UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

VLADIMIR MOSQUERA CERQUERA.

VÍCTOR HUGO MONCAYO RUIZ.

CONTROL DE APLICACIONES MEDIANTE COMANDOS ORALES RECONOCIDOS POR REDES NEURONALES

Monografía presentada como requisito para optar
al título de Ingeniero en Electrónica y Telecomunicaciones

Director
Ing. ELENA MUÑOZ ESPAÑA.

Departamento de Electrónica Instrumentación y Control.
Popayán, 2001

CONTROL DE APLICACIONES MEDIANTE COMANDOS ORALES RECONOCIDOS POR REDES NEURONALES

VLADIMIR MOSQUERA CERQUERA.

VÍCTOR HUGO MONCAYO RUIZ.

Monografía presentada como requisito para optar
al título de Ingeniero en Electrónica y Telecomunicaciones

Director
Ing. ELENA MUÑOZ ESPAÑA.

UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE ELECTRÓNICA INSTRUMENTACIÓN Y CONTROL.
POPAYÁN, 2001

A todas aquellas personas que creyeron en mi, mi familia, en especial a mi padre junto a quien me gustaría haber compartido este logro; mi abuelo quien esperó pacientemente este momento; Andrés, Alejandro y Nathaly, mis hermanos, quienes fueron uno de los motivos mas importantes de mi lucha y a mi madre por todo su amor y apoyo.

Víctor Hugo

Definitiva y exclusivamente a mis padres, por su amor y apoyo incondicional en el transcurso de mi vida. A mis hermanos por su colaboración, a mi sobrinita Laura por ser el motivo de alegería en mi hogar y a las personas que estuvieron conmigo en todo momento para lograr la meta propuesta.

Vladimir

Abstract

Se ha desarrollado un sistema software capaz de reconocer de un léxico limitado un número fijo de comandos orales de un locutor permitiendo de esta forma, realizar aplicaciones a nivel hardware y software según la relación existente entre el comando reconocido y la aplicación a ejecutar. El reconocimiento de los comandos orales es realizado haciendo uso de los Mapas Auto –organizados gracias a la capacidad de estas redes de clasificar los patrones obtenidos a partir de la señal de voz del locutor. Este es un sistema de voz personalizado, es decir, reconoce la voz de quien haya entrenado la red neuronal.

AGRADECIMIENTOS

Los autores expresan agradecimientos a :

Elena Muñoz España. Ingeniero en Electrónica y Telecomunicaciones. Por el apoyo brindado y por su valiosa gestión como directora del proyecto.

Oscar Andrés Vivas. Ingeniero en Electrónica y Telecomunicaciones. Por el apoyo brindado en la parte inicial del proyecto.

Carlos Felipe Rengifo. Magister en Automática y profesor de la Universidad del Cauca. Por la colaboración en la realización del proyecto y la evaluación al mismo.

Juan Fernando Flórez. Ingeniero en Electrónica y Telecomunicaciones. Por su colaboración en la evaluación del proyecto.

Oscar Mauricio Caicedo y John Alexander Mosso. Ingenieros en Electrónica y Telecomunicaciones. Por su valiosa colaboración en el desarrollo del proyecto.

Telésforo Moncayo. Por su valioso apoyo.

Jesús Mosquera. Por su paciencia y apoyo incondicional para lograr la meta propuesta.

Marisol López. Por su colaboración prestada.

A nuestros compañeros y amigos que de alguna forma aportaron al proyecto y a nuestra vida.

Contenido

Contenido de anexos

Introducción

En la actualidad existe una progresiva proliferación de aplicaciones basadas en el proceso automático del lenguaje hablado. Así, son cada vez más comunes las interfaces hombre-máquina controladas por voz, los sistemas de respuesta vocal interactiva, y la automatización de sistemas telefónicos.

El elevado número de aplicaciones posibles para los próximos años queda condicionado, además de los propios "factores humanos" que intervienen en el proceso de aceptación de una nueva tecnología, por los propios factores de la tecnología subyacente. Esta tecnología recibe la denominación común de Tecnología del Habla y se estructura en cuatro tecnologías básicas principales:

- El Reconocimiento de Voz o Reconocimiento del Habla

  Es el proceso de conversión de un mensaje hablado en texto, que permita al usuario una comunicación con la máquina. Se trata de la tecnología que mayor avance ha sufrido en los últimos cinco años, pasándose de poder reconocer tan sólo a un locutor, dentro de un vocabulario muy limitado, a prototipos capaces de reconocer a cualquier locutor sobre vocabularios flexibles compuestos por miles de palabras.

- La Conversión Texto - Voz

Se ocupa de la generación de mensajes hablados mediante la simulación del proceso de lectura de un texto escrito almacenado en formato electrónico. Esta tecnología también cuenta en la actualidad con unos excelentes resultados, planteándose como el mayor reto a conseguir, aumentar la calidad de la voz sintetizada.

- El Reconocimiento de Locutores

  Es el proceso de identificación o verificación de la identidad del locutor de forma automática a partir de la señal de voz. El grado de desarrollo de esta tecnología es inferior al de las anteriores, quizás como consecuencia de lo crítico que puede ser las aplicaciones en las que se inserte.

- La Codificación de Voz

  Es una tecnología que ha alcanzado un grado de madurez muy elevado, contando en la actualidad con un número importante de procedimientos estandarizados. Su objetivo es la búsqueda de representaciones eficientes en formato digital de la señal de voz para su almacenamiento y/o transmisión, persiguiendo obtener la mayor calidad posible, para el menor número de bits por muestra.

Podríamos, por tanto, situar a la Tecnología del Habla como receptora de un amplio conjunto de conocimientos y procedimientos de actuación sobre la información representada en la señal de voz. Conocimientos que se articulan con un alto grado de dificultad y especialización, ya que pertenecen a un marco científico-técnico multidisciplinar, donde se dan cita diferentes ramas del saber como son: fisiología, acústica, lingüística, procesado de señal, inteligencia artificial, teoría de la comunicación y de la información, y ciencia de la computación.

Lo expuesto hasta este punto sirve de referencia para poner de manifiesto las limitaciones que se nos plantean a la hora de desarrollar un trabajo como éste. Un trabajo que se presenta bajo el título de "Control de Aplicaciones Mediante Comandos Orales Reconocidos por Redes Neuronales", no puede pretender abarcar todos y cada uno de los desarrollos y últimas líneas de

trabajo abiertas en las diferentes áreas de interés. Por tanto, aún a pesar de introducir importantes simplificaciones, buscando una mayor claridad en la exposición nos limitaremos a resumir y estructurar el fondo común de los principales desarrollos clásicos.

Podríamos afirmar que, genéricamente, el principal objetivo que el Reconocimiento del Habla persigue es proporcionar una "apropiada" interacción hombre-máquina a través de órdenes habladas. Así, los resultados que esta tecnología proporcione deberán contrastarse con los derivados de otras alternativas como son: teclados, paneles, ratones, etc., en cuanto proporcionen un control de procesos de interacción hombre-máquina más o menos "apropiado". Las principales características que diferencian a los sistemas basados en Reconocimiento del Habla frente a otras alternativas son: la naturalidad que supone utilizar el habla en las operaciones de comando y control, y la precisión y robustez en la comunicación para diferentes usuarios y diferentes entornos. La primera de ellas debería representar la ventaja natural de los sistemas basados en la Tecnología del Habla. Aunque la experiencia nos ha enseñado que, si bien el habla es la forma natural de comunicación entre personas, en el diálogo hombre-máquina esto no parece obvio; por ejemplo, en los diversos estudios que reflejan el elevado número de personas incapaces de responder frente a una máquina, si bien es cierto que este tipo de rechazos va disminuyendo paulatinamente. Es la segunda de las características anteriores la que se muestra más crítica en las aplicaciones del Reconocimiento del Habla. El estado actual de la investigación en Reconocimiento del Habla nos muestra excelentes resultados de sistemas trabajando en entornos controlados de laboratorio. Sin embargo, una aplicación real de esta tecnología exige un funcionamiento en el mundo real donde el grado de dificultad de los problemas es un orden de magnitud mayor.

Por último, es necesario resaltar, que si bien son muchas las aplicaciones que pueden plantearse para la Tecnología del Habla, su consideración explícita corresponde a realizar acciones particulares para cumplir con los objetivos de este trabajo. No obstante, es claro que muchas de las restricciones que se derivan de las diferentes aplicaciones aparecen implícitamente como

aspectos que demandan solución y, por tanto, condicionan el desarrollo de las diferentes tecnologías.

Las aplicaciones que se llevan a cabo en este trabajo son acciones básicas en el campo de la tecnología del habla, que corresponde al reconocimiento de palabras aisladas de dos sílabas. Son acciones básicas porque el reconocimiento de un comando conlleva a la ejecución de operaciones que a nivel de software, corresponde a la invocación de las funcionalidades que permiten activar y desactivar dispositivos domésticos y controlar el funcionamiento de un motor de corriente continua igualmente por comandos de voz.

Este trabajo presenta la gran viabilidad de la incorporación de las redes neuronales artificiales en la Tecnología del Habla como método de reconocimiento de patrones para lograr la identificación de los comandos. Con esto se quiere demostrar que las redes neuronales son un método muy potente para solucionar el problema planteado ya que, con técnicas simples de procesamiento en la obtención de patrones, se obtiene un funcionamiento optimo que puede ser mejorado aún más si técnicas más complejas de procesamiento fuesen empleadas.

Este material esta compuesto por ocho capítulos que aportan un completo soporte teórico a la construcción del sistema y sobre las investigaciones realizadas al respecto, las cuales aportaron muchas alternativas de solución al problema planteado. El capítulo uno presenta aspectos muy interesantes de las funciones de la corteza auditiva en el proceso de identificación de la señal de voz, que inspiraron en gran parte varios de los procesos realizados en el sistema.

El capitulo dos expone la teoría sobre el procesamiento realizado a la señal de voz en este trabajo. En el caso del procesamiento espectral se realiza el análisis matemático a la transformada rápida de Fourier debido a que este concepto es el soporte al proceso realizado a la señal de voz a nivel frecuencial

El capítulo tres expone la teoría de las redes neuronales con características de auto – organización, dado la gran capacidad que tienen en el reconocimiento de patrones. En el capítulo cuatro se expone el desarrollo del sistema, donde explica la realización de los aspectos teóricos a las diferentes etapas que constituyen el sistema.

El capítulo cinco contiene el modelado del sistema. El capitulo seis es la especificación de las aplicaciones del control proceso y dispositivo doméstico, el cual contiene información sobre los recursos software y hardware necesarios que dan soporte a dichas aplicaciones.

El capítulo siete exalta aspectos notables obtenidos en la culminación de este trabajo mediante las conclusiones y recomendaciones, y el capitulo ocho es la referencia bibliográfica del trabajo para las personas que quieran saber más sobre este tema.

Este material además presenta cuatro anexos que complementan los aspectos teóricos del sistema desarrollado. El anexo A corresponde al manual de usuario. El anexo B es una exposición de las funciones de procesamiento de señal usadas en este trabajo. El anexo C es una descripción detallada del hardware que permite realizar las aplicaciones control proceso y dispositivo doméstico y el anexo D es el diseño detallado del software del sistema.

Capítulo Uno

El Sistema Nervioso: Mecanismos Auditivos Centrales

Este capitulo expone una breve introducción de la fisiología del cerebro humano y los procedimientos que éste realiza para la identificación de la información de voz. Los aspectos aquí comentados exhiben importantes propiedades, tales como la excitación de sonidos de alta y baja frecuencia a regiones de neuronas bien diferenciadas de la corteza auditiva, o los procesos realizados por la corteza auditiva primaria como dar la capacidad de oír y de interpretar tonos, o los realizados por la corteza de asociación auditiva tales como dar significado de asociación a las palabras reconocidas. Estos procesamientos a nivel neurofisiológico inspiraron el funcionamiento de varias fases en el sistema desarrollado.

1.1 EL SISTEMA NERVIOSO

El sistema nervioso, junto con el sistema endocrino, asegura las funciones de control del organismo. En general, el sistema nervioso controla actividades corporales rápidas, tales como contracciones musculares, cambios viscerales rápidos, o incluso la intensidad de la secreción de algunas glándulas endocrinas. El sistema endocrino, por el contrario, regula principalmente las funciones metabólicas del cuerpo.

El sistema nervioso es único en lo que concierne a la gran complejidad de las acciones de control que puede llevar a cabo. Recibe literalmente millones de datos de información procedentes de los distintos órganos sensoriales y los integra a continuación para determinar la respuesta corporal.

1.2 LA NEURONA DEL SISTEMA NERVIOSO CENTRAL – LA UNIDAD SENSORIAL BÁSICA

El sistema nervioso central esta compuesto por más de 100.000 millones de neuronas. La figura 1.1 ilustra una neurona típica de las que se encuentran en la corteza motora del cerebro.

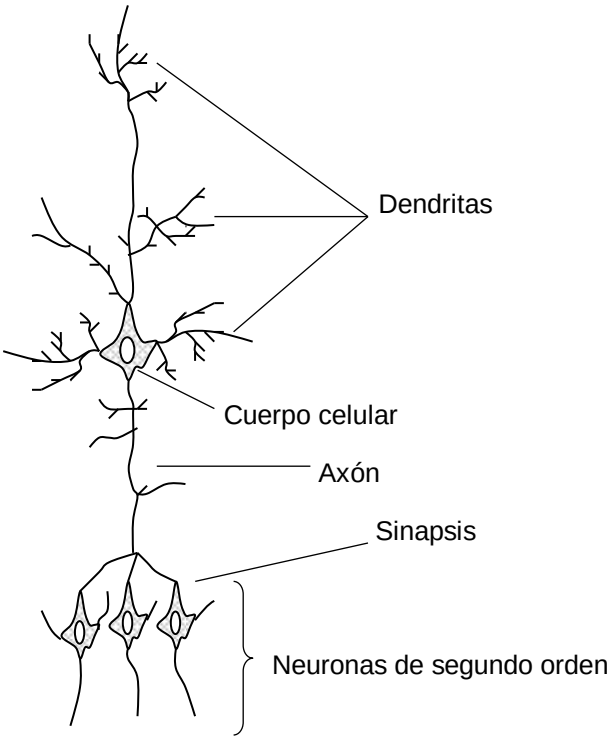


Figura 1.1. Estructura de una neurona.

La información de entrada llega a la célula casi por completo a través de sinapsis que establecen las dendritas del cuerpo neuronal; el número de estas conexiones sinápticas procedentes de las

fibras de entrada puede oscilar entre unos cientos y 200.000. Por otra parte, la señal de salida viaja a lo largo de un axón único, aunque este axón se ramifica hacia otras partes del cerebro, la médula espinal, o de la periferia corporal. Estas terminales establecen luego sinapsis con otra serie de neuronas o con células musculares o secretoras.

Una característica especial de la mayoría de la sinapsis es que la señal pasa normalmente sólo en la dirección de avance excepto en condiciones muy inusuales, lo que permite direccionar adecuadamente las señales para que lleven a cabo las funciones nerviosas necesarias. Las neuronas están organizadas en un gran número de redes neuronales que son las que determinan las funciones del sistema nervioso.

1.3 REFLEJO DE ATENUACIÓN

Cuando se transmiten sonidos fuertes por la cadena de huesecillos del oído medio hacia el sistema nervioso central, después de un periodo de latencia de 40 a 80 milisegundos tiene lugar un reflejo que permite que toda la cadena de huesecillos desarrolle un alto grado de rigidez, reduciendo así grandemente la conducción por la misma de sonidos de baja frecuencia, principalmente aquellos por debajo de 1000 ciclos por segundo.

Este *reflejo de atenuación* puede reducir la intensidad de la transmisión del sonido hasta en 30 ó 40 decibelios, que viene a ser la misma diferencia que existe entre una voz fuerte y el sonido de un susurro. La función de este mecanismo posiblemente sea doble:

1. Proteger al caracol de vibraciones lesivas producidas por sonidos excesivamente fuertes.
2. *Enmascarar* los sonidos de baja frecuencia en entornos muy ruidosos. Esto suele suprimir una gran parte del ruido de fondo y permite a la persona concentrarse en sonidos por encima de

los 1000 ciclos por segundo, frecuencia a la que se transmite la mayoría de la información de la comunicación oral.

1.4 DETERMINACIÓN DE LA FRECUENCIA DEL SONIDO

Los sonidos de baja frecuencia producen el máximo de activación de la membrana basilar cerca del vértice del caracol, los de alta frecuencia activan la membrana basilar cerca de la base del caracol, y las frecuencias intermedias lo hacen a distancias intermedias entre estos dos extremos. Es más, hay una organización espacial de las fibras nerviosas en el recorrido coclear, desde el caracol a la corteza cerebral. El registro de señales desde las cintillas auditivas del tallo cerebral y de los campos receptores auditivos de la corteza cerebral muestra que se activan neuronas especificas por sonidos de una frecuencia determinada. Por tanto, el método principal utilizado por el sistema nervioso para detectar frecuencias distintas es determinar que posición a lo largo de la membrana basilar resulta mas estimulada.

1.5 FUNCIÓN DE LA CORTEZA CEREBRAL EN LA AUDICIÓN

En la figura 1.2 se ilustran las áreas de proyección de la vía auditiva a la corteza cerebral, mostrándose que la corteza auditiva se halla principalmente en el *plano supratemporal de la convolución temporal superior,* pero que también se extiende por el *límite lateral del lóbulo temporal,* por gran parte de la corteza insular, e incluso hacia la porción más lateral del *opérculo parietal.*

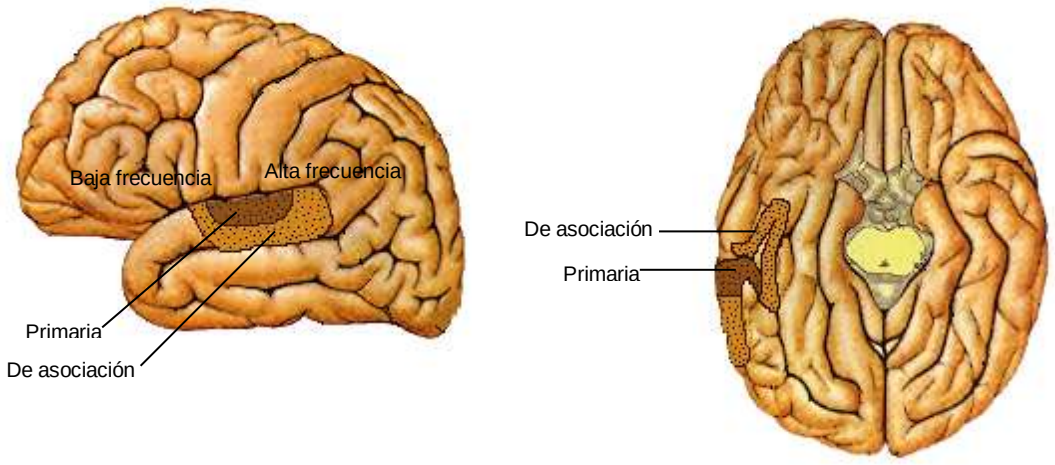


Figura 1.2. Corteza auditiva

En esta figura se muestran dos áreas diferenciadas: la *corteza auditiva primaria y la corteza de asociación auditiva (también llamada corteza auditiva secundaria).* La corteza auditiva primaria se excita directamente por proyecciones procedentes del cuerpo geniculado medial, en tanto que las áreas de asociación auditiva se excitan secundariamente por impulsos procedentes de la corteza auditiva primaria y por proyecciones de las áreas de asociación talámica adyacentes al cuerpo geniculado medial.

Percepción de la frecuencia del sonido en la corteza auditiva primaria

Se han encontrado al menos seis *mapas tonotópicos* en la corteza auditiva primaria y en las áreas de asociación auditiva. En cada uno de ellos, los sonidos de alta frecuencia excitan a neuronas de uno de los extremos del mapa, mientras que los de baja frecuencia excitan neuronas del extremo opuesto. En la mayoría de ellos, los sonidos de baja frecuencia se localizan en la parte anterior, como se muestra en la figura 1.2, y los de alta, en la posterior. Cada una de estas áreas diferenciadas extrae alguna característica específica de los sonidos. Por ejemplo, es casi seguro

que uno de los grandes mapas de la corteza auditiva primaria discrimina a las propias frecuencias sonoras y proporciona a la persona la sensación física del tono de los sonidos. Otro de los mapas se utiliza probablemente para detectar la dirección de la que proviene el sonido.

La membrana basilar cerca de la base del caracol es estimulada por sonidos de todas las frecuencias, y en los núcleos cocleares se encuentra esta misma amplitud de representación sonora. Con todo, para cuando la excitación ha alcanzado la corteza cerebral, la mayoría de las neuronas que responden al sonido lo hacen solo a una estrecha banda de frecuencias, en lugar de una banda ancha.

Una gran proporción de las neuronas de la corteza auditiva, especialmente de la corteza de asociación auditiva no responden a frecuencias específicas en el oído. Se cree que estas neuronas asocian entre sí distintas frecuencias sonoras o asocian información sonora con información procedente de otras áreas sensoriales de la corteza. Ciertamente, la porción parietal de la corteza de asociación auditiva se solapa en parte con el área sensorial somática II, lo cual podría brindar una buena oportunidad para la asociación de la información auditiva con la información sensorial somática.

Discriminación de patrones de sonido por la corteza auditiva

La extirpación bilateral completa de la corteza auditiva no evita que un gato o un mono detecten sonidos o reacciones de forma burda a los sonidos. No obstante, si reduce grandemente e incluso anula a veces su capacidad de discriminar tonos diferentes y especialmente *patrones de sonido.* Por ejemplo, un animal al que se haya entrenado a reconocer una combinación o una secuencia de tonos de sonido, formando un patrón determinado, pierde esta capacidad cuando se destruye

su corteza auditiva; es más, ya no puede volver a aprender este tipo de respuesta. Por tanto, la corteza auditiva es importante en la discriminación de *patrones de sonido secuenciales y tonales.*

Se dice que la destrucción total de ambas cortezas auditivas primarias en el ser humano reduce grandemente la sensibilidad auditiva de la persona, efecto bastante diferente del observado en animales inferiores. Sin embargo, esta información no esta del todo clara. En cambio, la destrucción de la corteza auditiva primaria de un solo lado del hombre tiene poco efecto sobre la audición debido a las muchas conexiones entrecruzadas de un lado a otro en la vía neuronal. A pesar de todo, sí afecta la capacidad de localizar el origen del sonido porque para esta función de localización sí que se necesitan señales comparativas en ambas cortezas.

Las lesiones en el humano que afectan a las áreas de asociación auditiva pero que no afectan a la corteza auditiva primaria no disminuyen la capacidad de la persona de oír y diferenciar tonos y de interpretar por lo menos patrones sencillos de sonido. Sin embargo, el individuo será a menudo incapaz de interpretar el significado del sonido escuchado, haciendo imposible que interprete el significado de las palabras aunque las oiga perfectamente bien e incluso pueda repetirlas.

Capítulo Dos

Técnicas de Procesamiento de la Señal de Voz

El factor más crítico para el diseño de sistemas de Reconocimiento de voz es la determinación de los parámetros o características sobre los que el sistema basará su funcionamiento. Idealmente deberán elegirse características fáciles de medir, estables con el paso del tiempo, y robustas frente a los diferentes entornos de trabajo del sistema.

Esta etapa investigativa, va desde la evaluación del modo más eficaz para la obtención de la señal de voz, examen de esta señal en caso de ser capturada, conversión a un formato digital, su análisis de espectro discreto y medición de los mismos, lo cual nos conlleva a la evaluación de los datos de todas las muestras, estudios de los valores arrojados por cada una de ellas y la opinión final respecto a su identificación. Su objetivo es la búsqueda de representaciones eficientes en formato digital de la señal de voz para su almacenamiento y procesado, persiguiendo obtener la mayor calidad posible, para el menor número de bits por muestra.

2.1 OBTENCIÓN DE LA SEÑAL DE VOZ

La señal de voz básicamente está constituida por ondas de presión producidas por el aparato fonador humano. La manera obvia de capturar este tipo de señal se realiza mediante un micrófono, el cual se encargará de convertir la onda de presión sonora en una señal eléctrica. La siguiente etapa será aquella que se encargue de amplificar las señales a niveles que sean manejables. A partir de la señal analógica obtenida se hace necesario el procesamiento mediante la tarjeta de sonido del PC para convertir la señal a formato digital, el cual es realizado mediante dos procesos: muestreo y cuantificación. Este proceso de dos etapas se conoce como Modulación por Código de Pulsos (PCM). Este tratamiento digital posibilita el almacenamiento dinámico de la señal de voz en un espacio de memoria del sistema bajo el formato de archivo wav para su posterior procesamiento temporal y frecuencial.

La señal vocal tiene componentes frecuenciales que pueden llegar a los 10 KHz, sin embargo la mayor parte de los sonidos vocales tiene energía espectral significativa hasta los 5 KHz, solamente los sonidos fricativos poseen componentes que pueden llegar a los 10 KHz. La frecuencia de muestreo dependerá del tipo de aplicación, para señales de voz se adopta un rango de 6 a 20 KHz dependiendo de la resolución que se desee. Otra consideración que se debe tener en cuenta es la cuantificación de la señal, la cual involucra la conversión de la amplitud de los valores muestreados a forma digital usando un número determinado de bits. El número de bits usados afectará la calidad de la voz muestreada y determinará la cantidad de información a almacenar. Para cada instante de muestreo, el conversor analógico digital compara la señal muestreada con una serie de niveles de cuantificación predefinidos. El número de niveles N a usar determina la precisión del análisis y por tanto el número de bits necesarios. Cada bit adicional que se agrega contribuye en mejorar la relación señal a ruido en aproximadamente 6 dB. La señal de

voz exhibe un rango dinámico de unos 50 a 60 dB. por lo que resultaría suficiente una cuantificación de 8 a 9 bits para una buena calidad de voz. Sin embargo generalmente se usa de 11 a 20 bits en aplicaciones de procesamiento de señales de voz de alta calidad.

2.2 FILTRADO

Esta sección describe como filtrar señales discretas en el dominio del tiempo. También se discute el diseño y análisis de las características de los filtros, incluyendo la respuesta al impulso. Este procesamiento de señales está basado según los procedimientos matemáticos que emplea la herramienta computacional MATLAB.

2.2.1 Convolución y Filtrado

El fundamento matemático del filtrado en el dominio del tiempo es la convolución. Si se tienen dos señales discretas temporales representadas en forma de vector, al convolucionar un vector con otro, obtenemos:

```
A = conv ( [1  1  1],[1  1  1] )
A =  [2  3  2  1]
```

La salida *y(n)* de un filtro digital está relacionada a su entrada *x(n)* por convolución con su respuesta al impulso *h(n)*:

$$y(n) = h(n) * x(n) = \sum_{m=-\infty}^{\infty} h(n-m)x(m) \qquad (1)$$

Si la respuesta al impulso *h(n)* de un filtro digital es de longitud finita, y la entrada *x(n)* es también de longitud finita, el filtro se implementa almacenando *x(n)* y *h(n)* en vectores independientes, y convolucionando estos dos vectores.

2.2.2 Filtros y Funciones de Transferencia

En general, la transformada *Z Y(z)* de la salida $y(n)$ de un filtro digital esta relacionada a la trasformada *Z X(z)* de la entrada por:

$$Y(z) = H(z)X(z) = \frac{b(1) + b(2)z^{-1} + ... + b(nb+1)z^{-nb}}{a(1) + a(2)z^{-1} + ... + a(na+1)z^{-na}} X(z) \qquad (2)$$

Donde *H(z)* es la función de transferencia del filtro. Aquí, las constantes *b(i)* y *a(i)* son los coeficientes del filtro y el orden del filtro es el máximo de *na* y *nb*.

Coeficientes y Nombres de los Filtros

Muchos nombres estandarizados para los filtros reflejan el número de los coeficientes a y b presentes:

- Cuando *nb*=0 (Esto es, b es un escalar), el filtro es de respuesta infinita al impulso (IIR), de todo polo, recursivo, o autoregresivo (AR)
- Cuando *na*=0 (Esto es, a es un escalar), el filtro es de respuesta finita al impulso (FIR), de todo cero, no recursivo, o de promedio movible (MA).
- Si *na* y *nb* son más grandes que cero, el filtro es un IIR, de polos y ceros, recursivo o autoregresivo de promedio movible (ARMA).

Los nombres AR, MA, y ARMA son usualmente aplicados a filtros asociados con procesos estocásticos de filtrado.

2.2.3 Proceso de Filtrado

Es simple volver a trabajar con una ecuación diferencial desde la relación de la transformada *z* mostrada anteriormente (ecuación 2). Se asume que *a(1)* = 1. Moviendo el denominador a mano izquierda y tomando la transformada *z* inversa:

$$y(n)+a(2)y(n-1)+...+a(na+1)y(n-na)=b(1)x(n)+b(2)x(n-1)+...+b(nb+1)x(n-nb) \quad (3)$$

En términos de entradas comunes y pasadas, y salidas pasadas, *y(n)* (salida del filtro) es:

$$y(n)=b(1)x(n)+b(2)x(n-1)+...+b(nb+1)x(n-nb)-a(2)y(n-1)-...-a(na+1)y(n-na) \quad (4)$$

Esta es la representación estandarizada en el dominio del tiempo de un filtro digital, calculada iniciando con *y(1)* y asumiendo condiciones iniciales iguales a cero. Esta representación de la progresión es:

$$\begin{aligned} y(1) &= b(1)x(1) \\ y(2) &= b(1)x(2)+b(2)x(1)-a(2)y(1) \\ y(3) &= b(1)x(3)+b(2)x(2)+b(3)x(1)-a(2)y(2)-a(3)y(1) \end{aligned} \quad (5)$$

.
.
.

