta we128
     Input #1, we_(i, j)

   Next
Next
Close #1
```

En este trozo se abre un archivo de texto con el valor de los pesos sinápticos de la capa 1 para el 1er. Set de letras compuesto por T, D, P, e I. Luego se distribuyen los valores completando la matriz de pesos para la 1ª. capa de la red por medio de los ciclos *for*. Este proceso se repite para la segunda y tercera capa de red copiando los valores de los pesos sinápticos correspondientes guardados en archivos de texto (*PesosDITDP1_2.txt* y *PesosDITDP1_3.txt*) y luego se repite todo el proceso para las restantes 3 redes.

Una vez cargadas las matrices de las redes con los pesos entrenados el sistema está en condiciones de reconocer letras. El objetivo en esta etapa es juntar las letras separadas y formar palabras, esto se realizó mediante el siguiente código:

```
For x = 0 To 127
        .CurrentY = ScopeHeight
        .CurrentX = x
 ScopeBuff.Line Step(0, 0)-(x, ScopeHeight – (Sqr(Abs(OutData(x * 2) \ Divisor)) +
Sqr(Abs(OutData(x * 2 + 1) \ Divisor))))
   aux(x + 1) = (Sqr(Abs(OutData(x * 2))) + Sqr(Abs(OutData(x * 2 + 1))))
Next
```

El código de arriba toma 128 componentes de la FFT correspondientes a una letra, para toda la palabra pronunciada, la une y la reconoce como un comando.

En el primer trozo se normalizan los valores de las componentes de frecuencia para ingresarlas a la red.

```
Cont=1
If cont = 1 Then
    n°n3 = 4
    ii = 1
    kk = 2
    a = aux(ii)
    Do While kk <= 128
    b = aux(kk)
      If a > b Then          pregunta si a es mayor que b, si es así pasa al siguiente
                             valor del arreglo, conservando el valor de a
        kk = kk + 1          incrementa en 1 el contador para pasar al siguiente elemento
                            del arreglo
      Else
      a = b                  si no es mayor guarda el valor de b en a y aumenta en uno el
                             contador para ir a la siguiente posición del arreglo
      kk = kk + 1
      End If
```

```
    Loop
For k = 1 To 128 '64
  x_(k) = aux(k) / a
Next k
```

Se ingresan las componentes de frecuencia a la red neuronal empezando por la primera capa, luego la oculta y la de salida. Esta está encargada de reconocer la letra ingresada.

```
For j = 1 To n°n1
  netE_(j) = 0
Next j
x_(0) = -1
For i = 1 To n°n1 '35        por cada una de las neuronas de la capa de entrada
  s = 0
  For j = 0 To n°n1 '35      por cada uno de los pesos desde el umbral hasta el 35
  netE_(i) = netE_(i) + x_(j) * we_(i, j) 'netE(i) = netE(i) + x(j) * we(i, j) se calcula la
                             suma de las entradas de la neurona
  Next j
Next i
For i = 1 To n°n1            por cada salida de la capa de entrada
  ye_(i) = 1 / (1 + Exp(-netE_(i))) se calcula la salida de la neurona
Next i
For j = 1 To n°n2
  netO_(j) = 0
  Next j
ye_(0) = -1
For i = 1 To n°n2            por cada una de las neuronas de la capa oculta
  s = 0
  For j = 0 To n°n1          por cada peso recibido de las 35 neuronas de la capa de entrada
  netO_(i) = netO_(i) + ye_(j) * wo_(i, j) se obtiene la suma de las entradas de las
  Next                              neuronas de la primera capa
```

```
Next
For i = 1 To n°n2 '35        para cada neurona de la capa oculta se calcula su salida
   yo_(i) = 1 / (1 + Exp(-netO_(i))) 'a partir de las sumas
Next
For j = 1 To n°n3
   netS_(j) = 0
Next j
yo_(0) = -1
For i = 1 To n°n3            por cada neurona de salida se calculan los pesos que vienen de las
   s = 0                     neuronas de la capa oculta
   For j = 0 To n°n2
   netS_(i) = netS_(i) + yo_(j) * ws_(i, j) 'se obtiene la suma de las salidas de la capa
   Next j                    oculta
Next i
For i = 1 To n°n3            se calcula la salida de la red para las 5 neuronas de salida
   ys_(i) = 1 / (1 + Exp(-netS_(i)))
Next i
For i = 1 To n°n3
    If ys_(i) > 0.85 Then
   ys_(i) = 1
   Else
   ys_(i) = 0
   End If
Next i
```

Esta parte del código es sumamente importante ya que es la encargada de distribuir los puntos ganados entre las distintas variables por cada letra reconocida. Dependiendo de la letra que se ha reconocido se agregan puntos a la variable que corresponde a la palabra que se desea reconocer. Así si se pronuncia el comando "DEJA" y efectivamente la primera letra reconocida es la "D" el programa sumará un punto a las variables cmd(1) y cmd(4) que representan los comandos "DERÉ" y "DEJA" respectivamente. Seguidamente se ingresarán los componentes de

la FFT de la siguiente letra a la segunda red entrenada con el 2º set de letras y se realizará el mismo proceso donde se sumarán puntos a la variable que corresponde en el caso que la letra reconocida sea la correcta. Esto se repite para las 3 o 4 letras que conforman el comando pronunciado.

```
Select Case 1
   Case Int(Round(ys_(1)))     'Letra D
      cmd(1) = cmd(1) + 1
      cmd(4) = cmd(4) + 1
      GoTo chao1

   Case Int(Round(ys_(2)))     ' Letra I
      cmd(2) = cmd(2) + 1
      GoTo chao1

   Case Int(Round(ys_(3)))     'Letra T
      cmd(3) = cmd(3) + 1
      GoTo chao1

   Case Int(Round(ys_(4)))     'Letra P
      cmd(5) = cmd(5) + 1
      GoTo chao1
chao1:
End Select
```

Finalmente se determina cual fue la variable con mayor puntuación:

```
ii = 1
kk = 2
Do While kk <= 5
   If cmd(ii) > cmd(kk) Then
      kk = kk + 1
   Else
      ii = kk
      kk = kk + 1
   End If
Loop
```

Y de acuerdo a ello se entrega el resultado del comando (o letra en el caso del Display) reconocida:

```
If ii = 1 Then
Text4.Text = "DERECHA"
Command17_Click
End If
If ii = 2 Then
Text4.Text = "IZQUIERDA"
Command16_Click
End If
If ii = 3 Then
Text4.Text = "TOMA"
Command15_Click
End If
If ii = 4 Then
Text4.Text = "DEJA"
Command14_Click
End If
If ii = 5 Then
```

```
Text4.Text = "PARA"
Command18_Click
End If
```

Donde el estamento "C*ommandxx_Click*" representa la orden que es enviada al puerto paralelo del computador y "TextX.Text=comando" visualiza la palabra reconocida en un cuadro de texto.

El envío de las órdenes al puerto paralelo se realiza de la siguiente manera:

```
Private Sub Command14_Click()
ClrPortBit &H37A, 0     'elige la grua
ClrPortBit outport, 7    'toma
ClrPortBit &H37A, 2
ClrPortBit &H37A, 1
End Sub

Private Sub Command15_Click()
ClrPortBit &H37A, 0     'eligue la grua
ClrPortBit outport, 7     'deja
SetPortBit &H37A, 2
SetPortBit &H37A, 1
End Sub

Private Sub Command16_Click()
SetPortBit &H37A, 0     'elige el motor de la base
SetPortBit outport, 7
SetPortBit &H37A, 2
SetPortBit &H37A, 1
End Sub
```

```
Private Sub Command17_Click()
SetPortBit &H37A, 0     'elige el motor de la base
SetPortBit outport, 7
ClrPortBit &H37A, 2
ClrPortBit &H37A, 1
End Sub

Private Sub Command18_Click()
SetPortBit &H37A, 0     'elige display
ClrPortBit outport, 7

SetPortBit &H37A, 2
ClrPortBit &H37A, 1
End Sub
```

Donde la instrucción "SetPortBit&DirecciónPuerto, Nº de bit" envía un uno lógico al puerto paralelo y "ClrPortBit& DirecciónPuerto, Nº de bit" envía un cero lógico. El número binario enviado por el puerto responde a la tabla vista en el apéndice donde se refiere a la asociación de un comando con un número.

IV.2 Fase de prueba del reconocedor

Ahora el sistema fue puesto a prueba. Para ello se realizaron los siguientes experimentos: para el conjunto de comandos se hizo andar el reconocedor y se pronunciaron 50 veces seguidas cada comando en un micrófono unidireccional, en una pieza sin ruido. La interfaz del programa usado en las pruebas para los comandos se puede ver más abajo:

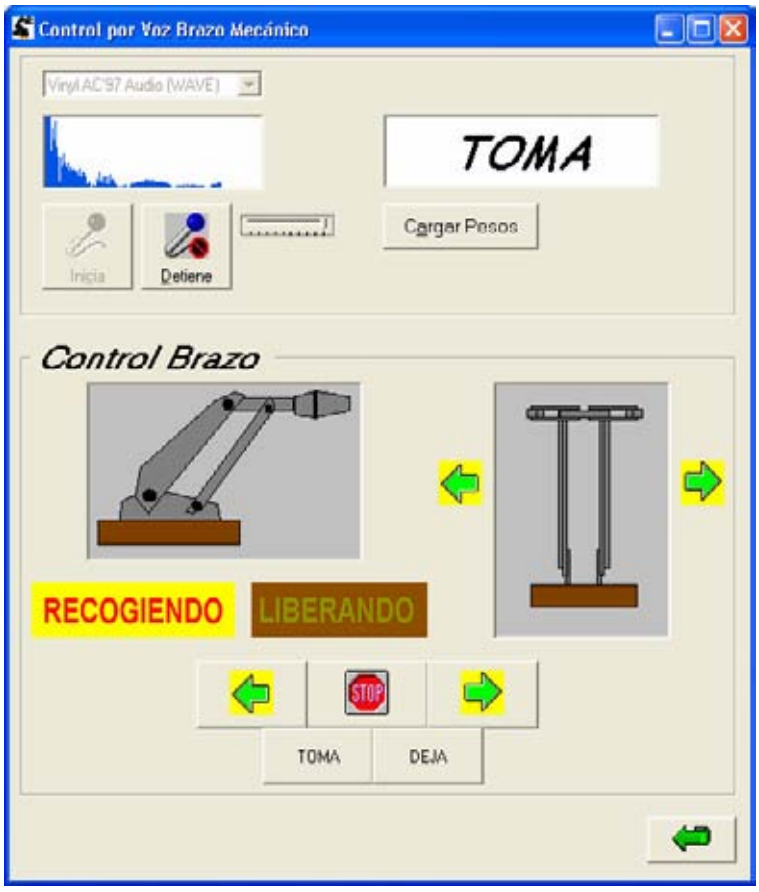


Fig.66

Los resultados del experimento a continuación:

Comandos pronunciados	**Comandos Reconocidos**							
	Deré	***Deja***	***Izké***	***Para***	***Toma***	**No Reconoce**	**Aciertos %**	**Error %**
Deré	**37**	5	1	2	5	0	**74%**	**26%**
Deja	7	**35**	2	2	4	0	**70%**	**30%**
Izké	3	2	**34**	1	10	0	**68%**	**32%**
Para	2	1	1	**41**	3	2	**82%**	**18%**
Toma	4	2	0	7	**37**	0	**74%**	**26%**
						Precisión Global	***73.6%***	

Para el caso de las 8 letras del abecedario la prueba realizada fue similar, se pronunciaron 50 veces cada letra en forma seguida en un micrófono. Se hizo andar el reconocedor, cuyo interfaz se muestra más abajo:

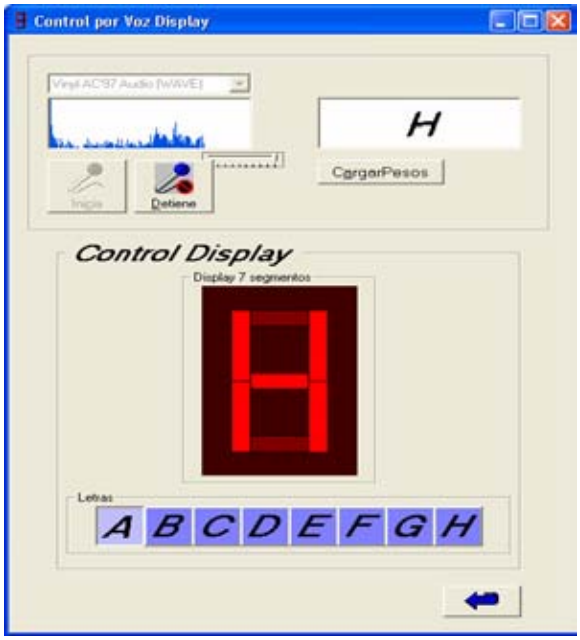


Fig.67

Los resultados de las pruebas para este caso a continuación:

Letras Pronunciadas	**Letras Reconocidas**										
	A	***B***	***C***	***D***	***E***	***F***	***G***	***H***	***Sin reconocer***	**Aciertos %**	**Error %**
A	**42**	0	0	0	4	1	0	2	2	**84%**	16%
B	3	**23**	2	19	1	2	0	0	0	**46%**	54%
C	1	2	**40**	0	1	0	5	0	1	**80%**	20%
D	2	20	5	**19**	2	1	0	1	0	**38%**	62%
E	2	3	0	1	**44**	0	0	0	0	**88%**	12%
F	2	9	5	0	10	**21**	0	0	3	**42%**	58%
G	0	2	3	0	1	1	**41**	0	2	**82%**	18%
H	8	4	6	2	5	2	3	**19**	1	**38%**	62%
									Precisión Global	***62.25%***	

En el caso de las pruebas de los comandos de voz, la precisión fue relativamente similar para todos los casos, siendo la precisión en términos globales aceptable. Sin embargo en el caso de las pruebas realizadas a las 8 letras se pudo apreciar grandes variaciones en relación a la precisión, obteniéndose buenos resultados en algunos casos como son las vocales solas y algunas consonantes, pero por otro lado grandes dificultades en distinguir por ejemplo la letra "D" de la "B" mostrando una precisión más pobre que en el caso anterior en términos globales.

El porcentaje de aciertos se calculó de acuerdo a la siguiente fórmula:

$$\%Aciertos = \frac{Palabras\,\mathrm{Re}\,conocidasCorrectamente}{TotaldePalabras\,\mathrm{Pr}\,onunciadas} x100\%$$

Y el porcentaje de errores de acuerdo a la siguiente fórmula:

$$\%Errores = \frac{PalabrasNo\,\mathrm{Re}\,conocidas + PalabrasMal\,\mathrm{Re}\,conocidas}{TotaldePalabras\,\mathrm{Pr}\,onunciadas} x100\%$$

IV.3 Desarrollo de la Interfaz gráfica del sistema de reconocimiento de palabras.

Con el fin de permitir una manipulación sencilla del sistema se implementó un programa que permite al usuario en forma intuitiva elegir la tarea que desee realizar. Este interfaz está compuesto de una pantalla principal con todas las opciones disponibles: control del brazo mecánico por voz y en forma manual, control de la visualización del display por voz y manualmente y un gráfico donde se puede entrenar una red neuronal y visualizar el avance de su entrenamiento. La pantalla principal se puede apreciar a continuación:

Fig.68 Pantalla principal Sistema de reconocimiento de voz

Se pueden seleccionar las opciones directamente desde los botones ubicados a la izquierda del programa, donde cada uno tiene un icono que representa la función que realiza. Esto también se puede realizar desde el menú colgante ubicado en la parte superior izquierda del programa.

Fig.69 Pantalla principal Sistema de reconocimiento de voz

Donde cada opción lleva a otra pantalla con la función descrita, empezando desde el botón de arriba se llega a la siguiente pantalla:

Este corresponde al modo manual para control de movimientos del brazo mecánico. Se puede apreciar dos figuras que representan el brazo, de frente y de lado. Por medio de los botones de la interfaz es posible ordenar al brazo que se mueva hacia la derecha e izquierda, que coja algo o que lo suelte. Esto también se puede realizar por medio de los botones del teclado, utilizándose las flechas "derecha" e "izquierda" para los movimientos horizontales y las letras "T" para coger y "D" para soltar. A medida que son realizados los movimientos, las flechas al lado de la figura frontal cambian de color para indicar que se está realizando un movimiento con esa dirección y al mismo tiempo envían al puerto paralelo la orden para que se realicen los movimientos adecuados. Asimismo cuando se ordena una acción de tomar o soltar algo los cuadros con las etiquetas "Recogiendo" y "Liberando" cambian de color según se haya ordenado realizar una u otra.

Fig.70 Modo manual control Brazo Mecánico

En el segundo modo se controlan los movimientos del brazo mecánico por voz. Para ello primero se debe seleccionar la tarjeta de sonido disponible y posteriormente es necesario inicializar el sistema, lo que se consigue presionando el botón etiquetado "Inicializar sistema". Una vez hecho esto se debe iniciar la escucha del micrófono, pues el sistema está listo para reconocer voz, para ello se debe presionar el botón "Escuchar". En este modo hay una pantalla que visualiza el espectro de la voz ubicada sobre el botón "Escuchar". Al reconocer una palabra ésta es visualizada en una pantalla ubicada a la derecha del visualizador de espectro y se envía la orden al puerto paralelo para que el programa realice las acciones que correspondan. Para detener la escucha presionar el botón "Detener" y para salir la flecha verde en la parte inferior de la pantalla. Los elementos comunes con el modo manual funcionan de la misma forma al activarse un comando de voz.

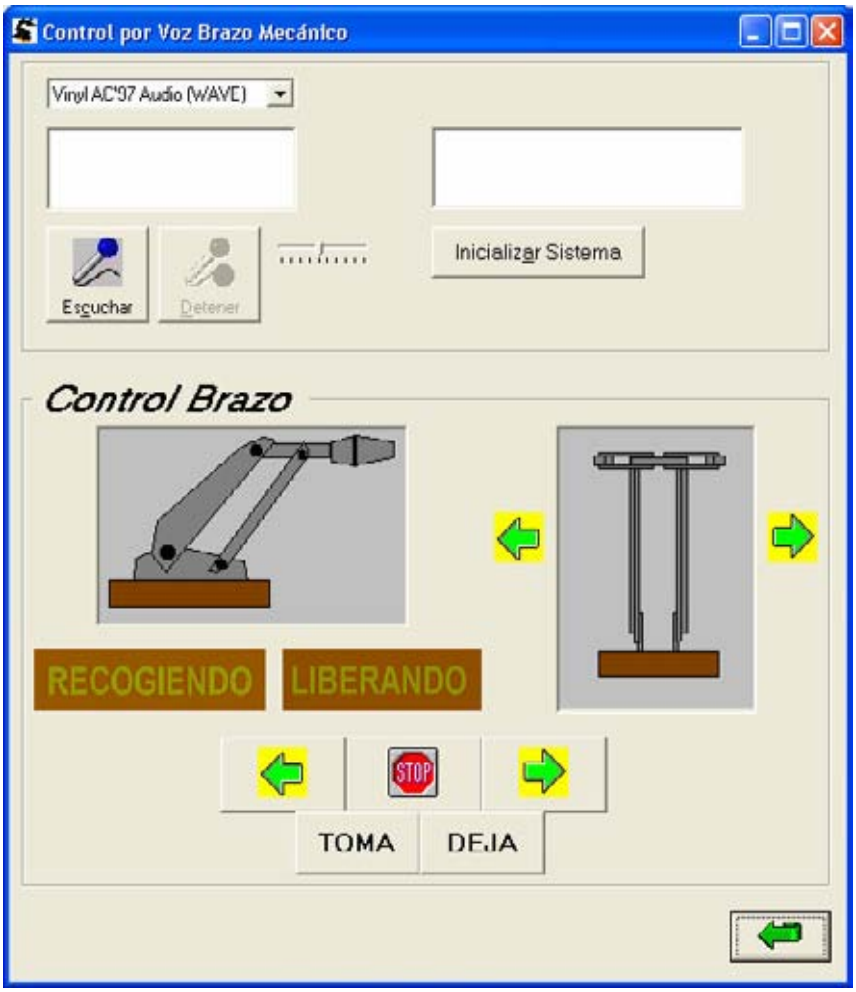


Fig.71 Modo de control por voz Brazo Mecánico

En el 3er. modo se controlan por medio del mouse o del teclado las letras visualizadas en el Display montado en el interfaz del sistema. Para ello el programa tiene 8 botones con las 8 primeras letras del abecedario. Estas pueden ser seleccionadas haciendo clic con el mouse sobre los botones o desplazándose con las flechas derecha, izquierda, arriba y abajo del teclado. Al seleccionar una letra esta aparece inmediatamente en el Display en pantalla y sobre el Display real.

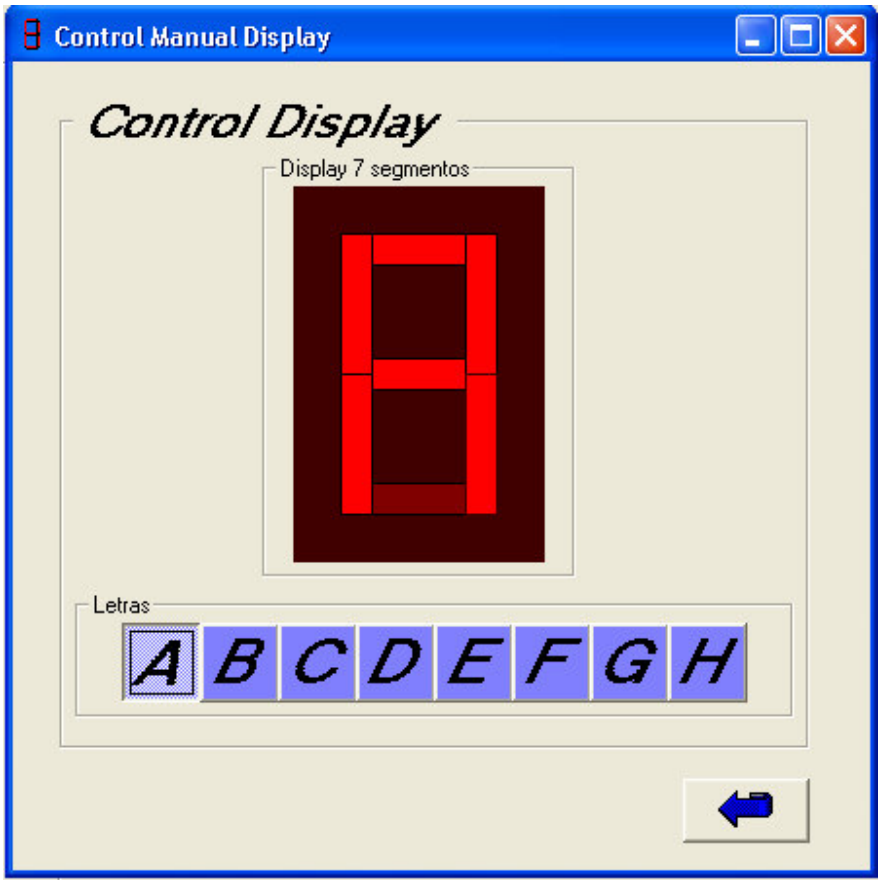


Fig.72 Control manual Display 7 segmentos

El 4º modo corresponde al control de voz de la visualización del Display. Su funcionamiento es idéntico al modo de control por voz para el brazo mecánico. Primero se debe inicializar el sistema y luego iniciar el micrófono para que la red pueda reconocer las letras pronunciadas. Al reconocer una letra esta se visualiza en el Display, y se envía la orden al puerto paralelo que se comunica con la interfaz para que este lo muestre en el Display real.

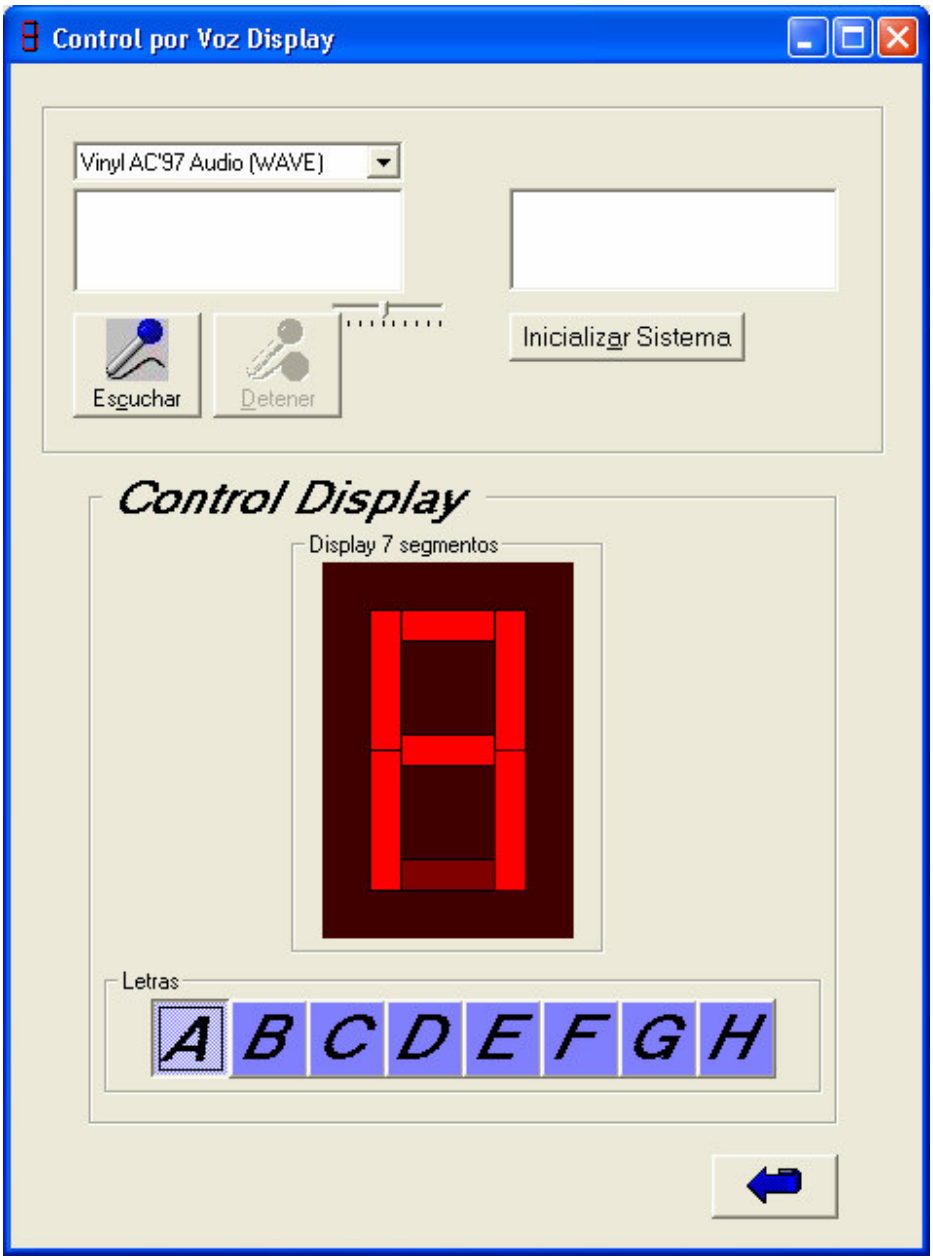


Fig.73 Modo de control por voz Display 7 segmentos

Y el último modo corresponde al de entrenamiento. En este se puede seleccionar un archivo con datos de entrenamiento, el número de neuronas de entrada, el número de neuronas de la capa oculta y de salida, el valor de la tasa de aprendizaje α y del momento β. Ya que se han seleccionado todos los valores anteriores se puede visualizar el aprendizaje de la red. Su operación es muy intuitiva, con texto explicativo en algunos elementos.

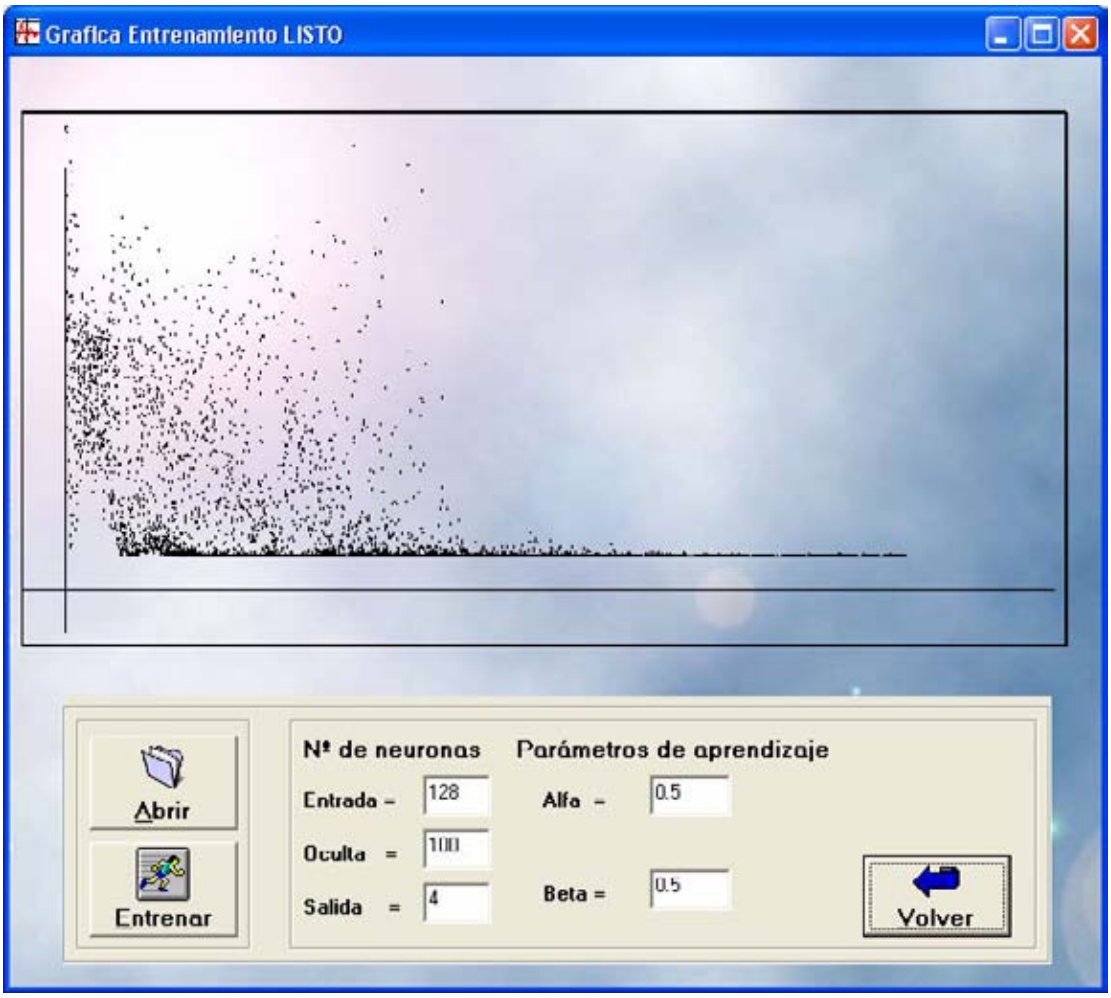


Fig.74 Modo de entrenamiento

CONCLUSIONES

- Es posible reconocer letras con la red *Backpropagation*, su capacidad de "aprender" patrones le permite formar una representación adecuada de las formas del espectro de un fonema y clasificarlo según se haya establecido un criterio. Esto se extiende para cualquier patrón de forma, como imágenes o caracteres.

- Para reconocer palabras completas la red *Backpropagation* no posee la capacidad de hacerlo directamente. Su arquitectura impide procesar señales que dependen de un instante anterior y que varíen en el tiempo. Su capacidad de procesamiento permite trabajar con patrones invariantes en el tiempo. A pesar de esto, a través de programación es posible utilizar sus capacidades para reconocer palabras en tiempo real adecuadamente.

- De acuerdo a pruebas realizadas con la red *Backpropagation* se consiguió reconocer hasta 10 patrones de formas distintos exitosamente (reconocedor de caracteres de números 0 a 9 con 9x7 neuronas) por lo que se presupone que utilizando otro método de extracción de características más eficiente que la FFT que revele información más relevante de cada fonema pronunciado, sería posible aumentar el número de letras capaz de reconocer traduciéndose esto directamente en un aumento en la cantidad de palabras manejadas por el reconocedor y además mejorar la eficiencia en el reconocimiento de fonemas cuya pronunciación es muy similar, perfeccionando su capacidad de reconocimiento.