Al realizar los cálculos en la progresión se obtiene tantas muestras de salida como entradas haya, esto es, la longitud del vector de salida *y(n)* es la misma que la longitud del vector de entrada *x(n).*

2.2.4 Especificación y Requerimiento de los Filtros

La meta de diseñar un filtro es hacer que la frecuencia sea dependiente de la alteración de una secuencia de datos. Un posible requerimiento podría ser remover el ruido por encima de 30 Hz de una secuencia de datos muestreada a 100 HZ. Una especificación más rigurosa podría requerir de una cantidad específica de señal pasante, de atenuación de banda rechazada o el ancho de la transición. Una especificación más precisa podría pedir la obtención de la respuesta deseada con el mínimo orden del filtro, o podría llamar una forma arbitraria de magnitud en su respuesta, o podría requerir un filtro FIR.

Los métodos para el diseño de filtros difirieren principalmente en como su funcionamiento es especificado. Para requerimientos poco específicos como en el primer caso, donde solo el orden y la frecuencia de corte son dados como parámetros de diseño, un filtro Butterworth IIR es a menudo suficiente.

Para requerimientos más rigurosos de filtrado, tradicionalmente se incluyen parámetros tales como la ganancia de la señal en la banda pasante (Rp, en dB), atenuación de la banda rechazada (Rs, en dB), y el ancho de la transición (Ws – Wp, en Hz.).

Una metodología para diseñar filtros Butterworth, Chebychev tipo I, Chebychev tipo II y elípticos es conocer sus especificaciones de funcionamiento, para obtener de esta manera los coeficientes que determinan sus respectivas funciones de transferencia.

2.2.5 Diseño de filtros IIR

La ventaja primaria de los filtros IIR sobre los filtros FIR es que ellos típicamente conocen un grupo dado de especificaciones con un orden de filtro mucho más bajo que un correspondiente filtro FIR. Aunque los filtros IIR tienen fase no lineal, el procesamiento de datos dentro de MATLAB es ejecutado comúnmente "fuera de línea", esto es, la entrada de la secuencia de datos está disponible antes de realizar el filtrado. Esto permite aproximarse a un filtrado no casual, de cero fase, el cual elimina la distorsión de fase no lineal de un filtro IIR.

Los filtros clásicos IIR Butterworth, Chebyshev tipo I y II, elíptico, y Bessel, todos aproximan la respuesta ideal en diferentes maneras. MATLAB proporciona funciones para crear todos esos tipos de filtros clásicos IIR en los dominios analógico y digital (Excepto el Bessel, para el cual solo el caso analógico esta soportado) en configuraciones pasa bajo, pasa alto, pasa banda y eliminador de banda. Para mas tipos de filtrado, se puede también encontrar el orden del filtro mas bajo que se ajuste a unas especificaciones dadas de filtrado en términos de la atenuación de la banda pasante y la banda eliminada, y el ancho de la transición en la frecuencia de corte.

2.2.5.1 Diseño de Filtros Clásicos IIR Usando un Prototipo Analógico

La principal técnica de diseño de filtros digitales IIR que provee MATLAB esta basada en la conversión de filtros pasa - bajos analógicos en sus equivalentes digitales.

Diseño Completo de Filtros IIR Clásicos

Se puede crear un filtro de cualquier orden con una configuración pasa bajo, pasa alto, pasa banda, o eliminador de banda usando las funciones de diseño de filtro:

Tabla 2.1: Funciones de MATLAB para el diseño de filtros.

Tipo de Filtro	**Función de Diseño**
Butterworth	[b, a] = butter (n, Wn, *opciones*) [z, p, k] = butter (n, Wn, *opciones*) [A, B, C, D] = butter (n, Wn, *opciones*)
Chebyshev tipo I	[b, a] = cheby1 (n, Rp, Wn, *opciones*) [z, p, k] = cheby1 (n, Rp, Wn, *opciones*) [A, B, C, D] = cheby1 (n, Rp, Wn, *opciones*)
Chebyshev tipo II	[b, a] = cheby2 (n, Rs, Wn, *opciones*) [z, p, k] = cheby2 (n, Rs, Wn, *opciones*) [A, B, C, D] = cheby2 (n, Rs, Wn, *opciones*)
Elíptico	[b, a] = ellip (n, Rp, Rs, Wn, *opciones*) [z, p, k] = ellip (n, Rp, Rs, Wn, *opciones*) [A, B, C, D] = ellip (n, Rp, Rs, Wn, *opciones*)
Bessel (sólo analógico)	[b, a] = besself (n, Wn, *opciones*) [z, p, k] = besself (n, Wn, *opciones*) [A, B, C, D] = besself (n, Wn, *opciones*)

Por defecto, cada una de esas funciones retorna un filtro pasa bajo; sólo se necesita especificar la frecuencia de corte Wn deseada en una frecuencia normalizada (frecuencia de Nyquist = 1 Hz). Para obtener un filtro pasa alto solo se necesita agregar la cadena `'high'` a la lista de parámetros de la función. Para un filtro pasa – banda o eliminador de banda, se especifica Wn como un vector de dos elementos conteniendo las frecuencias de borde de la banda pasante o eliminada, añadiendo '`stop`' para la configuración de banda eliminada.

Aquí están algunos ejemplos de filtros digitales:

```
[b, a] = butter (5, 0.4);                  %Filtro Butterworth pasa bajo de 5º orden y Wn = 0.4
[b, a] = cheby1 (4, 1, [0.4  0.7]);        %Filtro pasa banda Chebyshev Tipo I
[b, a] = cheby2 (6, 60, 0.8, 'high');      %Filtro pasa alto Chebyshev Tipo II
[b, a] = ellip (3, 1, 60, [0.4  0.7], 'stop');  %Filtro elíptico eliminador de banda
```

Para diseñar un filtro analógico, quizás para simulación, se debe usar en las funciones de diseño de filtros IIR una '`s`' adicional y especificar las frecuencias de corte en radianes/segundos:

```
[b, a] = butter (5, 0.4, 's');             %Filtro Butterworth analógico
```

Diseño de Filtros IIR según las Especificaciones de Frecuencia

MATLAB proporciona funciones de selección de orden que calculan el orden mínimo de filtrado que conoce un grupo dado de requerimientos:

Tabla 2.2: Funciones de MATLAB para el diseño de filtros según especificaciones de frecuencia.

Tipo de Filtro	Función de Diseño
Butterworth	[n, Wn] = buttord (Wp, Ws, Rp, Rs)
Chebyshev tipo I	[n, Wn] = cheb1ord (Wp, Ws, Rp, Rs)
Chebyshev tipo II	[n, Wn] = cheb2ord (Wp, Ws, Rp, Rs)
Elíptico	[n, Wn] = ellipord (Wp, Ws, Rp, Rs)

Las funciones para el diseño de filtros según las especificaciones de frecuencia en conjunto con las funciones para el diseño de filtros (Tabla 1) son muy útiles, ya que de las primeras se obtiene el orden del filtro y las frecuencias de corte según los requerimientos de filtrado, que son los

parámetros de entrada para el segundo grupo de funciones, las cuales obtiene finalmente los coeficientes de la función de transferencia del filtro requerido.

2.2.5.2 Comparación de los Tipos de Filtros Clásicos IIR

MATLAB proporciona cinco tipos diferentes de filtros clásicos IIR, cada uno de ellos óptimo en algún requerimiento. A continuación se muestra los prototipos básicos analógicos para cada uno y sus principales características.

Filtro Butterworth

El filtro Butterworth proporciona la mejor aproximación de la serie de Taylor para la respuesta ideal del filtro pasa bajo en $\omega = 0$ y $\omega = \infty$; para cualquier orden N, la magnitud de la respuesta cuadrada tiene 2N-1 derivadas cero en esas localizaciones (máxima planicidad en $\omega = 0$ y $\omega = \infty$).

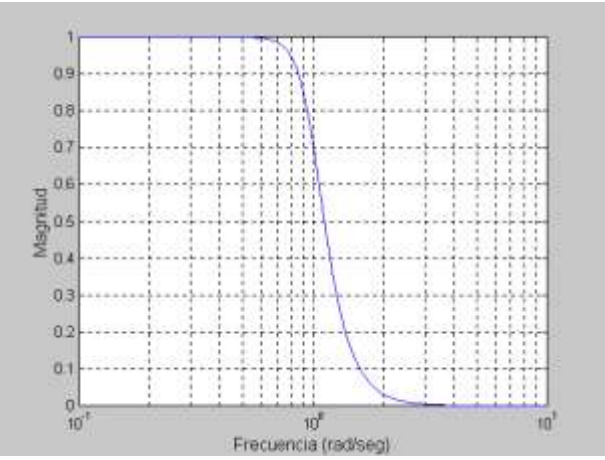


Figura 2.1. Respuesta de un filtro Butterworth pasa bajo de orden 5 con frecuencia de corte de 1 rad/seg.

La respuesta es completamente monótona, decreciendo suavemente de $\omega = 0$ a $\omega = \infty$. $|H(j\omega)|$=sqrt(1/2) en ω=1.

Filtro Chebyshev Tipo I

El filtro Chebyshev tipo I minimiza la diferencia absoluta entre la ideal y la actual respuesta de frecuencia sobre toda la banda pasante incorporando una ganancia igual de Rp dB en la banda pasante. La respuesta en la banda eliminada tiene máxima planicidad. La transición de la banda pasante a la banda eliminada es mucho más rápida que para el filtro Butterworth. $|H(j\omega)|=10^{-Rp/20}$ en ω=1.

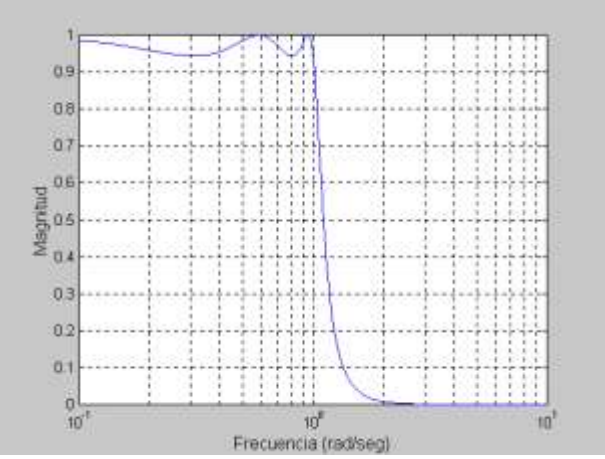


Figura 2.2. Respuesta de un filtro Chebyshev tipo 1 pasa bajo de orden 5 con frecuencia de corte de 1 rad/seg.

Filtro Chebyshev Tipo II

El filtro Chebyshev tipo II minimiza la diferencia absoluta entre la respuesta de frecuencia ideal y actual sobre la entrada de banda eliminada, incorporando una ganancia igual de Rs dB en la banda eliminada. La respuesta en la banda pasante es de máxima planicidad.

La banda eliminada no se aproxima a cero tan rápidamente como el filtro tipo I. La ausencia de ganancia en la banda pasante, sin embargo, es a menudo una ventaja importante. $|H(j\omega)|=10^{-Rs/20}$ en $\omega=1$.

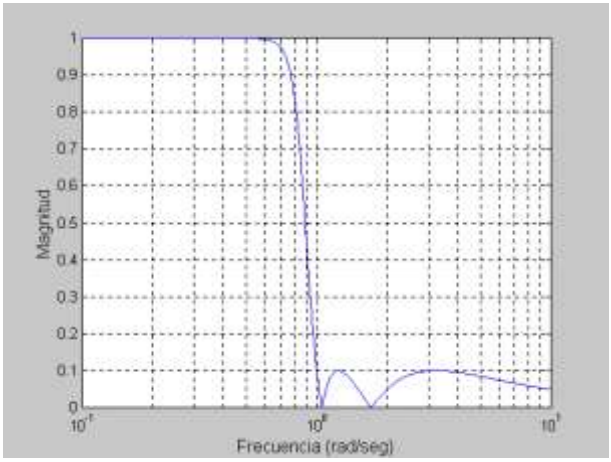
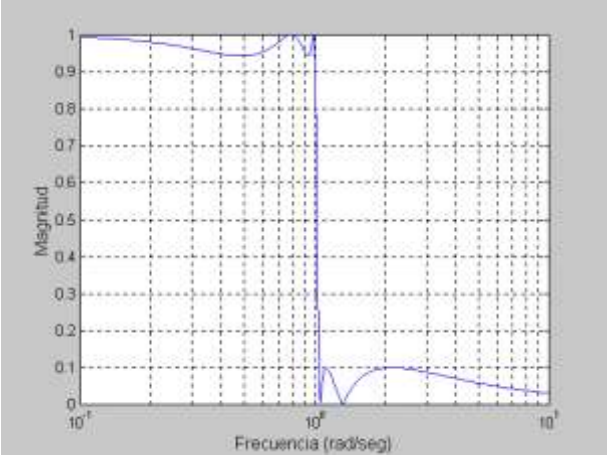


Figura 2.3. Respuesta de un filtro Chebyshev tipo 2 pasa bajo de orden 5 con frecuencia de corte de 1 rad/seg.

Filtro Elíptico

Los filtros elípticos tienen igual ganancia en la banda pasante y en la banda eliminada. Ellos generalmente conocen los requerimientos de filtrado con el menor orden que cualquier otro tipo de

filtro. Dado un filtro de orden n, ganancia Rp en dB de la banda pasante, y ganancia Rs en dB en la banda eliminada, los filtros elípticos minimizan el ancho de la transición. $|H(j\omega)|=10^{-Rp/20}$ en



$\omega=1$.

Figura 2.4. Respuesta de un filtro elíptico pasa bajo de orden 5 con frecuencia de corte de 1 rad/seg.

Filtro Bessel

Los filtros pasa bajos Bessel analógicos tienen en el grupo de retardo máxima planicidad a la frecuencia cero y retienen aproximadamente constante el grupo de retardo a través de toda la banda pasante. Las señales filtradas por consiguiente mantienen su forma de onda en el rango de frecuencias de la banda pasante. El mapeado de frecuencia y los filtros digitales Bessel, sin embargo, no tienen esta propiedad de máxima planicidad.

Generalmente los filtros Bessel requieren un orden de filtro mas alto que otros filtros para una atenuación satisfactoria de la banda rechazada. $|H(j\omega)|<1/\sqrt{2}$ en $\omega=1$ y decrece cuando n se incrementa.

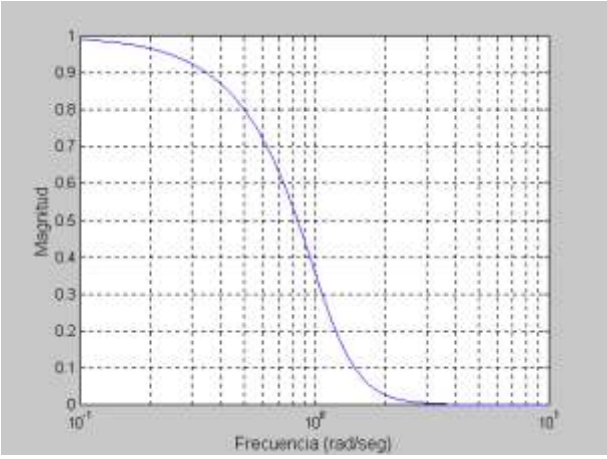


Figura 2.5. Respuesta de un filtro Bessel pasa bajo de orden 5 con frecuencia de corte de 1 rad/seg.

2.3 DELIMITACIÓN DE LA SEÑAL DE VOZ

Esta sección es la encargada de realizar una detección automática de los instantes de comienzo y final de la voz de entrada al sistema. Podemos decir que estamos ante un reconocedor de voz y silencio; de hecho, en algunas configuraciones no existe distinción alguna entre el reconocimiento de señal útil y el reconocimiento de ruido o silencio. Su funcionamiento es crítico y especialmente difícil en entornos ruidosos. Un error del detector de actividad puede suponer:

- La pérdida de parte del mensaje pronunciado por el usuario, sin posibilidad alguna de éxito en el proceso de reconocimiento.

- La aceptación de sonidos indeseados capaces de ser confundidos con unidades lingüísticas.

De los dos tipos de errores anteriores el primero es irrecuperable, por lo que el diseño de un detector de extremos suele hacerse buscando que se produzca el menor número de veces posible, aún a costa de permitir una mayor aceptación de sonidos indeseados, pasando por tanto, a jugar un papel importantísimo la incorporación de procedimientos de rechazo de estos sonidos.

El proceso de detección es realizado mediante el reconocimiento de la palabra del silencio o ruido de fondo que la rodea. Para lograr este propósito, las señales asociadas al silencio o ruido de fondo son eliminadas mediante el uso de un sencillo algoritmo de delimitación llamado *VicVlad*[1], el cual actúa sobre la señal de voz según su umbral de amplitud y el número de muestras de referencia.

2.4 PROCESAMIENTO ESPECTRAL

A lo largo de este apartado se desarrollará varios aspectos importantes del análisis de Fourier, debido a que el procesamiento espectral de la señal de voz en el sistema se soporta en estos conceptos. Aunque técnicas más eficientes para procesamiento espectral existen, se decidió utilizar Fourier para la extracción de los patrones dado a su sencillez conceptual y de implementación.

Una señal periódica se escribe como aquella cuyas características se repiten durante un intervalo largo de tiempo, teóricamente hablando. En el caso de las señales no periódicas, los efectos se

[1] Nombre dado al algoritmo delimitación diseñado por los autores de este trabajo.

concentran en un breve periodo. Estas señales pueden ser estrictamente limitadas en el tiempo, es decir que son cero fuera del intervalo de observación, o asintóticamente limitada en el tiempo, es decir que tienden a cero cuando el intervalo en que son observadas tiende de $-\infty$ hasta $+\infty$.

En el caso de señales periódicas o no periódicas, para que una señal tenga una representación mediante la transformada de Fourier su energía total debe estar bien definida, y medida en forma normalizada como:

$$E = \int_{-\infty}^{+\infty} |v(t)|^2 \, dt$$

Esta definición implica que si E es finita, tanto el valor promedio como la potencia promedio son iguales a cero.

2.4.1 Transformada de Fourier Discreta en el Tiempo

La representación en series de Fourier de las señales discutidas en el párrafo de arriba son determinadas, en general, por un número infinito de coeficientes. Dada la forma de onda, se puede determinar la secuencia de coeficientes. Dada una secuencia, se puede encontrar la forma de onda continua. Este ultimo procedimiento es el que genera la transformada de Fourier discreta en el tiempo (DTFT).

La DTFT proporciona una representación en el dominio de la frecuencia de una secuencia de datos que podría resultar, por ejemplo, de un muestreo de una forma de onda analógica cada *T* segundos. La diferencia distintiva entre el espectro de frecuencia de una señal analógica y el espectro derivado de una secuencia discreta en el tiempo es que el proceso de muestreo causa la

repetición del espectro analógico periódicamente en intervalos de f_S, donde $f_S = 1/T$ es la frecuencia de muestreo.

La tarea en este apartado es obtener una expresión matemática para la transformada de Fourier discreta en el tiempo. Inicialmente el proceso consiste de la evaluación numérica de la integral

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt \qquad (6)$$

la cual corresponde a la transformada de Fourier de una señal no periódica, y de la integral relacionada

$$a_n = \frac{1}{T} F_0(n\omega_0) = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) e^{-j\omega_0 t} dt \qquad (7)$$

Donde a_n es el coeficiente de Fourier n - ésimo en la representación en serie de Fourier de una señal periódica. La función $f(t)$ viene especificada ya sea gráficamente, por medio de una secuencia de datos o bien analíticamente.

Para el cálculo de $F(\omega)$ en valores discretos, se debe sustituir los limites infinitos en la ecuación (6) por limites finitos, se debe también aproximar la integral resultante con una suma y luego evaluar la suma para un conjunto de valores discreteos de ω. A continuación se presenta la teoría que permite dicho proceso.

Introduzcamos las funciones

$$\overline{f}(t)=\sum_{n=-\infty}^{\infty} f(t+nT), \qquad \overline{F}(\omega)=\sum_{n=-\infty}^{\infty} F(\omega+n\omega_1) \tag{8}$$

Obviamente, $\overline{f}(t)$ es periódica con periodo T y es igual a la suma de la función *f(t)* y de sus desplazamientos (figura 2.6). Análogamente, $\overline{F}(\omega)$ es periódica con periodo $\omega 1$ y es igual a la suma de $F(\omega)$ y de sus desplazamientos. Las constantes T y $\omega 1$ son arbitrarias.

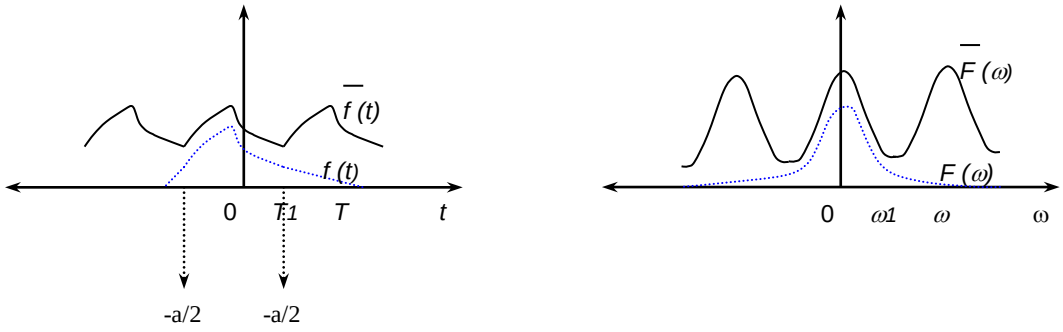


Figura 2.6

La ecuación

$$\overline{f}(t)=\frac{1}{T}\sum_{n=-\infty}^{+\infty} F(n\omega_0)e^{jn\omega_0 t} \quad , \quad \omega_0=\frac{2\pi}{T} \tag{9}$$

es un importante resultado conocido como *fórmula sumatoria de Poisson*, el cual constituye la base para la transformación de la integral impropia (6) en una integral con limites finitos.

Una formula semejante se cumple también para al función $\overline{F}(\omega)$

$$\overline{F}(\omega) = \frac{2\omega}{\omega_1} \sum_{n=-\infty}^{\infty} f(nT_1) e^{-jnT_1\omega} \qquad (10)$$

Dicha fórmula puede deducirse de la ecuación (9) y del teorema de la simetría. Se sigue de la formula de Poisson que las muestras $F(n\omega_0)$ de $F(\omega)$ son iguales a los coeficientes de la serie de Fourier de la función periódica $T\,\overline{f}(t)$, por lo tanto,

$$F(n\omega_0) = \int_{-\frac{T}{2}}^{\frac{T}{2}} \overline{f}(t) e^{-j\omega n_0 t} dt \qquad (11)$$

De esta forma, para valores de $\omega = n\omega_0$ la integral infinita (6) se reduce a la integral finita (11) en la que interviene la función $\overline{f}(t)$. Esta función puede determinarse exactamente del conocimiento de $f(t)$ para todo t.

Si $f(t)$ viene dada en un intervalo finito (-a/2, a/2) y $f(t) = 0$ para $|t| > a/2$, $\overline{f}(t)$ se conoce también exactamente pero ahora la suma de definición (8) contiene tan solo m términos si $a = mT$.

Como se ha observado, las muestras $F(n\omega_0)$ de $F(\omega)$ son los coeficientes del desarrollo en serie de Fourier de $T\overline{f}(t)$. Para su evaluación numérica es suficiente con considerar el problema de las series de Fourier. Las consideraciones siguientes permitirán la aproximación de la integral (11) con una suma finita.

Sea la función periódica

$$y(t) = \sum_{k=-\infty}^{\infty} c_k e^{jkw_0 t} \qquad (12)$$

donde *N* es un entero arbitrario y $T_1 = T/N$. Las muestras $y(mT_1)$ de $y(t)$ vienen dadas por

$$y(mT_1) = \sum_{k=-\infty}^{+\infty} c_k e^{jkw_0 mT_1} = \sum_{k=-\infty}^{+\infty} c_k w_N{}^{km}, \qquad w_N = e^{j2\pi/N} \qquad (13)$$

Ya que $\omega_0 T_1 = \frac{\omega_0 T}{N} = \frac{2\pi}{N}$. El número w_N es la *N* – ésima raíz primitiva de la unidad.

El entero *k* puede ser expresado como una suma

$$k = n + rN, \qquad n = 0,...,N-1, \quad r = ..., -1, 0, 1,... \qquad (14)$$

y al ser

$$w_N{}^N = 1, \qquad w_N{}^{(n+rN)m} = w_N{}^{mn}$$

la ecuación (13) se transforma en

$$y(mT_1) = \sum_{n=0}^{N-1} \sum_{r=-\infty}^{+\infty} c_{n+rN} w_N^{(n+rN)m} = \sum_{n=o}^{N-1} w_N^{mn} \sum_{r=-\infty}^{+\infty} c_{n+rN}$$

Si se realiza una definición de los coeficientes $\bar{c}_n$ como

$$\bar{c}_n = \sum_{r=-\infty}^{+\infty} c_{n+rN} \qquad (15)$$

Obtenemos

$$y(mT_1) = \sum_{n=0}^{N-1} \bar{c}_n w_N^{mn} \ , \qquad m = 0,..., N-1 \qquad (16)$$

De lo anterior se concluye: $y(mT_1)$ de una función periódica y los coeficientes $\bar{c}_n$ están relacionados por medio del sistema de *N* ecuaciones dado por (16).

Teorema fundamental

La evaluación numérica de la integral de Fourier (6) se basa en el resultado siguiente:

Sea T una constante arbitraria, N un entero también arbitrario y hagamos

$$T_1 = \frac{T}{N} \ , \qquad \omega_0 = \frac{2\pi}{T} \ , \qquad \omega_1 = \frac{2\pi}{T_1} = \omega_0 N \qquad (17)$$

se tiene entonces que para cualquier m

$$T\,\overline{f}(mT_1) = \sum_{n=0}^{N-1} \overline{F}(n\omega_0) w_N \ , \quad m = 0,..., N-1 \ , \qquad w_N = e^{j\frac{2\pi}{N}} \qquad (18)$$

en donde $\overline{f}(t)$ y $\overline{F}(\omega)$ son las funciones definidas en la ecuación (8) (figura 2.6).

El resultado anterior es consecuencia de las ecuaciones (9) y (16) En efecto, $\overline{f}(t)$ es una función periódica con coeficientes de Fourier $c_n = \frac{F(n\omega_0)}{T}$, por consiguiente

$$\bar{c}_n = \frac{1}{T}\sum_{r=-\infty}^{+\infty} F\left[(n+rN)\omega_0\right] = \frac{1}{T}\sum_{r=-\infty}^{+\infty} F\left(n\omega_0 + r\omega_1\right) = \frac{1}{T}\overline{F}\left(n\omega_0\right) \qquad (19)$$

y la ecuación (18) se desprende de (16). La ultima igualdad de la ecuación (19) es una consecuencia de (8).

Si se hace $m = 0,...,N-1$ en la ecuación (18) se obtiene un sistema de N ecuaciones cuya solución genera las muestras $\overline{F}\left(n\omega_0\right)$ de $\overline{F}\left(\omega\right)$ en función de las muestras $\overline{f}\left(t\right)$. En general, $F\left(n\omega_0\right)$ no puede determinarse en función de $\overline{F}\left(n\omega_0\right)$. Sin embargo, en el caso que

$$F\left(\omega\right) = 0 \text{ para } \quad \left|\omega\right| > \sigma \qquad (20)$$

se tiene que

$$F\left(\omega\right) = \overline{F}\left(\omega\right) \qquad \text{para} \quad \left|\omega\right| < \sigma \qquad (21)$$

y por lo tanto la solución de la ecuación (18) produce $F\left(n\omega_0\right)$. Si la función $F\left(t\right)$ no es de banda limitada como lo define la ecuación (20) pero ω_1 es lo suficientemente grande de forma que $F\left(\omega\right)$ pueda despreciarse para $\left|\omega\right| > \omega_1/2$, se tendrá que $F\left(n\omega_0\right)$ será aproximadamente igual $\overline{F}\left(n\omega_0\right)$ para $\left|n\right| < \dfrac{\omega_1}{2\omega_0}$ y por lo tanto podrá ser determinada a partir de la ecuación (18).