- La red neuronal se mostró incapaz de clasificar adecuadamente las formas espectrales de letras distintas, pero muy similares entre sí para el oído humano. Esto se explica porque la red al aprender un patrón no memoriza una serie de valores específicos, sino que asocia una respuesta a varios conjuntos de valores que agrupados presentan formas similares, por lo tanto al tener dos letras distintas pero con espectros parecidos la red neuronal trata de aprender una forma como generalización de la otra y viceversa a lo que además se asocia una salida distinta para cada patrón, generando oscilaciones en la red e impidiendo una asociación adecuada de los patrones. Dicho de otra forma la red neuronal crea zonas en

un espacio para cada patrón aprendido, pero en el caso de dos patrones similares asociados a letras distintas, la red crea zonas que se traslapan y no es capaz de decidir a qué categoría pertenece una u otra en el momento de clasificar.

- Como se puede apreciar en los resultados de los entrenamientos de la red con los set de letras, para cada grupo se tuvo que hacer ajustes en los parámetros de la red y en todos los casos estos fueron distintos, por ejemplo en algunos casos la red convergía más rápido con determinados valores, pero utilizando los mismos valores estos no entregaban la misma respuesta para otro set de entrenamiento, por lo que no se pudo encontrar una regla que permita encontrar fácilmente los valores óptimos para cada set de entrenamiento. Al parecer esto se debe a que el algoritmo *Backpropagation* es un algoritmo que busca un mínimo de la función de error en el espacio de pesos, pero estos a su vez dependen de los valores guardados en los set de entrenamiento, por lo que la forma de la función de error es inevitablemente una superficie distinta para cada sesión de entrenamiento y además va cambiando en cada iteración (procesamiento de un patrón de entrenamiento) impidiendo que se pueda determinar un valor para la tasa de entrenamiento y el momento con anterioridad al inicio de las iteraciones de la red ya que no se puede saber donde estarán ubicados los mínimos de la función.

- El funcionamiento del sistema se vio limitado entre otras cosas por la longitud de las palabras que podía reconocer. Este se diseñó para aprovechar el hecho de que las palabras de 2 sílabas al ser pronunciadas presentan una longitud temporal relativamente constante. Esto significa que al pronunciar palabras cortas su duración es en la mayoría de las veces la misma para varias pronunciaciones de una misma palabra, sin embargo al pronunciar palabras más largas es más difícil mantener esta duración constante. Este problema impidió que se utilizaran palabras más largas en los comandos de voz, por ejemplo “Derecha” en vez de “Deré” o “Izquierda” en lugar de “Izké”. Para mejorar esto implementar un mecanismo que tome la palabra completa y la normalice a una longitud manejable, permitiendo ingresar correctamente los trozos a reconocer por las distintas

redes utilizadas en esta tarea habría sido una posible solución, lo que abriría la posibilidad de reconocer palabras más largas.

- Los valores de inicialización de los pesos sinápticos de la red neuronal juegan un papel crucial en su aprendizaje. Muchos autores recomiendan iniciar los pesos sinápticos con valores pequeños aleatorios para realizar los entrenamientos, sin embargo se comprobó que utilizando esta premisa la red neuronal no aprendía. En las gráficas no se observaba ningún punto, o en ocasiones una línea continua en la parte alta del gráfico que no presentaba variación. Con este problema se implementó una técnica de inicialización aleatoria de los pesos que permitió un aprendizaje adecuado de la red. Esta técnica permite una distribución uniforme de los pesos en un cierto intervalo con lo que asegura un funcionamiento adecuado en la etapa de aprendizaje.

- Sin duda, la utilización de redes neuronales en el tratamiento de problemas que los algoritmos comunes no pueden solucionar está aumentando rápidamente, así como el desarrollo de esta tecnología que permite agregar cierta "inteligencia" a los sistemas de procesamiento de información que la computación tradicional no ha podido entregarles. Su aplicación a un sinfín de problemas, desde reconocimiento de caracteres, clasificación de electrocardiogramas, pronóstico del tiempo hasta en la aprobación de créditos a personas hacen de esta técnica una potente herramienta tecnológica que servirá para agregarle a las máquinas un toque de humanidad y raciocinio. Y esto se debe en gran medida a que el hombre ha buscado desde mucho tiempo crear aparatos que imiten la capacidad natural del cerebro humano, permitiendo una comunicación sencilla con estos, dotándolos de muchas de sus capacidades y liberando al hombre de tareas que muchas veces se vuelven tediosas. Con todo esto se vislumbra un futuro con máquinas similares al humano en su forma de actuar, pero con el poder de cómputo que las caracteriza, ayudando en múltiples áreas el trabajo del hombre y reemplazándolo en muchas otras.

- Las posibles aplicaciones de este proyecto son muchas, sin embargo al plantear la idea, ésta fue pensada con el fin de servir de ayuda a personas con problemas motrices en el

comando de aparatos electrónicos, tales como una silla de ruedas eléctrica, y también el control de sistemas mecánicos por ejemplo para la manipulación de materiales peligrosos o grúas de manera que no necesite de sus manos para realizar la acción y permita un lenguaje de comunicación más natural con las máquinas.

REFERENCIAS BIBLIOGRÁFICAS

- LIBROS:

J.R. Hilera, V.Martínez**.** *Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones*.

Ben Kröse, Patrick Van Der Smagt, *An Introduction to Neural Networks,* 1996.

James A. Freeman, David M. Skapura, *Neural Networks, Algorithms, Applications, and Programming Techniques*, Addison-Wesley Publishing Company 1991.

Haykin Simon, Neural Networks, *A Comprehensive Foundation*, segunda edición Prentice Hall.

Pedro Isasi Viñuela, Inés M. Galván L, *Redes Neuronales Artificiales. Un Enfoque Práctico*, Prentice Hall 2004

- REVISTAS:

Vicente Pablo Guerrero Bote, *Inteligencia Artificial y documentación*, 2001.

J. Orozco García, Carlos A. Reyes García, *Clasificación de Llanto del Bebé utilizando una Red Neuronal de Gradiente Conjugado Escalado*.

Gamal Mahmoud Ali Sowilan, *Aplicación de las redes neuronales en los sistemas de control vectorial de los motores de inducción*, 200.

Pedro Larrañaga e Iñaki Inza, *Redes Neuronales*.

Flavio D. García, *Procesamiento de imágenes mediante Redes Neuronales*.

David Novo, Pere Martí, *Algoritmo de FFT por decimación en tiempo utilizando notación matricial*, Universidad Autónoma de Barcelona.

Rodrigo Huerta Cortés, *Transformada Rápida de Fourier*, Universidad Federico Santa María, 2003.

Luis Miguel Bergasa Pascual, *Introducción a los Modelos Ocultos de Markov*, Universidad de Alcalá.

Grupo de tratamiento Avanzado de Señal, *Análisis Homomórfico de la señal de voz*, Universidad de Cantabria.

Joseph P.Campbell, *Speaker Recognition*, IEEE 1997.

Antonio J. Rubio A. José C. Segura Luna, *Reconocimiento de Formas y Análisis de Imágenes,* Universidad de Granada.

- WEBs:

J.J. Merelo, *Mapa autoorganizativo de Kohonen,*
http://geneura.ugr.es/~jmerelo/tutoriales/bioinfo/Kohonen.html

Transformada Rápida de Fourier,
http://www.arrakis.es/~ppriego/fourier/fourier.htm

Don Cross, *Time domain filtering techniques for digital audio*,
http://groovit.disjunkt.com/analog/time-domain/index.html.

Modelos Ocultos de Markov,
http://www.virtual.unal.edu.co/cursos/ingenieria/2001832/lecciones/hmm1.html

APÉNDICE

Transformada Rápida de Fourier

Una transformada de Fourier es una operación matemática que transforma una señal de *dominio de tiempo* a *dominio de frecuencia* y viceversa. La Transformada de Fourier tiene la característica de que entrega información característica del espectro de una señal y reduce considerablemente la cantidad de información original, pudiendo, a partir del espectro, reconstruir la señal original. Es por ello que se eligió como técnica de extracción de características. Para el tratamiento de señales discretas y su implementación en un computador existe lo que se llama la Transformada Rápida de Fourier. Este algoritmo es una forma rápida de calcular la Transformada Discreta de Fourier y su poder radica en que disminuye drásticamente la cantidad de cálculos a realizar. Por ejemplo si se desea calcular la DFT directamente el número de cálculos está dado por N^2 operaciones, en cambio el cálculo de la FFT implica $N*\log_2(N)$, siendo N el número de puntos tomados para realizar el cálculo.

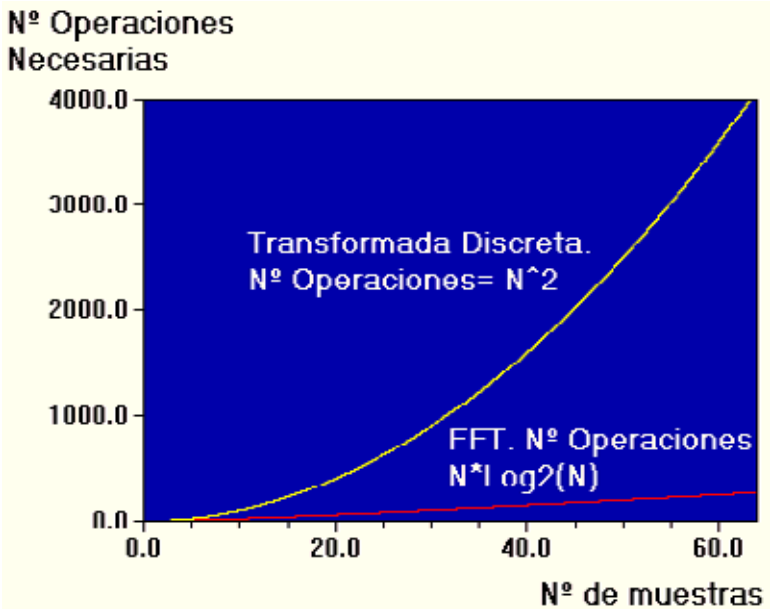


Fig.75 Gráfico comparativo de la eficiencia de FFT vs DFT.

La condición es que el número de muestras para el cálculo de la FFT debe ser igual a una potencia de 2. En la figura 75 se aprecia la diferencia en eficiencia.

A grandes rasgos el funcionamiento de la Transformada De Fourier consiste en descomponer el set de muestras de una DFT en N/2 puntos, luego esta DFT de N/2 se descompone en N/4 puntos, y así sucesivamente hasta llegar N/2 transformadas de 2 puntos. Una vez conseguido esto se reordenan los valores a la salida para obtener los coeficientes espectrales de la transformada. Para implementar la FFT existen dos procedimientos: diezmado en frecuencia (DIF del inglés Decimation In Frequency) y diezmado en el tiempo (DIT del inglés Decimation In Time). En este trabajo se utilizó el diezmado en frecuencia.

Estas son las ecuaciones que corresponden a la FFT.

$$X(2k) = \sum_{n=0}^{(N/2)-1} a(n) \cdot W_{N/2}^{nk} \qquad k = 0,1,\ldots,\left(\frac{N}{2}\right)-1$$

$$X(2k+1) = \sum_{n=0}^{(N/2)-1} b(n) \cdot W_N^n \cdot W_{N/2}^{nk} \qquad k = 0,1,\ldots,\left(\frac{N}{2}\right)-1$$

Parte del código utilizado para el cálculo de la FFT más abajo:

```
Do While BlockSize <= NumSamples
    DeltaAngle = AngleNumerator / BlockSize
    Alpha = Sin(0.5 * DeltaAngle)
    Alpha = 2! * Alpha * Alpha
    Beta = Sin(DeltaAngle)

    i = 0
    Do While i < NumSamples
      AR = 1!
```

```
        AI = 0!

        j = i
        For n = 0 To BlockEnd - 1
           k = j + BlockEnd
           TR = AR * RealOut(k) - AI * ImagOut(k)
           TI = AI * RealOut(k) + AR * ImagOut(k)
           RealOut(k) = RealOut(j) - TR

           ImagOut(k) = ImagOut(j) - TI
           RealOut(j) = RealOut(j) + TR
           ImagOut(j) = ImagOut(j) + TI
           DeltaAr = Alpha * AR + Beta * AI
           AI = AI - (Alpha * AI - Beta * AR)
           AR = AR - DeltaAr
           j = j + 1
        Next

        i = i + BlockSize
     Loop

     BlockEnd = BlockSize
     BlockSize = BlockSize * 2
  Loop
End Sub
```

Interfaz PC brazo

Para conseguir la comunicación entre el PC el brazo mecánico y el display fue necesario construir un interfaz que realice esta conexión, permitiendo el control del display y el gobierno preciso de los movimientos de los servomotores. La conexión se realizó a través del puerto paralelo del computador, enlazándose con un microcontrolador 16f84a. Para la alimentación del circuito se utilizaron 3 reguladores, 2 lm7805 y un lm7812, sin transformador ni puente de diodos. Previo a esto, las señales provenientes del PC pasan a través de 2 buffer cuya finalidad es proteger al puerto. Desde ahí, se distribuyen hacia el display y el microcontrolador.

En el esquema mostrado más abajo se puede entender mejor el funcionamiento del interfaz:

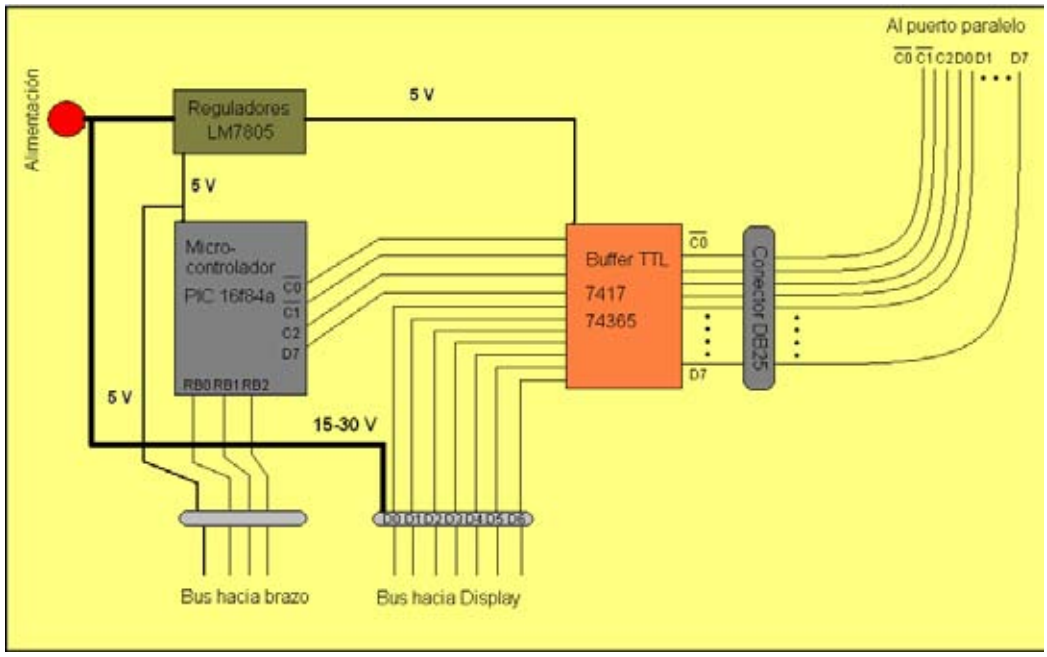


Fig.76 Esquema Interfaz PC-Brazo.

Los buffer utilizados fueron los modelos 7417 y 74365 cuya función era proteger el puerto paralelo de posibles cortocircuitos. Más abajo se muestran algunas fotos del interfaz.

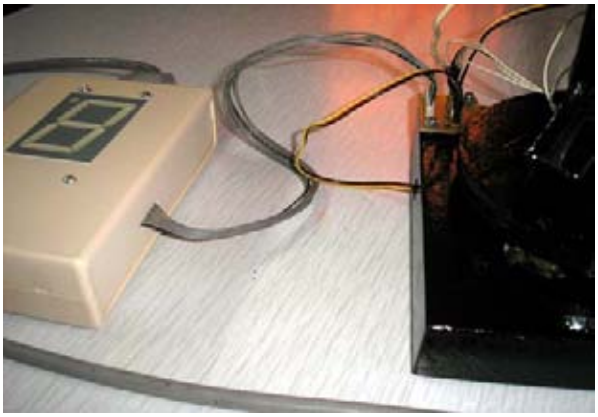
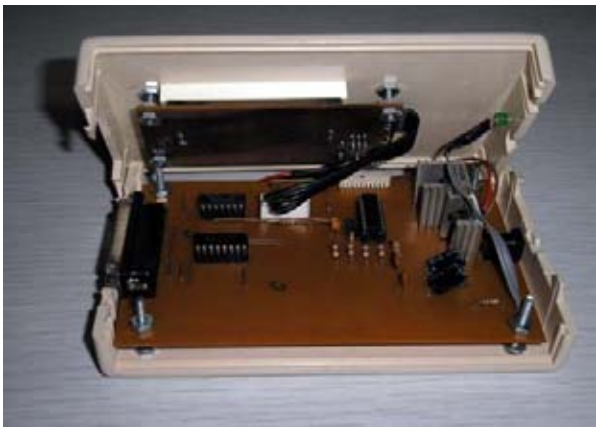


Fig.77 y 78



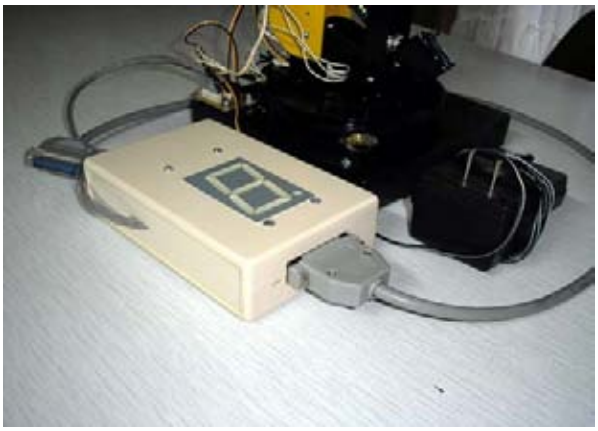


Fig. 79 y 80

El Display

El display de 7 segmentos es un modelo SA-XXXX, ánodo común. Los bits del puerto paralelo utilizados para el control del display son DB0, DB1, DB2,…DB6. Estos pasan directamente desde los buffer hacia el display, saltándose el microcontrolador. La generación de los caracteres se realizó de acuerdo a la siguiente tabla, considerando que para activar un led se debe enviar desde el puerto paralelo un 0 lógico y para apagarlo un 1 lógico.

Caracter formado	**D4**	**D3**	**D2**	**D1**	**D0**	**D5**	**D6**
A	0	0	0	1	0	0	0
B	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
D	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0
G	0	1	0	0	0	0	0
H	1	0	0	1	0	0	0

El esquema de abajo muestra el modo de conexión del display.

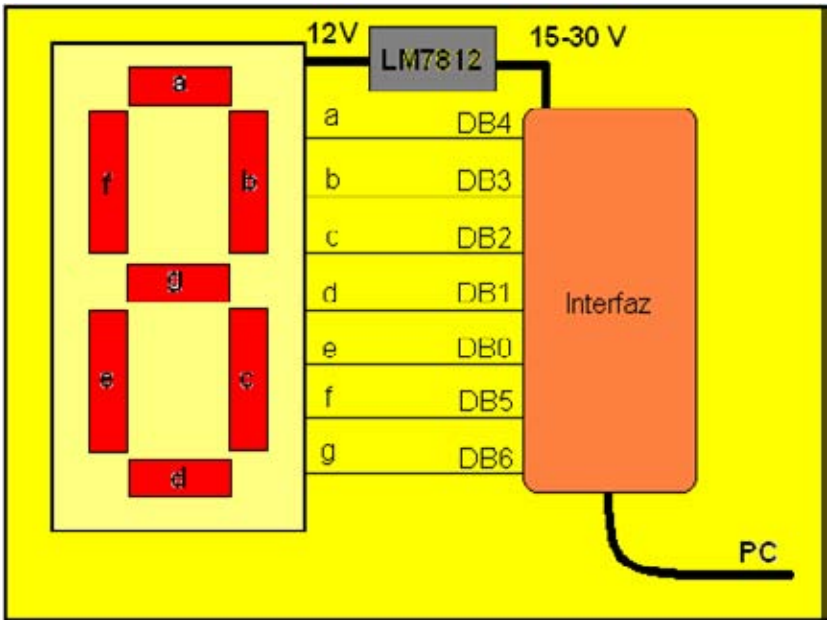


Fig.81 Diagrama de conexión

El display se montó en el mismo interfaz como se puede apreciar en las siguientes fotos.

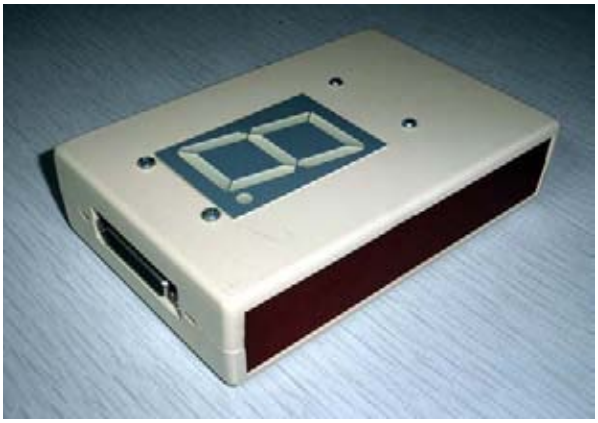



Fig.82 y 83

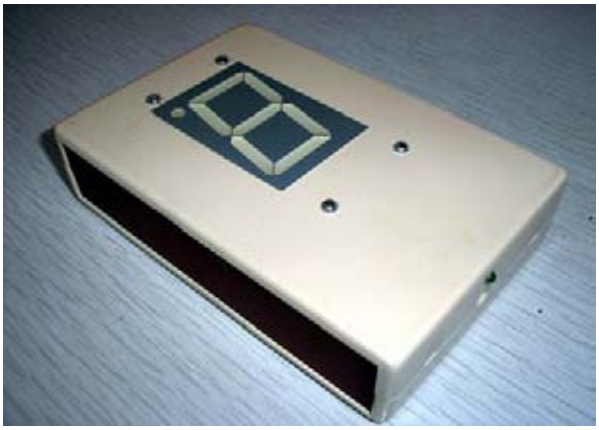
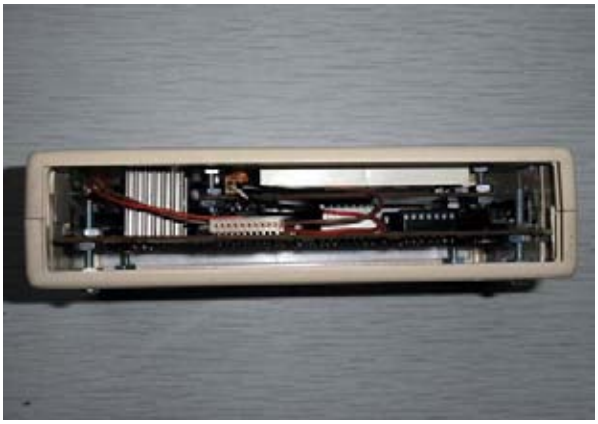



Fig.84 y 85

El Control del brazo

El brazo mecánico se construyó en base a 3 servomotores Hitec HS-311 de 3.0kg*cm y 5 volt de alimentación modificados para tener un recorrido de 360º libres y actuar como motor. El servo es un pequeño pero potente dispositivo que dispone en su interior de un pequeño motor con un reductor de velocidad y multiplicador de fuerza, también dispone de un pequeño circuito que gobierna el sistema. Los servomotores están fabricados para funcionar en posiciones precisas con un ángulo total de trabajo de 180º en la mayoría de ellos.

Para controlar un servo tendremos que aplicar un pulso de duración y frecuencia específicas. Todos los servos disponen de tres cables dos para alimentación Vcc y Gnd y otro cable para aplicar el tren de pulsos de control que harán que el circuito de control diferencial interno ponga el servo en la posición indicada por la anchura del pulso y en para este caso en particular se mueva hacia la derecha o izquierda .

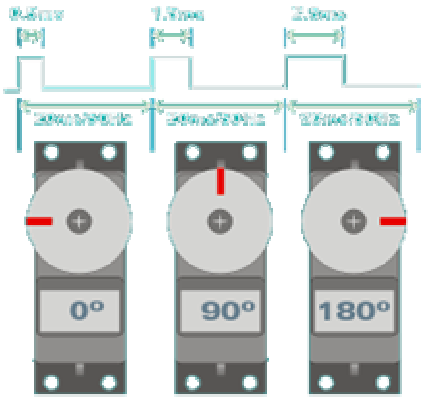


Fig.86 Movimientos Servomotor

En este caso los servomotores modificados actuarán de acuerdo al siguiente gráfico:

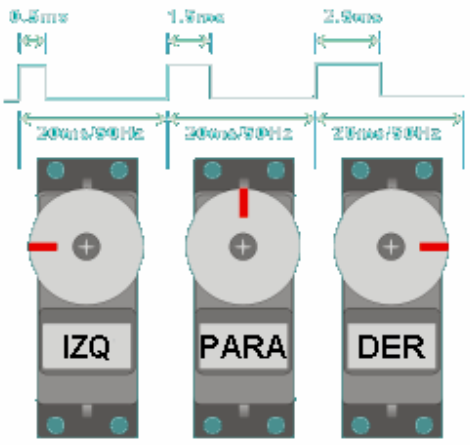


Fig.87 Movimientos Servomotor

Cuando el motor está modificado para funcionar libremente en 360º el pulso más pequeño provoca un giro a máxima velocidad hacia la izquierda, al contrario, cuando se genera un pulso con la duración más larga este hará que el motor gire rápidamente hacia la derecha, y a medida que los pulsos tanto el más largo como el más corto se van acercando al pulso central, va