Después de efectuar este proceso matemático, la evaluación de integrales y series de Fourier se reduce a la solución de un sistema de *N* ecuaciones de la forma

$$A_m = \sum_{n=0}^{N-1} a_n w_N^{mn}, \qquad m = 0,...,N-1, \quad w_N = e^{j\frac{2\pi}{N}} \qquad (22)$$

que se obtiene a partir de las ecuaciones (16) y (18). La solución de este sistema viene dado por

$$a_n = \frac{1}{N}\sum_{m=0}^{N-1} A_m w_N^{-mn} \quad , \quad n = 0,\ldots,N-1 \qquad (23)$$

Las ecuaciones (22) y (23) forman un *par de series discretas de Fourier* debido a que proporcionan las funciones que permiten el cálculo de la transformada de Fourier discreta y su inversa, respectivamente. La secuencia de datos de entrada es representada por a_n y su transformada es representada por la secuencia A_m Estas ecuaciones establecen una correspondencia uno a uno entre los *N* números a_n y los *N* Números A_m. Para designar esta correspondencia se emplea la notación

$$a_n \underset{N}{\longleftrightarrow} A_n$$

Si las ecuaciones (22) y (23) se cumplen para toda m y n, las secuencias A_m y a_n deben entonces ser periódicas

$$A_{m+N} = A_m, \quad a_{n+N} = a_n \qquad (24)$$

ya que

$$w_N^{\pm(m+N)n} = w_N^{\pm mn}$$

Así un par de series discretas de Fourier puede ser visto como una correspondencia entre dos secuencias periódicas a_n y A_m de periodo N satisfaciendo las ecuaciones (22) y (23).

2.4.2 Transformada Rápida de Fourier FFT

La transformada rápida de Fourier FFT es el algoritmo utilizado para el procesamiento espectral de la señal de voz en el sistema desarrollado. Este algoritmo es un método rápido para la evaluación numérica de integrales de Fourier; propiedad muy útil en este caso, dado al incremento en la rapidez del cálculo computacional.

A continuación se considera el problema computacional de la determinación de los N números A_m a partir de la ecuación (22). Las mismas consideraciones valdrán para la ecuación (24). Al ser las sumas normalmente más rápidas que las multiplicaciones, se cuenta solamente las multiplicaciones requeridas. En el cálculo de $A_0 = a_0 + a_1 + a_2 + \ldots + a_{N-1}$ no interviene multiplicación alguna. Para hallar A_m a partir de

$$A_m = a_0 + a_1 * w_N^m + a_2 * w_N^{2m} + \ldots + a_{N-1} * w_N^{m(N-1)}$$

se deben efectuar $N - 1$ multiplicaciones para cada m. Por lo tanto, el número total de multiplicaciones requeridas es igual a $(N-1)^2$. Dicho número puede ser reducido considerablemente por medio de una técnica conocida como FFT.

Para la determinación de los $2N$ coeficientes de A_m de una serie de Fourier discreta de orden $2N$

$$a_n \xleftrightarrow[2N]{} A_n \qquad (25)$$

basta con calcular los dos conjuntos, B_n y C_n, de coeficientes de serie discreta de Fourier discreta de orden N.

$$b_n \xleftrightarrow[N]{} B_n \qquad c_n \xleftrightarrow[N]{} C_n \qquad (26)$$

Teorema fundamental

Si

$$b_n = a_{2n} \qquad c_n = a_{2n+1} \qquad (27)$$

Son los componentes par e impar respectivamente de una secuencia dada a_n (figura 2.7), se tiene para cualquier m

$$A_m = B_m + w_{2N}^m C_m \qquad (28)$$

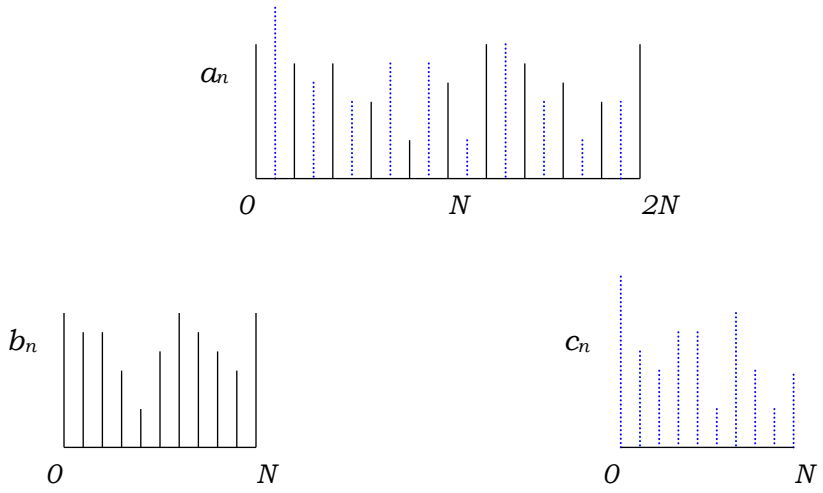


Figura 2.7

Demostración. Si sustituimos N por $2N$ en la ecuación (22) del apartado anterior y sumamos los valores pares e impares de n por separado, obtenemos

$$A_m = \sum_{n=0}^{2N-1} a_n w_{2N}^{mn} = \sum_{k=0}^{N-1} a_{2k} w_{2N}^{2km} + \sum_{k=0}^{N-1} a_{2k+1} w_{2N}^{(2k+1)m}$$

pero $w_{2N}^{2km} = w_N^{km}$, $w_{2N}^{(2k+1)m} = w_N^{km} w_{2N}^m$

$$\sum_{k=0}^{N-1} a_{2k} w_{2N}^{2km} = \sum_{k=0}^{N-1} b_k w_N^{km} = B_m$$

$$\sum_{k=0}^{N-1} a_{2k+1} w_{2N}^{(2k+1)m} = w_{2N}^m \sum_{k=0}^{N-1} c_k w_N^{km} = w_{2N}^m C_m$$

De lo que se desprende la ecuación (28).

El número total de multiplicaciones para la evaluación de B_m y C_m es igual a $2(N-1)^2$, por consiguiente para determinar A_m a partir de la ecuación (28) se necesitan $2(N-1)^2 + 2N-1 = 2N^2 - 2N+1$ ya que en esta ecuación se tiene una multiplicación $w_{2N}^m C_m$ para cada m>0. Si se determinan A_m de modo directo a partir de la ecuación (22) del apartado anterior, el número de multiplicaciones es igual a $(2N-1)^2 = 4N^2 - 4N+1$. Así, el empleo de la ecuación (28) trae consigo una reducción del número de multiplicaciones por un factor cercano a 2.

FFT para $N = 2^s$

En el caso de N par, el proceso anterior se repite, o sea B_m y C_m pueden a su vez determinarse a partir de cuatro series discretas de Fourier de orden N/2. Si $N = 2^s$ se puede empezar la primera iteración con N = 2. Para el cálculo del ahorro computacional resultante, llamemos F(N) al numero de multiplicaciones necesarias para una serie discreta de Fourier de orden $N = 2^s$ obtenida por repetidas iteraciones de la ecuación (28). Un teorema fundamental enuncia que:

$$F(2N) = 2F(N) + N \qquad (29)$$

En efecto, se necesita *2F(N)* multiplicaciones para la evaluación de B_m y C_m, y *N-1* multiplicaciones para el cálculo de A_m a partir de la ecuación (28).

Al ser *F(2)* igual a 1, se desprende de la ecuación (29) que *F(4)* es igual a 4, *F(8)* es igual a 12. Entonces, el número de multiplicaciones viene dado por:

$$F(N) = \frac{N}{2} \log_2 N \qquad (30)$$

La fórmula anterior se cumple para N = 2. Si esta fórmula es cierta para alguna N, en dicho caso, haciendo uso de la ecuación (5) se concluye que:

$$F(2N) = N \log_2 N + N = N \log_2 2N$$

Luego la ecuación (30) también es cierta para 2N.

En el sistema desarrollado, la teoría de la transformada rápida de Fourier y su resultado tiene una consecuencia que además de ser importante es conveniente. Para el procesamiento espectral de la señal de voz, la transformada de Fourier se evalúa a 256 puntos de frecuencia, involucrando un total de 60 025 multiplicaciones cuando el cálculo es realizado de forma ordinaria mediante la ecuación (22). Al usar el algoritmo de la transformada rápida de Fourier el número de multiplicaciones involucradas en el cálculo es de apenas 1024, observando que el ahorro computacional permite la obtención de mayor rapidez en el proceso de reconocimiento de los comandos de voz.

Capítulo Tres

Reconocimiento de Patrones: Redes Auto Organizadas

La auto - organización en redes es uno de los tópicos más fascinantes en el campo de las redes neuronales. Tales redes pueden aprender a detectar regularidades y correlaciones en sus entradas y adaptar sus respuestas futuras para que sean acordes a sus entradas.

Las neuronas de redes competitivas aprenden a reconocer grupos de vectores de entrada similares. Durante el entrenamiento cada neurona en la capa se acerca a un grupo de vectores de entrada ajustando su vector de peso hacia esos vectores. Eventualmente, si hay suficientes neuronas, cada grupo de vectores de entrada similares tendrá una neurona con salida 1 cuando un vector del grupo es presentado, mientras las demás son cero para las otras neuronas. De este modo la red competitiva aprende a categorizar la entrada.

La característica de los mapas auto - organizados (SOM) es aprender a clasificar vectores de entrada acorde a como están agrupados en el espacio de entrada. Ellos difieren de las capas competitivas en que neuronas vecinas en el SOM aprenden a reconocer secciones vecinas del

espacio de entrada. De este modo, los SOM aprenden la distribución (como lo hacen las capas competitivas) y la topología de los vectores de entrada en los que ellas son entrenadas.

En los SOM los vectores de entrada de un espacio de muchas dimensiones son proyectados, por así decir, sobre un mapa bidimensional, de tal manera que se mantenga el orden natural de los vectores de entrada. Esta reducción del número de dimensiones podría permitir visualizar fácilmente relaciones importantes entre los datos que, de otro modo, podrían pasar inadvertidas.

3.1 APRENDIZAJE COMPETITIVO

Las neuronas en una capa competitiva se distribuyen así mismas para reconocer frecuentemente vectores de entrada cuando estos son presentados.

3.1.1 Arquitectura

La arquitectura para una red competitiva se muestra abajo.

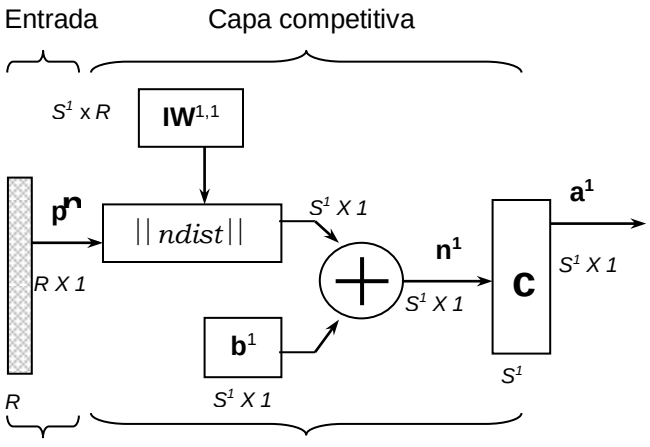


Figura 3.1. Arquitectura de una red competitiva.

La casilla $\|dist\|$ en esta figura acepta el vector de entrada **p** y la matriz de peso **IW**^{1,1} de entrada, y produce un vector que tiene S^1 elementos. Los elementos son los negativos de las distancias entre el vector de entrada y los vectores $_i$**IW**^{1,1} formados de las filas de la matriz de pesos de entrada.

La entrada neta **n**^{1} de una capa competitiva es calculada por hallazgo de la distancia negativa entre el vector de entrada **p** y los vectores de peso adicionando los bias **b**. Si todos los bias son cero, la máxima entrada neta **n** que una neurona puede tener es cero. Esto ocurre cuando el vector de entrada **p** es igual al vector de pesos de las neuronas.

La función de transferencia competitiva acepta un vector neto por capa y retorna salidas neuronales de cero para todas las neuronas excepto para la *ganadora*, la neurona asociada con el elemento más positivo de la entrada neta **n**^{1}. La salida de la neurona ganadora es uno. Si todos los bias son cero entonces la neurona cuyo vector de pesos es más cercano al vector de entrada tiene la *mínima* entrada neta negativa, y por esto gana la competición fijando su salida a uno.

3.1.2 Regla de Aprendizaje de Kohonen

Los pesos de la neurona ganadora (una fila de la matriz de pesos de entrada) son ajustados con la regla de aprendizaje de Kohonen. Suponiendo que la i - ésima neurona gana, la i – ésima fila de la matriz de pesos de entrada es ajustada como se muestra abajo.

$$_i IW^{1,1}(q) = {}_i IW^{1,1}(q-1) + \alpha\left[p(q) - {}_i IW^{1,1}(q-1)\right]$$

La regla de Kohonen asigna los pesos de una neurona para aprender un vector de entrada. Este procedimiento es útil en aplicaciones de reconocimiento.

De este modo, la neurona cuyo vector de pesos fue más cercano al vector de entrada es fijada para ser igual a la más cercana. El resultado es que la neurona ganadora tiene más probabilidad de ganar la competición en la próxima repetición cuando un vector similar le es presentado, y menos probabilidad de ganar cuando muchos vectores diferentes de entrada le son presentados. Cuando más y más vectores de entrada son presentados, cada neurona en la capa se acerca a un grupo de vectores de entrada ajustando sus vectores de pesos hacia esos vectores de entrada. Eventualmente, si hay suficientes neuronas en al capa, cada grupo de vectores similares de entrada tendrán una neurona con salida uno cuando un vector del grupo es presentado, mientras su salida es cero cuando vectores distintos son presentados. De este modo, la red competitiva aprende a categorizar los vectores de entrada que se le presentan.

3.2 MAPAS AUTO – ORGANIZADOS (SOM's)

Las neuronas presentes en una capa de un SOM son ordenadas originalmente en posiciones físicas de acuerdo a la topología del mapa, que puede ser en malla, hexagonal o aleatoria. Las distancias entre las neuronas guardan información muy importante sobre el espacio vectorial de entrada en el que ha sido entrenado el SOM.

La característica de una red SOM identifica una neurona ganadora i^* usando el mismo procedimiento como el empleado por una capa competitiva. Sin embargo, en lugar de fijar solo la neurona ganadora, todas las neuronas dentro de una cierta vecindad $N_{i^*}(d)$ de la neurona ganadora son fijadas usando la regla de Kohonen. Específicamente, todas las neuronas i son ajustadas tal que $i \in N_{i^*}(d)$ como sigue:

$$_{i}w(q) = {}_{i}w(q-1) + \alpha\left[p(q) - {}_{i}w(q-1)\right] \text{ ó}$$

$$_i w(q) = (1-\alpha)_i w(q-1) + \alpha p(q)$$

Aquí el vecindario $N_{i^*}(d)$ contiene los índices para todas las neuronas que se hallan dentro de un radio d de la neurona ganadora i^*:

$$N_i(d) = \{j, d_{ij} \leq d\}$$

De este modo cuando un vector p es presentado, los pesos de la neurona ganadora y sus vecinas cercanas se moverán hacia p. Consecuentemente, después de muchas presentaciones, neuronas vecinas tendrán cada una vectores similares aprendidos.

Para ilustrar el concepto de vecindario, la figura de abajo lo considera. El diagrama de la izquierda muestra un vecindario bidimensional de radio d =1 alrededor de la neurona 13. El diagrama de la derecha muestra un vecindario de radio d =2.

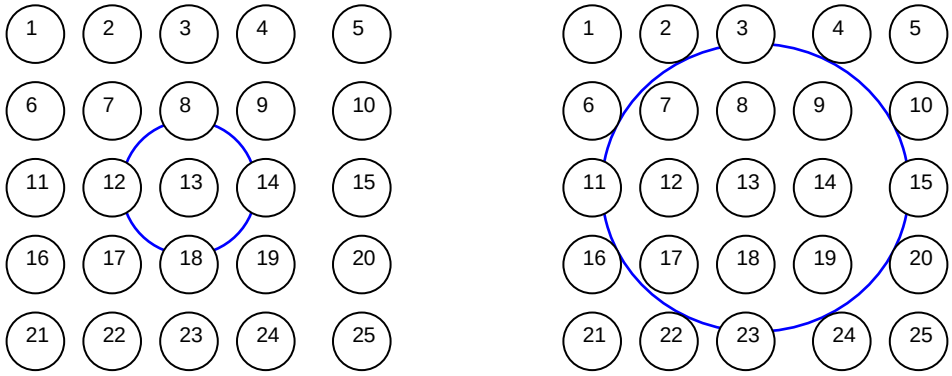


Figura 3.2. Concepto de vecindad.

Las neuronas en un SOM no necesariamente tienen que estar ordenadas en un patrón bidimensional. Un ordenamiento unidimensional, o igual de tres o más dimensiones puede ser usado. Para un SOM unidimensional, una neurona tendrá solo dos vecinas dentro de un radio uno

(o una sola vecina si la neurona esta en cualquiera de los extremos del mapa). La distancia puede ser definida de diferentes maneras usando ordenamientos rectangulares y hexagonales de neuronas y vecindarios según el caso. El funcionamiento de la red no es sensible a la forma exacta de los vecindarios.

La gráfica de abajo muestra una neurona central en una capa bidimensional de neuronas (topología en malla). La neurona central tiene vecindades de diámetro incrementado circundante. Una vecindad de diámetro uno incluye la neurona central y sus vecinas inmediatas. El vecindario de diámetro dos incluye las neuronas de diámetro uno y sus vecinas inmediatas.

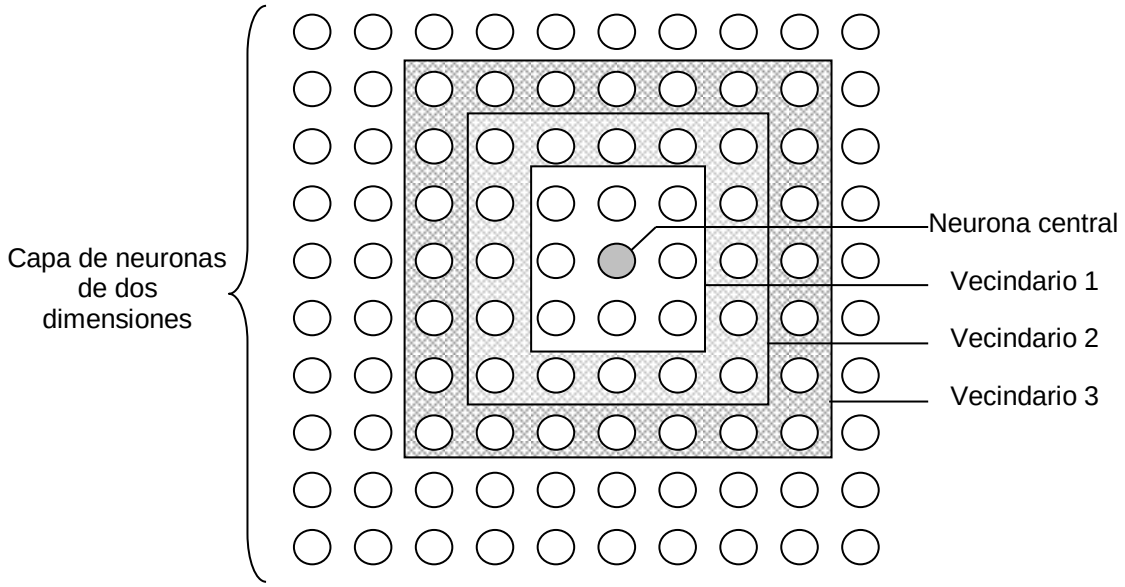
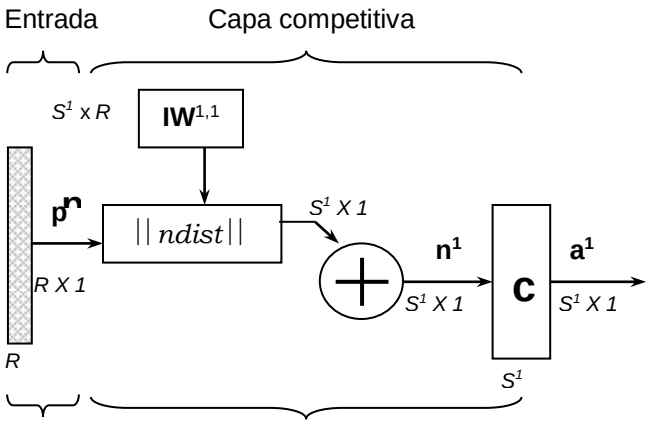


Figura 3.3. Vecindarios de la neurona central.

3.2.1 Arquitectura

La arquitectura para este SOM es mostrada abajo.



$$n_i^1 = -\left\| {}_i IW^{1,1} - P \right\|$$
$$a^1 = compet\left(n^1\right)$$

Figura 3.4. Arquitectura de una red SOM.

Esta arquitectura es parecida a la de una red competitiva, excepto en que el bias no se usa aquí. La función de transferencia competitiva produce un uno para el elemento de salida $\mathbf{a}^1{}_i$ correspondiente a i^*, la neurona ganadora. Todos los otros elementos de salida en $\mathbf{a}^1$ son cero.

Sin embargo, como se describió arriba, la neurona ganadora fija consigo a sus neuronas cercanas. Como se describió anteriormente, se puede elegir varias topologías de neuronas y métodos para calcular la distancia entre la neurona ganadora y sus neuronas cercanas.

3.2.2 Entrenamiento

El entrenamiento en un mapa de características auto - organizado ocurre para un vector en un tiempo independiente, así la red sea entrenada directamente o sea entrenada adaptativamente.

Primero la red identifica la neurona ganadora. Entonces los pesos de la neurona ganadora, y los pesos de las neuronas en su vecindario, son movidos para acercarse al vector de entrada en cada paso de aprendizaje. Los pesos de la neurona ganadora son alterados proporcionalmente a la mitad de la tasa de aprendizaje. La tasa de aprendizaje y la distancia de vecindad, usados para determinar cuales neuronas están en el vecindario de neuronas ganadoras, son alteradas durante el entrenamiento a través de dos fases.

Fase 1: Fase de Ordenamiento

Esta es la fase final para un número dado de pasos. La distancia de vecindad inicia como la máxima distancia entre dos neuronas, y decrece a la *distancia de vecindario sintonizado*. La tasa de aprendizaje inicia en la *tasa de aprendizaje de la fase de ordenamiento* y decrece hasta alcanzar la *tasa de aprendizaje de la fase de sintonización*. Como la distancia de vecindario y la tasa de aprendizaje decrecen sobre esta fase, las neuronas de la red se ordenarán ellas mismas en el espacio de entrada con la misma topología consiguiendo un ordenamiento físico.

Fase 2: Fase de Sintonización

Esta es la fase final para el resto del entrenamiento o adaptación. La distancia de vecindad se apoya en la *distancia de vecindario sintonizado* (incluiría solo los vecinos cercanos, por ejemplo

típicamente 1.0). La tasa de aprendizaje continua decrementandose desde *tasa de aprendizaje de la fase de sintonización*, pero muy lentamente. El pequeño vecindario y un decrecimiento lento en la tasa de aprendizaje sintonizan finamente la red, mientras mantiene el ordenamiento aprendido en la previa fase estable. El número de épocas para la parte sintonizada de entrenamiento (o números de pasos para adaptación) sería mucho más largo que el número de pasos en la fase de ordenamiento, porque la fase de sintonización usualmente toma mucho más tiempo.

En la tabla 1 se muestran los valores específicos de entrenamiento comúnmente usados en estas redes.

Tabla 3.1. Parámetros por defecto usados por Matlab en el entrenamiento de los SOM′s.

`LP.order_lr`	0.9	*Tasa de aprendizaje de la fase de ordenamiento.*
`LP:order_steps`	1000	*Pasos de la fase de ordenamiento.*
`LP:tune_lr`	0.02	*Tasa de aprendizaje de la fase de sintonización*
`LP:tune_nd`	1	*Distancia de vecindario de la fase de sintonización.*

De este modo, los vectores de peso de las neuronas inicialmente toman pasos largos todos juntos hacia el área del espacio de entrada donde los vectores de entrada están ocurriendo. Entonces como el tamaño del vecindario decrece a uno, el mapa tiende a ordenarse topológicamente sobre los vectores de entrada presentados. Una vez el tamaño de la vecindad es uno, la red estaría completamente bien ordenada y la tasa de aprendizaje es decrementada lentamente sobre un periodo largo, para dar a las neuronas tiempo para extenderse igualmente a través de los vectores de entrada.

Como con las capas competitivas, las neuronas en un mapa auto - organizado se ordenarán así mismas con distancias aproximadamente iguales entre ellas, si los vectores de entrada aparecen

completamente con igual probabilidad en una sección del espacio de entrada. También, si todos los vectores de entrada aparecen con frecuencia variable en el espacio de entrada, la característica de la capa del mapa, tenderá a asignar neuronas a un área en proporción a la frecuencia de vectores de entrada.

De este modo, los mapas de características, mientras aprenden a categorizar sus entradas, también aprenden la topología y distribución de sus entradas. Gracias a esta propiedad, estas redes son usadas con éxito en el sistema desarrollado para el reconocimiento de los comandos de voz, porque estos, después del procesamiento espectral, generan un espacio vectorial de entrada compuesto por grupos de vectores que son categorizados, debido a que cada grupo identifica a un comando particular.

Capítulo Cuatro

Desarrollo del Sistema

En este capitulo se expone el desarrollo básico de cada uno de los bloques constitutivos del "Control de Aplicaciones Mediante Comandos Orales Reconocidos por Redes Neuronales", cuyo soporte teórico fue dado en los capítulos anteriores. En este caso el desarrollo esta orientado a aspectos muy particulares del sistema en donde para la implementación de cada uno de los bloques fueron usadas funciones de MATLAB con el objetivo de evaluar su desempeño en esta fase de desarrollo.

Bajo estas consideraciones, el modelo genérico del sistema se representa en forma completa siguiendo una secuencia según las funciones que se realizan. En el modelo de la figura 4.1 se presentan los principales bloques funcionales que intervienen en el diseño y especificación del sistema. Estos bloques son los siguientes:

- Captura y digitalización
- Filtrado.
- Delimitación de la señal de voz.
- Procesamiento espectral de la señal de voz (Transformada de Fourier).

- Patrones de entrenamiento.
- Sistema neuronal de clasificación y decisión.
- Aplicaciones.

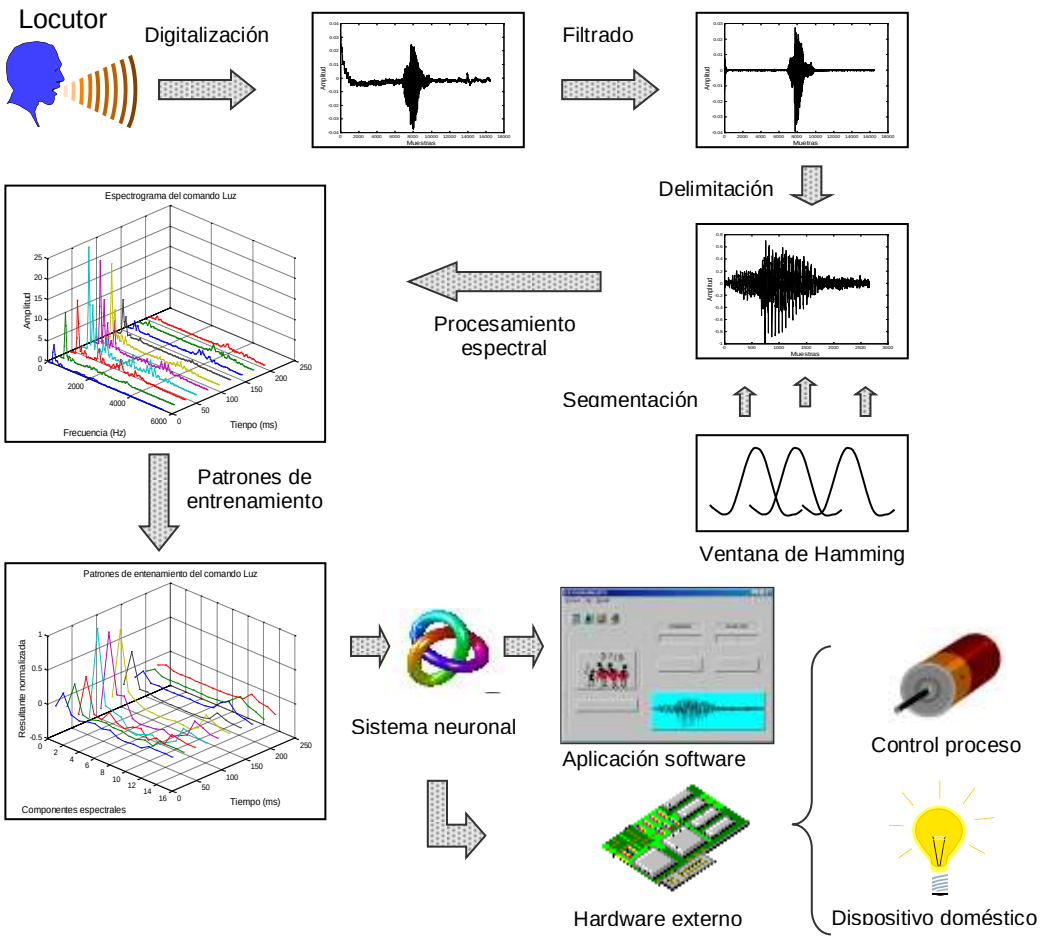


Figura 4.1. Representación general del proceso "Control de Aplicaciones Mediante Comandos Orales Reconocidos por Redes Neuronales".

Los detalles de diseño de los diferentes bloques que aparecen en la figura varían ampliamente de unos sistemas a otros. Además, hay que considerar que el secuenciamiento de funciones que

presenta la figura corresponde en gran parte al modo de operación real, aunque en muchos casos esta correspondencia no se cumple debido a la fuerte interacción entre módulos o funciones.

Teniendo presente las consideraciones anteriores, se pasará seguidamente a describir las funciones desempeñadas por cada bloque de la figura.

4.1 CAPTURA Y DIGITALIZACIÓN

Las pronunciaciones provenientes de un locutor son capturadas por un micrófono y procesadas por una tarjeta de sonido obteniendo una representación digital de la señal de voz, lo que posibilita su almacenamiento en el disco duro del PC para su posterior procesamiento.

La tarjeta de sonido utilizada es una Sound Blaster AWE 64. Como se mencionó, las pronunciaciones son guardadas bajo un formato PCM usando una frecuencia de muestreo de 11025 Hz, modo monofónico y una precisión de 16 bits por dato muestreado.

El primer proceso realizado a las señales de voz almacenadas consiste en efectuar una conversión a un formato tipo vector. Esta representación vectorial permite una manipulación matemática que facilita la ejecución de los diferentes procesos a la que debe ser sometida.

La captura de la señal de voz para el entrenamiento y simulación del sistema es parecido desde el punto de vista de los procesos que se realizan, pero difieren debido a que la captura para el entrenamiento se ejecuta mediante una grabadora de sonidos de Windows, mientras que para la simulación es realizada en forma automática, sin interrupciones.

4.1.1 Captura para el entrenamiento

Este procedimiento es realizado cuando el usuario inicia una sesión de entrenamiento para obtener las pronunciaciones de los comandos con los que el sistema será entrenado.

La captura es básicamente un proceso de adquisición de las pronunciaciones mediante el uso de la grabadora de sonidos de Windows. El proceso comienza cuando el usuario, mediante la ejecución del botón grabar, comienza a capturar sus pronunciaciones para luego almacenarlas en un archivo de extensión .WAV.

4.1.2 Captura para la simulación

Este procedimiento es activado cuando el usuario inicia la simulación del sistema para que, cuando se detecte un comando por el micrófono, el sistema realice la tarea asociada sin ningún tipo de intervención externa.

La captura de la señal de voz es un proceso cíclico de llenado controlado del buffer que almacena los datos capturados. El buffer es básicamente una cola FIFO, que en la figura aparece como un círculo dividido en sectores.

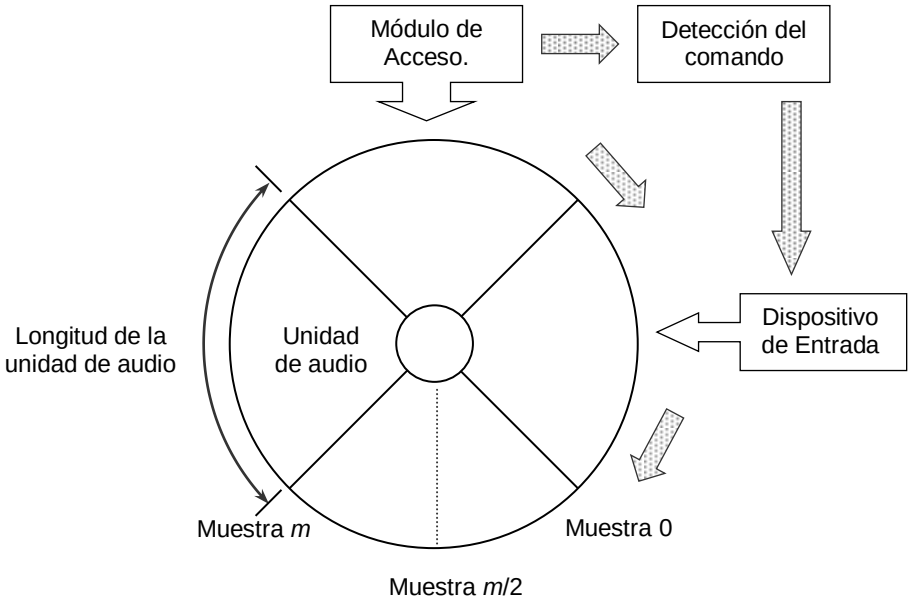


Figura 4.2. Buffer de almacenamiento de los datos capturados

Cada sector representa un paquete de audio y su longitud en bytes depende del número de muestras que contenga el sonido capturado. La cola FIFO es accesada por los procesos Módulo de Acceso y Dispositivo de Entrada.

Dispositivo de Entrada: Encargado de llenar cada uno de los sectores con *m* muestras de audio provenientes del dispositivo de entrada, almacenando las *m* muestras de cada sector del buffer en una sola operación. El dispositivo físico de entrada es el micrófono.

Módulo de Acceso: Accesa cada uno de los sectores de buffer para que el bloque de Detección del Comando examine la presencia o no de algún comando en información contenida en los sectores. El número de bits accesados es el producto de *m* y el número de bits por muestra.

Un aspecto importante es que el Módulo de Acceso no puede accesar sectores que no hayan sido llenados por el Dispositivo de Entrada. Es por eso que el Módulo de Acceso está colocado detrás del Dispositivo de Entrada.

La dinámica del proceso comienza cuando el Dispositivo de Entrada se encuentra frente a un sector, entonces lo llena y se mueve al siguiente sector. En este instante, inicialmente, el Módulo de Acceso se encontraba frente a un sector no procesado por lo que no realizó su función. Apenas se mueve el Dispositivo de Entrada, el módulo de Acceso puede avanzar un sector y entonces accesa su información. El ciclo se repite iterativamente.

Por cada operación del Módulo de Acceso la información de sonido del sector se envía a Detección del Comando. Detección del Comando corresponde a los bloques de filtrado y delimitación del sistema (que serán explicados más adelante) y su función básica consiste en detectar la presencia de algún comando en los sectores. Si en algún sector en consideración no se detecta presencia, el ciclo continúa, pero si es detectado el inicio de algún comando, la información es enviada al resto de los bloques funcionales del sistema. El ciclo de captura continúa hasta que se detecta la finalización del mismo deteniendo el ciclo.

El comando detectado se recupera accesando la información de todos los sectores que lo contienen, para luego ser procesado por el resto de los bloques del sistema para que este haga el reconocimiento y ejecución de la tarea asociada. Después de esto, el ciclo de captura continúa su repetición normal.

Teniendo en cuenta consideraciones empíricas, la longitud de cada segmento es de 1000 muestras, lo que proporciona cerca de un décimo de segundo de duración. La duración no puede ser muy larga para no incluir información innecesaria en caso de que la detección del inicio del

comando sea al final del segmento y la detección de la finalización sea al principio; tampoco puede ser muy corta por que la ejecución de los bloques de filtrado y delimitación se realizaría a una frecuencia alta, lo que aumenta el cálculo computacional.

El número de segmentos que forman el buffer está relacionado con la longitud de los comandos a reconocer. Con una longitud de segmento de aproximadamente un segundo, un número de 100 segmentos proporciona una duración de cerca de 9 segundos de longitud suficientes para nuestro propósito.

4.2 FILTRADO

Este bloque funcional es el encargado de efectuar a la señal de voz un proceso de filtrado digital en el dominio del tiempo. La razón por la cual la acción de filtrado es realizada como primer paso antes de continuar con el procesamiento obedece a la eliminación de ruido y componentes frecuenciales que no aportan información a la señal de voz. De esta forma se obtiene un suavizado de la señal que mejora notablemente el desempeño del algoritmo de delimitación, en contraste de un regular desempeño si un filtrado preliminar no es realizado.

El filtro utilizado es del tipo de respuesta al impulso infinita (IIR), específicamente un pasabanda Butterworth de orden 19 y banda pasante de 300 Hz a 5000 Hz. En la figura 4.2 se observa el efecto de filtrado con estas características sobre la señal correspondiente al comando "Luz".

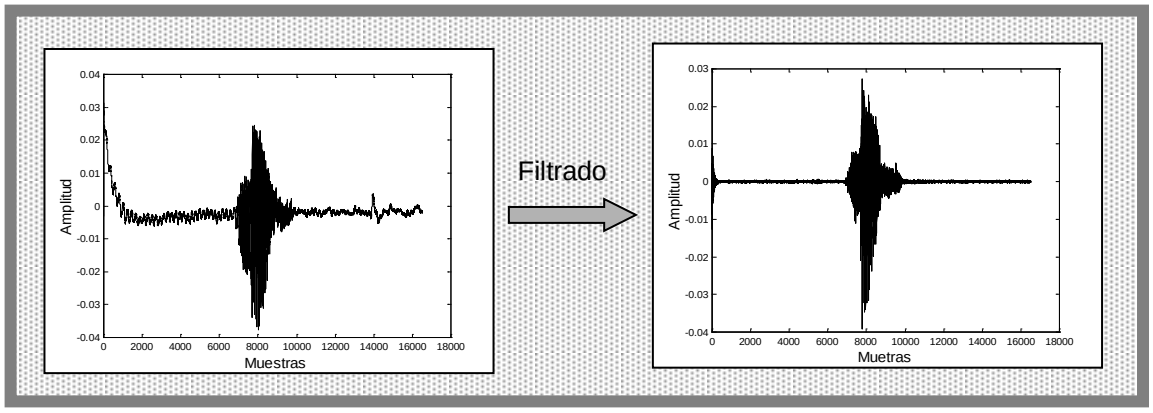


Figura 4.2. Filtrado realizado al comando "Luz".

El diseño del filtro se lleva a cabo bajo el ambiente de MATLAB, usando las funciones para el diseño de filtros según las especificaciones de frecuencia en conjunto con las funciones para el diseño de filtros (funciones `buttord` y `butter`, respectivamente), las cuales obtienen los coeficientes que determinan el funcionamiento descrito antes, el de eliminar componentes de la señal de voz que no aportan información. Una vez hallados los coeficientes, se procede a convolucionarlos con el vector que representa la señal a filtrar para obtener finalmente la señal filtrada (la convolución es el fundamento matemático del filtrado en el dominio del tiempo, que para el caso particular del filtrado es realizada mediante función `filter`).

El valor de los coeficientes de la función de transferencia del filtro que son obtenidos en una etapa inicial son almacenados en una estructura de datos del sistema, de tal forma que permita su utilización cada vez que sea requerida la realización del proceso de filtrado.

4.3 DELIMITACIÓN DE LA SEÑAL DE VOZ

Una vez realizado el filtrado se procede a detectar la palabra del silencio o ruido de fondo que la rodea. Para lograr este propósito, las señales asociadas al silencio o ruido de fondo son eliminadas mediante el uso del algoritmo de delimitación *VicVlad*, el cual actúa sobre la señal de voz según su umbral de amplitud y el número de muestras de referencia.

Este algoritmo básicamente consiste en hacer una exploración a la señal en toda su extensión por grupos de muestras de referencia, realizando un proceso de discriminación por grupo con respecto a un umbral de amplitud, para ser catalogado como señal útil o como silencio o ruido de fondo.

El número de muestras de referencia por grupo es de 45, debido a que este valor no puede ser muy grande o muy pequeño. Para el primer caso, cuando son tomados grupos muy grandes, el algoritmo pierde efectividad, porque puede ser catalogado como señal útil un segmento con muestras que realmente corresponden a un silencio o ruido de fondo. En el caso de los segmentos muy pequeños, se presenta inconvenientes de retardo debido al incremento en el número de segmentos a ser evaluados. Por estas razones, 45 muestras por grupo es el mejor compromiso obtenido, además con pruebas empíricas realizadas en la puesta a punto de este algoritmo.

El umbral de amplitud es determinado según los valores alcanzados por la señal en ausencia de las pronunciaciones hechas por el locutor y bajo condiciones normales de ruido ocasional. Este es calculado haciendo una prueba inicial del ambiente bajo las condiciones anteriormente expuestas, antes de que el sistema procese cualquier sonido.

El proceso inicia accediendo las componentes del vector que representa la señal filtrada en grupos de 45 muestras, extrayendo la mayor del grupo para compararla con el umbral de amplitud. Si la

máxima componente es menor que el umbral, este segmento no es tenido en cuenta para formar el nuevo vector de señal útil. Si es mayor, el segmento corresponderá a las primeras 45 componentes del vector de señal útil, almacenando en la variable `liminf` la posición de la primera componente de este segmento para indicar que en este punto inicia las componentes que forman el nuevo vector. El proceso continúa hasta encontrar un segmento en donde la mayor componente no supere el umbral, en cuyo caso se almacena en la variable `limsup` la posición de la ultima componente de este segmento para obtener de esta forma el límite superior del nuevo vector.

Un aspecto importante a tener en cuenta en la formación del nuevo vector es el caso de encontrar un silencio intermedio que hace parte del comando, el cual no debe ser eliminado. La variable `liminf` es la primera que se encuentra y permanece fija durante todo el desempeño del algoritmo, ya que este limite no debe ser calculado como consecuencia del silencio intermedio. Para la variable `limsup` se tiene un comportamiento temporal, debido a que al presentarse señal significativa después del silencio intermedio se necesita una actualización de esta variable a un nuevo límite superior por cada cumulo de información útil encontrado a medida que se avanza en los grupos de la señal filtrada hasta su finalización.

Los aspectos comentados anteriormente son esquematizados mediante la figura 4.3. La variable *i* representa el recorrido de la señal en tramos de 45 muestras, asignando su valor a las variables `liminf` y `limsup` cada vez que se presentan los cambios de amplitud comentados antes.

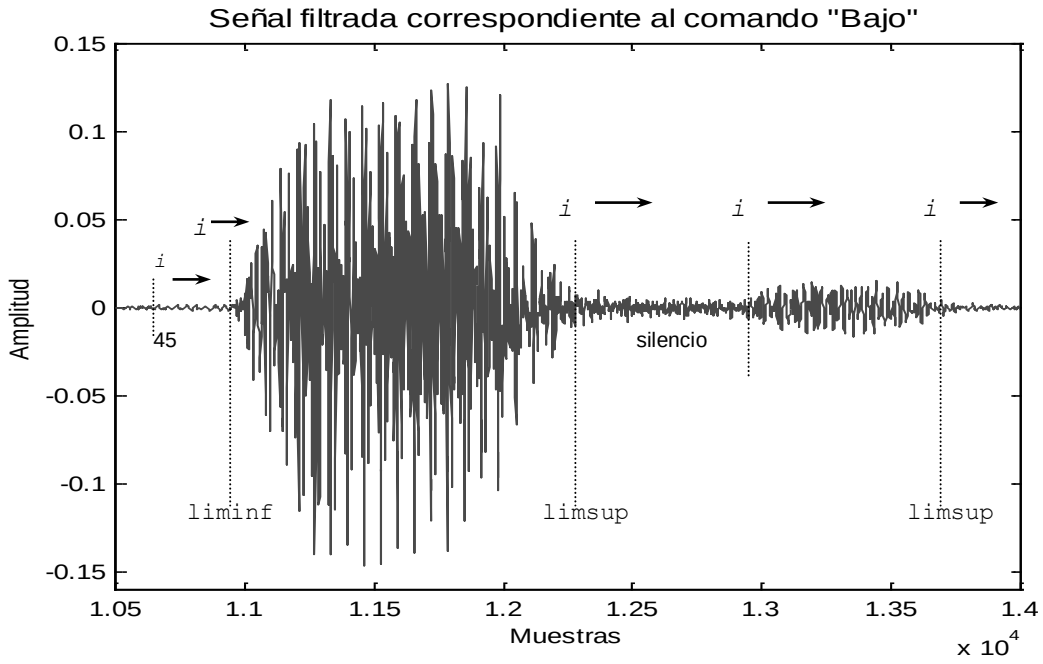


Figura 4.3. Detección del comando "Bajo".

4.4 PROCESAMIENTO ESPECTRAL DE LA SEÑAL DE VOZ

Después de obtener la señal de información útil se procede a realizar el análisis espectral con el objetivo extraer las características más relevantes de los comandos pronunciados, debido a que a nivel frecuencial se obtienen marcadas diferencias entre un comando y otro. Esto proporciona un método que permite la obtención de parámetros que identifican los comandos a reconocer.

La forma de onda en el dominio del tiempo es dividida en segmentos. Estos segmentos son ventanas Hamming con un ancho de 23.22 mseg. y 256 muestras temporales con una frecuencia de muestreo de 11025 Hz.

Se procedió a calcular la transformada de Fourier dependiente del tiempo a cada segmento de habla con el algoritmo de la transformada rápida de Fourier (FFT). Cada segmento transformado es definido por 256 coeficientes de Fourier complejos, donde los primeros 128 corresponden a un rango de frecuencia de 0 a 5.51 KHz. Los primeros 128 coeficientes son los utilizados como características para la diferenciación entre comandos, debido a que los restantes poseen la misma información frecuencial como consecuencia de la simetría par del espectro, como se comentó en el capítulo dos.

El número de coeficientes de Fourier complejos que definen cada segmento transformado (correspondientes a los puntos en frecuencias discretas en las que se calcula el espectro) debe ser definido de tal forma que el espectro resultante presente diferencias claras de un comando a otro. Debido a que el algoritmo de la transformada rápida de Fourier es efectuado cuando el número de puntos en la que se calcula es una potencia de dos, se decide utilizar un número de 256 puntos porque es el mejor compromiso entre la diferenciación espectral y rapidez de cálculo; con 128 puntos los espectros obtenidos no presentan una diferencia clara, y con 528 el proceso de cálculo empieza a tornarse lento.

La norma de cada uno de estos coeficientes representa el espectro de Fourier por ventana, lo que proporciona los vectores espectrales de 128 componentes. La figura 4.3 muestra este aspecto, donde cada gráfica representa a una ventana, que para este caso particular de la pronunciación del comando "Luz" son un total de trece.

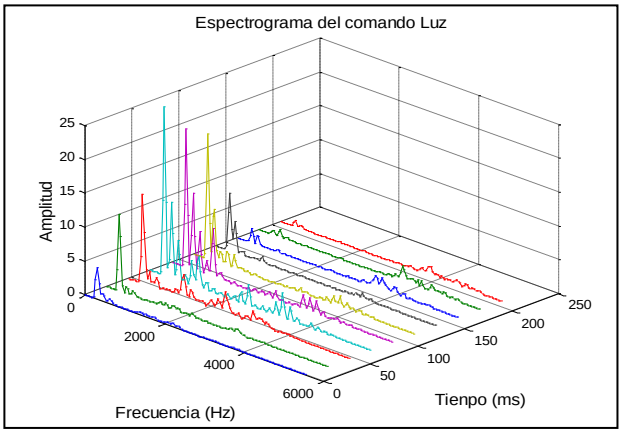


Figura 4.3. Representación espectral de l comando "Luz", aplicando la ventana de Hamming.

En la fase inicial de desarrollo todo el procesamiento espectral descrito es realizado mediante la función de MATLAB *specgram*, la cual retorna el espectrograma de las señales en consideración, y cuyos parámetros de funcionamiento son la señal a procesar, el número de puntos de la FFT por ventana, la frecuencia de muestreo de la señal, la función de ventana y el número de puntos de traslape entre ventanas. Para información completa sobre esta función, se debe consultar el anexo B "Funciones de Matlab para el procesamiento de señales".

4.5 PATRONES DE ENTRENAMIENTO

Una vez obtenidos los vectores espectrales de 128 componentes, se procede a realizar un algoritmo de compresión[1] para adaptarlos como entrada a un mapa auto - organizado. Esto se hace porque es inadecuado realizar una red neuronal con muchas entradas, y además porque para este caso, un vector de muchas componentes presenta información redundante que no contribuye en mucho con el desempeño de la red.

[1] Véase B. KOSKO, "NEURAL NETWORKS FOR SIGNAL PROCESSING". Ed Prentice Hall, 1992. En las paginas 28 y 29 se propone la base de este procedimiento.

Cada vector se agrupó en un nuevo vector real característico de dimensión dieciséis en un espacio euclidiano. La forma de obtener cada componente se explica a continuación.

Como sabemos, el vector de 128 componentes representa el contenido espectral en un rango de frecuencias discretas de 0 a 5.51 KHz. Este rango es dividido en dos regiones, de 0 a 3618 Hz (de la componente 0 a la 84) y de 3618 a 5512 Hz (de la componente 85 a 128). El rango de 0 a 3618 Hz es a su vez dividido en 12 regiones iguales y el rango de 3618 a 5512 Hz en cuatro regiones de iguales características. Siete coeficientes de Fourier representan cada región en el primer rango y once coeficientes de Fourier para el segundo. Luego se procedió a calcular el espectro de potencia promedio sobre cada región para formar el vector patrón de dimensión dieciséis. Todo el proceso descrito antes se repite para todas las ventanas que constituyen los comandos, para obtener finalmente el conjunto de patrones de dimensión dieciséis.

La figura 4.4 ayuda a visualizar el proceso de compresión, donde se destaca la relación existente entre los grupos de componentes del vector de dimensión 128 y las componentes del vector de dimensión dieciséis formado.

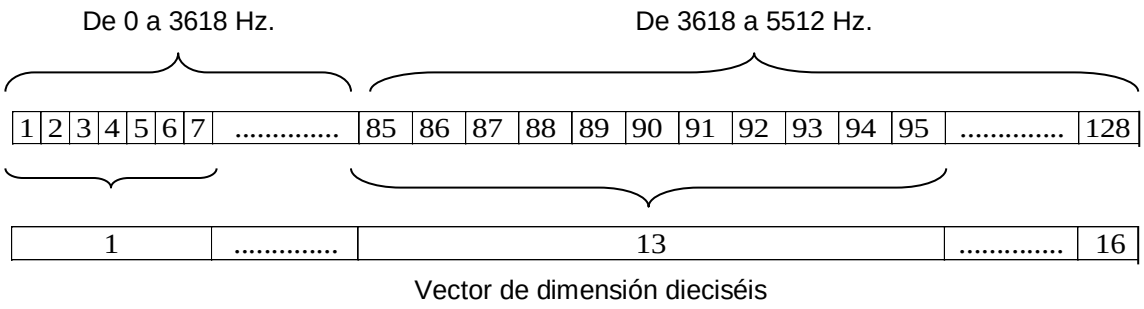


Figura 4.4. Proceso de compresión.

Después de la compresión, a las componentes de las bandas de 0 a 3618 Hz y de 3618 a 5512 Hz se les restó el valor medio de todas las componentes de su respectiva banda. Luego se procedió a normalizar el vector resultante usando como norma el valor de la máxima componente de todos los vectores resultantes que forman un comando particular. La figura 4.5 muestra los resultados obtenidos en este proceso.

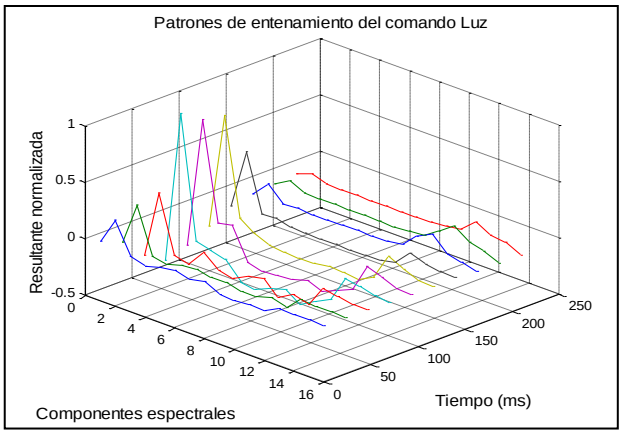


Figura 4.5. Patrones de entrenamiento de la pronunciación del comando "Luz". Las gráficas tienen un espaciamiento entre sí de 23,2 ms.

En el proceso de obtención de los vectores de entrenamiento, las funciones `norm` y `mean` provistas por MATLAB son empleadas. La primera función extrae la norma de cada una de las componentes del vector de dimensión 128 y la segunda el valor medio de las bandas de 0 a 3618 Hz y de 3618 a 5512 Hz. Para información completa sobre estas dos funciones, se debe consultar el anexo B "Funciones de Matlab para el procesamiento de señales".

Una vez obtenidos los vectores de entrenamiento se procede a ordenarlos en forma adecuada para ser presentados al sistema neuronal, debido a que el entrenamiento y simulación de este sistema es realizado por ventanas. Los vectores de entrenamiento obtenidos son columnas de dieciséis

elementos que forman matrices asociadas a cada pronunciación, pero por la razón expuesta antes, estos vectores se agrupan en matrices donde cada una contiene los vectores correspondientes a una ventana particular de todos las pronunciaciones. Las matrices así constituidas son las matrices de entrenamiento, y como consecuencia del proceso anterior, la i - ésima matriz de entrenamiento esta formada por todas columnas correspondientes a la j - ésima ventana de todas las pronunciaciones de un comando particular.

4.6 SISTEMA NEURONAL DE CLASIFICACIÓN Y DECISIÓN

Los patrones de entrenamiento obtenidos del procesamiento espectral son usados como entrada a un sistema neuronal de clasificación.

El sistema neuronal de clasificación (figura 4.6) está formado por 13 SOM, donde cada uno tiene una dimensión de 2X4 y vecindario 1. A cada SOM le es asociado uno de los 13 segmentos transformados de habla, cuyo contenido son los patrones que identifican los comandos a reconocer. La asociación se hace con el objetivo de que cada SOM haga el entrenamiento y reconocimiento de los patrones pertenecientes al segmento de habla asociado.

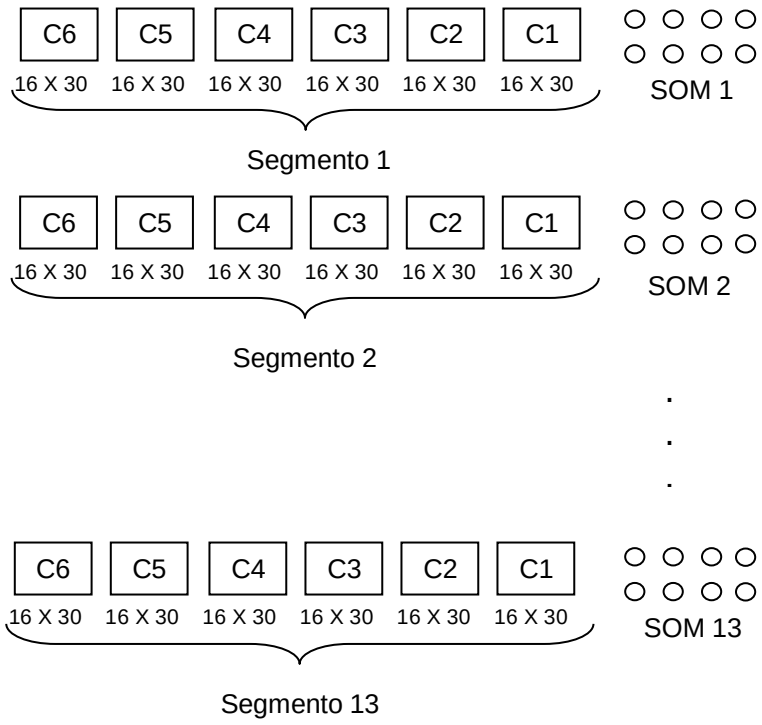


Figura 4.6. Sistema neuronal de clasificación.

Las clases obtenidas por los 13 SOM de clasificación sirven como patrones de entrada al sistema neuronal de decisión, el cual es un SOM de dimensión 4X2 y vecindario 1.

Cada SOM de clasificación reconoce clases de vectores asociados a los comandos a reconocer según el segmento al que esté asociado; de este modo, la respuesta conjunta del sistema de clasificación proporciona vectores que identifican completamente el comando a reconocer, y por lo tanto, se necesita de un sistema que decida el comando reconocido según la información presentada por este sistema. El SOM de decisión es precisamente el encargado de esta labor (figura 4.7).

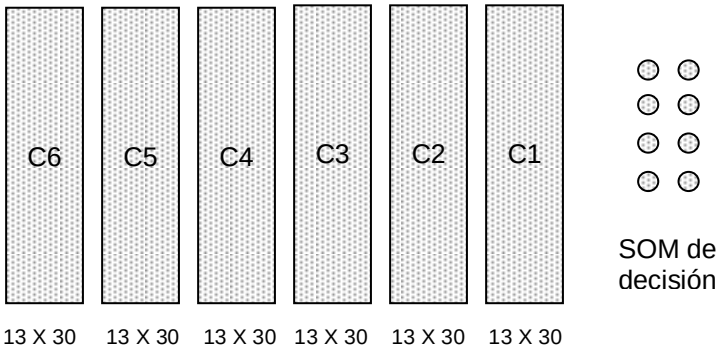


Figura 4.7. SOM de decisión.

En el SOM de decisión C1, C2, C3, C4, C5 y C6 son las clases que identifican cada comando, donde sus 13 filas son dadas por cada SOM del sistema neuronal de clasificación y sus 30 columnas por el número de veces que se repite un comando para el entrenamiento.

Los SOM del sistema neuronal de clasificación son entrenados con matrices cuyas columnas son vectores de dimensión dieciséis, los cuales representan la información espectral de un segmento de habla particular de los comandos que se le enseñan al sistema.

El SOM del sistema de decisión es entrenado con las clases suministradas por el sistema neuronal de clasificación, las cuales son vectores de dimensión trece.