disminuyendo paulatinamente su velocidad hasta detenerse completamente. Para el caso del motor Hitech se tienen los siguientes valores para su funcionamiento y colores para su conexión.

	Duración pulso (ms)				disposición de cables		
Fabricante	**min.**	**neutral.**	**máx..**	**Hz**	**+ batt**	**-batt**	**pwm.**
Futaba	0.9	1.5	2.1	50	rojo	negro	blanco
Hitech	0.9	1.5	2.1	50	rojo	negro	amarillo

Fig.88 Tabla de valores para los movimientos del servo y código de colores.

El microcontrolador utilizado para el control de los motores del brazo es un PIC 16F84a de Microchip, dotado de 20 pines, con 2 puertos de 8 y 5 pines, todos con funcionamiento digital. Se utilizó un microcontrolador por la razón de que los tiempos de duración de los pulsos deben ser muy precisos para conseguir el movimiento deseado y este dispositivo entrega la precisión necesaria. Al tratar de realizar esto directamente por medio del puerto del computador se encontraron problemas de sincronismo, entre el software Visual Basic y el puerto, resultando esto en pulsos con duración variable y respuestas indeseadas por parte de los motores. El microcontrolador su utilizó con un cristal de cuarzo de 10 Mhz

El diagrama de pines de este microcontrolador es el que sigue:

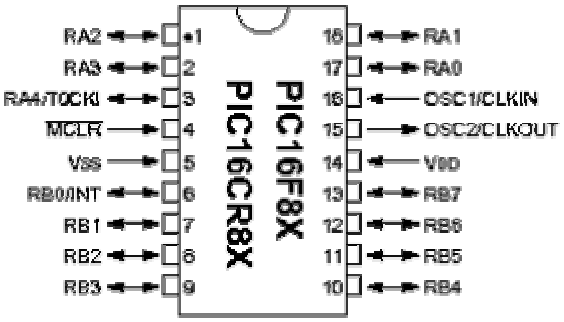


Fig.89 Diagrama de pines PIC 16f84a

El esquema a continuación muestra esquemáticamente la comunicación entre el microcontrolador y los motores.

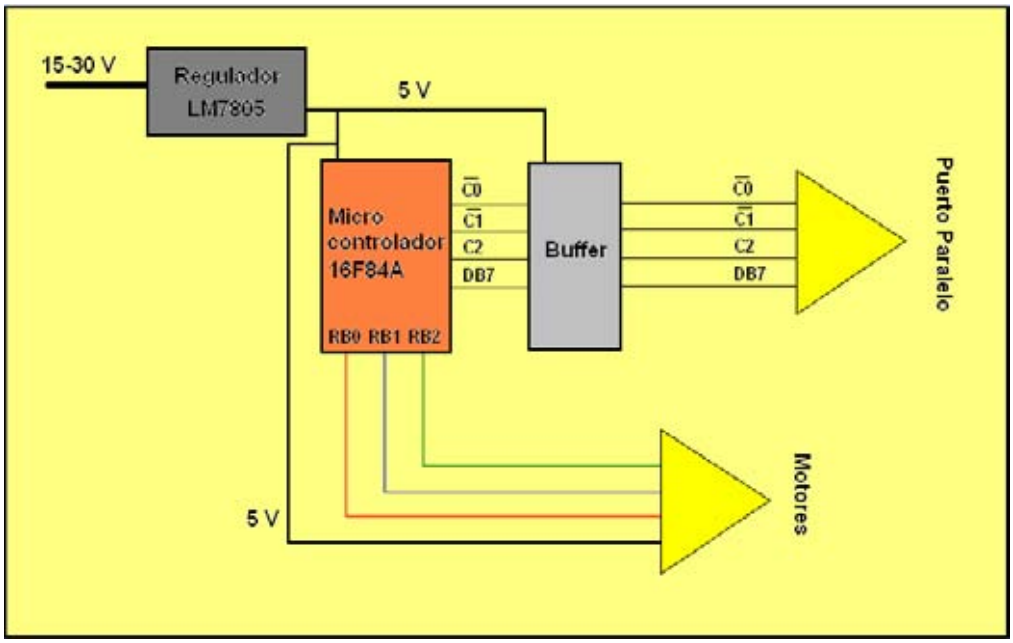


Fig.90 Diagrama Interfaz.

Y por el lado de los motores se puede ver esto:

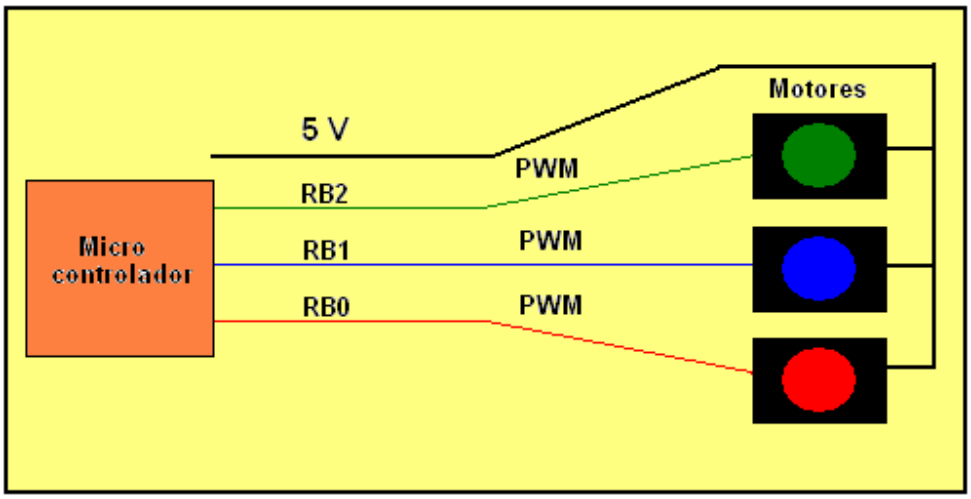


Fig.91 Diagrama Interfaz

El microcontrolador comanda los movimientos de los motores de acuerdo a las instrucciones que recibe desde el puerto paralelo, y esto lo realiza siguiendo las reglas de acuerdo a las siguientes tablas.

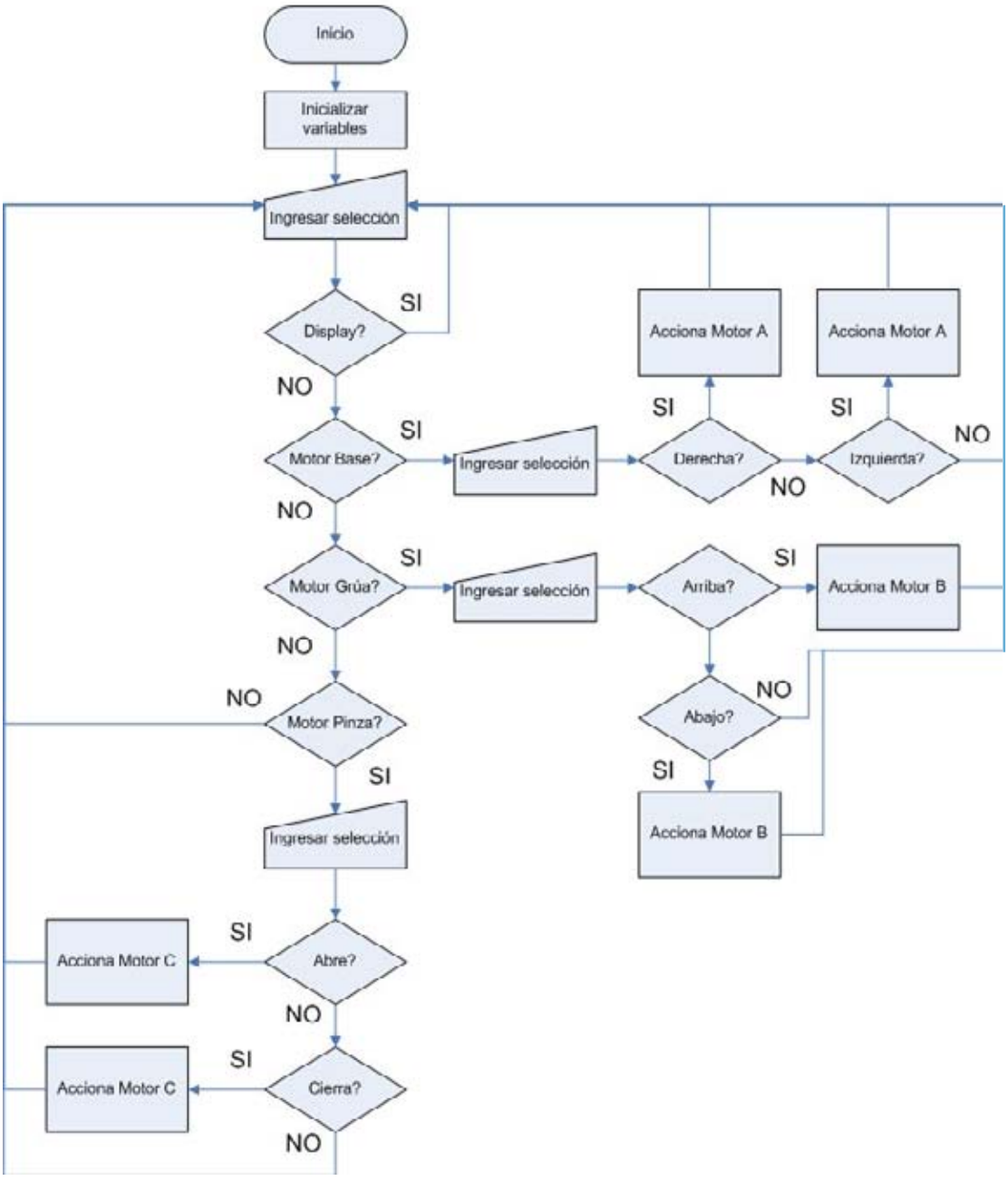
Dispositivo	**C0**	**DB7**	
	RA1	**RA0**	**Numero Decimal**
Display	0	0	0
Base	0	1	1
Grúa	1	0	2
Pinza	1	1	3

Cuando se ha seleccionado el display desde el PC, el microcontrolador solo queda a la espera de recibir una instrucción para controlar algún motor, mientras tanto no ejecuta ninguna acción. Cuando es seleccionado alguno de los 3 motores el micro revisa inmediatamente en qué dirección debe mover este motor. Esto lo realiza siguiendo las reglas de la siguiente tabla.

Acción	**C2**	**C1**	
	RA3	**RA2**	**Número Decimal**
Detiene	0	0	0
Derecha/Toma/Abajo	0	1	1
Izquierda/Suelta/Arriba	1	0	2
Detiene	1	1	3

Cuando se selecciona algún sentido de giro el microcontrolador genera un tren de pulsos para controlar por medio del PWM el sentido. A continuación el diagrama de flujo del funcionamiento del microcontrolador y posteriormente el código escrito en Assembler.

Programa PIC



Y el código en Assembler:

```
Monitor                 ;verifico si se ha seleccionado Display
        movlw 3         ;cargo el b '11' para hacer un and con los 2 primeros bit
                        del registro A
        andwf porta,0   ;guardo el resultado en el acumulador
        movwf comp      ;guardo el valor del acumulador en un registro auxiliar
        movlw 0         ;cargo un cero para compararlo con el valor del registro
                        auxiliar y determinar si es un cero.
        xorwf comp,1    ;realizo un xor entre el acumulador que tiene un cero y el
                        registro comp, si es igual a cero entonces el resultado de
                         la operacion es cero y el bit z del registro de estado esta
                        en 1
        btfss estado,2  ;pregunto si es uno el bit z del registro de estado
        goto  monitor2  ;revisa si se ha seleccionado el motor de la base
        goto  monitor   ;vuelve a monitorear todo.
```

Lo que se hace aquí es comprobar si se ha enviado un 0 a los 2 primeros bit del registro A, y en caso de que sea así se vuelve al comienzo, repitiendo la misma acción hasta que el valor enviado a los dos bit del registro sean distintos de 0. Este código se repite para el caso de selección de otros motores, difiriendo solo en el valor que se busca y las acciones que se realizarán en caso de que esto ocurra, por ejemplo en el caso de seleccionar el motor de la base se tiene lo siguiente:

```
Monitor2                ;verifico si se ha seleccionado el motor de la base

        movlw 3         ;cargo el b '11' para hacer un and con los 2 primeros bit
                        del registro A
        andwf porta,0   ;guardo el resultado en el acumulador
        movwf comp      ;guardo el valor del acumulador en un registro auxiliar
        movlw 1         ;cargo un uno para compararlo con el valor del registro
```