Al sistema le son enseñados seis comandos de trece segmentos de longitud, donde cada uno es repetido 30 veces, por que con menos repeticiones, la red no logra diferenciar muy bien los 6 comandos. Con mas repeticiones se aumenta el tiempo de procesamiento, dando como resultado un valor de 30 muestras. Esto significa que a cada SOM asociado a una ventana particular le son

enseñados 180 patrones, dando un total de 2340 patrones de entrenamiento para el sistema neuronal de clasificación.

Con 2340 patrones de entrenamiento, el sistema neuronal de clasificación responde con 180 vectores de dimensión trece (cada SOM genera una componente), los cuales corresponden a los patrones de entrenamiento del SOM de decisión.

Cada SOM del sistema neuronal de clasificación y el SOM de decisión fueron entrenados a 1000 épocas. El tiempo aproximado de entrenamiento empleado por el sistema de clasificación fue de 7 minutos y de 1 minuto para el SOM de decisión.

4.7 APLICACIONES

El resultado práctico del reconocimiento de comandos está relacionado a la ejecución de tareas cuando el sistema neuronal reconoce el comando asociado. Las tareas a realizar corresponden a la aplicación software y hardware. La aplicación software consiste en la invocación por comandos de voz de las funcionalidades que permiten la ejecución de la aplicación hardware, activar y desactivar dispositivos domésticos y controlar el funcionamiento de un motor de corriente continua, igualmente por comandos de voz. La aplicación software se especifica con detalle en el capitulo cinco "Modelado del Sistema" y las aplicaciones de control hardware se especifican con detalle en el capítulo seis "Especificación de las Aplicaciones del Control Proceso y Dispositivo Doméstico".

Capítulo Cinco
Modelado del Sistema

En este capitulo se realiza el modelado del sistema usando el lenguaje de modelado unificado UML, debido a la gran aceptación que ha tenido como notación gráfica unificada para representar los modelos de los sistemas desarrollados con el paradigma de la orientación a objetos.

Todos los aspectos relacionados con el modelado del sistema son realizados con el uso de la herramienta Rational Rose 98 Evaluation Edition.

5.1 DIAGRAMA DE CASOS DE USO

En esta fase inicial de modelado los casos de uso ayudan a comprender los requerimientos del sistema enfocado en un desarrollo orientado a objetos. Estos se han convertido en elemento primario de planificación y desarrollo de proyectos.

En los diagramas de casos de uso es muy importante identificar los actores del sistema para proceder con la descripción de los casos de uso. A continuación se identifican los actores del sistema

5.1.1 Identificación de actores y casos de uso

Los actores del sistema son:

Actor 1: Usuario

Actor encargado de ejecutar la aplicación para poder ejercer control sobre el software y sobre los dispositivos hardware (control del proceso y sobre el dispositivo doméstico).

Actor 2: Dispositivo doméstico

Dispositivo casero eléctrico o electrónico que será activado o desactivado por comandos orales. En este caso particular se activará o desactivará una bombilla.

Actor 3: Proceso controlado

Hardware externo sobre el que se realizará control por comandos orales. Consiste de un pequeño proceso con disturbios, específicamente un pequeño motor de corriente continua controlado por un sistema PI.

Actor 4: software controlado

En la aplicación, la invocación del software que realiza operaciones de control sobre el dispositivo domestico o sobre el proceso controlado será realizado mediante comandos de voz.

Después de identificar los objetivos del usuario y las funcionalidades que el sistema debe desarrollar para los actores se obtuvieron los siguientes casos de uso de interacción con el sistema para alcanzar los objetivos mencionados.

- Ingresar al sistema.
- Validación de ingreso.
- Desplegar gráficas.
- Procesamiento de señal.
- Captura de señal.
- Entrenamiento.
- Reconocimiento.
- Aplicaciones.

5.1.2 Descripción del modelo de casos de uso

Caso de uso 1: Ingresar al Sistema

Proceso ejecutado por el usuario del sistema que se encarga de validar su ingreso. Este caso inicia cuando el usuario de la orden de ingreso, la cual tiene dos opciones:

- Cuando el usuario llama una nueva sesión se solicita el *nombre* y *clave* iniciales y se prosigue con la sesión una vez registrado su *nombre* asociado.
- Cuando el usuario llama la continuación de una sesión (con la opción abrir del menú o mediante el icono asociado) el sistema solicita el *nombre* y *clave* al usuario. Los datos enviados por el usuario son verificados por este caso de uso comprobando si el *nombre* esta registrado y la correspondencia de la *clave* a este *nombre*.

Caso de uso 2: Validación de Ingreso

Encargado de recibir y asignar el nombre y la clave de un usuario cuando inicia una nueva sesión, logrando de esta forma registrar el *nombre* a la sesión asociada. El inicio de una nueva sesión incluye la configuración de los comandos con los que opera el sistema.

Caso de uso 3: Captura de señal

Este caso de uso es ejecutado por el actor usuario cuando este requiere que el sistema capture sus comandos orales para operaciones de entrenamiento o simulación.

Para el caso del entrenamiento, la captura de señal es realizada por la grabadora de sonidos de Windows, en la cual la captura de comandos es controlada por el usuario mediante el uso de los botones "grabar" para iniciar la captura, y alto para detenerla. Además presenta la facilidad de escuchar los comandos capturados mediante el uso del botón "reproducir".

En la simulación el usuario activa un *ciclo de captura automático* que posibilita al sistema responder a los comandos orales sin necesidad de que se tenga que hacer algún tipo de intervención externa.

Caso de uso 4: Procesamiento de señal

Este caso de uso es activado cuando el usuario ha ingresado las treinta pronunciaciones para obtener las matrices de entrenamiento de cada uno de los comandos a reconocer, o cuando el *ciclo de captura automático* detecta una pronunciación para obtener las matrices de

reconocimiento. Este caso de uso es muy importante en el sistema debido a la variedad de procesos que realiza:

- Carga de los archivos de sonido correspondientes a las pronunciaciones.
- Filtrado de cada una de las pronunciaciones.
- Extracción de la señal útil.
- Normalización.
- Procesamiento frecuencial de cada una de las pronunciaciones.
- Compresión de las componentes frecuenciales.
- Obtención de las matrices de entrenamiento y reconocimiento correspondientes a cada una de las pronunciaciones.
- Normalización de estas matrices.

Caso de uso 5: Entrenamiento

Caso de uso activado por el actor usuario cuando se han obtenido todas las matrices correspondientes a los patrones de entrenamiento de todos los comandos a reconocer.

En este caso de uso el entrenamiento es realizado primero sobre el *sistema neuronal de clasificación* para luego, con los resultados obtenidos en este, iniciar el entrenamiento del *sistema neuronal de decisión.*

Caso de uso 6: Reconocimiento

Caso de uso habilitado por el actor usuario cuando este ha tenido una sesión de entrenamiento exitoso. Este caso de uso utiliza a *procesamiento de señal* para obtener los patrones de reconocimiento de las pronunciaciones para realizar la simulación de los sistemas neuronales.

Caso de uso 7: Aplicaciones

Este caso de uso es activado por *reconocimiento* según el resultado del comando_reconocido. Cuando se reconoce un comando, este caso de uso interactúa con los actores asociados.

5.1.3 Elaboración del modelo de casos de uso

El diagrama de casos de uso se muestra en la figura 5.1

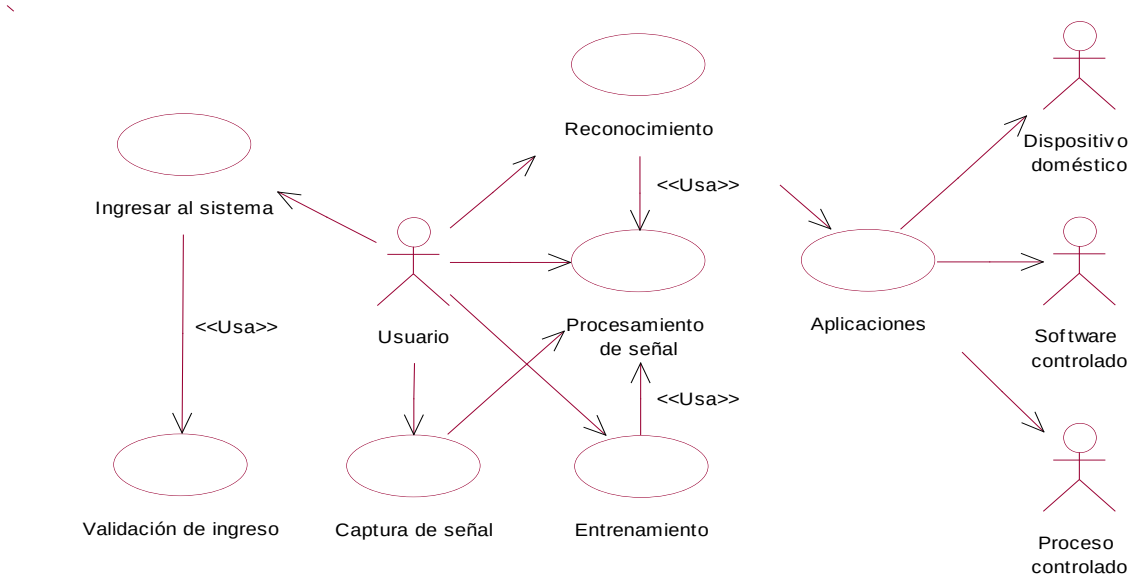


Figura 5.1. Diagrama de casos de uso.

5.2 ANÁLISIS DEL SISTEMA

En este apartado se presenta el análisis del "Control de Aplicaciones Mediante Comandos Orales Reconocidos por Redes neuronales", especificando de manera abstracta la forma de cómo el sistema es visto por los actores.

5.2.1 Identificación de clases

A continuación se presentan las clases que se obtienen a partir de la descripción del modelo de casos de uso:

- Sistema.
- Registro_Usuario.
- Sonido.
- Comunicación_DDE.
- Aplicación_SW.
- Control_PI.
- Matlab.

5.2.2 Elaboración de los diagramas de secuencia

La descripción dinámica del sistema en términos de la interacción entre los distintos objetos o clases que lo forman es realizada mediante los diagramas de secuencia. Esta interacción es llevada a cabo a través de mensajes.

A continuación se desarrollan los diagramas de secuencia para cada caso de uso.

Caso de uso 1: Ingresar al Sistema

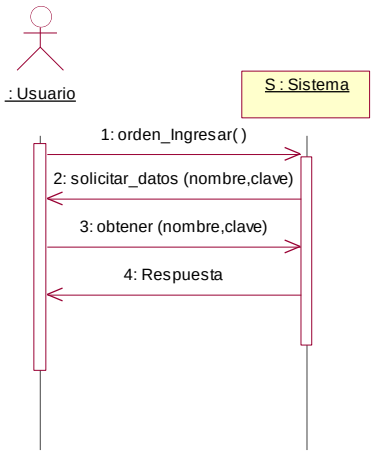


Figura 5.2. Diagrama de secuencia para el caso de uso Ingresar al Sistema.

Caso de uso 2: Validación de Ingreso

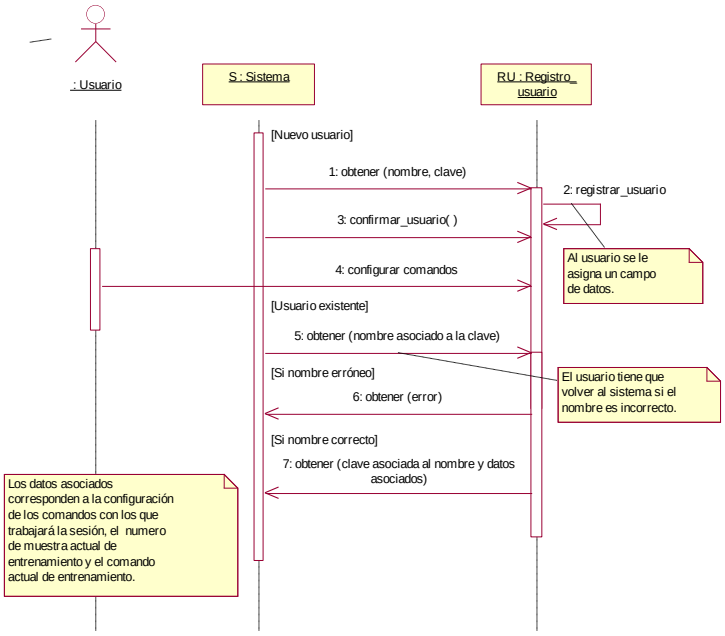


Figura 5.3. Diagrama de secuencia para el caso de uso Validación de Ingreso.

Caso de uso 3: Captura de señal

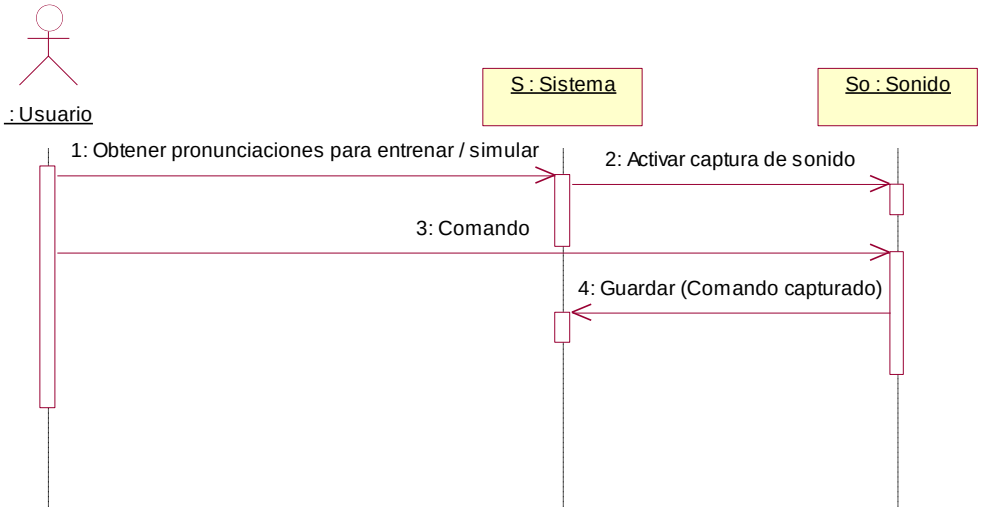


Figura 5.4. Diagrama de secuencia para el caso de uso Captura de Señal.

Caso de uso 4: Procesamiento de señal

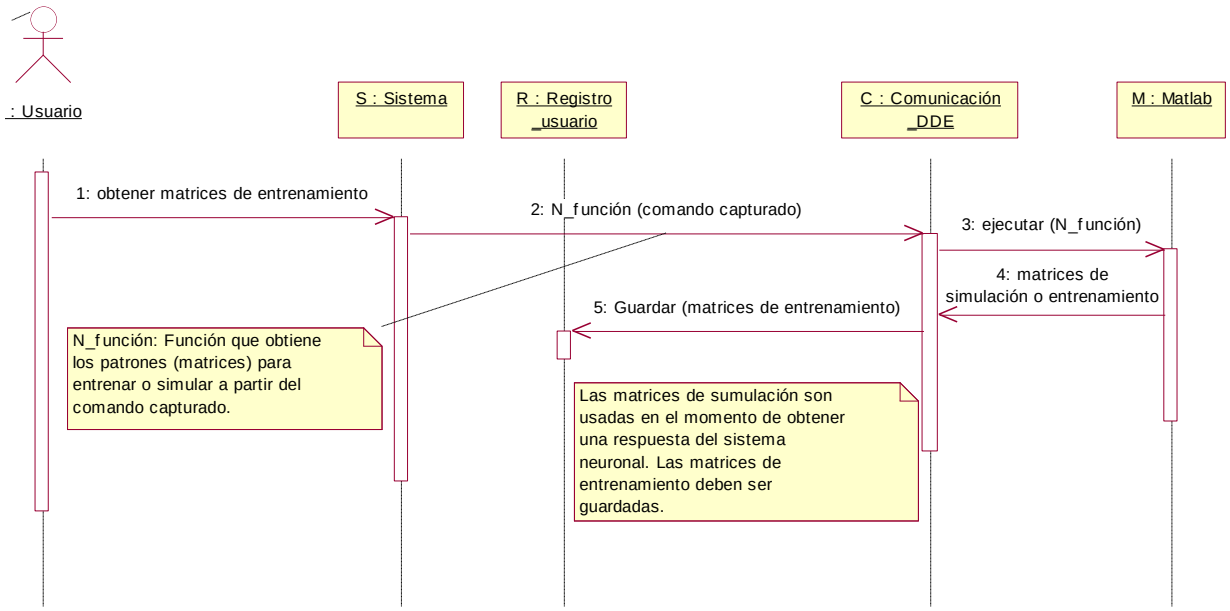


Figura 5.5 Diagrama de secuencia para el caso de uso Procesamiento de Señal.

Caso de uso 5: Entrenamiento

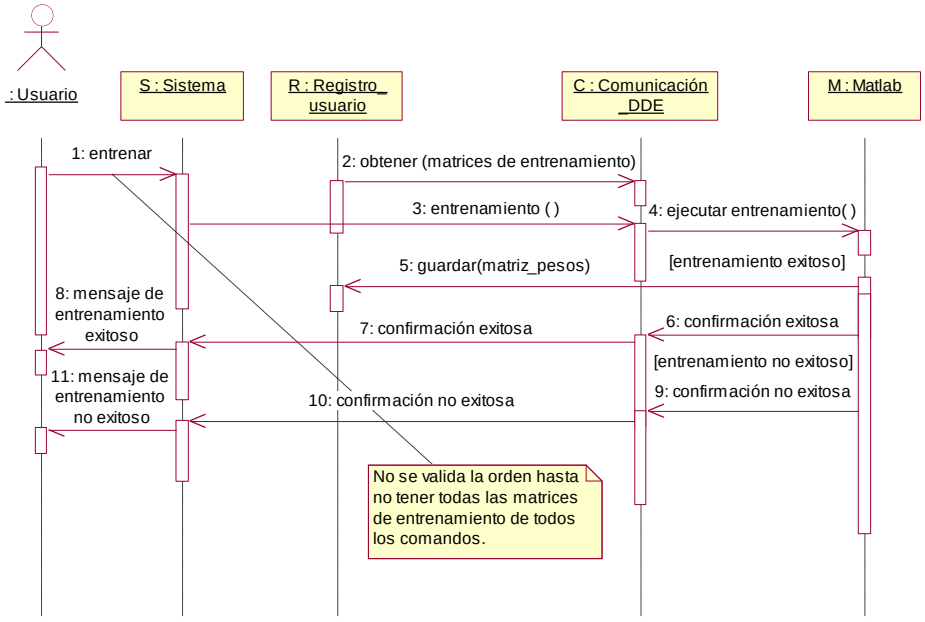


Figura 5.6. Diagrama de secuencia para el caso de uso Entrenamiento.

Caso de uso 6: Reconocimiento

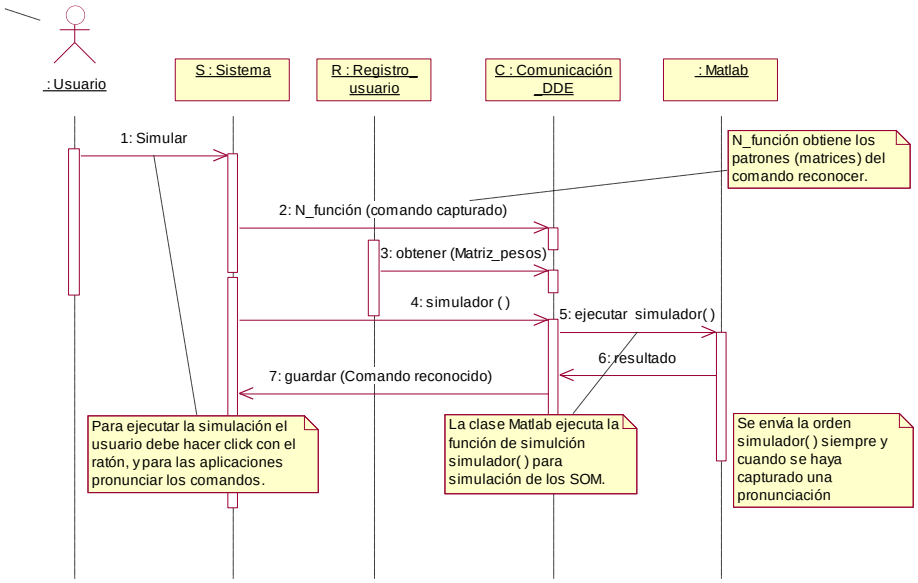


Figura 5.8. Diagrama de secuencia para el caso de uso Reconocimiento.

Caso de uso Aplicaciones

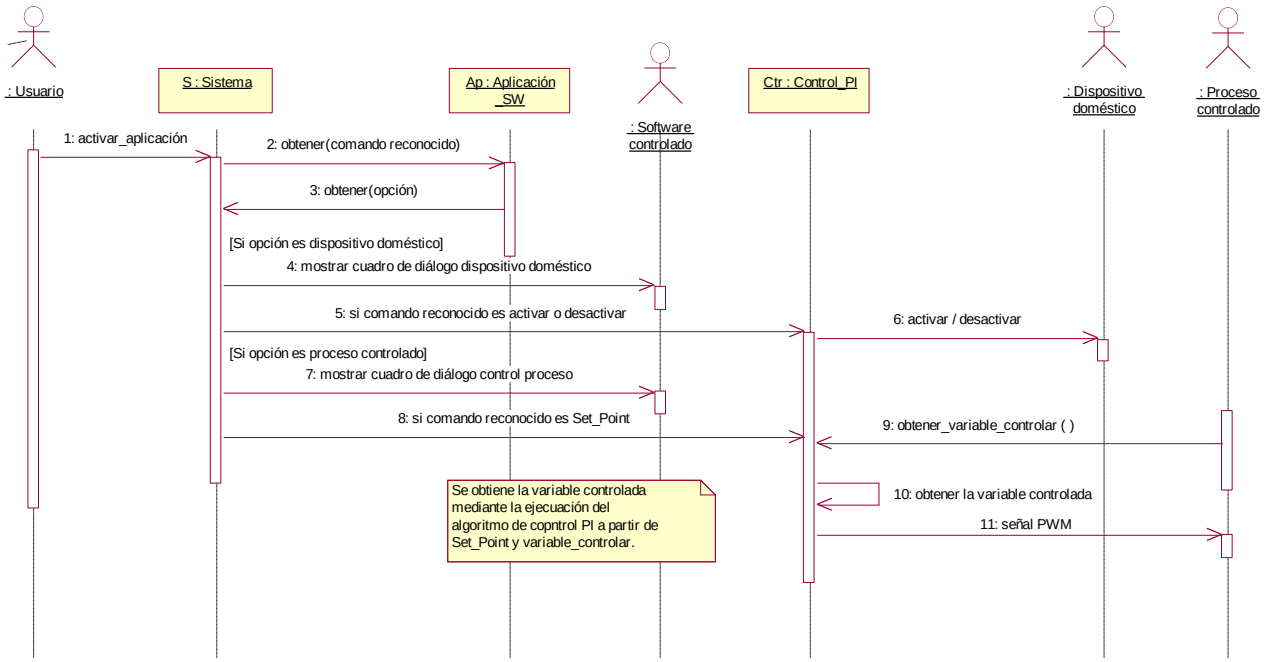


Figura 5.9. Diagrama de secuencia para el caso de uso Aplicaciones.

5.2.3 Identificación de atributos y operaciones

A continuación se hace la descripción de atributos y operaciones de cada una de las clases del sistema. Esta descripción esta basada en el papel que cada una desempeña en los diagramas de secuencia.

Clase Sistema

Esta clase realiza la administración de todos los procesos realizados por el "Control de Aplicaciones Mediante Comandos Orales Reconocidos por Redes Neuronales". Las acciones realizadas para administrar el funcionamiento de la aplicación son las siguientes:

- Validar el ingreso del usuario y mantener una sesión con este para el desarrollo de la aplicación.
- Detectar un nuevo usuario.
- Llevar a cabo las solicitudes del usuario para ingresar al sistema, solicitar gráficas, obtener las pronunciaciones para entrenar y reconocer, obtener las matrices de entrenamiento y reconocimiento, entrenar, simular y activar las aplicaciones.

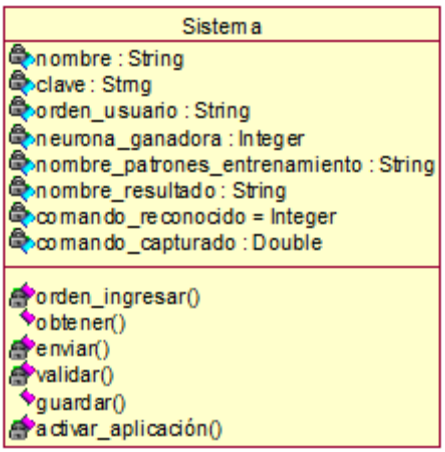


Figura 5.10. Diagrama de la clase Sistema.

Los atributos de esta clase son los siguientes:

- *nombre* identifica a los usuarios registrados.
- *clave* es el código de seguridad para el ingreso a la aplicación de los usuarios registrados.
- *orden_usuario* como su nombre lo indica, corresponde a las solicitudes que el usuario realiza para poder ejecutar las aplicaciones del sistema desarrollado.
- *neurona_ganadora* identifica la respuesta del *sistema neuronal de decisión* cuando un comando ha sido reconocido.
- *nombre_patrones_entrenamiento* corresponde a los nombres de los diferentes archivos que contienen las matrices de entrenamiento para un comando particular.
- *nombre_resultado* corresponde al nombre del archivo que guarda los resultados de entrenamiento (las matrices de peso y las neuronas ganadoras).
- *comando_reconocido* identifica al comando reconocido por el sistema.

- *comando_capturado* identifica al comando capturado por el sistema.

Las operaciones que realiza esta clase son:

- *orden_ingresar* ( ) es la orden de ingreso al sistema proporcionada por Usuario.
- *obtener ( )* es una operación orientada a la obtención de variables de las clases del sistema desarrollado. En este caso, obtiene *nombre* y *clave* del actor Usuario, y *error* de Registro_usuario en caso de que el usuario introduzca erróneamente su nombre asociado; si el nombre es correcto, obtiene de la clase Sistema la *clave* asociada al *nombre.* Además obtiene de la clase Aplicación_SW la *opción* de *dispositivo doméstico* o *control proceso.*
- *enviar( )* es la operación encargada de enviar *orden_usuario* a las clases que ejecutan las solicitudes del usuario que han de ser realizadas en el ambiente Matlab tales como realizar gráficas, obtener matrices de entrenamiento, simular y entrenar.
- *validar( )* es una operación que valida la correspondencia entre *clave* y *nombre* para controlar el ingreso de los usuarios al sistema desarrollado.
- *guardar( )* permite el almacenamiento en la Clase Sistema el *Comando_capturado* y el *Comando_reconocido.*
- *activar_aplicación( )* es la orden de Usuario para activar las aplicaciones del sistema desarrollado.

Clase Registro_Usuario

Interactúa con la clase sistema en caso de tener un nuevo usuario que inicia una sesión para asignarle *nombre* y *clave* iniciales. Además permite la configuración de los comandos con los que opera el sistema

Esta clase tiene el registro de todos los usuarios que tienen sesiones con el sistema mediante una base de datos, asociando a cada uno con su correspondiente nombre. Además, en caso de que el usuario se equivoque en su nombre, genera eventos de error.

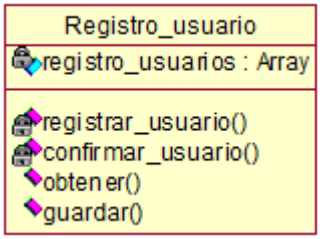


Figura 5.11. Diagrama de la clase Registro_Usuario.

La clase Registro_Usuario presenta como atributo *registro_usuarios*, el cual es el registro de los usuarios en el sistema desarrollado para asignarles a cada uno, un campo de datos para el almacenamiento de sus matrices de entrenamiento y matrices de peso.

Las operaciones que realiza esta clase son:

- *registrar_usuarios( )* registra los usuarios que interactúan con el sistema
- *confirmar_usuario( )* confirma el evento de registro de usuarios en el sistema.
- *obtener( )* es usada por esta clase para obtener de Sistema *nombre* y *clave* y el *nombre asociada a la clave.*

Clase Sonido

Clase que representa el almacenamiento y acceso de los datos de sonido en el buffer para la simulación, y la ejecución de la grabadora de sonidos para el entrenamiento. Sus atributos son las propiedades de los datos a almacenar y las operaciones, las acciones que se ejecutan en el acceso de los datos.

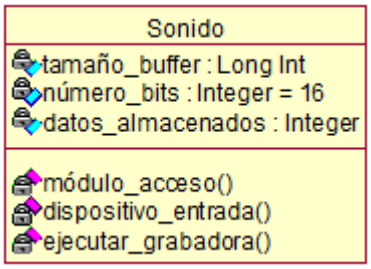


Figura 5.12. Diagrama de la clase Sonido.

Los atributos de esta clase son:

- *tamaño_buffer* es el tamaño del buffer de almacenamiento de los datos de sonido. El tamaño esta dado en el número *m* de muestras almacenadas, que corresponde a 1000.
- *número_bits* es la precisión de los datos de sonido almacenados, que corresponde a dieciséis bits por muestra.
- *datos_almacenados* corresponde a los datos de sonido capturados almacenados en el buffer.

Las operaciones realizadas por esta clase son:

- *módulo_acceso( )* es la operación encargada de accesar los datos de sonido del buffer.
- *dispositivo_entrada( )* es la operación encargada de capturar la información del micrófono y almacenarla en el buffer.
- *ejecutar_grabadora( )* realiza el llamado a la grabadora de sonidos de Windows para adquirir los comandos para el entrenamiento.

Clase Comunicación_DDE

Esta clase es muy importante debido a que es la responsable de establecer la comunicación entre el sistema desarrollado y la clase Matlab cuando son realizados procesos como:

- Las operaciones realizadas por el procesamiento de señales para obtener las matrices de entrenamiento y reconocimiento.
- El entrenamiento de los sistemas neuronales.
- El reconocimiento cuando se invoca la simulación de los sistemas neuronales.

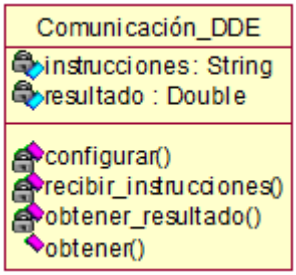


Figura 5.13. Diagrama de la clase Comunicación_DDE.

El atributo *instrucciones* de esta clase corresponde a la cadena de instrucciones que la clase Matlab debe ejecutar.

El atributo *resultado* es la variable donde se almacena los resultados obtenidos de las operaciones hechas por Matlab

Las operaciones de esta clase son:

- *configurar( )* realiza el establecimiento de la comunicación DDE para usar a Matlab como máquina de cálculos.
- *recibir_instrucciones( )* recibe de la clase Sistema la cadena de instrucciones de los procesos que se han de ejecutar en Matlab como plot, N-función(comando_capturado), simulador y entreno.
- *obtener_resultado( )* obtiene las variables calculadas por la clase Matlab como los resultados de la simulación de los sistemas neuronales, *matriz_pesos* y la confirmación de entrenamiento.
- *obtener( )* obtiene de la clase Registro_usuario las variables almacenadas en el campo de datos como *matrices de entrenamiento y matriz_pesos* cuando se hacen operaciones de entrenamiento y reconocimiento respectivamente.

Aplicación_SW.

Clase que gestiona el funcionamiento de la tarea software a realizar, que consiste en la invocación del software que controla la activación del dispositivo doméstico o el proceso controlado cuando el comando asociado a estas operaciones es reconocido.

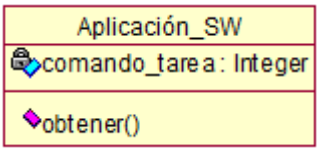


Figura 5.14. Diagrama de la clase Aplicación_SW.

El atributo *comando_tarea* identifica a los comandos que están asociados a las tareas a realizar. La operación *obtener( )* obtiene de la clase Sistema *comando_reconocido* para la ejecución de las tareas descritas.

Clase Control_PI.

La clase Control_PI se encarga de gestionar el funcionamiento del actor Dispositivo doméstico y del actor Proceso controlado mediante un controlador PI. Sus atributos son las variables de control de proceso y las operaciones, se efectúan sobre los disturbios para llevar a cabo la acción de control y sobre Dispositivo doméstico para activarlo o desactivarlo.

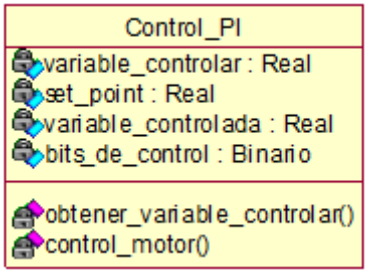


Figura 5.15. Diagrama de la clase Control_PI.

Los atributos que presenta esta clase son:

- *variable_controlar* corresponde en ente caso a la velocidad del giro del motor a controlar.
- *set_point* es el comando_reconocido que corresponde a la velocidad de giro del motor deseada por el usuario.
- *variable_controlada* como su nombre lo indica, corresponde a la variable controlada del proceso. En esta caso es la velocidad de giro del motor.
- *bits_de_control* corresponde a los datos suministrados al actuador del motor.

Las operaciones que realiza esta clase son:

- *obtener_variable_controlar( )* obtiene la variable a controlar del actor Proceso_controlado.
- *control_motor( )* es la operación encargada de ejecutar el algoritmo de control PI que obtiene la *variable_controlada.*

Clase Matlab

Esta clase es el soporte del procesamiento básico del sistema. Sus atributos son los datos que procesa y los datos que obtiene y las operaciones, las funciones en el ambiente Matlab que realiza.

Las funciones realizadas en el ambiente Matlab son invocadas en esta clase por la función *ejecutar( )*, la cual simboliza las instrucciones de la clase Comunicación_DDE.

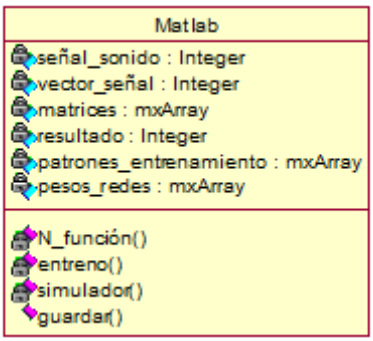


Figura 5.16. Diagrama de la clase Matlab.

Los atributos que presenta esta clase son los siguientes:

- *señal_sonido* corresponde a las señales de sonido a procesar.
- *vector_señal* corresponde a los vectores de la señal de sonido cuando recibe los procesos de filtrado, delimitación, procesamiento espectral y obtención de los patrones de entrenamiento.

- *matrices* corresponde a las matrices de entrenamiento y simulación obtenidas.
- *resultado* es la respuesta final de las redes neuronales cuando han terminado el proceso de entrenamiento y simulación.
- *patrones_entrenamiento* corresponde a las matrices de entrenamiento obtenidas de las pronunciaciones asignadas al entrenamiento de los sistemas neuronales.
- *pesos_redes* son las matrices de pesos de los sistemas neuronales cuando han sido entrenados.

En cuanto a las operaciones presentadas por esta clase tenemos:

- *N_función( )* es una función construida en el ambiente Matlab que obtiene las matrices para entrenar o simular a partir de *comando_reconocido*, el cual corresponde a los datos que representan las señales de sonidos obtenidas del acceso al buffer mediante la función *módulo_acceso( ).*
- *entreno( )* es una función de Matlab que realiza el entrenamiento de los sistemas neuronales.
- *simulador( )* también es una función de Matlab que realiza la simulación de los sistemas neuronales una vez que han sido entrenados.
- *guardar( )* es usada para varios propósito. Guarda en el campo de datos *Matrices de entrenamiento* y *Matriz_pesos* cuando se realiza la obtención de los patrones de entrenamiento y el entrenamiento de los sistemas neuronales respectivamente.

5.2.4 Identificación de relaciones

Al analizar las clases obtenidas y la secuencia de mensajes presentes en los diagramas de casos de uso, se obtienen las relaciones entre las clases de sistema.

La asociación es una relación que indica que existen enlaces entre las clases de los objetos relacionados. Por ejemplo, en el sistema desarrollado, los objetos de cada clase contienen referencias a los objetos de otra clase para acceder a sus atributos o para utilizar las operaciones que ellos ofrecen.

5.2.5 Diagrama de estructura estática

El diagrama de estructura estática se muestra en la figura 5.17, el cual muestra las relaciones existentes entre las clases del sistema.

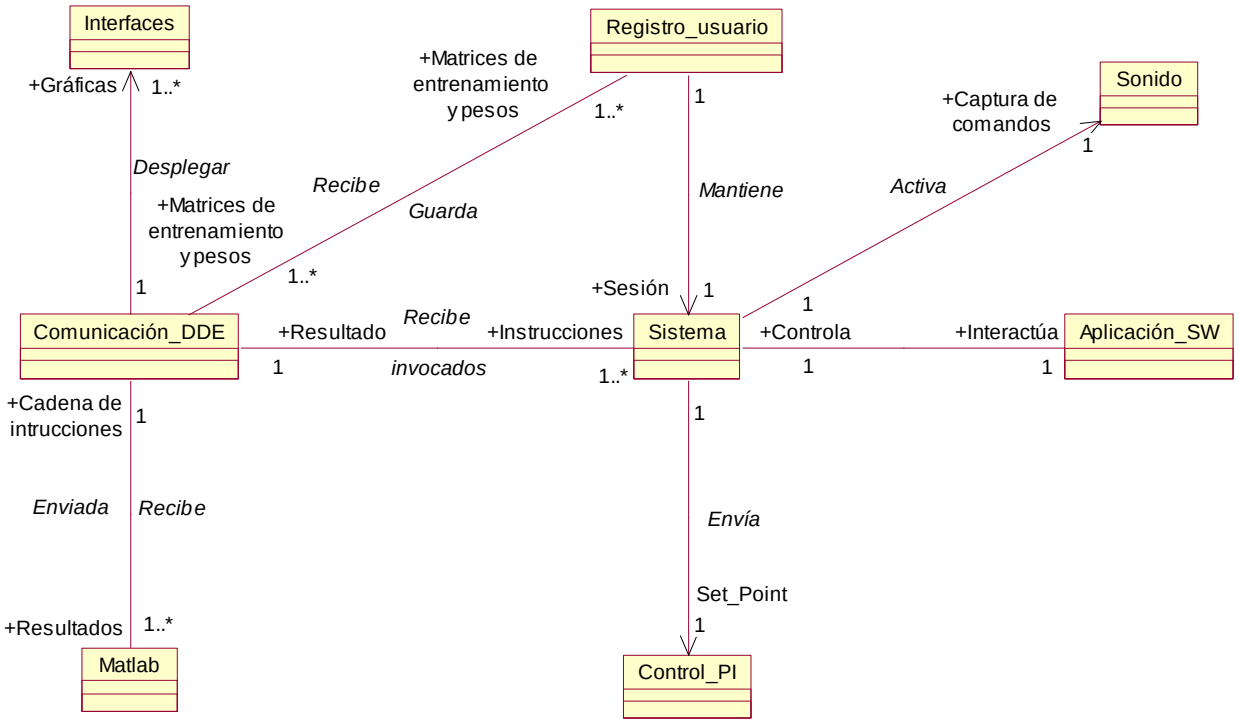


Figura 5.17. Diagrama de Estructura Estática.

Para mayor claridad, las relaciones entre las clases del diagrama de estructura estática son listadas para una correcta interpretación de la cardinalidad y de los roles desempeñados por cada clase.

Para la relación entre Comunicación_DDE, Sistema y Sistema, Comunicación_DDE se tiene respectivamente:

- Comunicación_DDE recibe una o varias instrucciones de Sistema.
- Los resultados obtenidos por sistema es realizado por una sola Comunicación_DDE.

Para la relación Sistema, Aplicación_SW y Aplicación_SW, Sistema se tiene respectivamente:

- El Sistema interactúa con una Aplicación_SW.
- Una Aplicación_SW se controla por un solo Sistema.

Para la relación Sistema, Control_PI se tiene:

- El sistema envía Set_Point a Control_PI.

Para la relación Registro_usuario, Sistema se tiene:

- Registro_usuario mantiene una sesión con Sistema.

Para la relación Comunicación_DDE, Matlab y Matlab, Comunicación_DDE se tiene respectivamente:

- Una Comunicación_DDE recibe una o varias instrucciones de Matlab.
- Una o varias cadenas de instrucciones recibidas por Matlab son enviadas por una Comunicación_DDE.

Para la relación entre Sistema y Sonido se tiene:

- El Sistema activa a Sonido para la captura de comandos.

Para la relación Comunicación_DDE, Registro_usuario y Registro_usuario, Comunicación_DDE se tiene respectivamente:

- Comunicación_DDE recibe una o varias matrices de entrenamiento y pesos de Registro_usuario.

5.3 DISEÑO DEL SISTEMA

La fase de diseño correspondiente al modelado del "Control de Aplicaciones Mediante Comandos Orales Reconocidos por Redes Neuronales" es realizado con detalle en el Anexo D "Diseño del Sistema".

Capítulo Seis

Especificación de las Aplicaciones del Control Proceso y Dispositivo Doméstico

El sistema desarrollado es un conjunto de componentes tanto a nivel de software como de hardware. Como ya se ha visto, los procesos en software además de dar al sistema el soporte matemático y la interacción con el usuario, junto al hardware asociado proporcionan soporte a las aplicaciones del control proceso y dispositivo doméstico.

Así como en el capitulo anterior se hizo la especificación del software del sistema, este corresponde a la especificación del hardware que da soporte a dichas aplicaciones. Primero se considera la tarea del control proceso, que como se sabe corresponde a ejercer control de velocidad de un motor de corriente continua mediante comandos de voz asociados al set point, y luego se considera la activación o desactivación de dispositivos domésticos mediante comandos de voz.

6.1 CONTROL DE VELOCIDAD DE UN MOTOR DE CORRIENTE CONTINUA MEDIANTE COMANDOS DE VOZ

En este proyecto el control de velocidad del motor de corriente continua se realiza mediante la asociación del comando reconocido al set point de un controlador proporcional integral PI. Como se ha explicado anteriormente, el reconocimiento de un comando conlleva a la ejecución de una tarea particular que para este caso corresponde al cambio del set point en el algoritmo de control PI.

Es importante tener en cuenta que la forma de implementar el control en cuanto a distribución física corresponde al uso de dos computadores: uno encargado del reconocimiento de los comandos que en caso de reconocer alguno asociado a un set point particular lo transmite, vía puerto paralelo al computador encargado de ejecutar el algoritmo software de control PI.

6.1.1 Obtención del modelo matemático del motor de corriente continua

Los motores de corriente continua tienen campos separadamente excitados. Son o bien controlados en el inducido (armadura) con campo fijo o de campo controlado con corriente de inducido fijo. En este caso, el motor de corriente continua utilizado es controlado en el inducido con campo fijo.

En la figura 6.1 se muestra el motor de corriente continua controlado en el inducido en donde,

R_a = Resistencia del devanado del inducido en ohmios.

L_a = Inductancia del devanado del inducido en henrios.

i_a = corriente del devanado del inducido en amperios.

e_a = Tensión aplicada a la armadura en voltios.

e_b = Fuerza contra - electromotriz en voltios.

ω = Velocidad del motor en radianes.

T = Par desarrollado por el motor en libras - pie.

J = Momento de inercia equivalente del motor y carga con referencia al eje del motor en libras- pie^2.

f = Coeficiente de fricción viscosa equivalente del motor y carga referenciado al eje del motor en libras – pie/rad/seg.

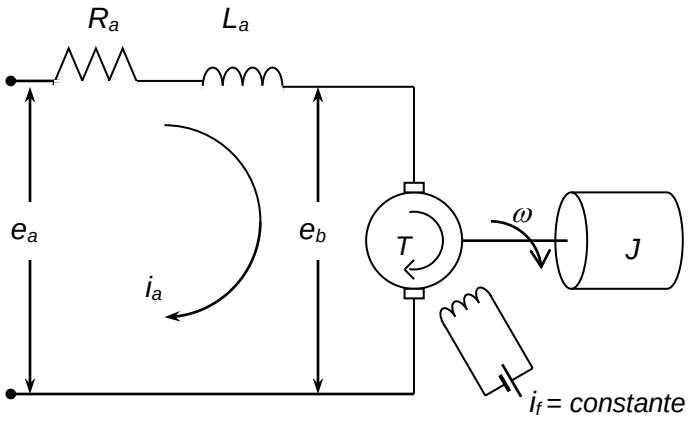


Figura 6.1. Diagrama esquemático de un motor de cc controlado por el inducido.

Con una corriente de campo i_f constante el par del motor se hace directamente proporcional a la corriente de inducido:

$$T = K * i_a \qquad (1)$$

Cuando el inducido está en rotación, se induce en él una tensión proporcional al producto del flujo por la velocidad angular. Para un flujo constante, la tensión inducida e_b es directamente proporcional a la velocidad angular del motor ω:

$$e_b = K_b \omega(t) \qquad (2)$$

Donde K_b es una constante de fuerza contra - electromotriz.

La velocidad del motor de corriente continua controlado por el inducido se controla por medio de la tensión de inducido e_a. La ecuación diferencial del circuito de inducido es:

$$L_a \frac{\partial i_a}{\partial t} + R_a i_a + e_b = e_a \qquad (3)$$

La corriente de inducido produce el par que se aplica a la inercia y fricción, por lo tanto:

$$j \frac{\partial \omega(t)}{\partial t} + f\omega(t) = T = K * i_a \qquad (4)$$

Suponiendo que todas las condiciones iniciales son cero, y tomando las transformadas de Laplace de las ecuaciones (2), (3) y (4) se obtienen las siguientes ecuaciones:

$$K_b W(s) = E_b(s) \qquad (5)$$

$$(L_a s + R_a) I_a(s) + E_b(s) = E_a(s) \qquad (6)$$

$$(Js + f) W(s) = T(s) = K I_a(s) \qquad (7)$$

Considerando a $E_a(s)$ como la entrada y $W(s)$ como la salida, se puede construir el diagrama de bloques de las ecuaciones (5), (6) y (7), como puede verse en la figura 6.2. Se ve el efecto de la fuerza contra – electromotriz en la señal de realimentación proporcional a la velocidad del motor la cual aumenta el amortiguamiento del sistema.

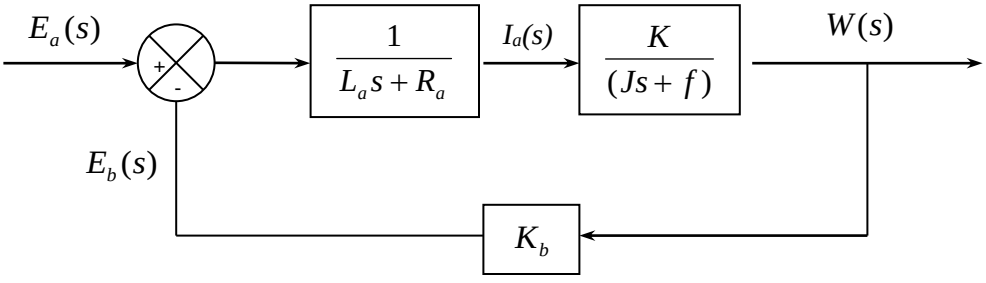


Figura 6.2. Diagrama a bloques de un motor de corriente continua controlado por el inducido.

La función de transferencia del sistema de la figura 6.2 se obtiene como:

$$\frac{W(s)}{E_a(s)} = \frac{K}{L_a J s^2 + (L_a f + R_a J)s + R_a f + KK_b} \qquad (8)$$

Debido a que la constante de tiempo del circuito de inducido $Ra - La$ es mucho menor que la constante de tiempo aportada por el momento de inercia equivalente del motor y carga J, la inductancia La se puede despreciar conllevando a la reducción de la función de transferencia dada por la ecuación (8) a:

$$\frac{W(s)}{E_a(s)} = \frac{K_m}{T_m s + 1} \qquad (9)$$

Donde

$K_m = \frac{K}{R_a f + KK_b}$ es la constante de ganancia del motor

$T_m = \frac{R_a J}{R_a f + KK_b}$ es la constante de tiempo del motor

6.1.1.1 Cálculo de la Constante de Tiempo del Motor.

La constante de tiempo del motor empleado en este proyecto es hallada experimentalmente usando el montaje del motor y un osciloscopio digital. Como puede verse en la figura 6.3, el montaje del motor esta formado por una estructura que soporta al motor, la masa con momento de inercia *J* y el sensor óptico, en donde el eje del motor esta acoplado mecánicamente a la masa a través de una rueda dentada de 40 orificios.

El sensor óptico junto a la rueda dentada se usan para medir la velocidad de giro del motor, gracias a los pulsos generados cuando el haz de luz infrarroja en el sensor pasa por los orificios de la rueda dentada.

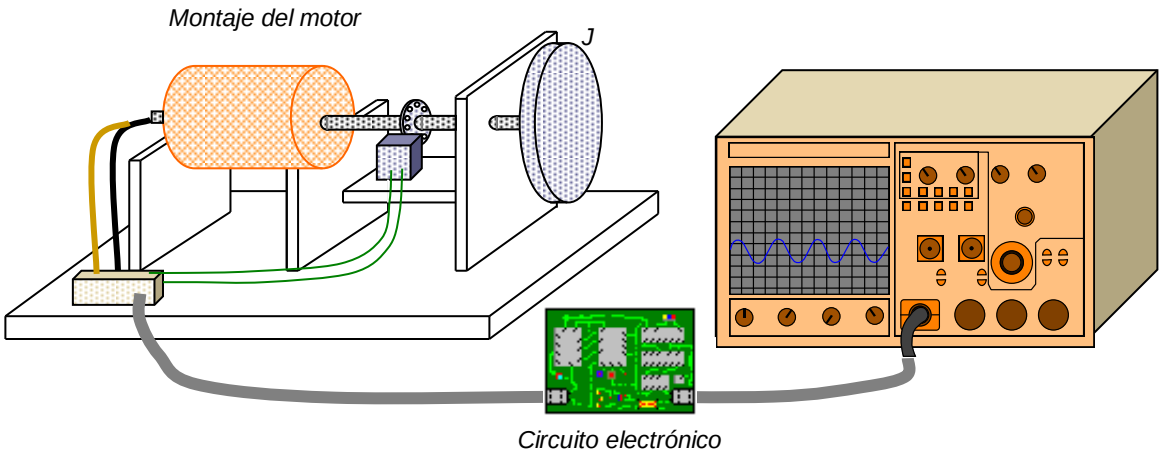


Figura 6.3. Elementos para hallar la constante de tiempo del motor cc.

Para dar mayor corriente a los pulsos del sensor óptico se utiliza un circuito electronico que se detella en el anexo C "Descripción Hardware del Sistema".

La forma de determinar el valor de la constante de tiempo del motor corresponde a la realización de varias pruebas en donde el motor es alimentado con señales escalón de diferente magnitud.

Para cada valor de magnitud de la señal escalón se mide la progresión de la frecuencia de los pulsos visualizados en el osciloscopio digital, desde que el escalón es aplicado hasta que se obtiene el 63% del valor de la frecuencia cuando el motor ha alcanzado estado estacionario. Gracias a la caracteristica de memoria del osciloscopio empleado la señal visualizada se almacena

permitiendo medir el tiempo transcurrido de la prueba, que corresponde a la constante de tiempo del motor.

Al observar la tabla 6.1, los valores de la constante de tiempo T_m obtenidos para cada valor de magnitud de la señal escalón estan alrededor de 170 ms, lo que permite establecer a T_m a este valor.

Un aspecto a tener en cuenta es la correspondencia entre la frecuencia de los pulsos entregada por el sensor óptico y la velocidad de giro del motor. Para el caso de velocidad angular ω en radianes por segundo se tiene:

$$\omega = \frac{f * 2\pi}{N}$$

Y para el caso de velocidad del motor υ en revoluciones por minuto:

$$v = \frac{f * 60}{N}$$

En donde f corresponde a la frecuencia entregada por los pulsos en Hz y N corresponde al número de orificios de la rueda dentada, que en nuestro caso particular es de 40.

6.1.1.2 Cálculo de la Constante de Ganancia del Motor

El valor de la constante de ganancia del motor K_m se obtiene a partir del valor de T_m hallado anteriormente. y esta dado por la relación:

$$K_m = \frac{v}{e_a}$$

Donde υ es la velocidad del motor en revoluciones por minuto en estado estacionario y e_a es la amplitud del escalón aplicado a la armadura. El calculo de K_m se efectúa para cada valor de T_m asociado a un valor de amplitud de la señal escalón aplicado. Como los valores de K_m están alrededor de 300 RPM/v, se toma este valor como la constante de ganancia del motor.

En la tabla 6.1 se observa los distintos parámetros involucrados en la obtención de las constantes de tiempo y ganancia del motor

Tabla 6.1: Parámetros del motor de corriente continua.

Amplitud del escalón (v)	Frecuencia de los pulsos en estado estacionario (Hz)	Velocidad del motor (rad/seg)	Velocidad del motor (RPM)	Constante de tiempo T_m (mseg)	K_m (RPM/V)
2	387.94	60.94	581.91	188.92	290.95
3	584.16	91.76	876.24	186.87	292.08
4	742.48	116.63	1113.72	185.04	278.43
5	921.03	144.67	1381.54	178.32	276.31
6	1166.1	183.17	1749.15	177.37	291.52
7	1455.82	228.68	2183.73	173.80	311.96
8	1706.5	268.06	2559.75	173.02	319.97
9	1893.9	297.49	2840.85	171.99	315.65
10	2051.3	322.22	3076.95	171.88	307.70
11	2331.6	366.25	3497.4	173.23	317.94
12	2554.2	401.21	3831.3	172.85	319.27
Valores aproximados				170	300

6.1.1.3 Comportamiento del Motor

Al realizar las gráficas de los valores involucrados en el cálculo de las constantes de tiempo y ganancia del motor se obtienen características muy importantes acerca de su funcionamiento.

En la figura 6.4a y 6.4b se observan respectivamente, las gráficas de los valores de la tabla 6.1 que relacionan las diferentes magnitudes de la señal escalón aplicada al motor Vs la velocidad de giro y la constante de tiempo T_m. De estas gráficas se observa que la velocidad de giro del motor presenta una buena linealidad y que T_m tiene un comportamiento constante para todas las amplitudes del escalón aplicado.

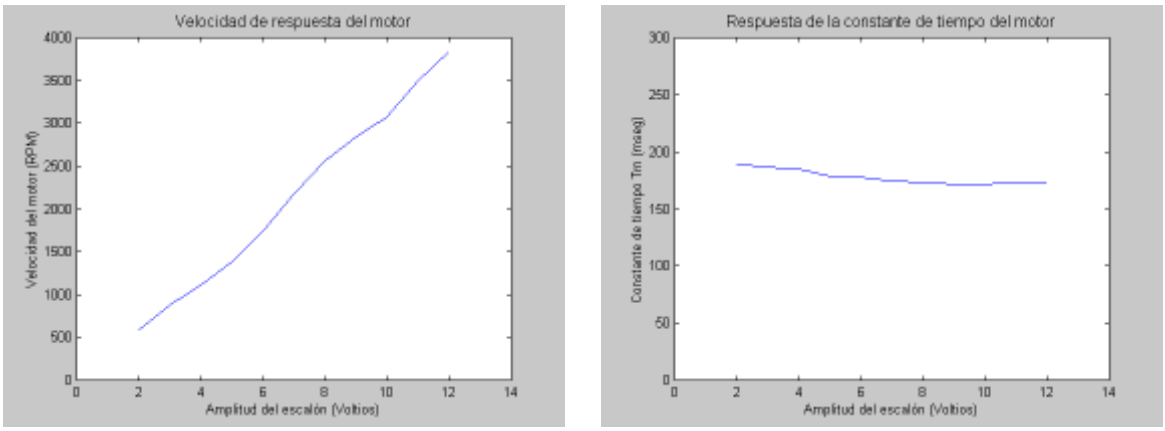


Figura 6.4. (a) Velocidad de respuesta del motor. (b). Respuesta de la constante de tiempo del motor.

Con la obtención de K_m y T_m queda definido el modelo matemático del motor de corriente continua en cuestión a $\frac{300}{170s+1}$. La respuesta al paso en lazo abierto de esta función representa en forma general su comportamiento. La figura 6.5 muestra este aspecto.

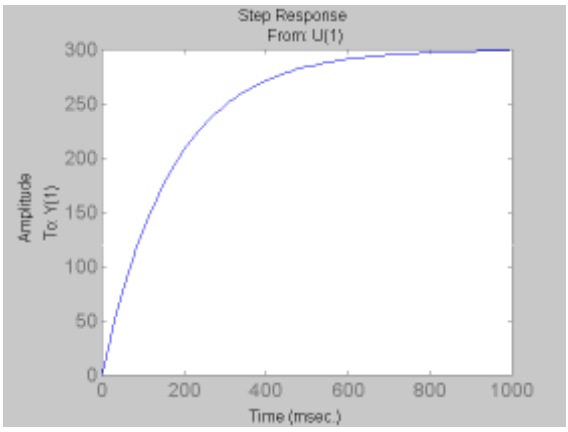


Figura 6.5. Respuesta al paso del modelo matemático del motor.

6.1.2 Control del motor de corriente continua mediante modulación por ancho de pulso PWM.

En la figura 6.6 se muestra el diagrama simplificado del sistema que maneja al motor. El motor es alimentado por una fuente DC a través de un circuito de disparo compuesto por un mosfet Q1 y un diodo de marcha libre D1.

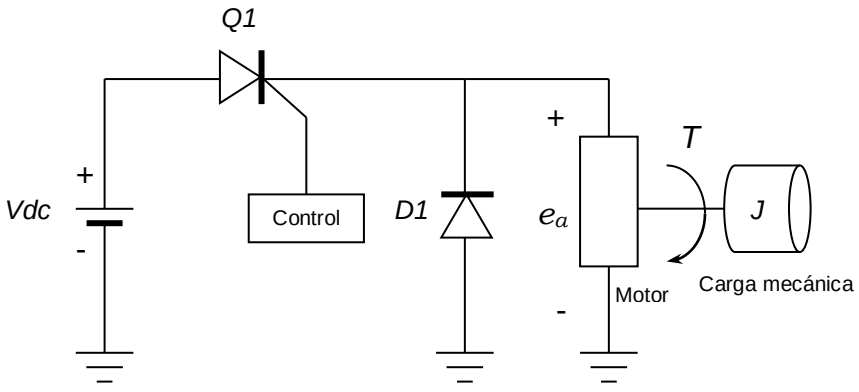


Figura 6.6. Diagrama simplificado del sistema.