```
                        auxiliar y determinar si es un uno
xorwf  comp,1           ;realizo un xor entre el acumulador que tiene un uno y el
                        registro comp, si es igual a cero entonces el resultado de
                        la operacion es cero y el bit z del registro de estado esta
                        en 1.
btfss  estado,2         ;pregunto si es uno el bit z del registro de estado
goto   monitor3         ;va a monitorear si se ha seleccionado el motor de la grua
goto   base             ;va a monitorear los bit 3 y 4 del registro A para
                        ;comprobar el sentido del movimiento.
```

Para los motores de la grúa y la pinza el código funciona del mismo modo y solo cambia el número que se busca y el pin por el cual se envía la señal PWM que controlará el motor seleccionado.

Siguiendo con el trozo de código anterior, suponiendo que se elige mover el motor de la base del brazo, una vez que se detecte la combinación de bits correcta el programa salta a la subrutina llamada Base, la que tiene por finalidad reconocer cual es el sentido del movimiento que se quiere dar al motor seleccionado. A continuación un pedazo de este código:

```
Base                    verifica si se ha ordenado la detención del motor

movlw  12               se carga un 12 en el registro de trabajo w.
andwf  porta,0          se realiza una operación AND con el puerto A  y el
                        registro de trabajo
movwf  comp             se traslada el contenido del registro w al registro comp
movlw  12               se carga el registro w con 12
xorwf  comp,1           se efectúa una operación x-or entre w y comp
btfss  estado,2         en caso de que el resultado sea 0 el bit z del registro
                        estado será 1 y saltará hacia monitor, iniciándose el
                        ciclo nuevamente.
goto   Base2
```

```
        goto    monitor

Base2                   verifica si se ha seleccionado que el motor gire hacia la
                        derecha
        movlw   12
        andwf   porta,0
        movwf   comp
        movlw   4
        xorwf   comp,1
        btfss   estado,2
        goto    Base3
        goto    derecha
```

En trozo de programa se puede ver como se monitorean los pines del puerto A en busca de las instrucciones que posteriormente realizará el micro. En la subrutina Base se monitorean los bits RA2 y RA3 con el fin de determinar si se ha enviado un "00" o un "11" que definen el estado en detención del motor. En caso de que esto se cumpla vuelve a la rutina Monitor, en caso contrario sigue a la siguiente subrutina Base2 donde se monitorea en búsqueda de la orden de girar hacia la derecha, si esta no es encontrada sigue a la subrutina Base3 que busca por la orden de giro hacia la izquierda, y en caso de no encontrarse vuelve a la rutina principal Monitor.

Más abajo un trozo de la rutina para mover un motor.

```
Derecha                 envía los trenes de pulsos necesarios para hacer girar al
                        motor  hacia la derecha.
        bsf portb, 0    envía un pulso
        call DEMORA3    mantiene el pulso por un tiempo determinado
        bcf portb,0     envía un 0
        call DEMORA4    mantiene el cero por un tiempo determinado
        goto    monitor vuelve a la rutina principal
```

El Puerto paralelo

Un puerto es un medio por el cual un computador puede comunicarse con otros dispositivos. El puerto paralelo lleva su nombre porque es capaz de enviar 8 bits al mismo tiempo. En sus comienzos los puertos paralelos fueron desarrollados para conectar impresoras y con el correr del tiempo este puerto se convirtió en un estándar con lo que muchos otros dispositivos hicieron uso de él para conectarse con otros aparatos. Cuando un PC envía datos a otro dispositivo usando el puerto paralelo, este manda **8 bits** de datos (1 byte) a la vez, de forma distinta al puerto **serie** el cual manda los 8 bits uno detrás de otro por el mismo cable. Un bit está representado por un nivel TTL presente en los pines del puerto. El puerto paralelo es capaz de transmitir datos de 150 kilobytes a 2 Mbytes de datos por segundo.

Este puerto consta de 25 pines, de los cuales 8 corresponden al registro de datos, 5 al registro de estado, 4 al registro de control y 8 pines a tierra. Cada registro posee una dirección que lo identifica, de esta forma si se desean escribir datos o leerlos de algún registro del puerto, se deben enviar a la dirección correspondiente.

Las referencias a cada registro del puerto se realizan de la siguiente forma:

- Base (datos) = base +0
- Estado = base +1
- Control = base +2

Para este caso la dirección hexadecimal base del PC es la 378h, entonces las direcciones del registro de datos, estado y control quedan de la siguiente forma:

- Datos = 378h
- Estado = 379h
- Control = 37Ah

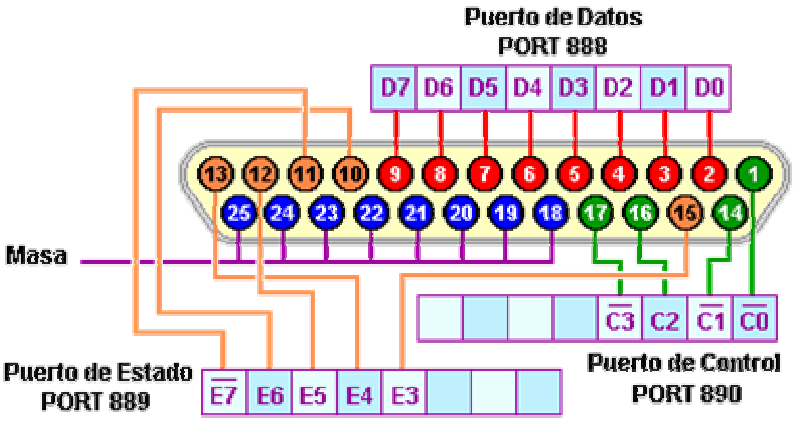


Fig.92 Diagrama de pines del puerto paralelo

Modos SPP, EPP Y ECP

El puerto paralelo puede funcionar en distintos modos de acuerdo al dispositivo al que se ha conectado, entre ellos:

Modo SPP: permite comunicación bidireccional full duplex, a velocidades de hasta 150 kilobytes por segundo

Modo EPP: diseñado para comunicar dispositivos de almacenamiento permite la comunicación bidireccional hasta a 2 Mbytes por segundo.

Modo ECP: permite comunicación bidirecciónal, hasta 2 Mbytes por segundo. Posee capacidades DMA, FIFO y compresiones RLE.

El Brazo Mecánico

Con el fin de visualizar una aplicación útil para el comando por voz se construyó un sencillo brazo mecánico. Este consta de 3 movimientos, uno horizontal para desplazamiento hacia la derecha e izquierda, otro vertical para bajar y subir el brazo y un movimiento de agarre por medio de una pinza que permite asir objetos al brazo. Estos movimientos los realiza gracias a 3 servomotores controlados por medio de un microcontrolador y 2 finales de carrera que permiten al microcontrolador detener el movimiento del motor vertical cuando la estructura ha llegado a un extremo. El primero de ellos, ubicado en la base realiza los movimientos horizontales, el segundo ubicado en la plataforma que sujeta al brazo tiene conectado un hilo que se une con el brazo a través del soporte principal de este, para controlar el movimiento vertical. El último motor está conectado a la pinza y controla apertura y cierre de ésta.