Como se había comentado antes, el control de velocidad del motor es efectuado mediante la variación del voltaje de armadura. Para este fin, se utiliza la fuente DC fija, a partir de la cual se produce un tren de pulsos con un ciclo de trabajo variable, como se ve en la figura 6.7.

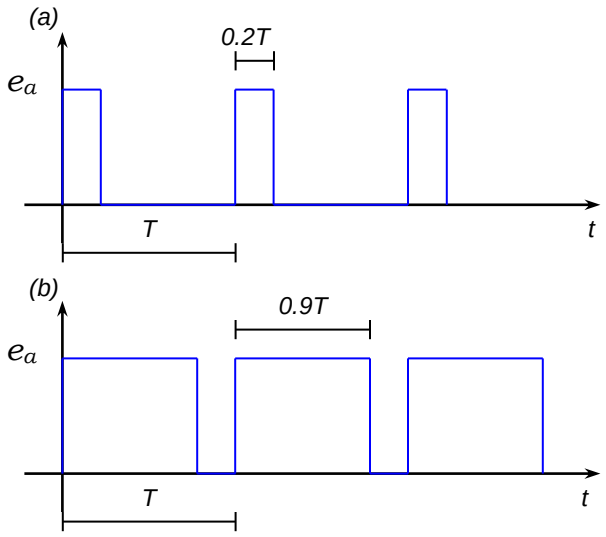


Figura 6.7. (a) Tren de pulsos con ciclo de trabajo del 20%. (b) Tren de pulsos con ciclo de trabajo del 90%.

Las variaciones del ciclo de trabajo se traducen en variaciones del voltaje promedio aplicado a la armadura del motor. De esta manera se llega a un control con modulación por ancho de pulso PWM, donde una cantidad variable de potencia es transferida a la carga.

El voltaje promedio entregado por un PWM esta dado por el área bajo la gráfica de la señal generada dividida entre su periodo. Esto es:

$$V_{prom} = \frac{V_{fuente} * CT * T}{T} = V_{fuente} * CT \qquad (10)$$

Donde CT es el ciclo de trabajo, expresado como un numero entre cero y uno (0% -100%). Aunque la expresión para el voltaje promedio es independiente del periodo de la señal, este no puede elegirse abiertamente. Este debe ser lo suficientemente pequeño para que el motor no pueda girar y detenerse en un periodo de tiempo debido a su inercia. Bajo esta condición, el motor tendría un comportamiento similar al que se obtendría al aplicar una fuente DC de valor variable demostrando que un tren de pulsos de alta frecuencia es visto por el motor como una fuente DC variable.

El tren de pulsos aplicado a la armadura del motor se logra mediante la conexión y desconexión de la fuente DC como se ve en la figura 6.6. Dicha conmutación se realiza a una frecuencia determinada de 2 KHz mediante el uso de un mosfet disparado por un control, de tal forma que se genera la señal PWM adecuada para controlar al motor. Debido a que el control es un proceso software, la frecuencia de la señal PWM generada depende del computador en que se ejecute.

Dado que el motor se comporta como una carga altamente inductiva, se ha conectado el diodo *D1,* figura 6.6, para que sirva de trayectoria a la corriente de carga cuando el mosfet pasa de estado de conducción al de no conducción. Con esto se logra proteger el circuito contra voltajes inversos provenientes del motor.

La descripción del circuito electrónico que genera la señal PWM se muestra en el anexo C "Descripción Hardware del Sistema".

6.1.3 Especificación del sistema de control digital para el motor de corriente continua.

El sistema encargado de controlar la velocidad del motor es implementado con un computador. Debido a esto, un algoritmo de control discreto es necesario para la realización de esta tarea.

En la figura 6.8 se muestra el diagrama de control digital del motor. La velocidad del motor se obtiene contando los pulsos entregados por el sensor óptico usando el puerto paralelo del computador.

El set point es impuesto por el computador remoto encargado de asociarlo según el comando que haya reconocido. Un aspecto importante a considerar es la conversión realizada por las funciones de transferencia que incluyen términos constantes. La conversión del rango de 0 – 255 al rango de 0 – 4000 RPM que realiza la función de transferencia que involucra el término 15.625 esta relacionado a que el computador que determina el set point lo hace usando los ocho bits bidireccionales del puerto paralelo. La función de transferencia que involucra el término 37.73, realiza la conversión del rango 0 – 106 de los pulsos entregados por el sensor óptico al rango 0 – 4000 RPM.

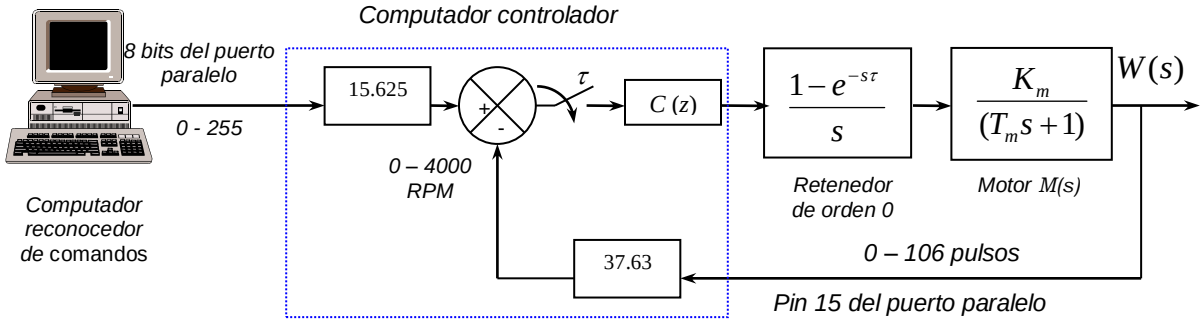


Figura 6.8. Diagrama de control digital de velocidad del motor.

6.1.3.1 Obtención del Modelo Discreto del Motor de Corriente Continua.

Como el motor es un proceso de naturaleza continua, es necesario obtener mediante la transformada Z un modelo discreto que permita la operación del controlador PI digital. En la figura 6.8 se observa el sistema en lazo cerrado usando el retenedor de orden cero.

A continuación se muestra el proceso matemático para obtener el modelo discreto del motor a partir de su modelo continuo usando el retenedor de orden cero.

$$M(z) = Z\left[\frac{1-e^{-s\tau}}{s}M(s)\right] = Z\left[\frac{1-e^{s\tau}}{s}\frac{K_m}{T_m s+1}\right] = \frac{z-1}{z}K_m Z\left[\frac{1/T_m}{s\left(s+1/T_m\right)}\right]$$

$$M(z) = \frac{z-1}{z}K_m Z\left[\frac{1}{s}+\frac{-1}{s+1/T_m}\right] = \frac{z-1}{z}K_m\left[\frac{z}{z-1}-\frac{z}{z-e^{-\tau/T_m}}\right] = \frac{z-1}{z}K_m\left[\frac{z(z-e^{-\tau/T_m})-z(z-1)}{(z-1)(z-e^{-\tau/T_m})}\right]$$

$$M(z) = K_m\frac{(1-e^{-\tau/T_m})}{(z-e^{-\tau/T_m})} \qquad (11)$$

La ecuación (11) corresponde al modelo discreto del motor de corriente continua usando el retenedor de orden 0, en donde T_m corresponde a la constante de tiempo del motor y τ al tiempo de muestreo del sistema.

6.1.3.2 Controlador Proporcional Integral PI Digital.

Considerando Integración trapezoidal, el controlador PI digital esta definido por:

$$C(z) = K_p\left[1+K_i\frac{T_i}{2}\left(\frac{z+1}{z-1}\right)\right] \qquad (12)$$

Donde

K_p = Constante proporcional del controlador.

K_i = Constante integral del controlador.

τ = Es el tiempo de integración, que es el mismo tiempo de muestreo.

Consideremos la función $\varepsilon(t)$ que describe la variación del error calculado por el sistema que controla al motor y calculemos el área bajo la curva que describe dicha función usando segmentos de área trapezoidales como se ve en la figura 6.9.

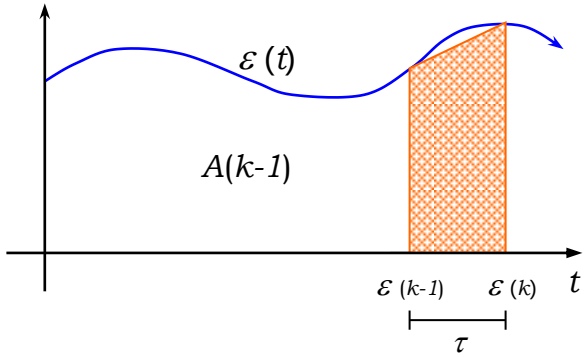


Figura 6.9. Calculo del área considerando trapecios.

El área bajo la curva de la función $\varepsilon(t)$ esta dada por:

$$A(K) = A(k-1) + \frac{\tau}{2}\left[\varepsilon(k-1) + \varepsilon(k)\right] \qquad (13)$$

Donde:

$A(k)$ = Área bajo la curva de la función $\varepsilon(t)$.

$A(k\text{-}1)$ = Área bajo la curva de la función $\varepsilon(t)$ hasta el instante $k\text{-}1$.

$\varepsilon(k\text{-}1)$ = Valor de la función en el instante $k\text{-}1$.

$\varepsilon(k)$ = Valor de la función en el instante k.

Teniendo en cuenta que k representa la variación discreta en el tiempo y usando la propiedad $Z[x(t-nT)] = Z^{-n}X(z)$, podemos convertir la ecuación (13) al dominio de Z:

$$A(z) = \frac{A(z)}{z} + \frac{\tau}{2}\left[\frac{e(z)}{z} + e(z)\right]$$

Y de esta ecuación obtenemos:

$$A(z)\left[1 - z^{-1}\right] = \frac{\tau}{2}\varepsilon(z)\left[z^{-1} + 1\right]$$

$$\frac{A(z)}{\varepsilon(z)} = \frac{T_i}{2}\left[\frac{z+1}{z-1}\right] \qquad (14)$$

La ecuación (14) corresponde a la parte integral de la ecuación (12), demostrando que en efecto la ecuación (13)describe el controlador PI digital cuando se considera integración trapezoidal.

La ecuación (13) es utilizada como formula de cálculo en el programa de computador que controla la velocidad del motor para obtener el valor de la integral de error.

6.1.3.3 Especificaciones de Desempeño.

En esta sección se realiza el cálculo de las constantes K_p y K_i del controlador PI digital descrito por la ecuación (12) por medio del método del lugar de las raíces a partir del modelo discreto del motor.

El sistema compensado debe tener un factor de amortiguamiento ξ de 0.707 y debido a que el motor se aproxima a un sistema de primer orden, el tiempo de establecimiento T_s es 5*T_m = 0.85 seg. Considerando T_m = 170 mseg, el tiempo de muestreo τ es de 40 mseg, cumpliéndose $\frac{T_m}{10} < \tau < \frac{T_m}{4}$.

La función de transferencia discretizada del motor en lazo abierto usando el retenedor de orden cero es $M(z) = \frac{62.898}{z - 0.790}$. Esta función indica que la planta tiene un polo en $z = 0.790$.

El controlador PI digital descrito por la ecuación (12) es objeto de un sencillo proceso matemático para obtener de este una representación equivalente más favorable:

$$C(z) = K_p\left[1 + K_i\frac{T_i}{2}\left(\frac{z+1}{z-1}\right)\right] = K_p\left[\frac{2(z-1) + K_iT_i(z+1)}{2(z-1)}\right] = K_p\left[\frac{z(2+K_iT_i) + (K_iT_i - 2)}{2(z-1)}\right]$$

$$C(z) = \frac{K_p}{2}\left[\frac{(2+K_iT_i)\left[z + \frac{K_iT_i - 2}{K_iT_i + 2}\right]}{(z-1)}\right] = \frac{K_p(2+K_iT_i)}{2}\left[\frac{z+C}{z-1}\right]$$

Finalmente se obtiene:

$$C(z) = K'\left[\frac{z+C}{z-1}\right] \qquad (15)$$

La ecuación (15) indica que el controlador tiene un polo en *Z=1* y un cero C en donde:

$$C = \frac{K_iT_i - 2}{K_iT_i + 2} \qquad (16)\ \text{define el cero del controlador.}$$

$$K' = \frac{K_p(2+K_iT_i)}{2} \qquad (17)\ \text{define la ganancia equivalente del controlador.}$$

Para las condiciones de desempeño se determina la ubicación de los polos deseados del proceso:

$$S_1 = -\xi\omega_n + j\omega_n\sqrt{1-\xi^2} = -0.707 * 6.656 + j6.656\sqrt{1-0.707^2} = -4.706 + j4.707$$

$$S_2 = -\xi\omega_n - j\omega_n\sqrt{1-\xi^2} = -0.707 * 6.656 - j6.656\sqrt{1-0.707^2} = -4.706 - j4.707$$

Donde $\omega_n = \frac{4}{T_s * \xi} = \frac{4}{0.85 * 0.707} = 6.656 \frac{rad}{seg}$.

Mediante la transformación $z = e^{\tau s}$ en donde τ es el periodo de muestreo, ubicamos los polos deseados en el plano Z:

$$Z_1 = e^{0.4(-4.706 + j4.707)} = e^{-1882} e^{j1.882} = 0.828 \angle 0.188 = 0.814 + j0.155$$
$$Z_2 = e^{0.4(-4.706 - j4.707)} = e^{-1882} e^{-j1.882} = 0.828 \angle -0.188 = 0.814 - j0.155$$

En la figura 6.10 se muestra la ubicación en el plano Z del polo de la planta $M(z)$, el polo de C(z) y los polos deseados $Z1$ y $Z2$.

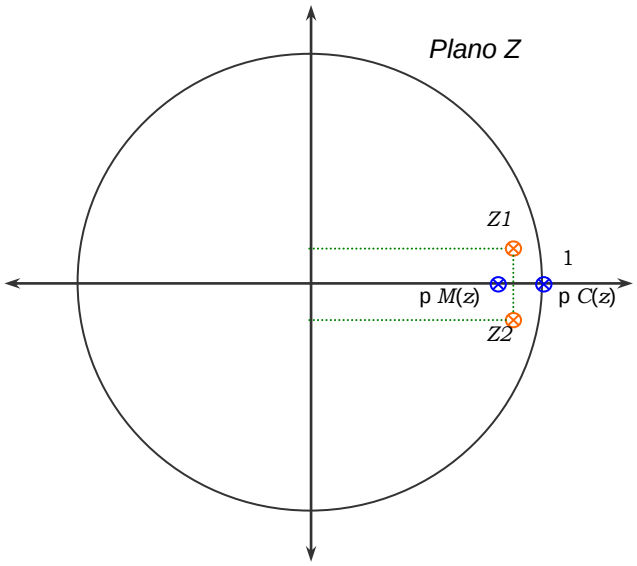


Figura 6.10. Ubicación en el plano Z de los polos del sistema.

A continuación se determina la condición de ángulo del sistema para hallar el ángulo del cero del controlador PI digital. La figura 6.11 ayuda a visualizar la posición del cero en cuestión.

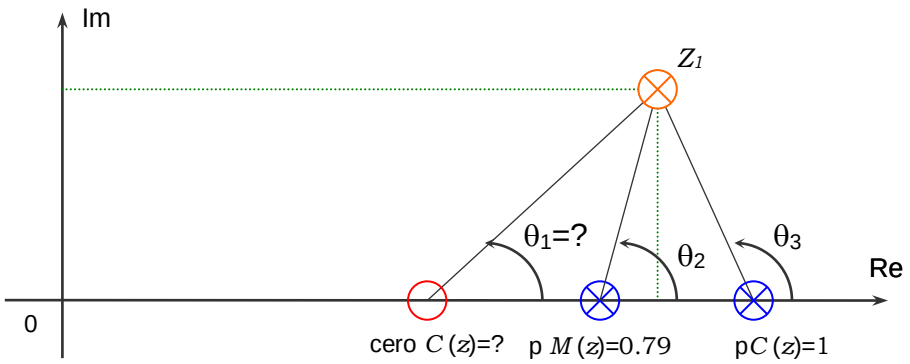


Figura 6.11. Posición del cero del controlador PI digital.

$$\theta_1 - [\theta_2 + \theta_3] = -180^0$$

$$\theta_1 - \left[arctan\left(\frac{0.155}{0.814 - 0.79} \right) + \left(180^0 - arctan\left(\frac{0.155}{1 - 0.814} \right) \right) \right] = -180^0$$

$$\theta_1 - [81.404^0 - 140.216^0] = -180^0$$

$$\theta_1 = 41.621^0$$

Después de calcular el ángulo del cero del controlador se procede a calcular su ubicación:

$$ceroC(z) = 0.814 - \left(\frac{0.155}{tan(41.621^0)} \right)$$

$$ceroC(z) = 0.639$$

Con la ubicación del cero del controlador y a partir de la ecuación (16) se obtiene la constante de integración K_i , en donde el valor de τ es fijado a 40 mseg. después de ser calculado empíricamente:

$$K_i = \frac{2C + 2}{T_i - CT_i} = \frac{2 * (-0.639) + 2}{40E - 3 - (-0.639) * 40E - 3} = 11.013$$

El sistema compensado resultante es $M(z)C(z) = \frac{62.898K´(z-0.639)}{(z+0.790)(z-1)}$. Se halla ahora la ganancia equivalente del proceso $K´$ para la ubicación de los polos deseados Z_1 y Z_2.

$$K´ = \left| \frac{(z+0.790)(z-1)}{62.898(z-0.639)} \right|_{Z_1 = 0.814 + j0.155} = |0.0065 - 0.02507i| = 0.0265$$

Una vez hallada la ganancia equivalente del proceso $K´$ y con el valor de K_i calculado anteriormente, se obtiene finalmente la constante proporcional K_p a partir de la ecuación (17):

$$K_p = \frac{2K´}{2 + K_i T_i} = \frac{2 * 0.0265}{2 + 11.013 * 40E - 3} = 0.0217$$

De esta forma, el controlador PI digital descrito por la ecuación (12) queda definido en su totalidad:

$$C(z) = 0.0217\left[1 + 11.013 * \frac{0.1}{2}\left(\frac{z+1}{z-1}\right)\right]$$

6.1.3.4 Especificación del Software de Control digital.

Los procesos realizados por el computador controlador son básicamente cuatro: La obtención del set point proveniente del computador reconocedor de comandos, cálculo de la variable controlada, generación de la señal PWM y la lectura de la velocidad de giro del motor. Dado que la generación de la señal PWM debe ser un proceso ininterrumpido para garantizar funcionamiento uniforme del motor, su generación integra la lectura de la velocidad de giro. Con esto se logra que el tiempo

empleado en el proceso de lectura no afecte el desempeño de la señal PWM. En la figura 6.12 se muestra la forma como se ejecutan los procesos que realizan el control de velocidad.

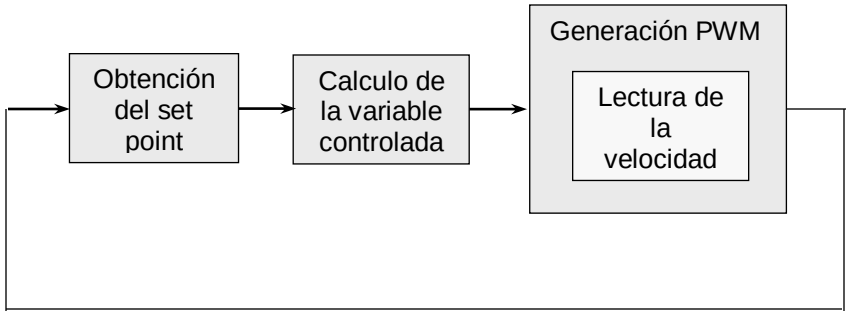


Figura 6.12. Procesos software del controlador PI digital.

A continuación se muestra el código en Lenguaje C++ que implementa el control de velocidad para el motor de corriente continua en cuestión.

Declaración de variables:

```
#define TOPE                    255.0  // Determina un periodo de la
                                       //señal PWM.

/*La Constante proporcional y la constante de integración son calculadas
usando el método del lugar de las raíces para definir el controlador
PI.*/

#define Constante_proporcional  0.0217 // Es la constante proporcional
                                       //del controlador.
#define Constante_integracion   11.013 // Es la constante integral del
                                       //controlador.
#define Tiempo_muestreo         40E-3  // 40 milisegundos corresponde a
                                       //Tm.

struct      _timeb timebufferinicial; // Estructura en donde se almacena
                                      //el tiempo inicial.
struct      _timeb timebufferfinal;   // Estructura donde se almacena el
                                      //tiempo final.
double Error;  // Es el error calculado en el proceso de control.
int Set_Point; // Recibe el set point del Pc reconocedor de comandos.
```

```
double Resultado; // Es el valor entregado después de realizar el cálculo
                  //PI.

double Voltaje; // Voltaje define la modulación de la señal PWM. Por
                // ejemplo, si Voltaje = 5.0, en el pin 16 del puerto
                // paralelo en la dirección H37A del Pc controlador se
                // obtiene una señal PWM con un ciclo de trabajo del 100%
                // y de 5 voltios de amplitud. Si Voltaje = 2.5, se
                // obtiene por este pin una señal PWM con un ciclo de
                // trabajo del 50% y 5 voltios de amplitud, pero con un
                // voltaje promedio de 2.5 voltios.


double  K1 = 800; // Constante que convierte del rango de 0 - 4000 RPM de
                  // la variable Resultado al rango de 0 - 5 voltios,
                  // para obtener el valor de Voltaje.

double  K2 = 37.74; // Constante que convierte el rango de los pulsos
                    // entregados por el sensor óptico (de 0 – 106
                    //pulsos) al rango de 0 - 4000 RPM.

double  K = 15.625; // Convierte el rango del set point recibido del Pc
                    // reconocedor de comandos (de 0 - 255) al rango
                    // 0 - 4000 RPM.

int  m_Tinicial, m_Tfinal;
int  Umbral         = 0; // Define el ciclo de trabajo.
int  Ciclos_prueba  = 0; // Cuenta los ciclos a los que se calcula su
                         // duración.
int  ciclos         = 0; // Cada número de ciclos se obtiene la velocidad
                         // del motor.
int  nciclos        = 0; // Cuenta los ciclos de la señal PWM.
int  Cuenta_Estados = 0; // Cuenta los pulsos entregados por el sensor
                         // óptico.
int  velocidad      = 0; // Velocidad del motor en RPM.
int  Dato_inicial   = 0; // Corresponde al dato que se lee inicialmente
                         // del Pc reconocedor de comandos.
bool Detener        = 0; // La variable Detener controla el encendido del
                         // motor. Si es igual a cero, el motor se
                         //prende.
                         // Si es igual a uno el motor se apaga.
```

Cuerpo de la función:

```
UINT motor(LPVOID pParam) // Esta es la función que controla al motor.
{

static double Error_anterior = 0, Voltaje_Anterior = 0, Integral_anterior
= 0;
```

```
double Integral; //Integral define la parte integrativa del controlador.

velocidad = 0;
nciclos   = 0;


  do
  {


    _outp(0x37A,32);   // Configura al puerto paralelo Bidireccional para
                       // recibir del Pc reconocedor de comandos el set
                       // point en el rango de 0 - 255.

    Set_Point= (int)(K*(_inp(0x378))); // Se obtiene el set point del Pc
                                       // reconocedor de comandos. La
                                       // constante K convierte al número
                                       // real de RPM a partir del rango
                                       // de 0 _ 255 recibido.


    Error     = (Set_Point - velocidad);
    Integral = Integral_anterior + Constante_integracion*0.5
               *(Tiempo_muestreo)*(Error + Error_anterior);
    Resultado = (Constante_proporcional * ( Error + Integral ));
    Voltaje   = (1/K1)*Resultado;

    //Actualización de variables.
      Voltaje_Anterior   = Voltaje;
      Error_anterior     = Error;
      Integral_anterior  = Integral;

      if (Voltaje > 5) Voltaje = 5;
    else if (Voltaje < 0) Voltaje = 0;

    //Conversión a ancho de pulso.
    Umbral = (int)(Voltaje * (TOPE / 5.0)); // Umbral define el ciclo de
                                            // trabajo de la señal PWM.

    //Generación de la señal PWM.
    register int Actual;
    for(int i=0; i <= TOPE; i++)
      {
            Actual = _inp(0x379); // Lee el estado de los pulsos
                                  //entregados por el sensor óptico.
            if (i <= Umbral) _outp(0x37A,0x4);
            else  _outp(0x37A,0x00);

            if (Actual != _inp(0x379)) // Determina si hay cambio de
                                       // estado en los pulsos entregados
                                       // por el sensor óptico.
```

```
////////////////////////////////////////////////////////////////////////
      Cuenta_Estados++; // Incrementa el contador de cambio de estados
                        // leídos del sensor óptico.
     if (nciclos == ciclos ) // Verifica que se cumpla el numero de
                             // ciclos que satisfacen Tm.
     {
       velocidad = (int)(Cuenta_Estados*K2); // Aquí se obtiene la
                                // velocidad del motor mediante número
                           // de pulsos entregados por el sensor óptico.
       Cuenta_Estados = 0; // Cuenta_Estados se hace igual a cero cuando
                           // ya se ha obtenido la velocidad del motor
                           // para iniciar nuevamente el proceso de
                           // obtención de la velocidad del motor en el
                           //próximo intervalo de 40 mseg.
       nciclos=0;
      }
////////////////////////////////////////////////////////////////////////

      }

      nciclos++;

  } while (Detener != 1);

_outp(0x37A,0x00); // Detiene el motor cuando finaliza la aplicación y
                   // además configura la dirección H378 del puerto
                   // paralelo en los pines del 2 al 9 como entradas.
return 0;
}
```

Para hacer que el tiempo de muestreo sea independiente de la rapidez del computador controlador, se obtiene la velocidad de giro del motor cada número de ciclos de la señal PWM cuya duración conjunta es de 40 mseg. Esto se logra mediante un sencillo procedimiento que se ejecuta antes de cualquier acción de control, el cual evalúa la duración de 2048 ciclos para obtener el periodo de cada uno . A continuación se muestra el código encargado de hacer esta labor.

```
// TODO: Add extra initialization here

// Se genera 2048 ciclos de señal PWM con ciclo de trabajo del 50% para
// calcular el número de ciclos cuya duración es el periodo de muestreo
// Tm.
    double tiempo,periodo;
    _ftime( &timebufferinicial );
    m_Tinicial=timebufferinicial.millitm; // Se toma el tiempo inicial de
                                          //los 2048 ciclos de señal PWM.
```

```
    Umbral = (int)(2.5 * (TOPE / 5.0));
         do
            {


              for(int i=0; i <= TOPE; i++)
              {

              if (i <= Umbral) _outp(0x37A,0x4);
               else  _outp(0x37A,0x00);

               }Ciclos_prueba++;

            }while(Ciclos_prueba<2048);

     _ftime( &timebufferfinal );
     m_Tfinal=timebufferfinal.millitm; // Se toma el tiempo final de los
                                       // 2048 ciclos de prueba.

     tiempo = m_Tfinal - m_Tinicial; // Se calcula la duración de 2048
                                     //Ciclos_prueba
       if(tiempo < 0) // Condición en caso de tener un tiempo negativo
                      // para volverlo al valor real mediante la fórmula
                      // de calculo de este if.
       {
            tiempo = (1000 -  m_Tinicial) +  m_Tfinal;
       }
     periodo=2048/tiempo; // Aquí se obtiene el periodo de los ciclos de
                          // prueba.
     ciclos = (int)(40/periodo); // Se obtiene el numero de ciclos cuya
                                 // duración es 40 mseg, es decir, cuya
                                 // duración es el tiempo de muestreo Tm.
Detener = 0; // El motor inicialmente se prende.

      return TRUE;  // return TRUE  unless you set the focus to a control
```

6.2 ACTIVACIÓN O DESACTIVACIÓN DE DISPOSITIVOS DOMÉSTICOS MEDIANTE COMANDOS DE VOZ

La activación o desactivación de dispositivos se realiza mediante la asociación del comando reconocido al dispositivo que el usuario desea activar o desactivar. En esta aplicación se ha concebido que el dispositivo es un artefacto de uso corriente en un hogar tal como una bombilla, un

sistema de sonido, un televisor; etc., siendo esta una aplicación que puede ser de mucha cotidianidad en cuanto al uso casero.

En la figura 6.13 se muestra el diagrama del sistema encargado de esta aplicación. El computador que reconoce los comandos orales en caso de detectar alguno asociado, invoca un procedimiento software que activa o desactiva cíclicamente el dispositivo a medida que se reconoce repetidamente el comando para tal fin.

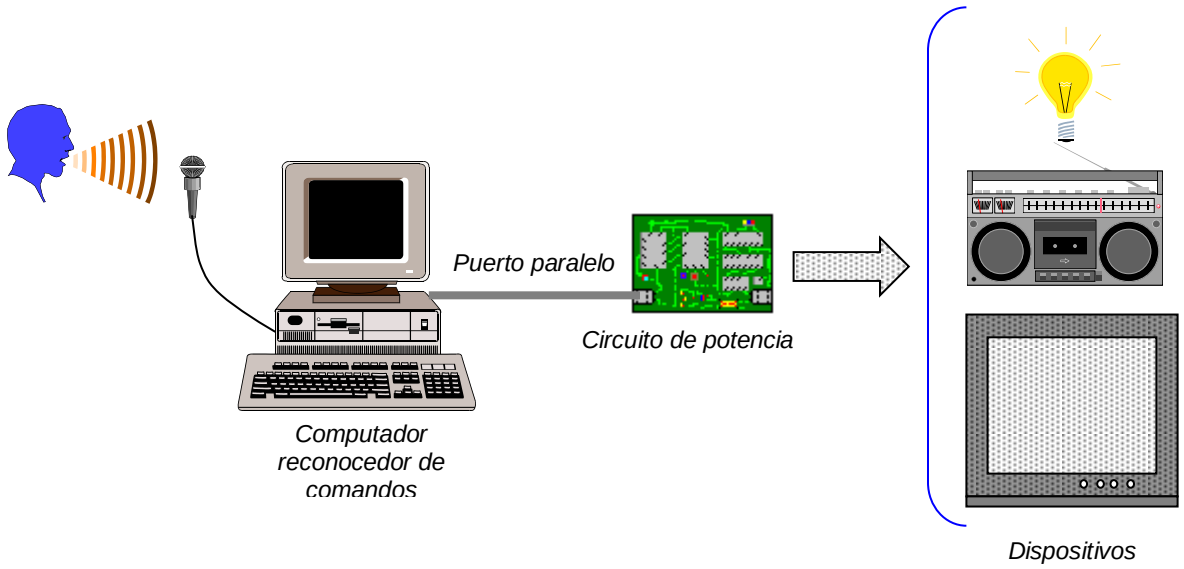


Figura 6.13. Activación o desactivación de dispositivos domésticos mediante comandos de voz.

El software mencionado hace uso del bit 2 en la dirección 37 AH del puerto paralelo para poder realizar su acción: a través de un circuito de potencia con un 1 lógico se activa el dispositivo, con un cero lógico lo desactiva. La descripción del circuito de potencia que suple la corriente para los dispositivos se muestra con detalle en el anexo C "Descripción Hardware del Sistema".

Capítulo Siete

Resultados, conclusiones y recomendaciones

Este capítulo presenta un sumario de los resultados del trabajo realizado. Inicialmente se analiza el desempeño del sistema neuronal diseñado y de los SOM′s que lo componen, como base del reconocimiento de los patrones de voz. Luego se procede entonces a realizar un análisis sobre los procesos realizados a la señal de voz, el control de velocidad del motor y las herramientas de programación usadas.

7.1 EL SISTEMA NEURONAL

La topología

Como se comentó en el capítulo cuatro, la red neuronal encargada del reconocimiento esta compuesta por 13 redes tipo SOM que forman el sistema neuronal de clasificación y de una red SOM que forma el sistema neuronal de decisión. La experiencia obtenida con el número de neuronas que integran cada SOM y su topología demuestra que la selección de estos parámetros influye en el desempeño del entrenamiento y por consiguiente, en el número de comandos que es capaz de reconocer el sistema.

El número de neuronas que integran los SOM tiene correspondencia directa al número de grupos de vectores a reconocer. La cantidad de neuronas debe ser suficiente para que cada grupo de vectores similares tengan una neurona ganadora cuando un vector del grupo es presentado. Esta es la razón por la que el número de neuronas en el SOM debe ser mayor al número de grupos de vectores a reconocer, pero esta relación no debe ser muy superior porque se tiene el problema de asociar varias neuronas ganadoras a un grupo de vectores tornando ambigua la respuesta del SOM.

Como es conocido, las neuronas de las redes SOM están dispuestas en puntos discretos en una matriz bidimensional que al ser cambiados de posición generan resultados de entrenamiento completamente distintos, que en algunos casos no eran muy favorables a la hora de reconocer los diferentes grupos de vectores y en otros casos si permitía un reconocimiento óptimo. Esta propiedad permite que con el cambio de la topología se llegue la escogencia de la mejor respuesta.

Cada SOM del sistema neuronal de clasificación y el SOM de decisión están encargados del reconocimiento de un número de grupos de vectores igual al número de comandos a reconocer (como se explico en el capítulo cuatro). Después de realizar muchas pruebas en la variación de la topología y el número de neuronas se optó por una configuración de 2 x 4 para cada SOM del sistema neuronal de clasificación y una configuración de 4 x 2 en el SOM de decisión, ya que con esta configuración se logra que cada grupo de vectores del espacio de entrada quede representado por una neurona particular optimizando el reconocimiento del sistema a seis comandos.

El entrenamiento uniforme

Existe una relación directa entre la longitud de los comandos a reconocer y el entrenamiento de cada uno de los trece SOM′s del sistema neuronal de clasificación. Como se explicó en el capitulo cuatro, después del procesamiento espectral y del algoritmo de compresión se generan vectores de dimensión dieciséis por cada segmento que conforma el comando procesado que se asocian a cada SOM particular. En el caso de tener palabras cortas, el número de vectores generados no alcanza a corresponder a todos los SOM′s y en el caso de palabras largas, solo existe correspondencia para los primeros trece vectores obtenidos.

Cuando el entrenamiento es realizado en forma uniforme, es decir, con palabras de longitud similar que permiten que a cada uno de los trece SOM′s o a un número parcial de estos le sean enseñados vectores en forma regular, se obtiene una buena clasificación por parte del sistema neuronal.

El sistema tiene un funcionamiento aceptable cuando la longitud de las palabras no es regular, sin llegar al punto de considerar una diferencia extrema de longitudes entre estas palabras. En el segundo caso se tiene que las últimas redes van a tener pocos ejemplos de entrenamiento (se tiene el caso de un número muy superior de neuronas para identificar pocos grupos de entrada) que en el momento de simular hace que el sistema neuronal de clasificación entregue al SOM de decisión una representación errónea de las clases que identifican a cada comando con la consecuencia en la disminución de efectividad en el reconocimiento.

El sistema fue probado entrenándolo con los comandos *Luz, Alto, Medio, Bajo, Casa* y *Control.* Dado a que la longitud de éstos es aproximadamente igual (aproximadamente de 250 mseg, excepto para el caso de *Luz* que es el más corto) se obtiene un reconocimiento muy aceptable.

La tasa de aprendizaje y la distancia de vecindad en el entrenamiento

Como conocemos del capitulo tres, en los SOM′s el entrenamiento tiene lugar en dos fases en donde la tasa de aprendizaje y la distancia de vecindad son modificados a medida que se realiza este proceso.

Una vez superado el problema del entrenamiento no uniforme fueron usados los valores que Matlab proporciona por defecto a los parámetros involucrados en las dos fases de entrenamiento de los SOM′s (como se observa en la tabla 3.1 del capítulo tres). El resultado obtenido en el entrenamiento fue satisfactorio debido a que las neuronas de los SOM′s de los sistemas neuronales de decisión y clasificación presentan activaciones por cada grupo de vectores a reconocer, reduciendo el problema de que un grupo particular fuera a activar a mas de una neurona en el SOM de decisión.

7.2 LA CAPTURA DE SONIDO

La captura es un aspecto muy representativo del sistema sin el cual hubiese sido imposible el funcionamiento de todo el proyecto ya que corresponde a la primera etapa de este. Por esto se tuvo especial cuidado a la hora de implementar la forma en que esta etapa se amoldara al resto del sistema.

Lo primero que se encuentra en esta primera fase es el micrófono. El dispositivo usado fue uno de tipo capacitivo Electret, de uso estándar, con el cual se obtuvo un desempeño aceptable a lo largo de las pruebas realizadas al sistema. No se procedió a usar uno de mejor calidad ya que esta clase de micrófonos está presente en la mayoría de los computadores que gozan de multimedia, y así aseguraremos que el sistema logre funcionar en casi todo lugar donde se ejecute. Con el paso del tiempo saldrán al mercado micrófonos de mejor calidad con los cuales se obtendrán mejores resultados en la captura de sonido, tarea para el cual está designado.

El sistema se divide en dos partes, entrenamiento y simulación. Para cada una, la captura de señales se trató de diferentes formas según el requerimiento por parte de los programadores, grabadora de sonidos de Windows y captura automática, respectivamente.

Ahora bien, como del entrenamiento de los comandos configurados por el usuario depende en gran medida el éxito del sistema, el proceso de captura se realizó con la ayuda de la grabadora de sonidos de Windows por varias razones. Una de las más elementales fue la facilidad que esta herramienta ofrece a la hora de poder reproducir el último comando introducido al igual que editarlo. Esto es esencial puesto que las muestras que se tomen en ese momento, son las que el sistema va a utilizar para obtener las matrices para el entrenamiento, y es por eso que el usuario dispone de estas ayudas de la grabadora para asegurar que los comandos que introduce se encuentren dentro de un rango aceptable para el usuario y reconocibles para el sistema.

Otra de las razones para usar esta grabadora fue la recursividad. Se aprovechó la existencia de esta herramienta que ofrece Windows para así no desperdiciar mucho tiempo al implementar en este sistema algo que ya existe en todo computador que tenga Windows instalado. Además, esto no corresponde con los objetivos planteados al iniciar con este proyecto.

En la etapa correspondiente a la simulación, existen varios aspectos que valen la pena resaltar. El más significativo fue la forma en que se implementó esta fase, mediante un proceso de captura automática, un desarrollo completamente distinto al presentado en la etapa de entrenamiento. Se implementó de esta forma, debido a que el sistema debe permanecer en un estado estacionario de "escucha", es decir, estar atento a cualquier pronunciación hecha por el usuario para capturar y procesar posteriormente dicha pronunciación.

La implementación de este método de captura, se realizó gracias al manejo de Api, consistentes en aplicaciones hechas por anteriores programadores las cuales contienen líneas de código usadas en diversos casos, que para nuestro propósito, consistió en la captura de señales mediante micrófono usando Buffers para el almacenamiento de las pronunciaciones obtenidas en el proceso de detección.

Con lo mencionado anteriormente, existen dos aspectos importantes en la captura, el micrófono y los Buffers. La parte pertinente al micrófono se refiere a su configuración, es decir, los parámetros que se manipulan para lograr un correcto funcionamiento. Entre los mas destacados se encuentran los bits por muestra y la frecuencia de muestreo. En cuanto a los buffers, los parámetros que se manejan son el número y tamaño de estos, aspectos que influyeron en el proceso de captura.

Trabajando con los bits por muestra, en la configuración de captura del micrófono, intervinieron dos valores, 8 bits y 16 bits. Tras diversas pruebas, operando con 16 bits se obtiene un mejor resultado de reconocimiento o simulación en comparación con 8 bits. Esto se debe a que en 16 bits, se deposita más información de la señal, logrando que el sistema pueda diferenciar con mayor facilidad un comando con otro.

Se invirtió mucho tiempo tratando de acoplar en forma aproximada los parámetros del buffer. El primero, el número de estos, debió ser grande para asegurar la captura completa de un comando. El segundo parámetro, su tamaño, donde se hace referencia al tamaño por celda, fue un parámetro difícil de establecer. No puede ser muy grande debido a que en el proceso de llenado utilizaría mucho tiempo, conllevando a un desfase entre el proceso de escritura y lectura del buffer, teniendo como resultado celdas incompletas que serán procesadas posteriormente. En cambio, si se escoge un tamaño demasiado pequeño, se aumentaría la carga computacional, llevando al sistema a trabajar en forma pesada. Es así, que para adquirir valores aceptables, se debe tener un poco de paciencia y desarrollar el mayor número de ensayos posibles.

7.3 EL PROCESAMIENTO DE LA SEÑAL DE VOZ

Las técnicas de procesamiento aplicadas a la señal de voz son procedimientos de relativa sencillez en comparación de otras técnicas mucho más elaboradas, pues muchas de las manipulaciones realizadas a la señal de voz fueron diseñadas por nosotros a partir del tratamiento general común que tienen todos los sistemas clásicos de procesamiento. A pesar de tener conocimiento sobre estos aspectos, la implementación específica de un sistema a otro presenta diferencias y además la información sobre estos desarrollos no es muy difundida.

El uso de estas técnicas, permitió tener resultados muy favorables. El mejoramiento de estas mismas podría optimizar el sistema, llevando a cumplir con éxito la tarea de las redes neuronales artificiales en el proceso de entrenamiento y simulación, presentándose estas redes como una buena alternativa para abordar esta clase de proyectos.

Dentro de los procesos realizados a la señal de voz, el filtrado y la delimitación son definitivos en la obtención de un buen reconocimiento de los comandos. El filtrado es importante porque elimina el ruido introducido en la etapa de captura y elimina componentes frecuenciales que no contribuyen en forma importante a la señal de voz preparándola para que la delimitación sea en forma exitosa y garantice que los patrones a reconocer obtenidos sean una correcta aproximación de los comandos a los que corresponden.

7.4 EL CONTROL DE VELOCIDAD DEL MOTOR

En esta aplicación, la forma de realizar el algoritmo computacional encargado del control de velocidad del motor consiste en integrar la generación de la señal PWM que alimenta al motor con los proceso que de lectura de la velocidad del motor. La razón de esto, obedece a que si la obtención del la velocidad del motor y la generación de la señal PWM son realizados en forma independiente, el retardo generado por el proceso de lectura y la duración del ciclo de trabajo se hacen comparables, provocando que la señal PWM pierda exactitud.

Antes de realizar el algoritmo de control integrado, tratamos de hacer que los procesos que obtienen el ciclo de trabajo y la generación de la señal PWM fueran ejecutados en forma independiente para tener mayor portabilidad en el código generado. Para esto se necesito trabajar con la programación multitarea, pero lastimosamente los resultados no fueron los mejores. La cuestión es que a pesar de que la ejecución de los procesos se hace en forma compartida, el intervalo de tiempo tomado por cada uno genera el retardo comentado. En cambio, la programación multitarea si tuvo éxito en la implementación del ciclo `do – while(Detener != 1)` que cierra el ciclo en el algoritmo de control integrado, evitando que este ciclo infinito se adueñe de la aplicación de control Dado a que la generación de la señal PWM debe ser un

proceso ininterrumpido, otra solución puede ser generarla mediante un hardware externo. Este método implica el uso ocho bits adicionales del puerto paralelo para representar el valor del ciclo de trabajo calculado y un conversor digital analógico para obtener una señal adecuada que varíe los parámetros de funcionamiento del generador de PWM. Dado que este método incrementa considerablemente el hardware, nos pareció más práctico usar el algoritmo de control integrado debido a que por un solo bit del puerto paralelo y sin ningún tipo de hardware externo se obtiene la señal PWM calculada por software.

Como habíamos comentado con anterioridad, el proceso que obtiene la velocidad de giro del motor es el que presenta mayor retardo y por consiguiente afecta sensiblemente la generación de la señal PWM. La razón del retardo se debe a que para realizar este proceso, se debe leer a través de un bit del puerto paralelo la cuenta de los pulsos generados por el sensor óptico en un intervalo de tiempo, y como las operaciones de lectura del puerto paralelo son relativamente lentas en Visual C, se debe tomar un intervalo de tiempo de 40 mseg. para obtener una buena lectura de velocidad. Precisamente por esta razón, el control de velocidad no funciona para valores altos del set point, porque después de realizar pruebas se comprobó que para valores superiores a 3000 R.P.M. no se tiene la suficiente velocidad para contar los pulsos entregados por el sensor óptico.

Una forma de eliminar el retardo en el proceso de obtención de la velocidad del motor es mediante el uso de un hardware externo encargado de esta labor, con un conversor de frecuencia a voltaje conectado a un conversor analógico digital para obtener un dato de ocho bits que represente la frecuencia de los pulsos entregados por el sensor óptico. El problema es que además de la incorporación de nuevos dispositivos y de requerir ocho bits adicionales del puerto paralelo para leer el dato obtenido, nos parece más conveniente realizar el proceso mediante software porque tan solo se necesita un bit del puerto, el tiempo de leer la velocidad se logra reducir a la cuarta

parte del tiempo de establecimiento del motor y porque el problema del retardo se soluciona en gran parte por la incorporación del algoritmo de control integrado.

Una vez colocado a punto el control PI de velocidad del motor se procedió a incorporarlo al sistema principal confirmando que la ejecución del control de velocidad del motor, especialmente el proceso encargado de la generación de la señal PWM fue afectado fuertemente de la forma ya conocida por la aplicación encargada del reconocimiento de comandos. Se trato solucionar este problema usando la multitarea, pero como en caso anterior este método de solución no tuvo resultados satisfactorios porque la ejecución de las dos aplicaciones se realizan una tras otra cuando a cada una se le asigna un intervalo de participación. El problema no es este, pues consideramos que es cuestión de velocidad porque después de realizar pruebas en un PC Pentium III de 500 Mhz se obtuvo cierta mejoría, pero no suficiente como para garantizar el correcto funcionamiento del motor. Después de enfrentar todas estas experiencias, nos pareció más práctico implementar un PC remoto encargado de controlar el motor, porque con esto garantizábamos un funcionamiento óptimo y además, porque en un ambiente industrial real se tienen arquitecturas distribuidas en donde las acciones particulares de control son realizadas por dispositivos como los PLC.

7.5 HERRAMIENTAS DE PROGRAMACIÓN

Este apartado hace mención a las herramientas computacionales utilizadas en el proyecto, muchas de las cuales son de uso común entre la comunidad universitaria y personas que estén familiarizadas con el uso de los lenguaje de programación. Estas herramientas son, Visual C++ y MatLab, apoyadas con algoritmos que ayudan a disponer mejor sus funciones.

Debido a que el sistema esta caracterizado por ser de uso general, el lenguaje de programación Visual C++, utilizado para el desarrollo del Software del sistema es una de las herramientas con una sintaxis sumamente compacta y de alta portabilidad, ya que muchas de sus operaciones se realizan por medio de llamadas a funciones contenidas en librerías externas. Además, esta herramienta goza de ayudas para orientar al programador a través de toda la etapa de implementación para ofrecer al usuario final un producto que satisfaga todas las expectativas requeridas desde un principio.

Otra de las herramientas, MatLab versión 5.3, fue importante proporcionando diversidad de funciones empleadas en el procesamiento del sistema. Además es el encargado de efectuar la mayoría de operaciones con señales tales como obtención, filtrado, normalización, delimitación, y el trabajo con las redes neuronales.

En un principio se dio la idea de desarrollar el proyecto íntegramente en Visual C++, tanto la parte de calculo: procesamiento de señales, entrenamiento y reconocimiento, como la parte visual que interactúa con el usuario. Pero al contar con MatLab, que nos ofrece exactitud y rapidez con estas operaciones, se opto por delegar esta tarea a esta poderosa herramienta, y a Visual C++ todo lo relacionado a la interfaz gráfica. Teniendo así, dos lenguajes implementados en un solo sistema interactuando en forma dinámica y eficaz.

Para no desechar el trabajo realizado en los primeros meses, ya que en un principio se realizo todo el trabajo en MatLab, se busco la forma de “unir” estas dos aplicaciones, tanto MatLab como Visual C++ pudieran comunicarse e intercambiar datos que pudieran ser de gran utilidad. MatLab proporcionando funciones que le permiten acceder a otras aplicaciones de Visual C y para que otras aplicaciones de Visual C accedan a MatLab en un amplio rango de contextos. Se llego a la conclusión de implementar el Intercambio Dinámico de Datos (DDE), el cual habilita a las

aplicaciones de Microsoft Windows comunicarse entre si, donde la aplicación que inicia la conversación es llamada “cliente” y la aplicación que responde a la solicitud de la aplicación “cliente” es llamado “servidor”. Esta comunicación se realiza en forma rápida, no representando retardos significativos.

7.6 RECOMENDACIONES

Una primera recomendación, retomando la etapa de captura de sonido, para estudiantes o personas interesadas en la continuidad de este trabajo, sería la implementación de una grabadora de sonidos. Si bien se comentó, que no era necesario hacerla debido a que se podía utilizar desde Windows, resultaría interesante crear, por parte del programador, una de estas grabadoras para el sistema, logrando un poco menos de dependencia hacia Windows, dado el caso que algún usuario no disponga de una grabadora de sonidos instalada en su PC.

Haciendo referencia a la tarjeta de sonido con la cual se va a trabajar, la cual es la responsable de la distribución de los bits por muestra, es importante configurarla de una forma correcta para que permita trabajar con el valor requerido, 16 bits por muestra . Pero al igual que se recomiende el valor especificado anteriormente, en un futuro se pueden explorar otras alternativas para las cuales el sistema podría funcionar en una forma más satisfactoria.

Una de las sugerencias más significativas que se pueden exponer es la exploración al máximo de las funciones que nos suministra Visual C. Esta herramienta es muy completa, la cual permite crear diversas aplicaciones, y es importante que desde un inicio se deba documentar de todas estas funciones que se nos ofrecen para mas adelante no perder demasiado tiempo intentando encontrar alguna función que nos ayude a solventar alguna necesidad del programa.

Un aspecto importante que se debe recalcar, es la debida escogencia desde un comienzo, de todas las herramientas a utilizar. No se debe presentar mucha pérdida de tiempo indagando otras alternativas de desarrollo que al final lo único que lograron fue desviar por momentos los verdaderos caminos que desde un principio se plantearon en los objetivos esbozados en el anteproyecto. Así, al iniciar cualquier proyecto, los integrantes de grupo deben ponerse de acuerdo en los elementos a utilizar, informándose previamente de todas las alternativas que les ofrecen la universidad.

Por otro lado, uno de los recursos a tener en cuenta en el desempeño del sistema, tanto del entrenamiento como para el reconocimiento, es el procesador. De este depende el buen desempeño del sistema. Entre más rápido sea, el sistema responderá de la misma manera, disminuyendo el tiempo de respuesta del sistema y por ende, forjando a que todos los procesos presentes corroboren con lo requerido por el usuario. En cambio, si el procesador es lento, se presentarán excesivos retardos en el momento de concebir toda clase de cálculos, empobreciendo el sistema y no permitiendo asegurar que tanto los procesos de entrenamiento como los de simulación, partes en las cuales está dividido el sistema, hagan su respectivo trabajo.

Respecto al control de velocidad del motor de corriente continua, ya hemos considerado el problema del retardo generado por el proceso que obtiene la velocidad de giro en el algoritmo de control PI. Pero debemos tener en cuenta que el retardo es relativo pues tiene relación directa con la velocidad del proceso. En nuestro caso particular inicialmente se consumían 250 mseg. en la obtención del dato de velocidad, siendo un periodo de tiempo muy grande para un proceso veloz como este, en donde el tiempo de establecimiento del motor es de 170 mseg. Afortunadamente se consiguió reducir este tiempo de 250 a 40 mseg. (aproximadamente la cuarta parte del tiempo de establecimiento), observando una mejora considerable en el funcionamiento del motor. Esta reducción se obtuvo mediante la incorporación de un algoritmo que hace independiente la

obtención de la velocidad de giro del motor de la tecnología del procesador en el que este se ejecute, mediante la obtención del número de ciclos de la señal PWM que satisfacen la duración de 40 mseg, para obtener cada número de ciclos calculados la velocidad de giro. Para procesos de control de temperatura o control de nivel que son más lentos, la obtención de la variable a controlar en 250 mseg. puede resultar en cambio muy favorable.

Una forma de hacer menos crítico el problema de la obtención de la velocidad es no usar motores muy pequeños (de esos que vienen en los juguetes que funcionan con pilas) porque su poca inercia hace que el tiempo de establecimiento sea pequeño exigiendo más rapidez del algoritmo de control. El uso de motores pequeños también genera dificultades cuando su tiempo de establecimiento es calculado experimentalmente, porque se requiere también de mayor velocidad en los instrumentos de medida, que en nuestro caso es el osciloscopio. Por estas razones, no recomendamos trabajar con motores de corriente continua de estas características.

Una sección del capitulo seis esta dedicada a determinar experimentalmente el valor del tiempo de establecimiento del motor usado en la aplicación, debido a que este método resulto ser satisfactorio. Inicialmente se quiso hallar analíticamente mediante el uso de las ecuaciones que definen las constantes de ganancia y de tiempo, obteniendo malos resultados debido a los errores cometidos en las mediciones de las características físicas del motor y al generado cuando se prescinde del coeficiente de fricción viscosa, ya que este no es fácil de medir.

Para terminar, hay varias consideraciones que se deben tener en cuenta con la clasificación de los grupos de vectores que representan las palabras para el reconocimiento de las mismas. En este dominio de aplicación, la clasificación de los vectores se basa en la información contenida en los vectores en sí y en la relación temporal entre estos. Es decir, El vector anterior, o el posterior, afecta la clasificación del vector de entrada en curso. Pero teniendo en cuenta que los SOM′s no

están condicionados para clasificar una secuencia particular de vectores de entrada ¿por qué los usamos para el reconocimiento de los comandos?. Precisamente por su particularidad de identificar los grupos de entrada y no su secuencia, se dispuso que cada uno de los trece SOM′s del sistema neuronal de clasificación se hicieran cargo de un segmento particular de todos los comandos a reconocer, porque simplemente la relación de dependencia entre los comandos no existe. La forma como se consigue resolver los aspectos temporales consiste en asociar en forma *ordenada* cada uno de los segmentos del comando a un SOM particular; de esta forma, cuando una secuencia de vectores que represente un comando es presentada al sistema neuronal de clasificación, se tiene en cuenta la ubicación de estos para el reconocimiento.

Lo importante es tener en cuenta la clasificación espacio – temporal de los grupos de vectores para realizarla con las redes neuronales. Para el caso de los SOM′s, nosotros proponemos este método, pero otras alternativas deben existir de solución cuando se trabaja con estas redes. Recomendamos explorar otros caminos para el reconocimiento de comandos, en especial el uso de la red de avalancha desarrollada por Grossberg, que esta diseñada para estos propósitos; habría que implementar una red por cada comando a reconocer y unir los vectores uno tras otro para presentárselos, o calcular el espectro de Fourier sin segmentación para obtener una sola secuencia del comando. La dificultad radica en que esta red no esta implementada en Matlab (al menos en la versión 5.3) y surge como tarea adicional su implementación en algún lenguaje de programación como Visual C++.

El trabajo ha brindado resultados satisfactorios considerando sobre todo la poca experiencia que se tiene en la facultad sobre los sistemas de reconocimiento de voz y en especial sobre el procesamiento de señales para este tipo de aplicaciones. Pese a estas limitaciones, el objetivo inicial de reconocer palabras de una sílaba se superó pues se logro el reconocimiento de palabras

más largas. Dejamos esta base para que personas interesadas en el tema continúen y perfeccionen este trabajo.

BIBLIOGRAFíA

B. KOSKO. "NEURAL NETWORKS FOR SIGNAL PROCESSING". Ed. Prentice Hall, 1992.

FREEMAN, James y KAPURA, David. Redes Neuronales, Algoritmos, aplicaciones y técnicas de programación. Addison Wesley, 1993.

MEMORIAS DEL PRIMER CONGRESO COLOMBIANO DE NEUROCOMPUTACION. Universidad Nacional, 1996.

D. ELLIOT. "HANDBOOK OF SIGNAL PROCESSING ENGINEERING APPLICATIONS". Ed. Rockwell International Corporation, 1987.

BRUCE CARLSON A. "SISTEMAS DE COMUNICACIÓN". Ed. McGraw Hill, 1997.

OGATA Katsuhiko. "INGENIERÍA DE CONTROL MODERNA". Prentice Hall Hispanoamericana, 1996.

VIVAS Andrés y GONTCHAROV Andrei. "CONFERENCIAS SISTEMAS DE CONTROL". Universidad del Cauca, 1999.

VIVAS Andrés. "ANÁLISIS Y DISEÑO DE SISTEMAS DE CONTROL UTILIZANDO MATLAB". Universidad del Cauca, 1997.

RENGIFO Carlos. "SIMULACIÓN EN MATLAB DE UN SISTEMA DE CONTROL DE VELOCIDAD PARA UN MOTOR DC.". Revista Electrónica y Computadores N_o 65. Cekit - Pereira.

CASTAÑO andrés. "MODULADOR POR ANCHO DE PULSO PWM". Revista Electrónica y Computadores N_o 64. Cekit - Pereira.

H. LILEN. "TIRISTORES Y TRIACS". Ed. Marcombo, 1994.

SCHILDT, Herbert. C GUÍA DE AUTOENSEÑANSA. Ed. McGraw Hill, 1998.

GUREWICH Nathan y GUREWICH Ori. APRENDIENDO VISUAL C++5 EN 21 DÍAS. Ed. Prentice Hall, 1996.

KRUGLINSKI, David. PROGRESE CON VISUAL C++. Ed. McGraw Hill, 1994.

KRUGLINSKI, David. PROGAMACIÓN AVANZADA CON VISUAL C++. Ed. McGraw Hill, 1996.

"NEURAL NETWORK TOOLBOX USER'S GUIDE". The Math Works Inc, 1998.

"SIGNAL PROCESSING TOOLBOX USER'S GUIDE". The Math Works Inc, 1998.

"MATLAB REFERENCE GUIDE". The Math Works Inc, 1998.