Fig.93 Base del Brazo Mecánico

En la base su puede apreciar el motor en el centro, y 4 ruedas que sujetan la plataforma al girar. También se puso una pequeña placa con 4 conectores para los motores y los finales de carrera, y un pequeño puerto para conectar con la interfaz en la izquierda superior de la figura 93.

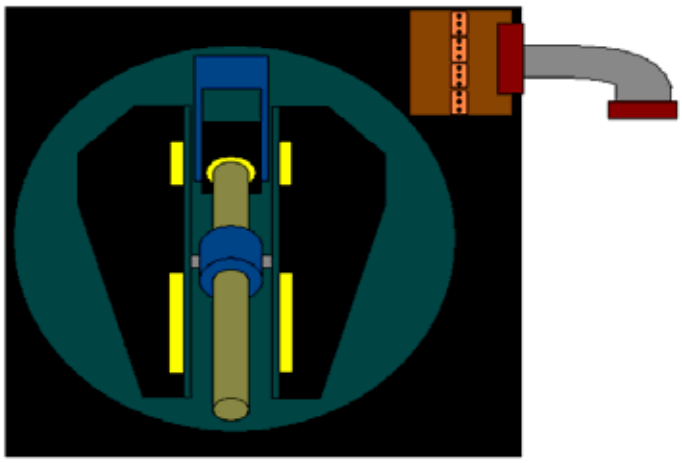


Fig.94 Vista superior Base con plataforma incluida.

En la figura 94 se puede ver el mecanismo para el movimiento vertical del brazo, donde el motor ubicado en contra del brazo tiene conectado a su eje un hilo que a su vez está acoplado por medio de un puente y una tuerca al soporte principal del brazo. Este motor no puede moverse verticalmente, pero si tiene un movimiento circular para dar espacio al movimiento de la estructura. De esta forma al girar el motor que tiene una posición vertical fija se hace girar el hilo que hace subir o bajar a la tuerca que se encuentra conectada a los soportes, consiguiendo el movimiento vertical de toda la estructura. Esto permite un movimiento vertical suave y aprovecha todo el torque del motor.

Se puede ver un soporte paralelo al principal, pero más pequeño, en color negro y cuya función es mantener en posición horizontal la pinza en todo momento. De esta forma al bajar y coger por ejemplo un vaso con agua, este siempre se mantendrá en posición horizontal de manera que no se vierte su contenido cuando la estructura suba y se mueva.

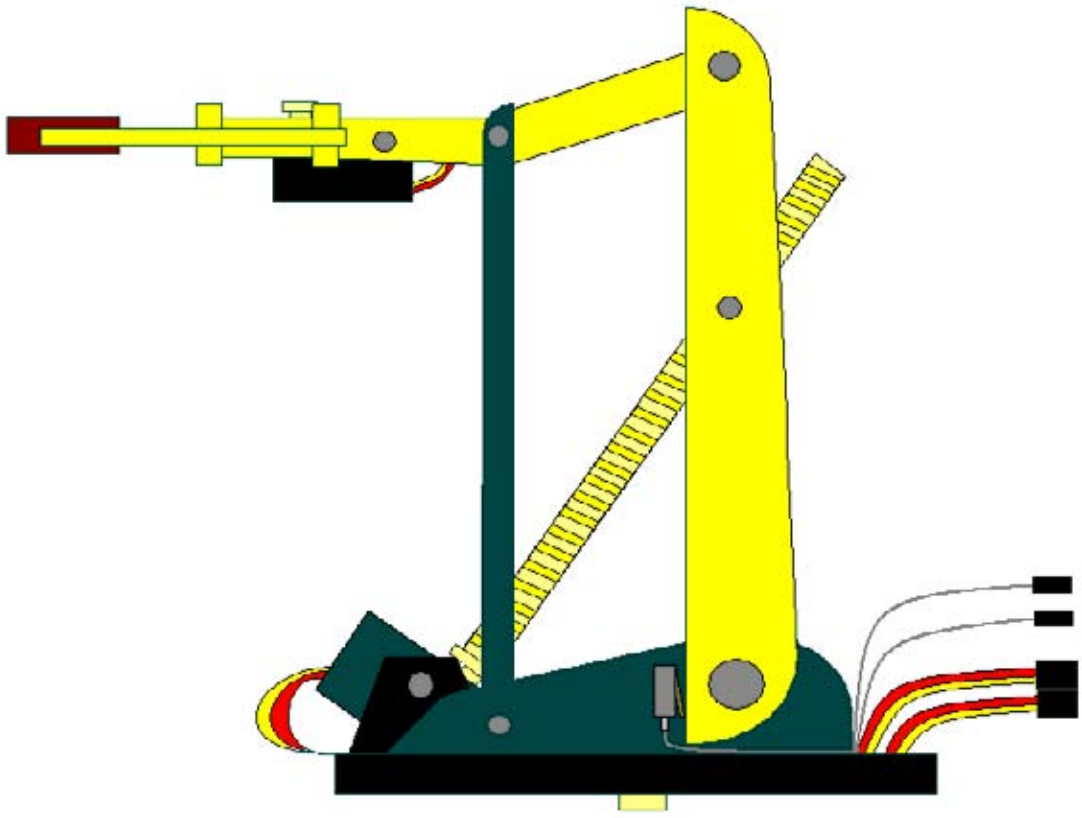


Fig.95 Plataforma y Brazo Mecánico

En la figura 95 se observa el motor ubicado en contra del brazo fijo en la plataforma con el hilo conectado a su eje y este conectado a los soportes principales del brazo por medio de una tuerca.

Para realizar los movimientos de agarre se construyó una pinza como la que se ve en la figura 96. Esta a través de un mecanismo de engranajes se conecta a un motor posicionado en su parte inferior que controla los movimientos de apertura y cierre.

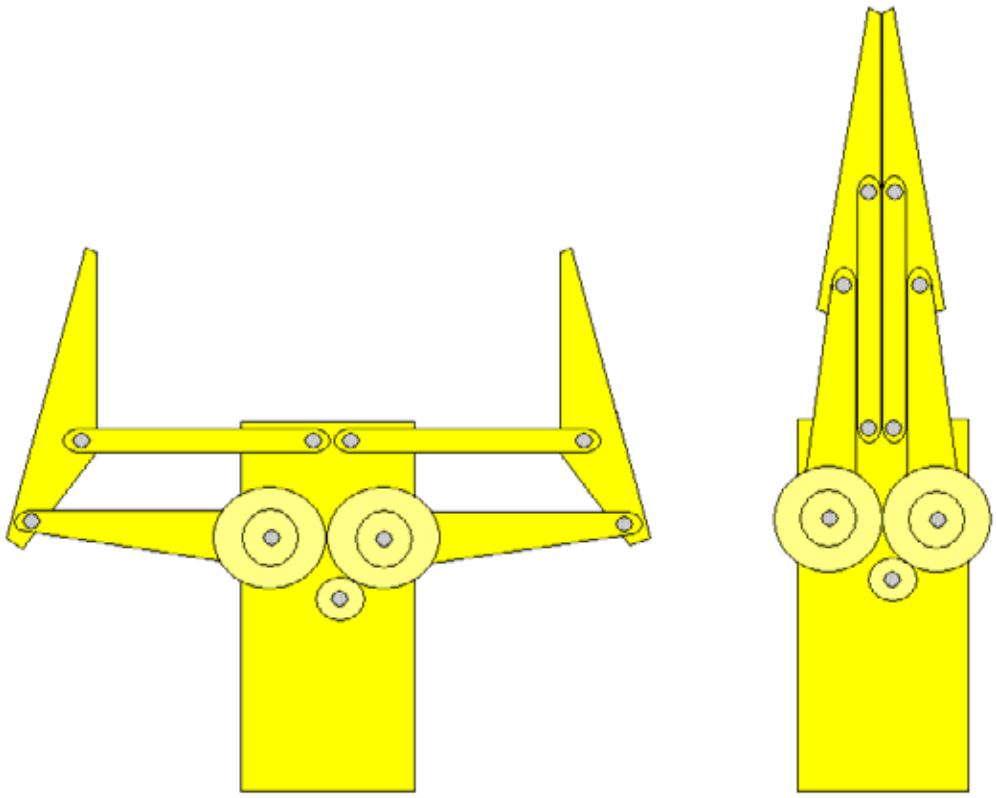


Fig.96 Pinza

Juntando todas las partes, el brazo mecánico queda constituido de acuerdo a la siguiente figura donde se observa como la plataforma se conecta al eje del motor en la base y las ruedas la sostienen.

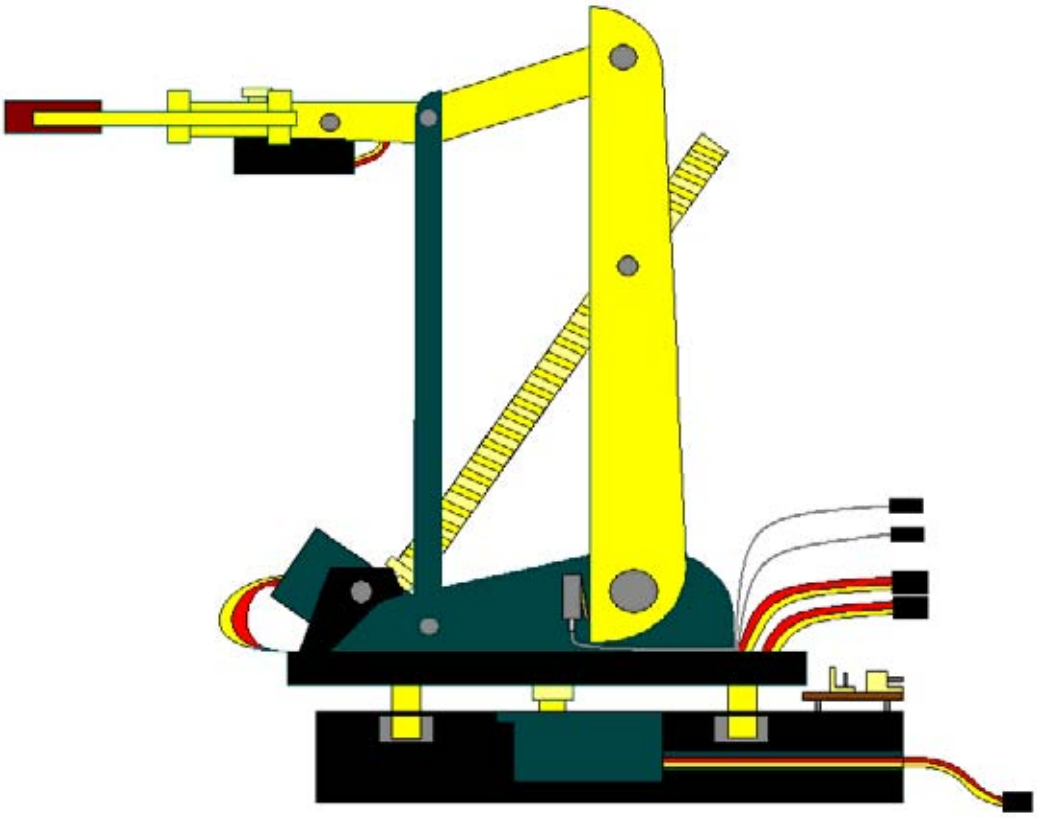


Fig.97 Brazo Mecánico completo

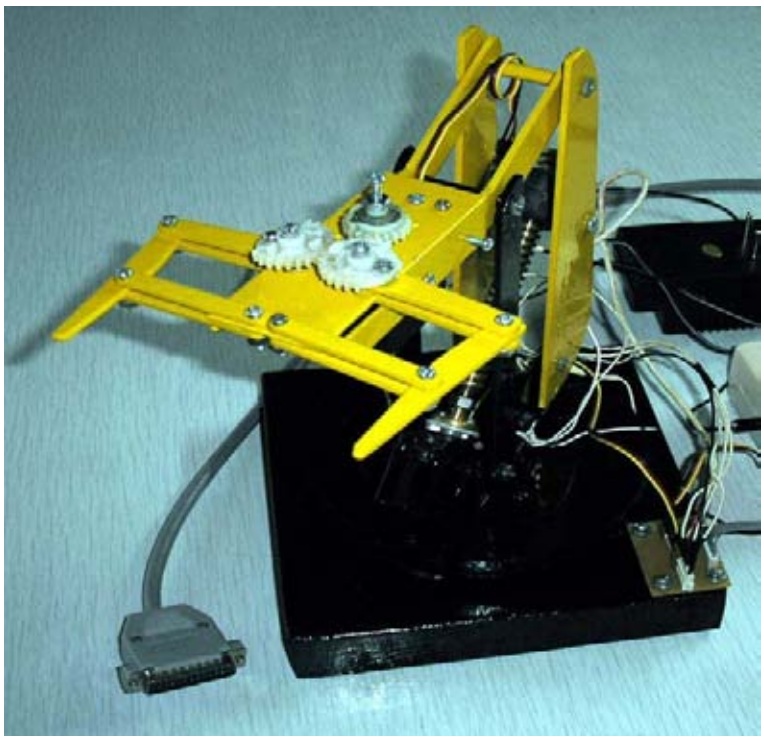


Fig.98 Brazo Mecánico Real

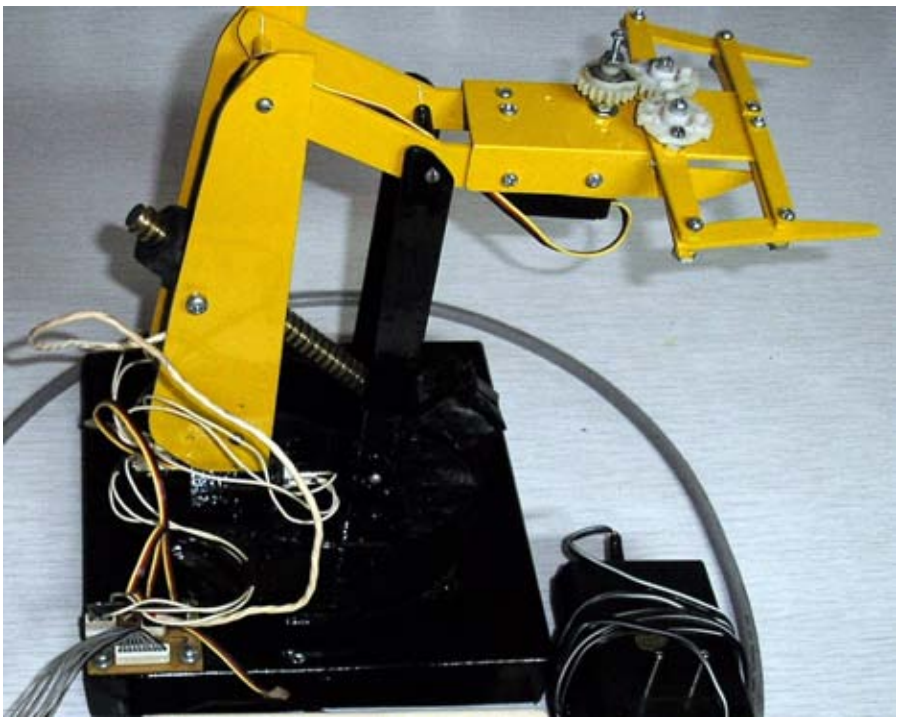


Fig.99 Brazo Mecánico

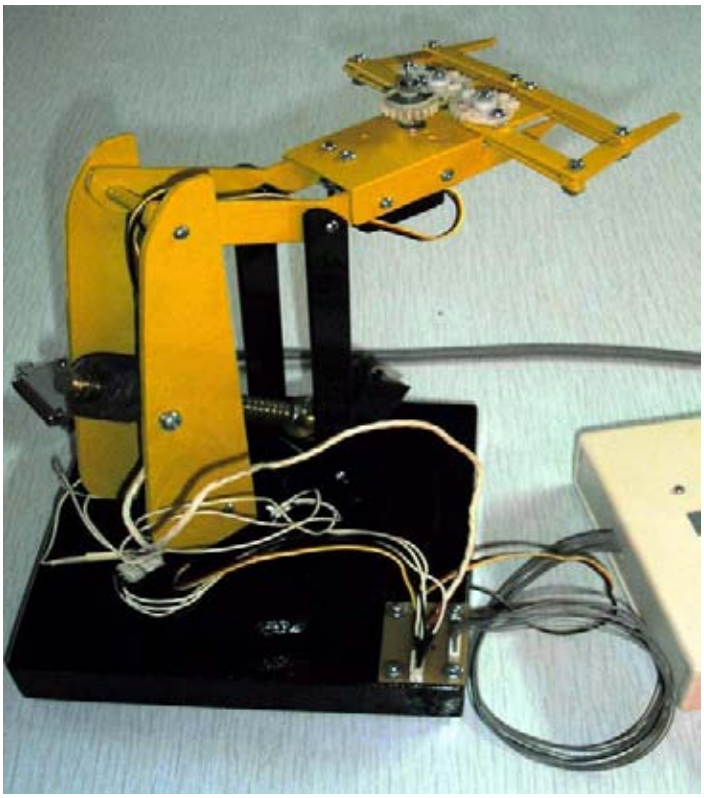
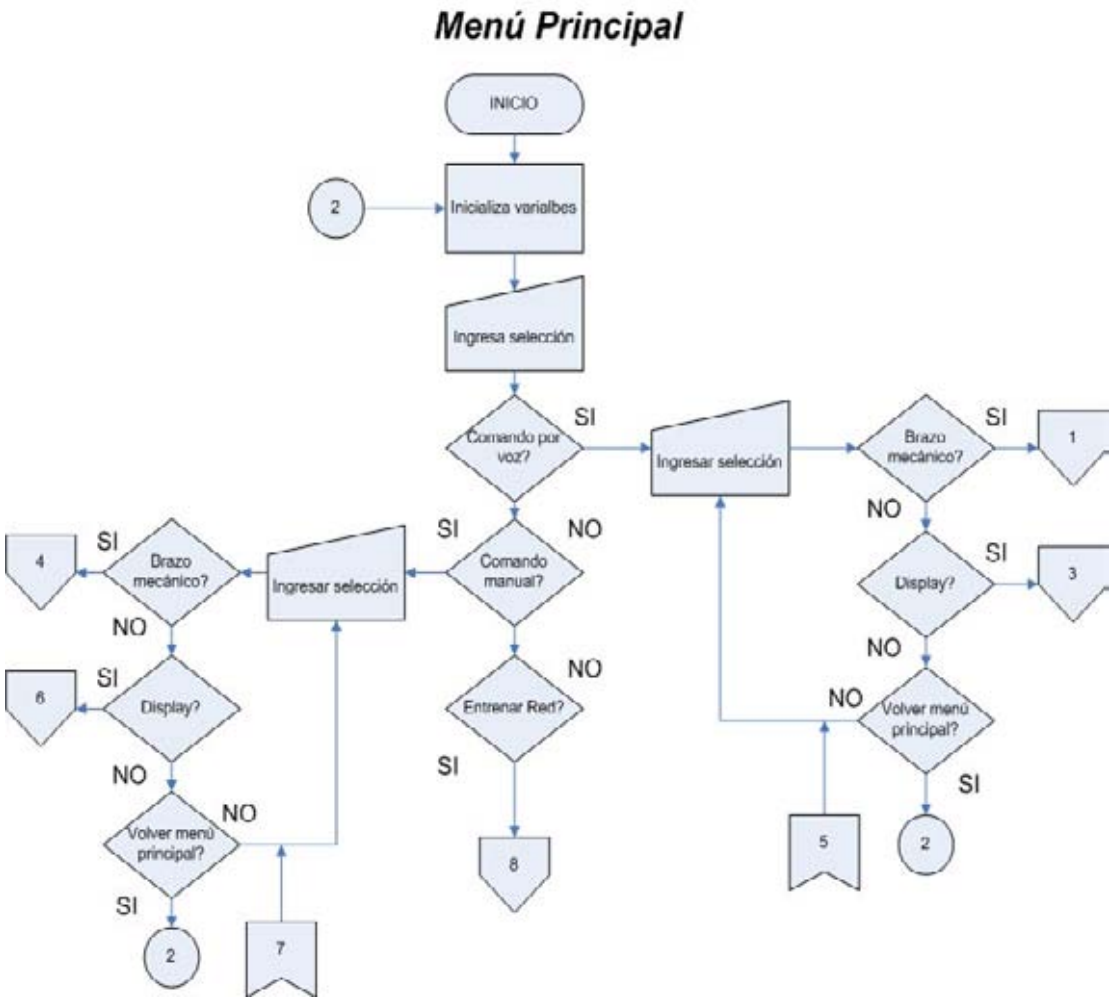


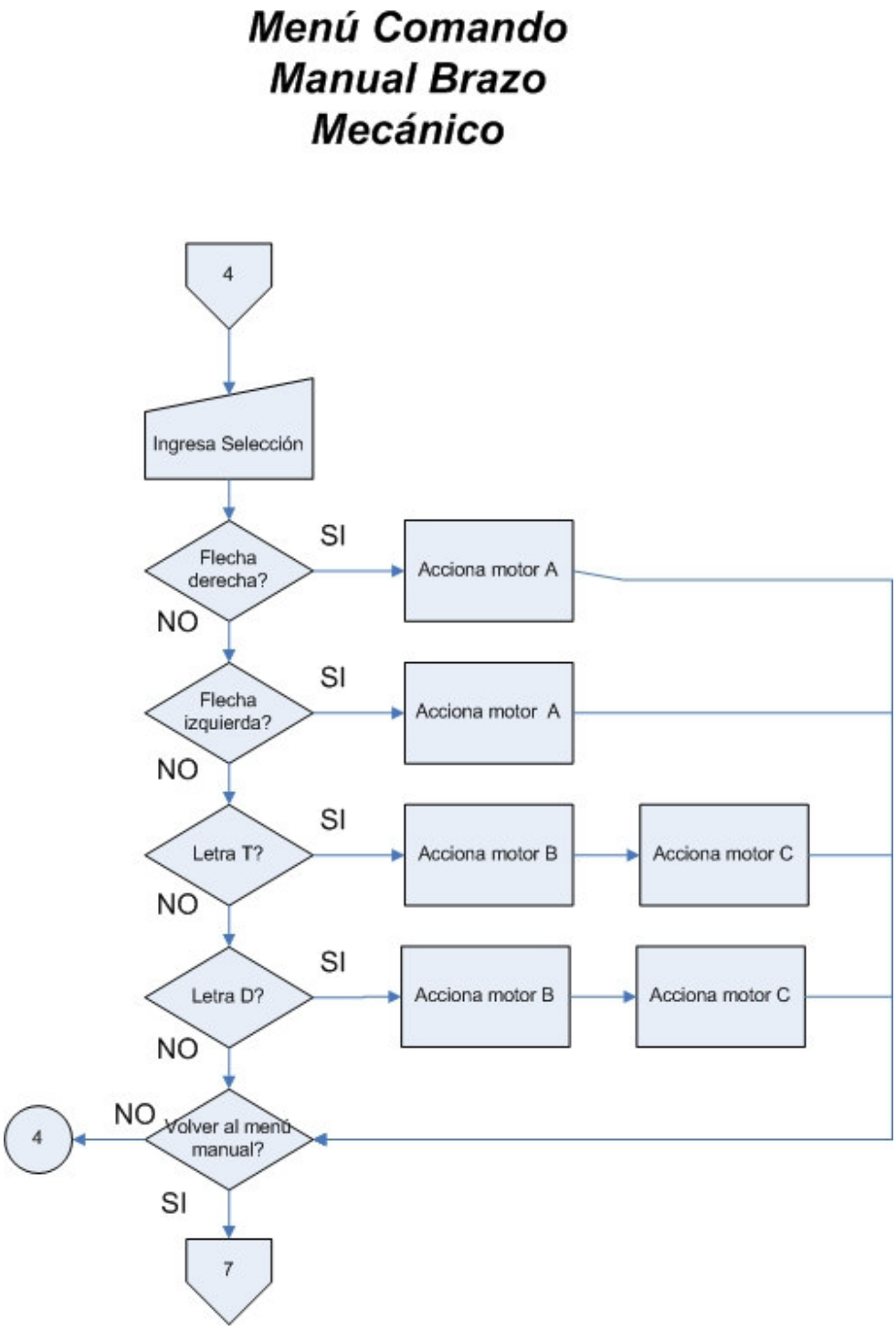
Fig.100 Brazo Mecánico

Diagrama de flujo del interfaz del sistema

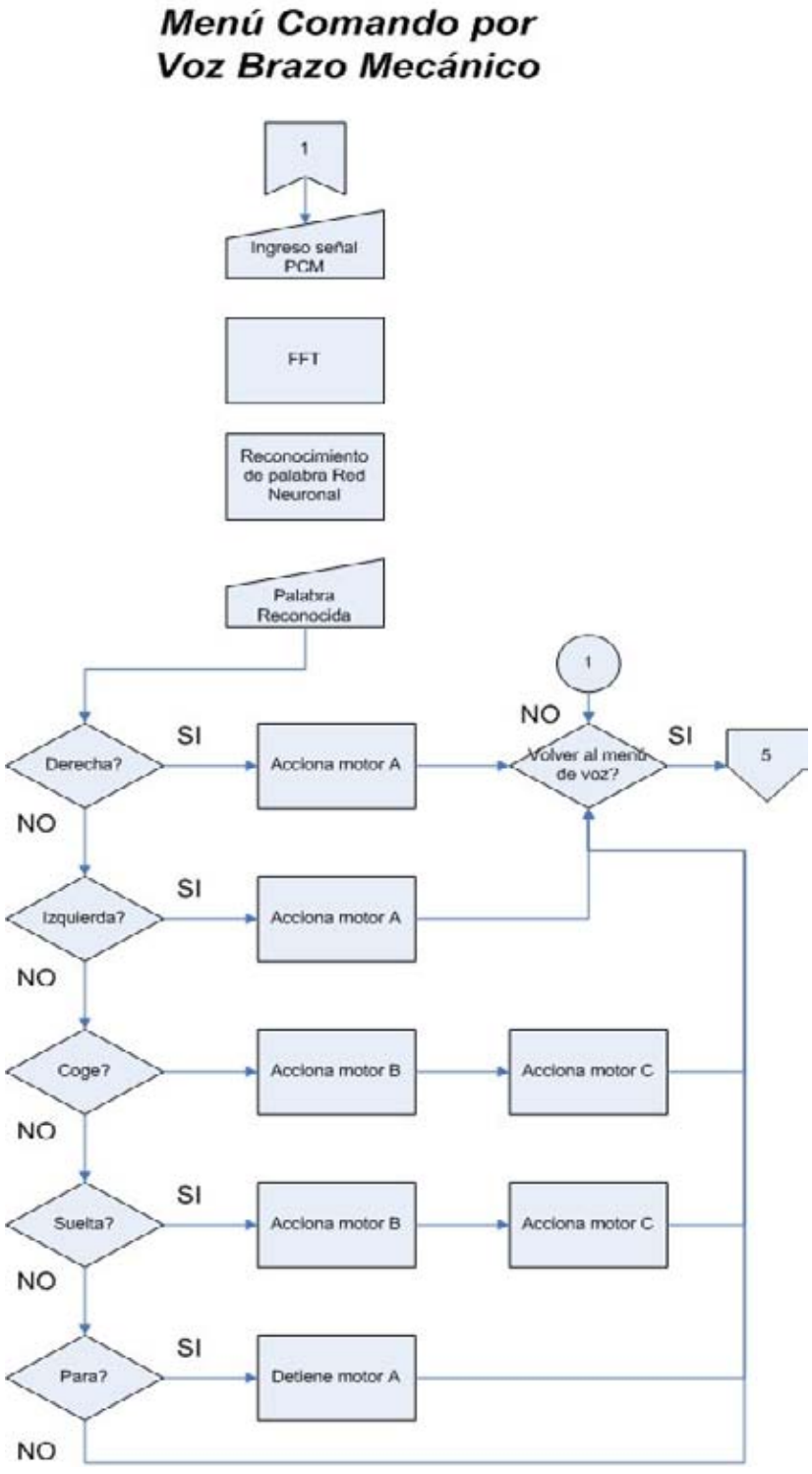
El programa principal sigue el siguiente esquema:



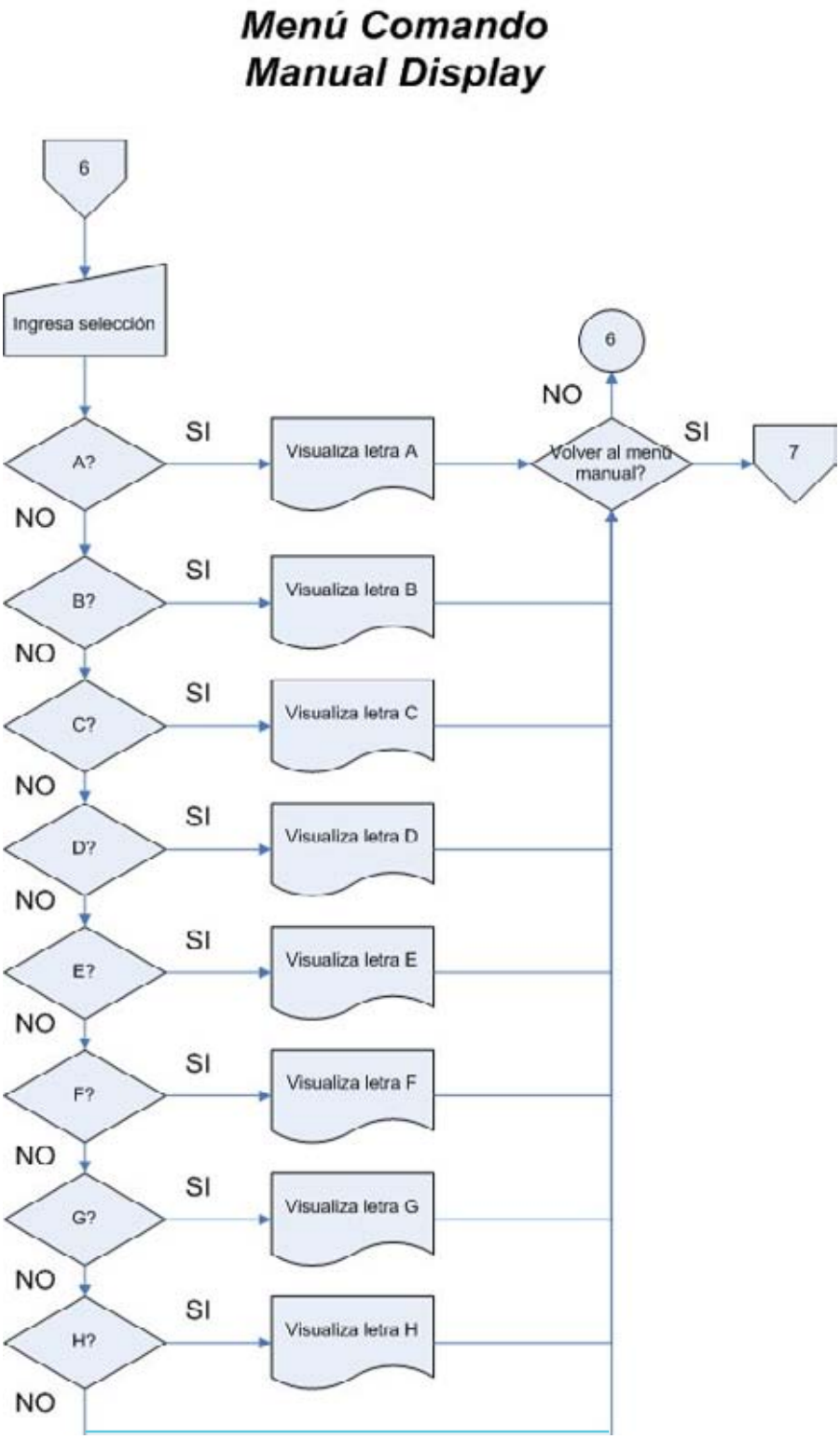
Para el caso del control del brazo mecánico en forma manual se siguió el siguiente esquema:



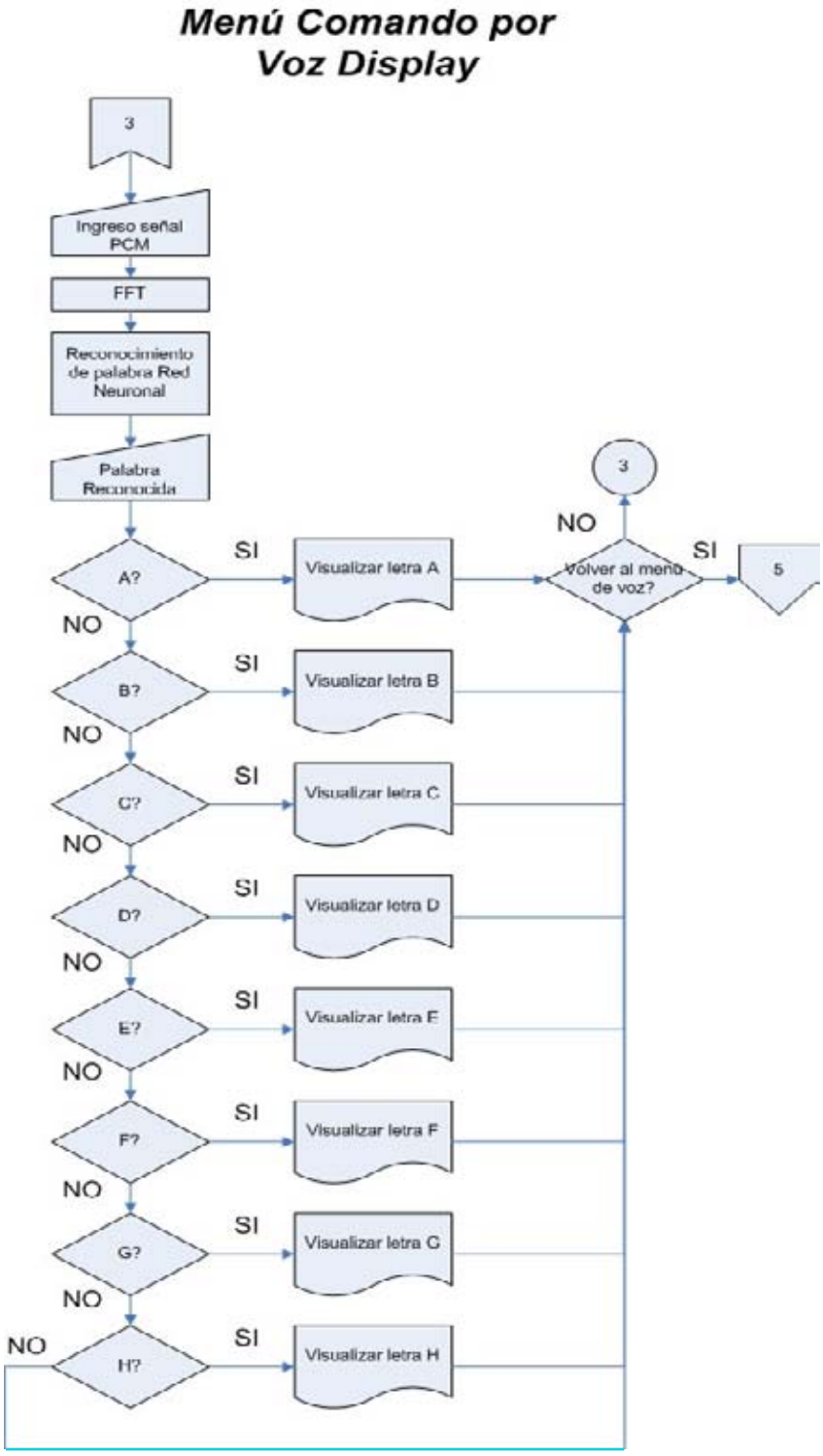
El menú para el control del brazo en modo de reconocimiento de comandos sigue el siguiente diagrama:



Para el caso del control manual del Display se siguió el siguiente diagrama:



Y para el modo de control por comandos de voz del Display se siguió el siguiente diagrama:



Para el modo de entrenamiento se utilizó el siguiente diagrama: