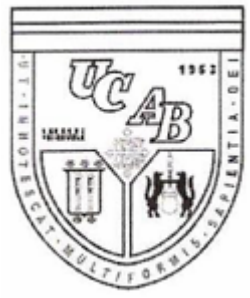


UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES

DISEÑO DE UN SISTEMA DE RECONOCIMIENTO DEL HABLA PARA CONTROLAR DISPOSITIVOS ELÉCTRICOS

TRABAJO ESPECIAL DE GRADO

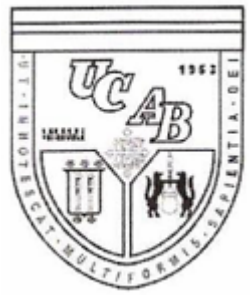
presentado ante la

UNIVERSIDAD CATÓLICA ANDRÉS BELLO

Como parte de los requisitos para optar al título de

INGENIERO EN TELECOMUNICACIONES

REALIZADO POR	*Dayana Karina Salcedo Cherubini* *Alejandro Patricio Teixeira Gómez*
PROFESOR GUÍA	*María Cristi Stefanelli*
FECHA	Caracas, 7 de julio de 2006.



UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

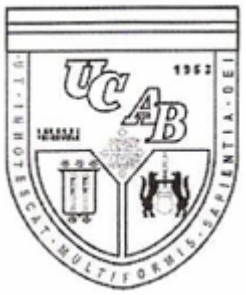
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES

DISEÑO DE UN SISTEMA DE RECONOCIMIENTO DEL HABLA PARA CONTROLAR DISPOSITIVOS ELÉCTRICOS

REALIZADO POR — *Dayana Karina Salcedo Cherubini*
Alejandro Patricio Teixeira Gómez

PROFESOR GUÍA — *María Cristi Stefanelli*

FECHA — Caracas, 7 de julio de 2006.



UNIVERSIDAD CATÓLICA ANDRÉS BELLO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES

DISEÑO DE UN SISTEMA DE RECONOCIMIENTO DEL HABLA PARA CONTROLAR DISPOSITIVOS ELÉCTRICOS

Este jurado; una vez realizado el examen del presente trabajo ha evaluado su contenido con el resultado:_____

J U R A D O E X A M I N A D O R

Firma:	Firma:	Firma:
Nombre: María Cristi Stefanelli	Nombre: Javier Barrios	Nombre: Iñaki Mendizábal

REALIZADO POR	*Dayana Karina Salcedo Cherubini* *Alejandro Patricio Teixeira Gómez*
PROFESOR GUÍA	*María Cristi Stefanelli*
FECHA	Caracas, 7 de julio de 2006.

Resumen

Dada la creciente necesidad que tiene el ser humano de desenvolverse en un ambiente cómodo, flexible y moderno, se plantea el reto de desarrollar un sistema que, usando el reconocimiento del habla como herramienta, permita el control de dispositivos eléctricos de una forma sencilla y sin ambigüedades. El desarrollo de este proyecto se basa en un estudio teórico sobre la naturaleza, parametrización, características y procesamiento de señales de voz, además de un estudio práctico que contempla simulaciones en MATLAB de distintas señales de voz, para analizar sus semejanzas y diferencias de acuerdo a la persona que las genera. Se realizó la evaluación teórica y práctica del desempeño de las técnicas de parametrización de la señal de voz Mel Frequency Cepstrum Coefficients (MFCC), Linear Frequency Cepstrum Coefficients (LFCC) y Linear Predictive Cepstrum Coefficients (LPCC) dentro del esquema del reconocimiento del habla, la programación de un microcontrolador como elemento de procesamiento dentro del hardware encargado de controlar los dispositivos eléctricos, para interpretar los comandos de la aplicación y convertirlos en acciones de encendido y apagado de los mismos; además se recolectó un total de 300 señales de voz entre 20 participantes: 200 de ellas para la creación de una base de datos de entrenamiento y las 100 restantes para la base de datos de pruebas del sistema. Entre los resultados más resaltantes se destacan la escogencia del MFCC como técnica de parametrización por su alto porcentaje de acierto en el reconocimiento del habla, además del "Sistema de reconocimiento del habla para controlar dispositivos eléctricos" completamente funcional.

Palabras clave: Procesamiento digital de la voz, parametrización de la voz, reconocimiento del habla

Dedicatoria y Agradecimientos

Le dedicamos el fruto de este arduo trabajo, primero a Dios por ser nuestro guía y orientador en cada uno de nuestros pasos, haciendo posible que lográramos con éxito todas las metas que nos hemos trazado, no sólo en el desarrollo de este trabajo, sino en nuestras vidas.

Dedicamos también este trabajo a nuestros padres: Beatriz Gómez y Carlos Teixeira (padres de Alejandro), así como a Miroslava Cherubini y Sali Salcedo (padres de Dayana) por estar siempre allí, en las buenas y malas, por ser nuestro pilar, nuestro apoyo incondicional; les agradecemos su guía, su ánimo y motivación para hacernos cada vez mejores personas y profesionales. A ustedes queremos dedicarles éste y todos los logros de nuestra vida.

Nuestro más profundo agradecimiento y reconocimiento a la Universidad Católica Andrés Bello y a nuestros profesores, por darnos una excelente formación profesional y la oportunidad de ser los pioneros de la carrera de Ingeniería de Telecomunicaciones en Venezuela, en especial a la Profesora María Cristi Stefanelli por sus orientación, su ayuda y apoyo en el desarrollo de este trabajo.

Índice General

Índice de tablas

Índice de figuras

Introducción

El diseño e implementación de un sistema que, mediante el uso de las telecomunicaciones permita satisfacer ciertas necesidades del ser humano es el fundamento sobre el cual se sostiene este Trabajo Especial de Grado, el cual pretende explicar mediante un texto estructurado en capítulos y a través de numerosos ejemplos gráficos, la metodología seguida para el desarrollo de un "Sistema de Reconocimiento del habla para controlar dispositivos eléctricos".

El capítulo uno define el planteamiento del proyecto, con sus alcances, limitaciones, objetivos y una explicación general de su funcionamiento; en el segundo capítulo se describen los resultados de la revisión bibliográfica realizada sobre: la naturaleza, características, pre-procesamiento y parametrización de la señal de voz, así como las técnicas empleadas para el reconocimiento del habla.

En el tercer capítulo se describen de forma detallada las fases en la metodología seguidas para el diseño e implementación del proyecto. Este capítulo se divide en: las fases de recopilación de información, diseño de la aplicación encargada del reconocimiento del habla, recolección de señales de voz para crear la base de datos de entrenamiento y pruebas del sistema, pruebas comparativas entre las distintas técnicas de parametrización basadas en el cálculo de los coeficientes cepstrales (MFCC, LFCC y LPCC) y el diseño del hardware encargado de controlar los dispositivos eléctricos conectados a él.

El cuarto capítulo corresponde a los resultados obtenidos en las fases intermedias y finales del desarrollo del proyecto, tales como: programación de la aplicación encargada del reconocimiento, escogencia de MFCC como técnica de parametrización de acuerdo a un estudio comparativo realizado entre las distintas

técnicas estudiadas y la implementación del hardware encargado del control de los dispositivos eléctricos.

Por su parte, en el capítulo cinco se recogen las conclusiones sobre los logros, resultados y experiencias obtenidos del proyecto realizado, además de ciertas recomendaciones sobre aspectos que pueden cambiarse en el sistema. Adicionalmente, se encuentran los anexos y apéndices, en los cuales se recoge información complementaria a la investigación realizada y que puede ser usada para futuros desarrollos que mejoren el desempeño del sistema ya implementado.

Capítulo I
Planteamiento del Proyecto

El reconocimiento del habla tiene como objetivo principal permitir la comunicación hablada entre el ser humano y el computador, para ejecutar acciones determinadas mediante comandos de voz. El principal reto que se plantea en un sistema da reconocimiento del habla es el de interpretar un conjunto de informaciones que proceden de diversas fuentes de conocimiento (acústica, fonética, fonológica, léxica, sintáctica, semántica y pragmática), en presencia de ambigüedades e incertidumbres, para obtener una interpretación aceptable del mensaje acústico recibido. La finalidad de uso del reconocimiento del habla en el desarrollo del sistema desarrollado en este Trabajo Especial de Grado es la del control de dispositivos eléctricos utilizando comandos de voz.

El principal objetivo de este proyecto es la implementación de un sistema que, utilizando el reconocimiento del habla como herramienta, permita controlar dispositivos eléctricos usualmente encontrados en un ambiente doméstico y hacer de éste un espacio cómodo, flexible y moderno. Además, este estudio pretende traer beneficios adicionales para las personas discapacitadas, las cuales requieren de una ayuda extra para realizar tareas cotidianas como encender una luz, mover una cama clínica o apagar la televisión.

Para la implementación del sistema que se muestra en la figura 1 se desarrolló una aplicación encargada del reconocimiento del habla, basada en los resultados obtenidos del estudio de la voz, sus características y técnicas de parametrización y que corre en un PC, bajo el ambiente de desarrollo MATLAB. Además, se implementó el hardware encargado del encendido y apagado de los dispositivos

eléctricos conectados a él, de acuerdo al comando de voz procesado por la aplicación encargada del reconocimiento del habla.

Los dispositivos que el sistema puede manejar son aquellos cuyo encendido y apagado no depende de un circuito electrónico y que al activarse un relé que permite o impida el paso de corriente hacia ellos, se enciendan o apaguen dependiendo del caso. Algunos ejemplos de este tipo de dispositivos son: televisores y lámparas con perilla de ENCENDIDO/APAGADO, cargador del celular y cargador del laptop.

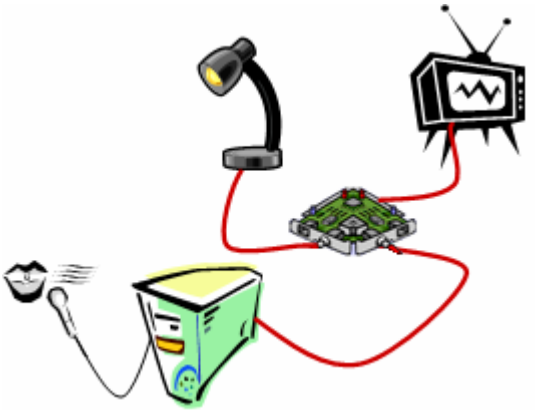


Figura 1. Componentes del Sistema de Reconocimiento del habla para el control de dispositivos eléctricos

La aplicación encargada de realizar el reconocimiento del habla está diseñada para el idioma español y para reconocer la palabra clave que constituye el comando de voz, no quién la dice (hombre, mujer, niño o niña). El sistema descrito maneja cinco comandos de voz correspondientes a las vocales a, e, i, o, u. Además, contempla una única implementación de hardware de encendido y apagado de los dispositivos eléctricos, el cual se conecta al computador mediante el puerto serial, utilizando la interfaz EIA – RS232.

Capítulo II
Marco Teórico

Para la realización de cualquier tarea que involucre el procesamiento de la señal de voz es fundamental el conocimiento de ciertos aspectos que ayudan a crear las bases necesarias para la comprensión de las características de la voz y del tratamiento que se le debe dar de acuerdo al fin deseado. Para la aplicación particular de reconocimiento del habla, el esquema de procesamiento que se sigue es similar al que se muestra en la figura 2

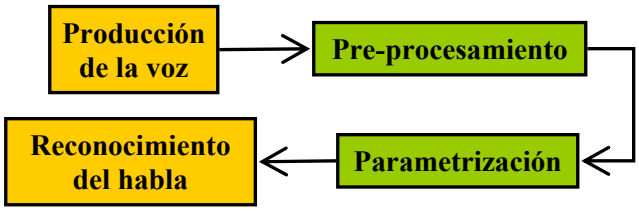


Figura 2. Esquema de procesamiento de la voz para el reconocimiento del habla

Para el establecimiento de las bases necesarias para el reconocimiento del habla, es necesario entender las fases mostradas en la figura 2, cada una de las cuales se describe más detalladamente en las secciones siguientes.

II.1 Naturaleza de la señal de voz.

Reyes y Herrera (2005) indican que la voz se obtiene por la acción conjunta de varias regiones de órganos: el *tracto pulmonar*, formado por los pulmones y la tráquea, los cuales controlan la amplitud de los sonidos. Por otra parte se encuentra *la laringe*, donde se sitúan las cuerdas vocales, que controlan la entrada de los sonidos y cuya tensión afecta la frecuencia (tono) de la señal de voz. Por último se encuentra *el tracto vocal*, que se encarga de la articulación de la voz, principalmente por la lengua, los labios y la mandíbula baja (Grupo PAS – Universidad de Deusto, n.d). En la

figura 3 se puede observar un diagrama de la cavidad bucal y los órganos que participan en el proceso de generación de la señal de voz (http://webschoolsolutions.com/patts/systems/mouth.gif, 2000).

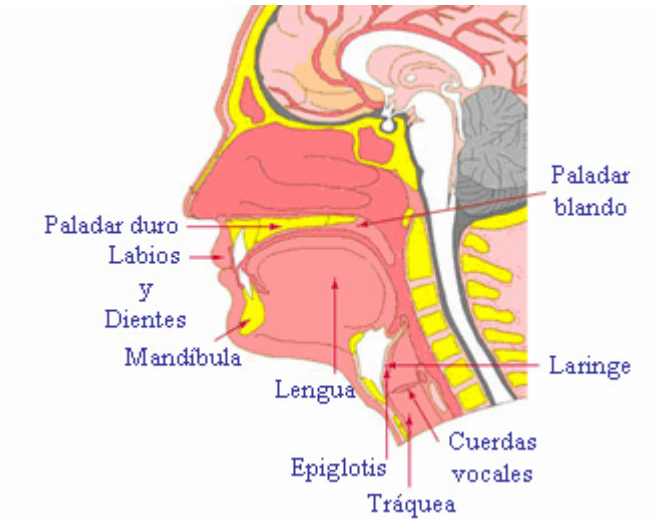


Figura 3. Diagrama de la cavidad bucal y los órganos que la conforman

Dependiendo del comportamiento de la cavidad bucal y de la configuración de los órganos mostrados en la figura 3, la señal que se genera se clasifica en dos tipos:

- *Señal sonora:* Se produce por la vibración de las cuerdas vocales, las cuales se abren y cierran modificando el área de la tráquea, produciendo una señal modulada y cuasi-periódica, con un período o frecuencia fundamental llamado *pitch.* Según López (1999), este tipo de señal se caracteriza por tener alta energía y su rango de frecuencias está entre 300 Hz a 4000 Hz. Un ejemplo de señal sonora generada por un niño de 9 años se puede apreciar en la figura 4

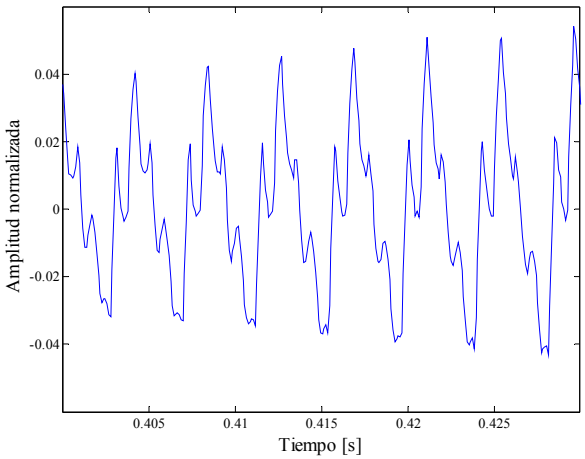
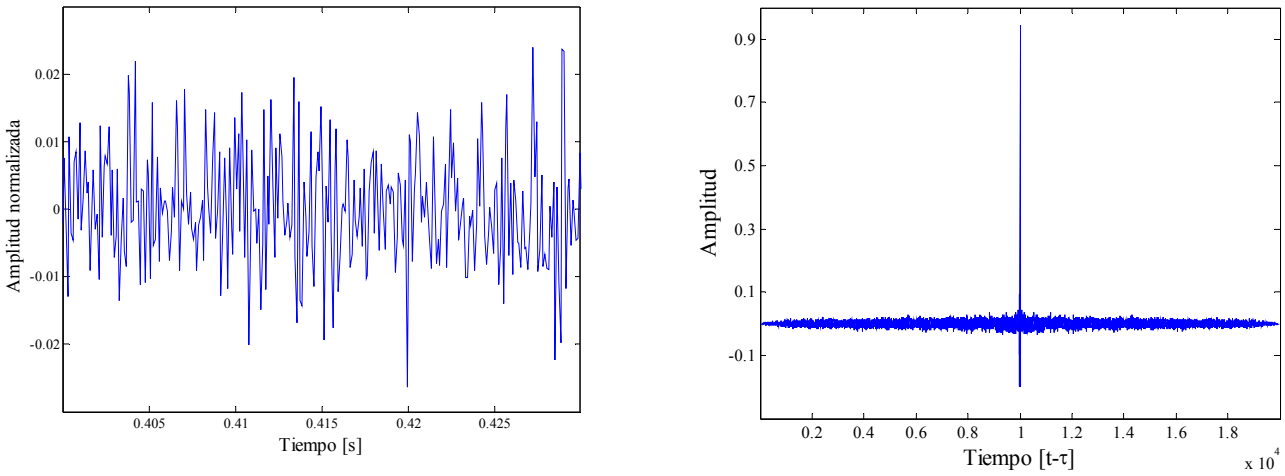


Figura 4. Ejemplo de la señal sonora correspondiente a la vocal "u"

En figura 4 se puede observar que la señal cumple con cierto patrón de periodicidad, el cual se produce por la modulación que el movimiento de las cuerdas vocales genera sobre el flujo de aire que atraviesa la cavidad vocal.

- *Señal no sonora*: En su generación el aire fluye libremente, al permanecer abiertas las cuerdas vocales hasta alcanzar el tracto vocal, por lo que la señal generada se constituye de una contribución desordenada de componentes frecuenciales. Se caracteriza por tener baja energía y componente frecuencial uniforme, presentando aleatoriedad similar a la del ruido blanco, tal y como se muestra en la figura 5, la cual corresponde a un hombre de 22 años



(a) *(b)*

Figura 5. *(a)* Ejemplo de la señal no sonora correspondiente a "sss", *(b)* Autocorrelación de la señal no sonora correspondiente a "sss"

De la figura 5 se puede observar que una señal no sonora se asemeja mucho al ruido blanco gaussiano, lo cual se comprueba mediante el cálculo de su autocorrelación, en la que se puede observar un Delta de Dirac en el origen, lo que representa el hecho de que la señal analizada carece en su totalidad de periodicidad.

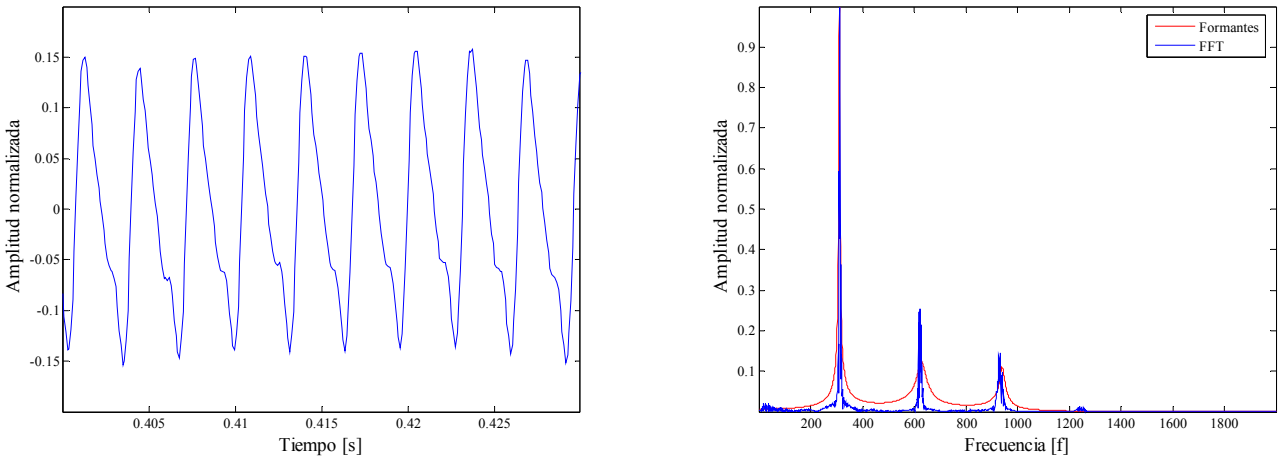
Para las señales sonoras, el tracto vocal actúa como una cavidad resonante, estando centradas las frecuencias de resonancia generalmente en 500 Hz y sus armónicos pares. Esta resonancia produce grandes picos en el espectro resultante, a los cuales se les llama *formantes*. Por su parte, para las señales no sonoras, el tracto vocal presenta una estructura ruidosa y aleatoria tanto en el dominio temporal como en el frecuencial, por lo que no se tienen formantes. Un *formante* es el pico de intensidad en el espectro de un sonido, el cual representa la concentración de energía que se da en una determinada frecuencia y viene determinado por el proceso de filtrado que se produce en el tracto vocal, dependiendo de la configuración de los articuladores (lengua, mandíbula, labios, velo del paladar) (Enciclopedia Wikipedia, 2006).

El formante con la frecuencia más baja se llama f_1, el segundo f_2, el tercero f_3 y así sucesivamente. Normalmente sólo los dos primeros son necesarios para caracterizar una vocal, pero los formantes posteriores determinan propiedades acústicas como el timbre. En la tabla 1 se muestra la región en la que se encuentran los principales formantes de las vocales a, e, i, o, u (Enciclopedia Wikipedia, 2006, http://es.wikipedia.org/wiki/Formante).

Vocal	Principal región de las formantes (Hz)
U	200 a 400 Hz
O	400 a 600 Hz
A	800 a 1200 Hz
E	400 a 600 y 2200 a 2600 Hz
I	200 a 400 y 3000 a 3500 Hz

Tabla 1 Principal región de las formantes de las vocales

En la figura 6 se pueden observar la representación en tiempo de la señal correspondiente a la vocal "u" y sus formantes, las cuales fueron calculadas mediante la aplicación de la técnica LPC con 18 coeficientes, que consiste en el cálculo de los coeficientes del filtro que modela el tracto vocal a través de predicción lineal y que en frecuencia representan la envolvente del espectro de Fourier, por lo que mediante esta técnica se pueden ubicar claramente las formantes.



(a) *(b)*

Figura 6. *(a)* Ejemplo de la señal correspondiente a la vocal "u", *(b)* Transformada de Fourier de la señal *a* con la ubicación de sus principales formantes

En la figura 6 se puede observar un patrón casi-estacionario similar al que se desprende de la figura 4, además de la clara diferenciación de las principales formantes de la letra "u", que son las frecuencias en las cuales existe mayor concentración de energía para esa señal. De forma más general, en la figura 7 se muestra el espectrograma de las vocales "i", "u" y "a" respectivamente, en el que se puede observar que las zonas más densas representan las primeras dos formantes de cada señal (Enciclopedia Wikipedia, 2005, http://es.wikipedia.org/wiki/Formante).

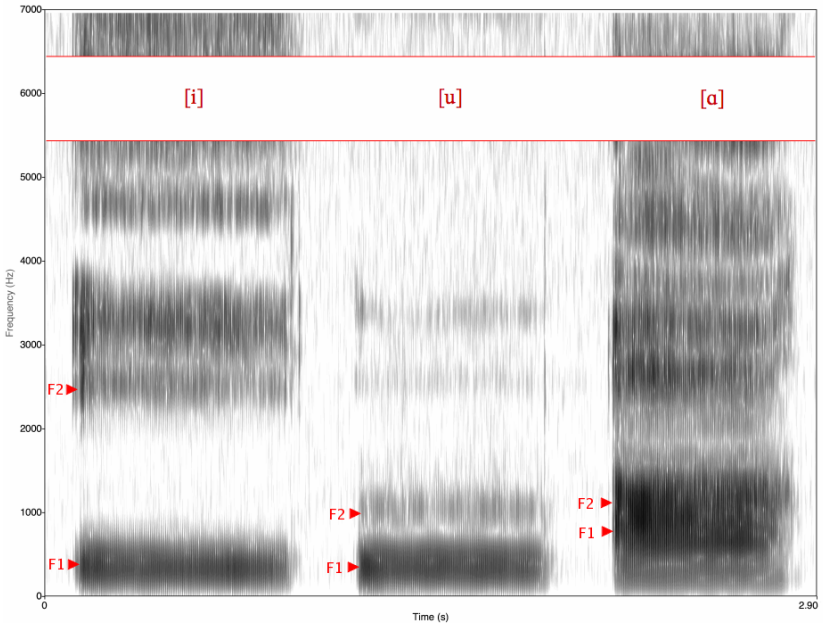


Figura 7. Espectrograma correspondiente a las vocales “i”, “u” y “a”

La Transformada de Fourier es una herramienta muy importante para el análisis de las señales de voz en el dominio de la frecuencia, ya que permite obtener información de las mismas que no es evidente en el dominio del tiempo. Una de las características que la Transformada de Fourier permite extraer es el contenido frecuencial de la señal, lo cual es muy importante para establecer el muestreo adecuado para la misma según el Teorema de Nyquist, que establece que la frecuencia de muestreo óptima para cualquier señal debe ser por lo menos dos veces el valor de su frecuencia máxima. En la figura 8 se presentan dos señales grabadas a una frecuencia de muestreo de 10 KHz, correspondientes a la palabra Televisor dicha por dos personas diferentes: una mujer y un hombre de 21 años.

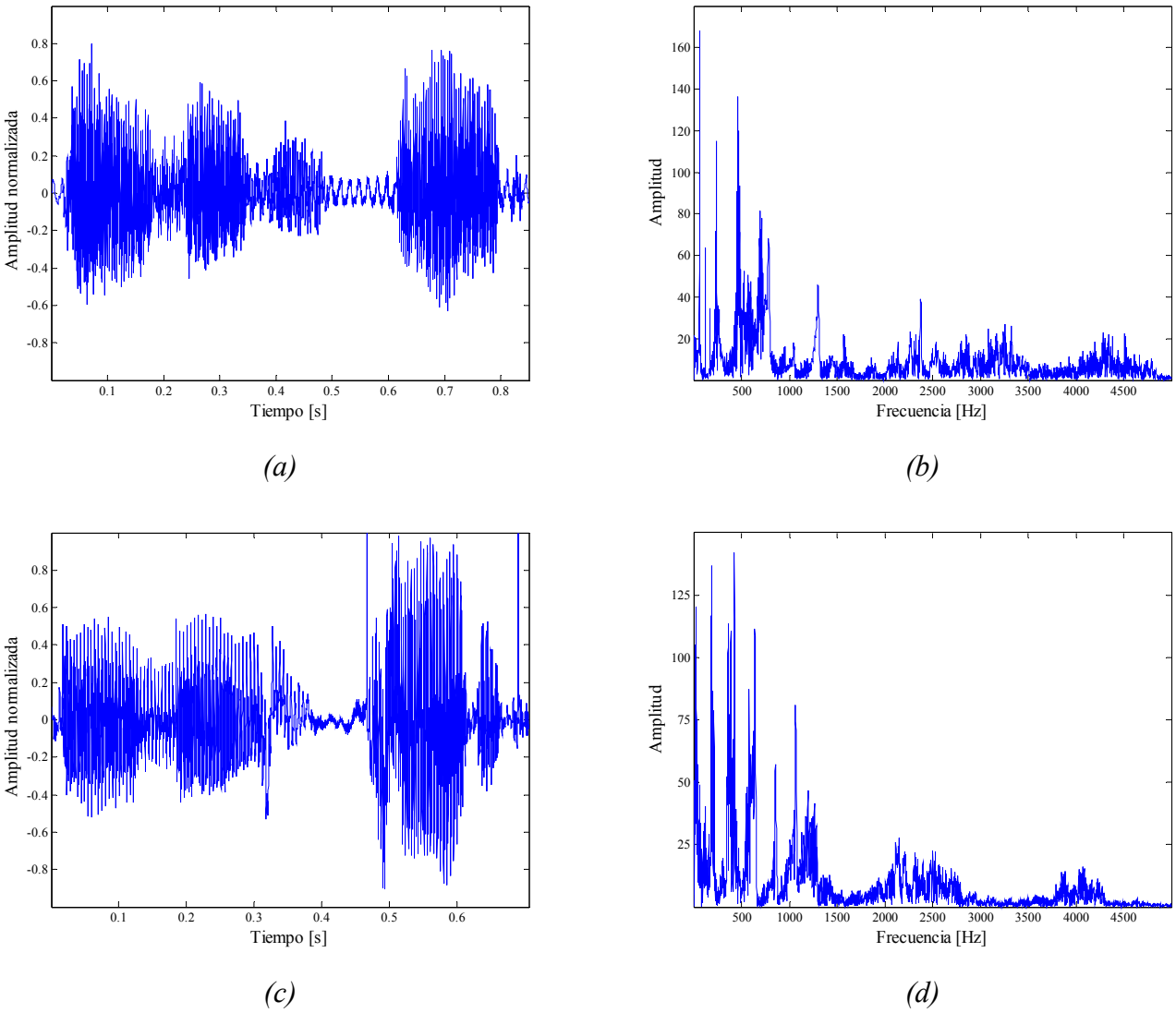


Figura 8. Representación en tiempo de la palabra Televisor: (a) Mujer de 21 años, (c) Hombre de 21 años. Representación en frecuencia de la palabra Televisor: (b) Mujer de 21 años, (d) Hombre de 21 años.

En la figura 8 se puede observar que las señales correspondientes al hombre y la mujer para la misma palabra son similares, aunque en su representación en frecuencia se observa que la señal correspondiente al hombre tiene mayor concentración de energía en las bajas frecuencias dado que la voz del hombre por lo general es más grave que la voz de la mujer.

II.2 Modelo de producción de la voz.

El modelo de producción de la voz es útil para muchas aplicaciones, entre ellas, la síntesis de la voz y la extracción de ciertos parámetros que permiten identificarla, ya que el tracto vocal se representa como un filtro que varía en el tiempo, dependiendo de la acción que se realiza al pronunciar una palabra. Al obtener los coeficientes que definen tal filtro, se obtienen por lo tanto, ciertas características de la señal de voz que posteriormente permitirán identificarla.

El filtro que modela el tracto vocal tiene dos posibilidades para la señal de entrada, que dependerán de si la misma es sonora o no sonora. Para señales sonoras, la excitación será un tren de impulsos de frecuencia controlada (pitch), mientras que para las señales no sonoras la excitación será ruido blanco, tal y como se muestra en el modelo de la figura 9 (De la Torre, n.d)

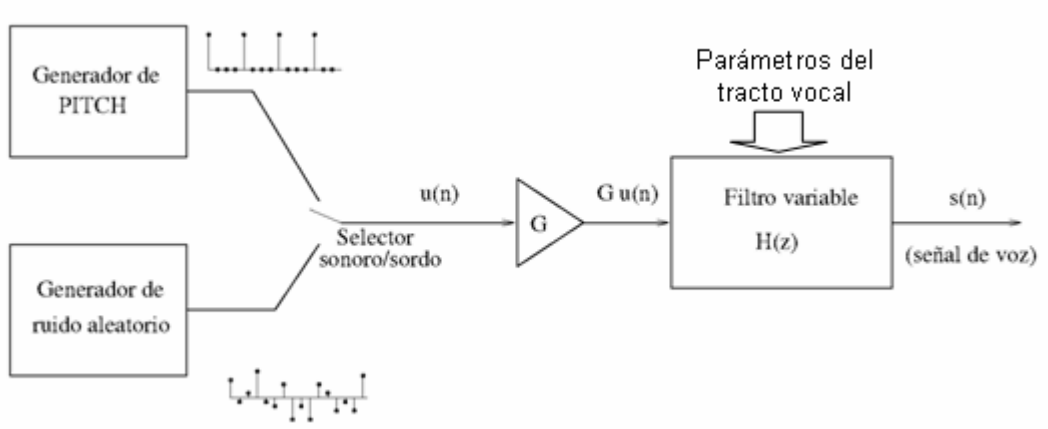


Figura 9. Modelo simplificado de producción de la voz

Según el modelo presentado en la figura 9, la señal de entrada puede ser un tren de impulsos periódicos o ruido aleatorio, dependiendo de la naturaleza de la señal. Además, se cuenta con una ganancia definida por la forma del tracto vocal y un filtro variable en tiempo que modela a este último y cuyos coeficientes dependen directamente de sus parámetros. Se supone además que las dos señales pueden

separarse sin considerar ninguna interacción entre ellas, lo que en la práctica no es aplicable pero puede ser ignorado, resultando el modelo lo suficientemente adecuado (Flores, 1998).

El tracto vocal, por poseer una característica de cavidad resonante, manifiesta un número muy grande de resonancias, de las cuales se consideran sólo las tres o cuatro primeras (formantes), cubriendo éstas un rango de frecuencias entre 100 y 3500 Hz, ya que las resonancias de alta frecuencia son atenuadas por la característica frecuencial del tracto que tiende a actuar como un filtro pasabajo con una caída de aproximadamente -12 dB por octava (Iosu, 1999).

II.3 Pre-procesamiento de la señal de voz.

La etapa de pre-procesamiento de la señal de voz corresponde a los pasos previos a la parametrización, necesarios para resaltar sus características más importantes y luego poder analizarlas con la técnica de parametrización escogida para tal fin. La figura 10 muestra un diagrama de bloques del pre-procesamiento aplicado a una señal de voz antes de la fase de parametrización.

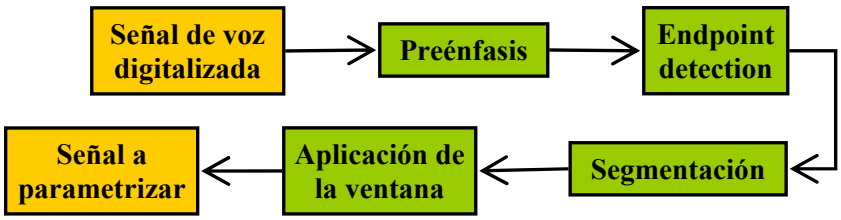


Figura 10. Esquema de pre-procesamiento de la señal de voz

La señal digitalizada que se tiene en el computador después de la grabación ya ha pasado por los siguientes procesos: Conversión de la señal de voz (onda sonora) a señal eléctrica (micrófono) y conversión análoga-digital (muestreo, cuantificación y

codificación). Las fases siguientes: preénfasis, endpoint detection, segmentación y aplicación de la ventana se describen a continuación.

II.3.1 Preénfasis

Después de que la señal de entrada se tiene digitalizada, la etapa de *preénfasis* se realiza para hacer el procesamiento de la señal menos susceptible a truncamientos, aplanarla espectralmente y para compensar la caída de 6 dB que experimenta la señal al pasar a través del tracto vocal (Kornhauser, 1999). El filtro utilizado puede tener coeficientes fijos o ser adaptativo, pero generalmente se usa un filtro digital de primer orden cuya función de transferencia es la que se muestra en la ecuación 2.1 (Flores, 1 de octubre de 1998)

$$H(z) = 1 - a \cdot z^{-1} \qquad (2.1)$$

Donde $0,9 \le a \le 0,95$, valor que se escoge cercano a la unidad a fin de que la estructura de los formantes mayores sean acentuadas. La representación en frecuencia del filtro de preénfasis mencionado se muestra en la figura 11

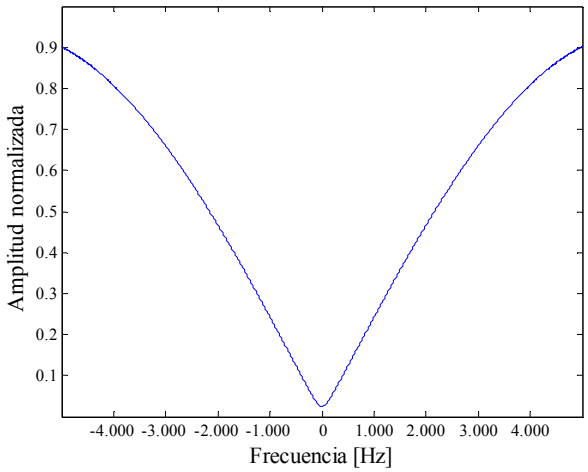


Figura 11. Representación en frecuencia del filtro de preénfasis

La salida del sistema de preénfasis $\tilde{s}(n)$ está relacionada a la entrada del sistema $s(n)$ mediante la ecuación 2.2

$$\tilde{s}(n) = s(n) - a \cdot s(n-1) \qquad (2.2)$$

II.3.2 Endpoint Detection.

Una vez que la señal de voz se tiene grabada en el computador, es importante determinar el comienzo y final de la parte útil de la señal, para lo cual se utiliza la técnica denominada *endpoint detection*, que mediante un algoritmo busca los puntos de la señal donde se concentra la energía y extrae sólo ese fragmento. (Slavinsky, 1999)

II.3.3 Segmentación y aplicación de la ventana.

Después de que se tiene la información útil de la señal de voz, se hace necesario dividir la señal en tramas de N muestras, donde N es un valor que se escoge tomando en cuenta que la señal de voz es estacionaria a "trozos"; condición necesaria para poder realizar el análisis de Fourier en tiempo corto (utilizado para la mayoría de las aplicaciones que involucran señales de voz). El intervalo de tiempo en el que la señal se considera estacionaria depende de la velocidad de cambios del tracto vocal y las cuerdas vocales y comúnmente se establece un valor entre 20 y 40 ms (Grupo PAS, n.d).

En la figura 12 se muestra un ejemplo de segmentación utilizado comúnmente en las aplicaciones que involucran el procesamiento de la voz. Por lo general se utilizan tramas de 256 muestras por ser un valor que establece un equilibrio entre la resolución en tiempo y en frecuencia para una señal de voz. En el caso particular de la

figura 12, esta cantidad de muestras corresponde a 25,6 ms de señal, ya que la frecuencia de muestreo es de 10 KHz.

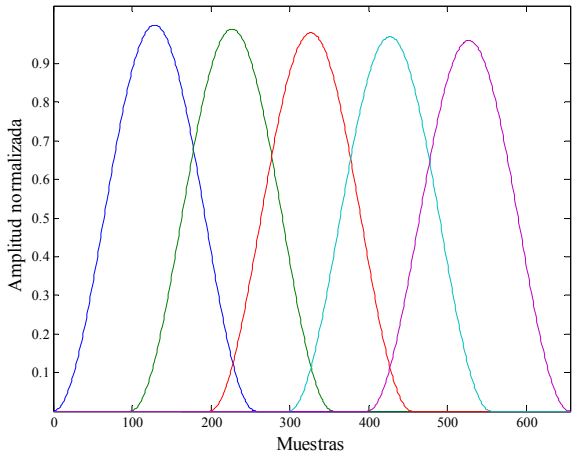


Figura 12. Segmentación de la señal de voz en tramas solapadas entre ellas

La siguiente fase de la etapa de pre-procesamiento corresponde a la escogencia del tipo de ventana que se le aplica a cada trama. Dicha escogencia es muy importante para analizar el efecto de la misma sobre la resolución espectral de la señal, la cual depende del ancho del lóbulo principal de la ventana y la atenuación de los lóbulos secundarios respecto al principal. Lo ideal es que el lóbulo principal sea muy estrecho y los lóbulos secundarios sean lo más pequeños posible (similar a un Delta de Dirac).

Tanto las ventanas rectangulares, como las no rectangulares tienen sus ventajas y desventajas en cuanto a los criterios mencionados anteriormente, pero en el caso específico de la ventana Hamming se cumple con el requerimiento de que la atenuación de los lóbulos secundarios sea brusca, a pesar de que su lóbulo principal es más ancho que para la ventana rectangular, lo que permite obtener una mejor resolución espectral. La ventana Hamming se define matemáticamente por la ecuación 2.3

$$W(nT) = 0,54 - 0,46 \cdot \cos\left(2 \cdot \frac{n}{N}\right) \qquad (2.3)$$

Donde $0 < n < N$

Otro aspecto que es importante analizar es el solapamiento entre tramas consecutivas, el cual para la figura 12 es de 156 muestras y se realiza con dos propósitos principales: el primero consiste en que las mismas guarden relación entre sí para que el análisis de cada trama no sea aislado y el segundo consiste en compensar el hecho de que para la ventana Hamming, la mayor concentración de energía se encuentra en el centro de la trama *n* y para compensar la caída que tiene la ventana en los bordes de la trama, se utiliza la ventana aplicada a la trama *n*+1.

II.4 Técnicas de parametrización de la señal de voz.

La selección de la mejor representación paramétrica de la señal de voz es una tarea importante en el diseño de cualquier sistema de reconocimiento del habla. Los objetivos principales de esta representación son los de comprimir los datos correspondientes a la señal de voz, eliminando información no pertinente al análisis fonético de la información y extraer esas características de la señal de voz que contribuyen significativamente con la detección de las diferencias fonéticas y que no son apreciables mediante un simple análisis en tiempo o en frecuencia, sino usando un análisis más exhaustivo como el Cepstrum, el cual se explica en la sección II.4.1

II.4.1 Cepstrum.

El Cepstrum es una herramienta muy utilizada para la representación paramétrica de las señales de voz y se define como la Transformada de Fourier del espectro logarítmico de la señal, por lo que existe un Cepstrum complejo y un Cepstrum real

dependiendo de si la función logarítmica está definida para valores reales o complejos. La diferencia entre uno y otro radica en el hecho de que el cepstrum complejo permite reconstruir la señal y el real no, ya que se pierde la información correspondiente a la fase.

Para entender las bases matemáticas del cálculo del Cepstrum es necesario considerar una secuencia estable $x[n]$, cuya Transformada Z se puede expresar en coordenadas polares según la ecuación 2.4

$$X(z) = |X(z)| \cdot e^{j \measuredangle X(z)} \qquad (2.4)$$

Donde $|X(z)|$ y $\measuredangle X(z)$ representan la magnitud y el ángulo respectivamente de la Transformada Z de $x[n]$

Como la señal $x[n]$ es estable, la región de convergencia para $X(z)$ incluye el círculo unitario y la Transformada de Fourier de $x[n]$ existe y es igual a $X(e^{j\omega})$. El cepstrum complejo correspondiente a $x[n]$ se define como una secuencia estable $\hat{x}[n]$, cuya Transformada Z se puede definir según la ecuación 2.5

$$\hat{X}(z) = \log[X(z)] \qquad (2.5)$$

Como se requiere que $\hat{x}[n]$ sea estable, la región de convergencia incluye el círculo unitario, por lo que el cepstrum complejo se puede representar usando la Transformada Inversa de Fourier, tal y como se observa en la ecuación 2.6

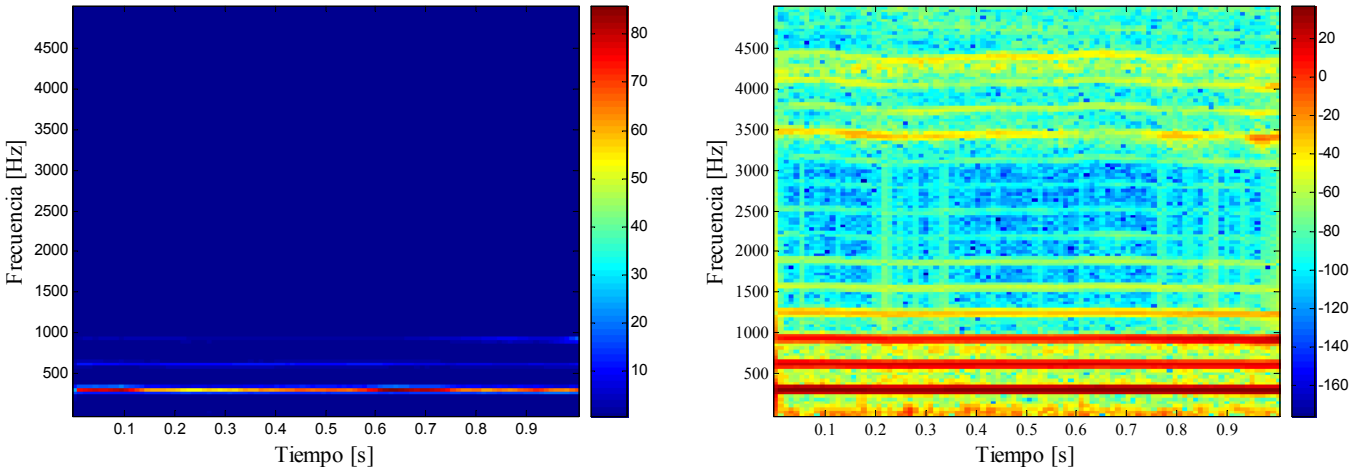
$$\hat{x}[n] = \frac{1}{2\pi} \cdot \int_{-\pi}^{\pi} \log[X(e^{j\omega})] \cdot e^{j\omega n} \cdot d\omega \qquad (2.6)$$

En contraste con el cepstrum complejo, el $c_x[n]$ o *cepstrum real* de una señal, es definido como la Transformada Inversa de Fourier del logaritmo de la magnitud de la Transformada de Fourier, tal y como se muestra en la ecuación 2.7

$$c_x[n] = \frac{1}{2\pi} \cdot \int_{-\pi}^{\pi} \log\left|X\left(e^{j\omega}\right)\right| \cdot e^{j\omega n} \cdot d\omega \qquad (2.7)$$

El cepstrum real se utiliza en muchas aplicaciones y como no depende de la fase de $X\left(e^{j\omega}\right)$ es mucho más fácil de calcular que el cepstrum complejo, aunque $x[n]$ no puede ser recuperada a partir de $c_x[n]$. (Oppenheim & Schafer, 1989)

El análisis cepstral es comúnmente utilizado para obtener información de la señal de voz que permita parametrizarla para luego ser usada en la fase de reconocimiento. Mediante el cepstrum se puede separar la señal de excitación del sistema que modela la producción de la voz y la función de transferencia que modela el tracto vocal. Por esta razón es que al analizar el espectrograma mostrado en la figura 13b se pueden observar componentes frecuenciales que no se aprecian en la figura 13a



(a) *(b)*

Figura 13. Espectrograma: *(a)* Señal de voz original, *(b)* Señal de voz con cepstrum

Se puede decir que esta transformada detecta las periodicidades que se observan en un espectro logarítmico como el de la figura 13b. Dichas periodicidades, para las señales sonoras, son de dos tipos: unas rápidas, debidas a la estructura armónica del espectro, que se repiten en los múltiplos de la frecuencia fundamental Fo, y unas fluctuaciones mucho más lentas, no periódicas, que proporcionan la envolvente espectral. Estas fluctuaciones lentas se manifiestan en la parte baja del Cepstrum y caracterizan la forma del tracto vocal. De hecho, también se reducen a unos pocos datos numéricos (del orden de 15 o 20), que parametrizan la información articulatoria. La información correspondiente a la fuente excitadora se sitúa, por el contrario, en la parte alta del Cepstrum, y corresponde básicamente a la periodicidad de Fo (detección de periodicidad y período correspondiente) y a la energía global de la señal. Esta separación de las dos informaciones permite un proceso de desconvolución de la señal de voz, para recuperar separadamente la excitación y la respuesta impulsiva del tracto vocal, en caso de que la aplicación así lo requiera (Martí, 1998).

Las representaciones paramétricas más utilizadas para trabajar con señales de voz y que están basadas en el análisis cepstral de la misma, pueden ser divididas en dos grupos: aquellas basadas en la predicción lineal del espectro y aquellas basadas en el espectro de Fourier, cada una de las cuales se explican en las secciones siguientes (Davis &Mermelstein, 1980).

II.4.2 Técnicas basadas en la predicción lineal del espectro.

Una de las técnicas más usadas en el procesamiento de señales de voz es el análisis de predicción lineal. Esta técnica ha probado ser muy eficiente debido a la posibilidad de parametrizar la señal con un número pequeño de patrones con los cuales es posible, entre otras cosas, reconstruirla adecuadamente (Herrera, n.d). Mediante esta técnica se puede representar la señal de voz usando parámetros que

varían en el tiempo y que están relacionados con la función de transferencia del tracto vocal y las características de la fuente sonora.

II.4.2.1 Predicción Lineal de los Coeficientes Cepstrales: LPCC.

La técnica LPCC permite estimar los coeficientes cepstrales mediante el uso del algoritmo LPC, el cual establece un modelo que permite calcular la próxima muestra de la señal mediante la función de transferencia que se define mediante la ecuación 2.8

$$H(z) = \frac{G}{1 - \sum_{k=1}^{p} a_k \cdot z^{-k}} \qquad (2.8)$$

Donde G es la ganancia del filtro, que depende de la naturaleza de la señal y a_k son los coeficientes del filtro que modela el tracto vocal (San Martín, 2004).

El análisis anterior se basa en el modelo de producción de la voz presentado en la figura 9 y su idea fundamental es que la voz puede modelarse a través de una combinación lineal de p muestras anteriores más una señal de excitación (periódica o ruido blanco dependiendo de la naturaleza de la señal), tal y como lo demuestra la ecuación 2.9 (GTAS, n.d)

$$s[n] = \sum_{k=1}^{p} a_k \cdot s[n-k] + G \cdot u[n] \qquad (2.9)$$

Donde $u[n]$ es la entrada del filtro que modela el tracto vocal, por lo que, dada la señal $s[n]$, el problema consiste en determinar los coeficientes de predicción a_k y la ganancia G del filtro.

Los coeficientes de predicción se usan en el proceso de parametrización para calcular los coeficientes cepstrales. Por lo tanto, dada una señal $s[n]$ (considerada estacionaria en el período de evaluación) un predictor de orden p se define según la ecuación 2.10

$$\tilde{s}[n] = -\sum_{k=1}^{p} a_k \cdot s[n-k] \qquad (2.10)$$

La determinación de los coeficientes de predicción a_k se realizará minimizando el error de predicción de orden p que se comete cuando se intenta realizar la aproximación de la señal y cuya representación se puede obtener mediante la ecuación 2.11

$$e[n] = s[n] - \tilde{s}[n] = s[n] + \sum_{k=1}^{p} a_k \cdot s[n-k] \qquad (2.11)$$

Donde $e[n]$ es la señal de error y $s[n]$ la señal de voz

Los coeficientes de predicción se calculan minimizando la media del error cuadrático medio con respecto a cada uno de los coeficientes. La ecuación 2.12 representa $\tilde{e}$, que se define como el error cuadrático total

$$\tilde{e} = E\{e^2[n]\} \qquad (2.12)$$

Donde $E\{.\}$ es el operador valor esperado

Para obtener el mínimo error de predicción se calcula la derivada de $e[n]$ con respecto a los coeficientes a_k y se obtiene la ecuación 2.13

$$\sum_{k=1}^{p} a_k \cdot E\{s[n-k] \cdot s[n-i]\} = -E\{s[n] \cdot s[n-i]\}, \quad 1 < i < p \qquad (2.13)$$

El error de predicción mínimo viene definido por la ecuación 2.14

$$\tilde{e}_{mín} = E\left\{y^2[n]\right\} + \sum_{k=1}^{p} a_k \cdot E\left\{y[n] \cdot y[n-k]\right\} \qquad (2.14)$$

Según lo anterior, el error medio cuadrático mínimo se puede escribir en función de la autocorrelación según la ecuación 2.15

$$\tilde{e}_{mín} = R(0) + \sum_{k=1}^{p} a_k \cdot R(k) \qquad (2.15)$$

Estas ecuaciones son llamadas ecuaciones normales o de Yule-Walker y los coeficientes a_k del predictor óptimo se obtienen resolviendo las p ecuaciones con p incógnitas (Stefanelli, 1985).

Como resultado de todo este análisis se puede observar que mediante el uso del LPC como técnica de predicción lineal, de todos los posibles juegos de parámetros que se pueden obtener a partir del modelado de la voz como un proceso autorregresivo, los coeficientes a_k que modelan el tracto vocal, pueden ser utilizados para el cálculo de los coeficientes cepstrales o LPCC's mediante la ecuación 2.16, cuya demostración se puede encontrar en el Anexo A

$$c_i = a_i + \sum_{k=1}^{i-1} \left(\frac{k-i}{i} \right) \cdot c_{i-k} \cdot a_k \qquad (2.16)$$

De esta manera se puede observar cómo la predicción lineal de los coeficientes del filtro que modela el tracto vocal puede ser utilizada para estimar por ejemplo la densidad espectral de potencia de la señal de voz o como en este caso, para estimar los coeficientes cepstrales utilizados para el reconocimiento del habla.

II.4.3 Técnicas basadas en el espectro de Fourier.

Para esta técnica de parametrización, las características espectrales de la señal de voz se derivan del análisis de Fourier de tiempo corto, definido por la ecuación 2.17

$$S_x(t,f) = \int_{-\infty}^{\infty} x(\tau + t) \cdot w(\tau) \cdot e^{-j2\pi f \tau} \cdot d\tau \qquad (2.17)$$

Donde $x(\tau)$ es la señal de voz y $w(\tau)$ representa la función de la ventana de análisis (por ejemplo, la ventana Hamming). De esta forma se realiza un análisis localizado de la señal mediante la aplicación de una ventana $w(\tau)$ a la señal, alrededor del instante de tiempo *"t"*, analizada a todas las frecuencias consideradas *"f"*. (Mahanad, 2005)

En la siguiente sección se muestran dos aplicaciones de las técnicas de parametrización basadas en el espectro de Fourier: MFCC y LFCC

II.4.3.1 Coeficientes Cepstrales de Frecuencia Mel: MFCC.

Una familia de coeficientes directamente relacionada con los LPCC son los llamados mel-cepstrum o MFCC (Mel-Frequency Cepstrum Coefficients), los cuales son de gran utilidad en la extracción de los parámetros de la señal de voz, ya que están basados en la variación conocida de los anchos de banda de las frecuencias críticas del oído. Los filtros que se le aplican a la señal en la técnica MFCC están espaciados linealmente para frecuencias menores a 1000 Hz y logarítmicamente para frecuencias mayores de 1000 Hz, con el fin de capturar las características fonéticamente importantes del habla (Do, M). A esta escala se le denomina "Escala Mel" y su fórmula matemática se describe en la ecuación 2.18

$$Mel(f) = 2595 \cdot \log\left(1 + \frac{f}{700}\right) \qquad (2.18)$$

En la figura 14 se muestra la correspondencia que existe entre la frecuencia en Hz y la escala de frecuencia mel

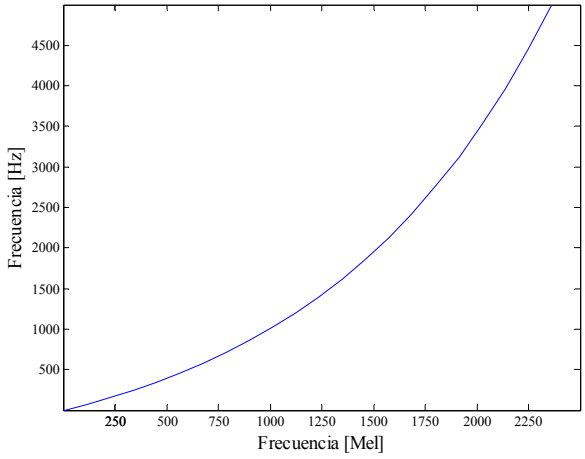


Figura 14. Correspondencia entre la frecuencia en Hz y la frecuencia mel

Los pasos necesarios para el cálculo de los MFCC's se muestra en la figura 15

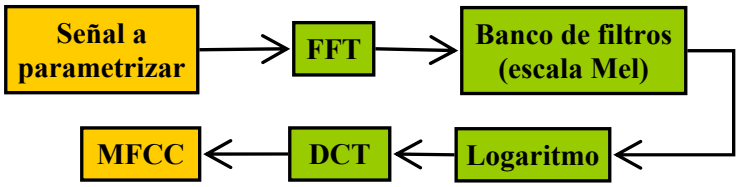


Figura 15. Diagrama de bloques para el cálculo de los MFCC's

Cada una de las fases mostradas en el diagrama de la figura 15, se describen a continuación

1) Se calcula la Transformada de Fourier de Tiempo Corto $X(n,\omega_k)$ a cada una de las tramas obtenidas de la etapa de pre-procesamiento mediante la ecuación 2.19

$$X(n,\omega_k) = \sum_{m=-\infty}^{\infty} x(m)\cdot w(n-m)\cdot e^{-j\omega_k m} \qquad (2.19)$$

Donde: $\omega_k = \frac{2\pi}{N}\cdot k$

2) El cuadrado de la magnitud de $X(n,\omega_k)$ es ponderado por una serie de filtros distribuidos sobre la escala Mel para luego calcular la llamada "log-energía" del filtro l-ésimo mediante la ecuación 2.20

$$E_{Mel}(n,l)=\frac{1}{A_l}\cdot\sum_{k=L_l}^{U_l}\left|V_L(\omega_K)\cdot X(n,\omega_K)\right|^2 \qquad (2.20)$$

Donde L_l y U_l son las frecuencias de corte inferior y superior del filtro l-ésimo.

El banco de filtros linealmente espaciado en la escala Mel tiene la forma que se muestra en la figura 16 y los filtros que lo conforman pueden ser triangulares o tener otras formas, tales como Hamming, Hanning o Kaizer, pero el triangular es el más utilizado.

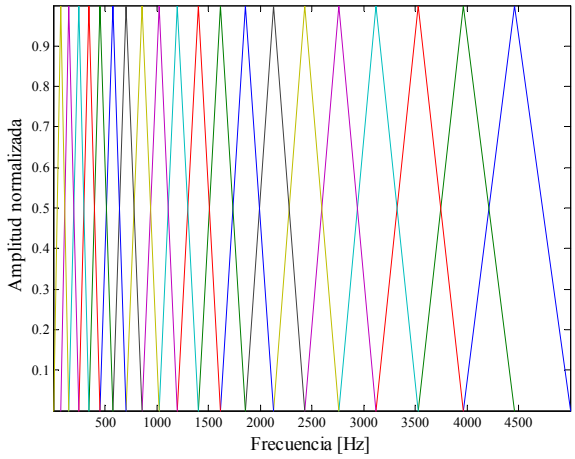


Figura 16. Banco de filtros espaciados linealmente en la escala de frecuencia Mel

3) Finalmente se convierte el espectro logarítmico Mel nuevamente al dominio del tiempo usando la Transformada Discreta de Coseno, dado que los coeficientes cepstrales son números reales. El cálculo de estos coeficientes se realiza mediante la ecuación 2.21

$$C_{Mel}[n,m]=\frac{1}{R}\cdot\sum_{l=1}^{R}\log\left\{E_{Mel}(n,l)\right\}\cdot\cos\left[n\left(l-\frac{1}{2}\right)\cdot\frac{\pi}{l}\right], \qquad (2.21)$$

$$n = 1, 2, 3....K$$

Donde K es el número de coeficientes cepstrales, que por lo general se escoge entre 10 y 20.

Mediante el proceso descrito anteriormente, para cada trama de voz de duración aproximada igual a 30 ms con solapamiento, se calcula un conjunto de coeficientes cepstrales. Este es el resultado de la Transformada Discreta de Coseno de la Densidad Espectral de Potencia expresada en la escala mel. A este conjunto de coeficientes se le denomina *Vector Acústico*, por lo que cada entrada es transformada mediante este proceso en una secuencia de Vectores acústicos que representan las características más importantes de la voz, necesarias para el proceso de reconocimiento del habla. Un ejemplo del vector acústico generado por el algoritmo que calcula los MFCC se muestra en la figura 17, para el cual se utilizó un banco de 20 filtros

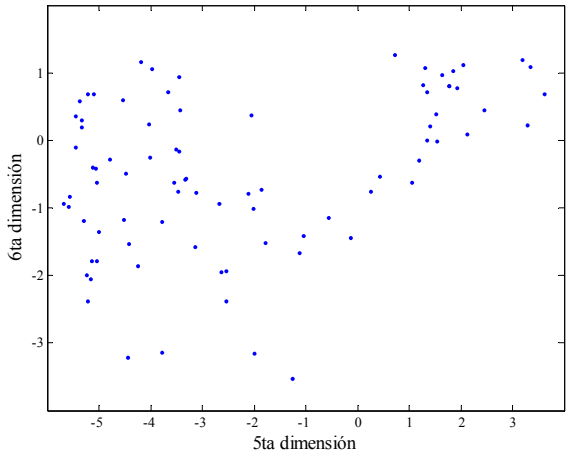


Figura 17. Vector acústico generado mediante el cálculo de los MFCC's

II.4.3.2 Coeficientes Cepstrales de Frecuencia Lineal: LFCC.

El esquema bajo el cual funciona esta técnica es similar al mostrado en la figura 15, la única diferencia es que los filtros que conforman el banco de filtros que se le

aplica a la señal, se encuentran sobre una escala lineal, es decir: $f = f_{Mel}$. El banco de filtros que se genera con esta técnica se muestra en la figura 18

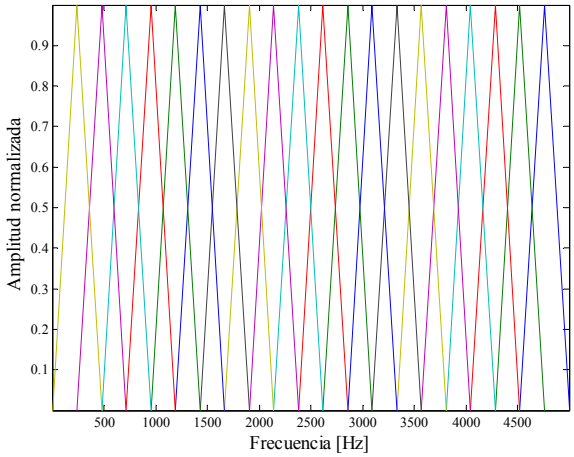


Figura 18. Banco de filtros generado para el cálculo de los LFCC's

Por su parte, el Vector acústico generado por el algoritmo que calcula los coeficientes ceptrales mediante este método muestra similitudes al derivado del cálculo de los MFCC's y se observa en la figura 19

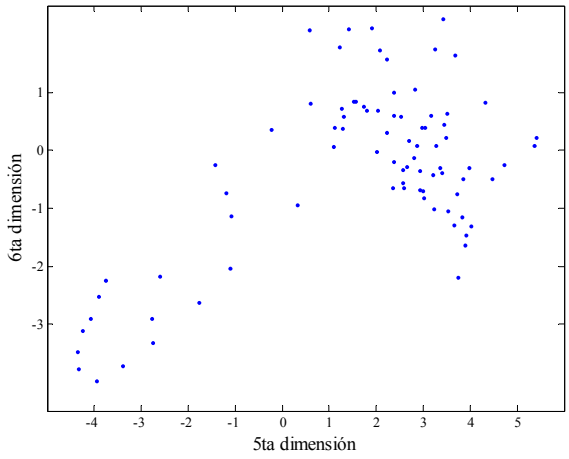


Figura 19. Vector acústico generado por el cálculo de los LFCC's

A continuación se muestra en la figura 20 los vectores acústicos generados mediante el cálculo de los MFCC's, LPCC's y LFCC's

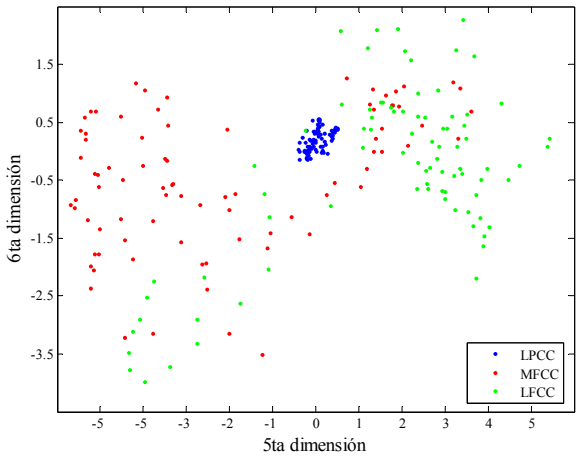


Figura 20. Vectores acústicos generados mediante el cálculo de los MFCC's, LPCC's y LFCC's

Se puede observar en la figura anterior que los vectores acústicos correspondientes a los MFCC's y los LFCC's presentan diferencias en cuanto a su ubicación, pero el vector acústico de los LPCC's se destaca, ya que se encuentra restringido a los valores entre 0 y 1 por características propias del algoritmo. Cabe destacar que la diferencia entre ellos se debe principalmente a que cada uno se utiliza para la extracción de distintas características de la voz

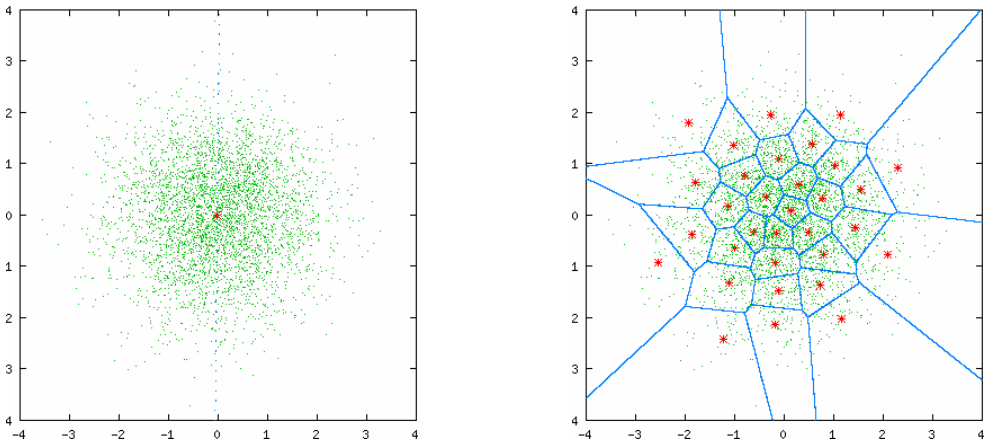
II.5 Reconocimiento del habla.

Las técnicas de parametrización explicadas en la sección II.4 tienen como finalidad generar una serie de coeficientes que representan las características de la señal de voz, que pueden ser usadas en la fase de reconocimiento del habla y que no se obtienen mediante un análisis temporal o frecuencial. El tamaño de la matriz obtenida del proceso de parametrización depende directamente de la longitud (variable) de la señal de voz, la cual tiene relación con la palabra en sí y el hablante. Por esta razón, se hace necesaria la estandarización de la matriz que contiene los coeficientes cepstrales calculados, para que el tamaño de las matrices usadas para el reconocimiento del habla sea el mismo.

La estandarización de la matriz de coeficientes cepstrales constituye el primer paso a realizar en el proceso de reconocimiento del habla y se denomina *Cuantización Vectorial.* El paso siguiente corresponde al cálculo de la diferencia entre la señal de voz del hablante y las señales que se encuentran en la base de datos de entrenamiento del sistema; dicha diferencia se obtiene mediante el cálculo de la *Distancia Euclidiana* en varias dimensiones. Cada uno de estos procedimientos se explica en las secciones siguientes.

II.5.1 Cuantización vectorial

La esencia de la cuantización vectorial, aplicada al caso particular del reconocimiento del habla, es la de obtener a partir de una matriz cualquiera de coeficientes cepstrales, una matriz de tamaño fijo que se parezca lo más posible a la original. Para ello, el espacio generado por los coeficientes cepstrales es dividido en un conjunto de regiones convexas mutuamente excluyentes y para cada una se calcula el *centroide,* que en dos dimensiones se representa como un punto que se encuentra a la menor distancia de todos los puntos (coeficientes cepstrales) que pertenecen a esa región. El conjunto de centroides obtenidos de la partición del espacio generado por los coeficientes cepstrales se denomina *codebook.* Por lo tanto, para caracterizar cada palabra se calcula su codebook correspondiente (Faúndez y Rodríguez, 2005). En la figura 21 se representan dos etapas del cálculo de los centroides (Extraída desde: http://www.data-compression.com/vq.shtml)



(a) *(b)*

Figura 21. División del espacio generado por los coeficientes cepstrales de la vocal "a": *(a)* Primer centroide generado, *(b)* Codebook obtenido

En la figura 21a se observa la ubicación del primer centroide calculado, mientras que en la figura 21b se aprecia el codebook correspondiente a todos los centroides generados mediante la Cuantización Vectorial. Uno de los algoritmos utilizados para la obtención del codebook es el LBG (Linde, Buzo and Gray), el cual se describe como un procedimiento recursivo que sigue los pasos que se describen a continuación (Do, M).

1. Diseñar un vector genérico del codebook, éste será el centroide del conjunto entero de vectores de entrenamiento.

2. Duplicar el tamaño del codebook dividiendo cada codebook existente de acuerdo a la ecuación 2.20

$$y_n^+ = y_n(1+\varepsilon)$$
$$y_n^- = y_n(1-\varepsilon) \qquad (2.20)$$

Donde *n* varía desde 1 hasta el tamaño actual del codebook, y ε es el parámetro de separación (normalmente se elige ε =0,01).

3. Buscar el vecino más cercano: Para cada vector de entrenamiento se debe buscar el codeword en el codebook actual más cercano (en términos de distancia métrica), y asignarlo a la correspondiente celda (asociada con el codeword más cercano).

4. Actualización del centroide: actualizar el codeword en cada celda usando el centroide del vector de entrenamiento asignado a esa celda.

5. Iteración 1: Repetir los pasos 3 y 4 hasta que la distancia media supere el presente umbral.

6. Iteración 2: Repetir los pasos 2,3 y 4 hasta que el tamaño del codebook haya alcanzado el tamaño designado.

En resumen, el algoritmo LBG crea una matriz codebook en varias etapas. Se inicia creando un vector genérico y luego empleando una técnica de separación de los codewords, se inicia la búsqueda del nuevo codebook. Manteniendo este proceso hasta que el tamaño del codebook sea el designado (Ejemplo: 16 centroides), se obtiene la matriz correspondiente al codebook de dicha palabra.

La finalidad de la Cuantización Vectorial es la de obtener una matriz que represente de la manera más exacta a la matriz original y que además permita hacer una comparación razonable entre la señal parametrizada del hablante y las señales contenidas en la base de datos del sistema de reconocimiento del habla, ya que todas las matrices tienen el mismo tamaño y es posible emplear un técnica como la distancia euclidiana para determinar la diferencia entre las mismas y realizar el proceso de identificación de la palabra.

II.5.2 Distancia euclidiana

La distancia euclidiana se emplea, para el reconocimiento del habla, como método para calcular la diferencia existente entre el codebook obtenido de la palabra dicha por el hablante y el resto de codebooks almacenados en la base de datos de entrenamiento del sistema. El resultado final de dicha comparación es un valor numérico que representa la distancia entre dos matrices de iguales dimensiones. El codebook perteneciente a la base de datos de entrenamiento del sistema cuya distancia al codebook generado para representar la palabra dicha por el hablante sea menor que el nivel de comparación establecido por el programador de la aplicación, identifica la palabra con la que existe mayor semejanza.

La fórmula matemática empleada para el cálculo de la distancia euclidiana en dos o más dimensiones se muestra en la ecuación 2.21

$$d(v,w) = \sqrt{(v_1 - w_1)^2 + (v_2 - w_2)^2 + (v_3 - w_3)^2 + ... + (v_n - w_n)^2} \qquad (2.21)$$

Donde $v = [v_1\ v_2\ v_3 .... v_n]$ y $w = [w_1\ w_2\ w_3 .... w_n]$ son matrices de longitud n

Se acostumbra a emplear dicha métrica para el cálculo de la distancia entre dos puntos, pero su aplicación va mas allá, permitiendo calcular la distancia entre dos matrices (Enciclopedia Wikipedia, 2006)

Capítulo III
Marco Metodológico

Las fases en la metodología que se siguieron para el desarrollo e implementación del sistema de reconocimiento del habla para controlar dispositivos eléctricos se describen a continuación:

III.1 Recopilación de información y realización de pruebas con señales de voz.

La búsqueda y recopilación de información realizada se basó en una consulta bibliográfica en sitios de Internet, así como en artículos de la IEEE Transactions on Acoustics, Speech, and Signal Processing y bibliografía especializada en el procesamiento de señales y sistemas como el libro Signals and Systems de Alan V. Oppenheim, entre otros.

Los aspectos estudiados en esta fase se basan en el esquema que se muestra en la figura 22

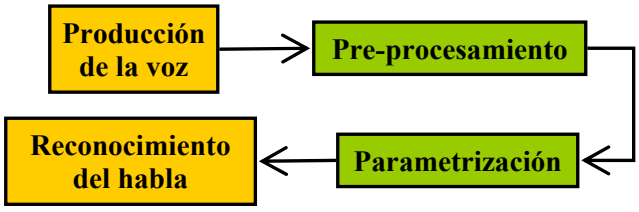


Figura 22. Bloques del sistema de reconocimiento del habla

Para comprender el proceso de producción de la señal de voz, fue necesario investigar sobre su naturaleza y características más importantes. Además, se realizaron simulaciones en MATLAB para observar el comportamiento periódico o aleatorio de la señal de voz, dependiendo de su naturaleza (señal sonora o no sonora).

Otro tema investigado en esta fase fue el modelo de producción de la señal de voz, ya que éste permite no sólo simular el comportamiento del tracto vocal como un filtro variable en el tiempo, sino que proporciona información sobre unos coeficientes o parámetros que caracterizan la señal de voz y que pueden ser usados para establecer criterios de comparación para la fase del reconocimiento.

Por su parte, luego de conocer la naturaleza y características de la señal de voz, la etapa de pre-procesamiento es de gran importancia en el esquema planteado, ya que mediante ella se le da un tratamiento a la señal de voz para resaltar sus características más importantes y prepararla para la etapa de parametrización. En esta fase de la investigación se recopiló información y se realizaron simulaciones en MATLAB sobre: la etapa de preénfasis y el filtro a utilizar para la misma, el endpoint detection como herramienta para detectar el comienzo y final de la información útil de un segmento grabado de señal de voz, la segmentación en tramas de la voz para poder realizar el análisis de Fourier en tiempo corto y la aplicación de la ventana cuya forma representa el equilibrio necesario para obtener la mejor resolución en tiempo y frecuencia posible.

Una vez obtenida la información sobre la fase de pre-procesamiento de la señal, se estudiaron los beneficios de realizar el análisis cepstral a la misma para caracterizar la forma del tracto vocal mediante una serie de pocos coeficientes que permiten parametrizar la señal de voz, extrayendo características que no son apreciables a través de un análisis en tiempo o en frecuencia. Por su parte, entre las representaciones paramétricas de la señal de voz, basadas en su análisis cepstral, se estudiaron dos grupos: aquellas basadas en la predicción lineal del espectro (LPCC) y aquellas basadas en el espectro de Fourier (MFCC y LFCC), cada una de las cuales se analizaron con mayor profundidad en la etapa de comparación entre las técnicas de parametrización mostrada en la sección III.4

III.2 Diseño de la aplicación encargada del reconocimiento del habla.

El código desarrollado para la programación de la aplicación encargada del reconocimiento del habla (ver Apéndice B) está conformado por un bloque principal denominado “POcoNE", desde el cual se hacen llamadas a varios bloques secundarios conformados por funciones que realizan las tareas de pre-procesamiento, parametrización y reconocimiento de la señal de voz. Esto se hace con el fin de no sobrecargar el bloque principal con un código muy largo y tener cada bloque del sistema de reconocimiento por separado para así hacer más fácil la detección de errores en los algoritmos.

Cabe destacar que para el desarrollo de la aplicación encargada del reconocimiento del habla se utilizaron funciones de MATLAB (las cuales se nombran según sea el caso) y también se crearon funciones propias. Para el bloque principal de la aplicación, la primera acción que se ejecuta es el entrenamiento del sistema mediante la función *train*, la cual lee todas las señales de voz guardadas en la base de datos de entrenamiento del sistema y les aplica los algoritmos correspondientes al pre-procesamiento y parametrización, los cuales serán explicados con más detalle en la sección III.2.1. La finalidad de hacer el entrenamiento del sistema cuando se inicializa la aplicación es que no se tenga que realizar esta acción cada vez que el usuario quiera probar el sistema.

El menú principal contenido en este bloque se basa en tres opciones, enumeradas del 1 al 3, donde el usuario puede escoger entre:

1. **Agregar:** Le proporciona al usuario la opción de grabar las señales de voz correspondientes a los comandos a,e,i,o,u y agregarlas a la base de datos de entrenamiento del sistema. Para agregar esta nueva entrada a la base de datos se creó una función denominada *addnew*, la cual se encarga de pedirle al usuario que

grabe las señales de voz, calcula la energía promedio de la señal mediante la función *calenergia,* para determinar si la señal tiene la energía suficiente para formar parte de la base de datos y guarda la señal grabada como un archivo en formato .wav.

2. **Reconocer:** Permite al usuario probar el sistema de reconocimiento al comparar el comando de voz que éste dice en el momento que escoge la opción, con la información guardada en la base de datos con la que se entrena al sistema. Para este fin, se creó una función llamada *test*, la cual a su vez hace una llamada a la función encargada del pre-procesamiento y parametrización de la señal de voz y hace la comparación entre la señal de voz dicha por el usuario y las guardadas en la base de datos del sistema mediante una función llamada *disteu*, cuyo funcionamiento será explicado con más detalle en la sección III.2.2

3. **Salir:** Opción para salir de la aplicación de reconocimiento del habla.

El funcionamiento y estructura de cada una de las principales funciones creadas para realizar las tareas de pre-procesamiento, parametrización y reconocimiento se describirá a continuación. Para las dos primeras tareas, se creó una función llamada *parametrizar* y para la etapa de reconocimiento se creó la función *test*.

La función *parametrizar* recibe como principales parámetros de entrada: la señal de voz *"s",* la frecuencia de muestreo *"fs"*, la longitud *"n"* de la trama, la longitud *"m"* del solapamiento a utilizar para la segmentación, el número *"p"* de filtros (para las técnicas MFCC y LFCC) o polos (para el caso de LPCC) a utilizar para la parametrización y la variable *"cual"*, que define la técnica de parametrización (MFCC, LFCC o LPCC). El funcionamiento detallado de cada uno de los bloques de esta función se explica a continuación

III.2.1 Pre-Procesamiento

Tal y como se mencionó en el Capítulo II y se muestra en la figura 23, la etapa de pre-procesamiento de la señal de voz se basa en el esquema siguiente

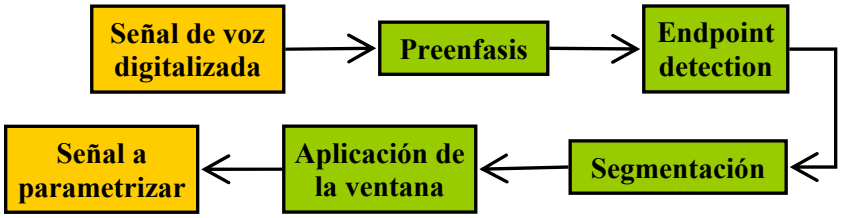


Figura 23. Esquema de pre-procesamiento de la señal de voz

- *Preénfasis:* Se realizó usando una función denominada *prenfasis,* en la que se le aplica un filtro pasaalto (con función de transferencia H(z) como la que se describe en la ecuación 2.1) a la señal de voz con la finalidad de prepararla para la detección de la información útil mediante el endpoint detection. Dicho filtro se genera en MATLAB mediante el uso de la función *filter*, para la que se definió un valor de 0,5 al coeficiente.

- *Endpoint detection:* La finalidad de esta etapa es la de eliminar los silencios usualmente presentes al comienzo y final de la señal de voz que graba el usuario. Se realizó mediante la función *energia* que divide la señal de voz en tramas (sin solapamiento), calcula la energía de cada una de las mismas y guarda esos valores en un vector E. Luego se establece un valor de comparación, que representa un porcentaje de la energía de la señal por encima del cual se considera que ésta es suficiente como para tomar la decisión de donde empieza y termina la información útil de la señal grabada, es decir, se determina el intervalo donde la energía de la señal supera el valor de comparación establecido y se extrae ese intervalo de la señal de voz original (sin preénfasis), para guardarlo como información útil.

- *Segmentación:* Se realizó mediante la función *framing* que divide la señal en tramas de 256 muestras (que representan 25,6 mseg de voz a una frecuencia de muestreo de 10 KHz) para considerar la señal de voz como estacionaria en ese intervalo de tiempo y crea una matriz *"M"* que tiene en cada una de sus columnas las muestras correspondientes a cada trama en la que se dividió la señal de voz. Para la segmentación se utilizó un solapamiento entre tramas consecutivas de 156 muestras.

- *Aplicación de la ventana Hamming:* Una vez obtenida la matriz M de la etapa de segmentación, mediante la función *windowing* se creó una matriz h que en su diagonal contiene los valores de una ventana Hamming y que multiplicada por M, devuelve una matriz "*M2"* que corresponde a la ventana aplicada a cada una de las tramas contenidas en M.

Una vez que ya se tienen las tramas correspondientes a la señal de voz con la ventana aplicada a cada una de ellas, la parametrización es el paso siguiente en el esquema de reconocimiento del habla y su implementación se muestra en la sección III.2.2.

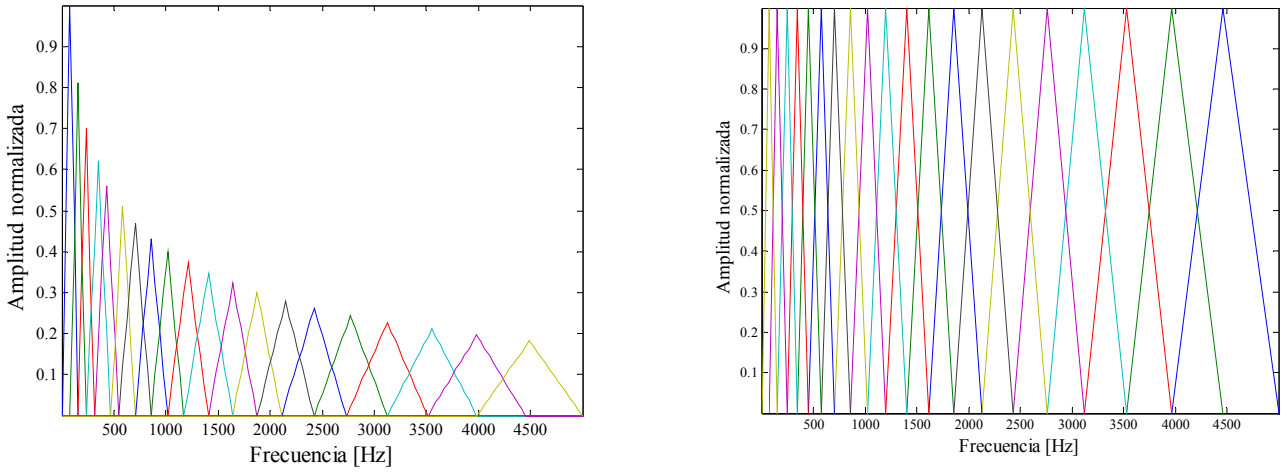
III.2.2 Parametrización.

El bloque preliminar de la aplicación, correspondiente a la parametrización de la señal de voz, se basa en el desarrollo de tres algoritmos básicos que representan cada una de las técnicas de parametrización estudiadas, ya que esta fase corresponde a las pruebas que se realizaron con cada una de ellas y no a la escogencia de un método en particular. La estructura de los algoritmos correspondientes a las técnicas MFCC y LFCC es similar, siguiendo el esquema que se muestra en la figura 15, mientras que para el LPCC la estructura es totalmente diferente. La explicación del funcionamiento de cada uno de los algoritmos desarrollados se muestra a continuación

III.2.2.1 Coeficientes Cesptrales de Frecuencia Mel y Lineal

- *Cálculo de la FFT:* Mediante la función *fftframe* se calcula la Transformada de Fourier a cada columna de la matriz M2 (que corresponde a la señal de salida de la etapa de pre-procesamiento), obteniéndose la matriz *"fframe"* que contiene a la FFT de cada trama. Esto se hace con el fin de realizar el análisis de Fourier de tiempo corto y para poder aprovechar las características estacionarias de la voz en intervalos cortos de tiempo.

- *Creación del Banco de filtros:* La creación del banco de filtros se realiza mediante la función *filterbank*, la cual tiene la opción de crear un banco de filtros separados linealmente en la escala Mel (MFCC) o separados linealmente en frecuencia (LFCC). Para el primer caso, se crea una escala de frecuencias Mel de acuerdo a la ecuación 2.18 y se calcula cada una de las frecuencias en las cuales se centrarán los filtros del banco. Para el segundo caso, las frecuencias centrales de los filtros se calculan mediante una correspondencia directa y no logarítmica. Para ambos casos se utilizó un banco de 20 filtros, ya que se determinó que para esta aplicación, 20 coeficientes o filtros es un número que establece un equilibrio entre la rapidez del algoritmo y la calidad de la parametrización. Luego de que la escala de frecuencias está establecida, se crea el "Banco de filtros triangulares de escala Mel" o "Banco de filtros triangulares de escala Lineal" según sea el caso, además se incluyó la opción de que la amplitud de los filtros pueda tener caída logarítmica o mantenerse constante (por razones de pruebas). La función descrita devuelve una matriz *"fb"* de 20 filas x 129 columnas, correspondiente al banco de filtros. La representación gráfica de esta matriz para cada caso se muestra en la figura 24



(a) *(b)*

Figura 24. Banco de Filtros triangulares: *(a)* Con caída logarítmica en amplitud, *(b)* Con amplitud constante

- *Cálculo de la Densidad Espectral de Potencia y aplicación del banco de Filtros a la señal:* Este corresponde al siguiente paso en el cálculo de los MFCC's o LFCC's y se hace determinando la Densidad Espectral de Potencia como el cuadrado de la magnitud de la matriz obtenida del cálculo de la FFT de *fframe*, y multiplicando el resultado de este cálculo por el banco de filtros *fb*, de lo cual se obtiene la matriz *"z"*. La razón para utilizar el cepstrum real como herramienta para el cálculo de los coeficientes cepstrales, radica en el hecho de que ésta representa una forma más fácil y rápida de parametrizar la señal de voz, ya que, aunque se pierde información sobre la fase de la señal y, a partir de estos coeficientes no se puede recuperar la señal de voz, se obtienen características de la misma que son especialmente útiles para las posteriores tareas de reconocimiento. Además es importante resaltar que el filtro generado no tiene fase, por lo que es imprescindible que se trabaje en este bloque con la densidad espectral de potencia (para asegurar que la FFT de la señal de voz sólo tenga componente real).

- *Cálculo de la Transformada Discreta de Coseno (DCT):* Este cálculo corresponde a pasar toda la información obtenida de la etapa previa, al dominio del tiempo, lo cual es un paso fundamental para la obtención de los coeficientes cepstrales. En este caso se aplica la DCT, que representa la parte real de la Transformada Inversa de Fourier, ya que la matriz "z" es totalmente real en el dominio de la frecuencia y mediante la DCT se lleva al dominio del tiempo.

III.2.2.2 Predicción Lineal de los Coeficientes Cepstrales: LPCC

El cálculo de los LPCC's se realiza dentro de la función *parametrizar* en un bloque que se basa la ecuación 2.16, en la cual se puede observar que, partiendo del cálculo de los LPC's de una señal, se pueden obtener sus LPCC's. Usando la función *lpc* de MATLAB se calcularon los Coeficientes de Predicción Lineal a la matriz "*M2"* y luego, a partir de esta matriz *"a"* se realizó el cálculo de los LPCC's, los cuales se guardaron en la matriz *"v"*.

III.2.3 Reconocimiento del habla

La etapa de reconocimiento del habla se realiza dentro de la función *test*, en la cual se realiza la comparación entre la señal de voz dicha por el usuario y las señales pertenecientes a la base de datos de entrenamiento del sistema. Dicha función basa su funcionamiento en dos grandes bloques:

- *Cuantización vectorial:* Este proceso corresponde a la estandarización de la matriz generada por la función encargada de la parametrización de la señal y se realiza mediante el uso de la función *VQLBG*, la cual sigue los pasos que se describen en la sección II.5.1 para el algoritmo LBG, utilizado para el cálculo de los centroides. La matriz correspondiente al codebook de la palabra parametrizada tiene 20 filas y 16 columnas, éstas últimas correspondientes al número de centroides calculados por el LBG. La idea fundamental de este proceso es la de

generar matrices que posean las mismas dimensiones, independientemente de la longitud de las señales de voz grabadas, con el fin emplear la distancia euclidiana entre matrices de igual tamaño para cuantificar la diferencia entre las señales comparadas.

- *Distancia euclidiana:* Este cálculo se realiza mediante el uso de la función *disteu* y representa el último paso para el reconocimiento de la señal de voz. Dicha función devuelve un valor numérico que representa la menor distancia entre la voz del hablante parametrizada y la señal de voz guardada en la base de datos de entrenamiento del sistema que más se parece a la primera. Normalmente las distancias obtenidas cuando existe gran semejanza entre las palabras comparadas es pequeño, por lo que para evitar falsos reconocimientos debidos a ruido u otros factores, se estableció un nivel de comparación igual a 6, por debajo del cual se considera que el reconocimiento ha sido exitoso.

III.3 Recolección de señales de voz

La recolección de las voces para el entrenamiento y pruebas del funcionamiento del sistema de reconocimiento del habla se hizo con un total de 20 personas, de las cuales 11 eran hombres con edades comprendidas entre 20 y 24 años y 9 eran mujeres con edades comprendidas entre 20 y 22 años. Dicha recolección se realizó en ambientes con diferentes niveles de ruido, por lo que se obtuvo una gran diversidad entre las señales de voz grabadas.

La metodología seguida para la creación de las bases de datos consistió en la grabación de las vocales desde la a hasta la u un total de tres veces: las primeras dos secuencias para agregarlas a la base de datos de entrenamiento del sistema y la tercera secuencia para agregar a la base de datos usada para simular el funcionamiento "en vivo" del sistema de reconocimiento.

Posterior al proceso de obtención de las señales de voz, se le aplicó a cada una de ellas el algoritmo correspondiente al endpoint detection para guardar en las bases de datos correspondientes, sólo la información útil de la señal de voz.

III.4 Pruebas comparativas entre los métodos de parametrización estudiados

El criterio establecido para realizar la comparación entre los diversos métodos de parametrización desarrollados fue el porcentaje de error que arroja cada uno en cuanto a la exactitud del reconocimiento de los vocablos. Para realizar esta comparación se utilizaron las señales obtenidas en la fase de recolección de señales de voz para entrenar al sistema y para simular el funcionamiento "en vivo" del mismo. La rapidez de los algoritmos no se tomó en cuenta para la escogencia de un método en particular porque la diferencia entre unos y otros no era tan notable como para que éste fuera un aspecto importante para la selección de uno de los métodos.

III.5 Hardware encargado del control de los dispositivos eléctricos

Se diseñó un hardware que, mediante la recepción de señales provenientes del PC, fuese capaz de realizar el encendido o apagado de los dispositivos eléctricos mediante relés controlados por transistores que se comportan como suiches, que se activan o desactivan de acuerdo a las instrucciones recibidas por un microcontrolador. Para la implementación de dicho hardware se siguieron varias fases de diseño, entre ellas: La programación de un microcontrolador que fuese capaz de comunicarse con el PC a través de su puerto serial empleando la interfaz EIA232, el circuito de encendido y apagado de los dispositivos eléctricos y el circuito impreso en el que se integran los diferentes módulos que conforman el hardware. Además, fue necesaria la implementación de una fuente regulada para alimentar al circuito de encendido y apagado de los dispositivos eléctricos.

III.5.1 Programación del microcontrolador

El papel del microcontrolador dentro del hardware encargado del control de los dispositivos eléctricos es el de interpretar las señales provenientes del puerto serial del PC y transformarlas en salidas que el circuito de encendido y apagado pueda entender. El modelo del microcontrolador empleado fue el PIC 16F873 (ver Anexo C o visitar: http://ww1.microchip.com/downloads/en/DeviceDoc/30292c.pdf para más información), el cual posee un módulo de comunicación serial y varios puertos de entrada y salida, de los cuales sólo 5 de ellos fueron empleados para controlar el encendido y apagado de los dispositivos eléctricos conectados al hardware.

El código desarrollado para la programación del PIC se muestra en el Apéndice A y se realizó en el ambiente de desarrollo MPLAB, siguiendo una estructura que se basa en los pasos que se describen a continuación:

1. **Rutina de inicialización:** Proporciona al PIC la configuración con la cual debe trabajar, es decir, se especifican los 5 puertos que funcionan como salidas (uno por cada comando de voz que maneja el sistema)

2. **Habilitación de interrupciones:** Se habilitan las interrupciones relacionadas con la recepción serial mediante las cuales se activa una bandera que produce la ejecución de cierta rutina una vez que el buffer de recepción serial está lleno.

3. **Revisión del contenido del búffer:** El PIC revisa el contenido del búffer de recepción serial y lo compara con los caracteres especificados para cada dispositivo, es decir, cada comando de voz tiene asociado a él una palabra de 8 bits que corresponde a la representación en formato ASCII de la vocal identificada por el sistema. Dependiendo del contenido que el PIC encuentra en el búffer, genera una salida que cambia el estado de un relé u otro.

4. **Rutina de chequeo de los dispositivos eléctricos:** Una vez que el microcontrolador identifica la palabra de 8 bits, determina si el dispositivo eléctrico está encendido o apagado verificando el valor de un registro donde se almacena la información del estado del dispositivo (inicialmente se asume que todos los dispositivos están apagados). Dependiendo del contenido de dicho registro, se activa o desactiva el puerto de salida asociado al dispositivo en cuestión mediante la puesta a 0 o 5 voltios del mismo (5 voltios significa encender y 0 voltios significa apagar)

Por su parte, las características configuradas en el PC para la comunicación serial fueron las siguientes: 8 bits de data, 1 bit de inicio, 1 bit de parada y una velocidad de transmisión de 9600 bps. Pero la diferencia entre los niveles de voltaje que maneja la interfaz serial del PC (± 12 voltios) y los que maneja el PIC (0 y 5 voltios) para representar el 0 y 1 lógico, hace necesario el empleo de un conversor HIN232 de TTL a RS232 (ver Anexo B ó visitar: http://info.hobbyengineering.com/specs/INTERSIL-HIN232.pdf para más información) para lograr la comunicación entre el PC y el microcontrolador.

III.5.2 Circuito de encendido y apagado de los dispositivos eléctricos

Un circuito aparte del que se implementó para el microcontrolador fue necesario para generar el suministro de corriente necesario para la polarización de los relés encargados del encendido y apagado de los dispositivos. Dicho circuito está basado en el funcionamiento de un transistor como suiche, un diodo y un relé, tal y como se muestra en la figura 25

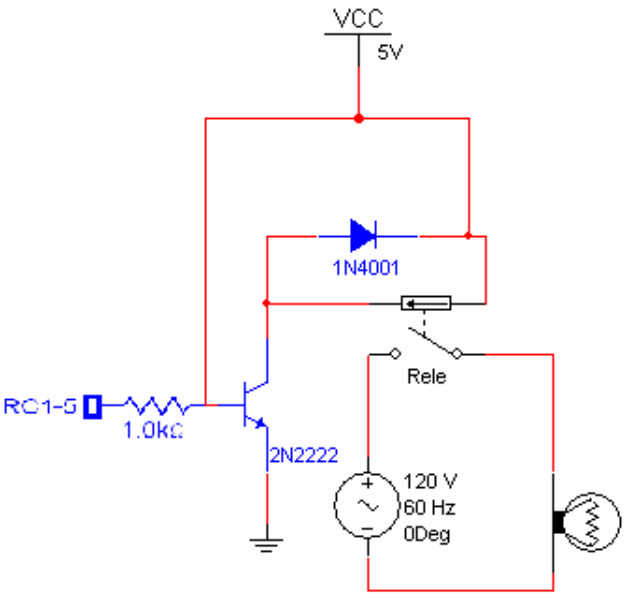


Figura 25. Diagrama circuital del módulo de encendido y apagado de los dispositivos eléctricos

El diagrama mostrado en la figura 25 representa la configuración circuital correspondiente a una de las salidas del microcontrolador, por lo que el módulo contiene cinco configuraciones similares, una por cada comando de voz que maneja el sistema. El funcionamiento de dicho circuito es el siguiente: Se polariza la base del transistor con 5 voltios y el colector cae a tierra. Por su parte, la diferencia de potencial en el diodo es de 5 voltios (provenientes directamente de la fuente regulada) y la corriente suministrada por éste es suficiente para cambiar el estado del relé y, por ende, encender el dispositivo eléctrico.

III.5.3 Circuito impreso para el hardware de control de los dispositivos eléctricos

El montaje del hardware para el control de los dispositivos eléctricos se realizó en una placa de cobre, sobre la cual se diseñó un circuito impreso mediante el uso del software PCBExpress. Se diseñó e implementó un circuito impreso en lugar de

utilizar un protoboard o baquelita de cobre para evitar el problema de la capacitancia parásita que usualmente se genera entre las diferentes líneas de éstos.

El circuito impreso incluye en su diseño una fuente de alimentación de 5 voltios necesaria para la polarización de los componentes del circuito de encendido y apagado, para prescindir de una fuente DC adicional. Además, en él se recogen todas las conexiones necesarias para la unión de cada uno de los módulos que confiorman el hardware para controlar los dispositivos eléctricos.

III.5.4 Componentes utilizados para la implementación del hardware

En la tabla 2 se recoge el listado de los componentes usados para la implementación del hardware encargado del control de los dispositivos eléctricos conectados a él

Cantidad	Componente		Cantidad	Componente
1	Condensador cerámico - 1000pF (500v)		1	PIC16F873 – 20/CP
1	Condensador electrolítico - 1000µF (16v)		1	Conversor TTL/232 - HIM232CP
1	Condensador poliester - 0,1µF (50v)		6	Condensador electrolítico - 1µF (50v)
1	Transformador - 120v / 8VAC @ 850mA		5	Diodo - 1NH42 1A / 1000v
1	Regulador de 5v - 7805		5	Transistor - 2N706
4	Diodos - 1N4004 - 1A / 400v		5	Relé - 5vDC - 2A/120v AC
2	Porta fusibles		6	Resistencia - 1K
1	Caja de fusibles - 1A		1	Resistencia - 10K
1	Pulsador ON/OFF		1	Cristal de cuarzo -20 MHZ
3	Regleta - 3 pin		2	Condensador cerámico - 22pF
7	Regleta - 2 pin		1	Zócalo - RJ45
1	Base P/IC - 28 pin		1	Conector - RJ45
1	Base P/IC - 18 pin		1	Conector - DB9 (hembra)

Tabla 2. Listado de componentes utilizados para la implementación del hardware

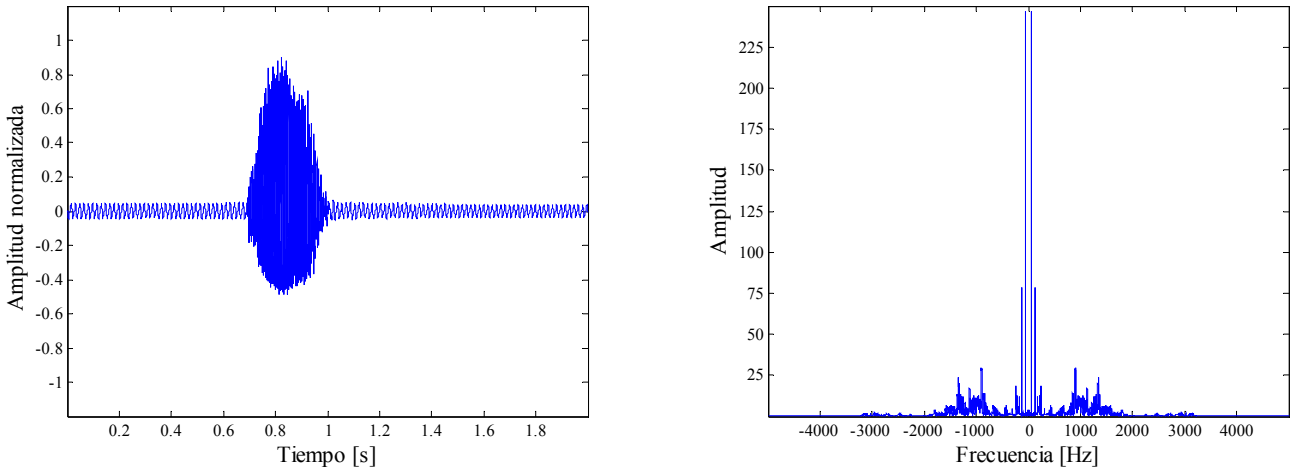
Capítulo IV

Resultados

Los resultados obtenidos de las fases seguidas en la metodología para el desarrollo e implementación del sistema de reconocimiento del habla para controlar dispositivos eléctricos se describen a continuación:

IV.1 Diseño en MATLAB de la aplicación encargada del reconocimiento del habla.

La estructura de la aplicación encargada del reconocimiento del habla se basa en el esquema mostrado en la figura 22. Para ejemplificar los resultados obtenidos de las etapas que conforman el esquema mencionado, se utilizó la señal de voz que se muestra en la figura 26.



(a) *(b)*

Figura 26. Señal de voz correspondiente a la letra "a": *(a)* En tiempo, *(b)* En frecuencia

Los resultados obtenidos de cada uno de los bloques correspondientes al esquema mencionado, partiendo desde la etapa de pre-procesamiento de la señal de voz, se muestran a continuación.

IV.1.1 Pre-Procesamiento

La primera etapa del pre-procesamiento corresponde a la aplicación del filtro de *preénfasis* a la señal de voz. El resultado de aplicar el filtro de preénfasis de la figura 11 a la señal que se usó como prueba para mostrar los resultados, se observa en la figura 27, donde se puede apreciar que el nivel del ruido disminuye.

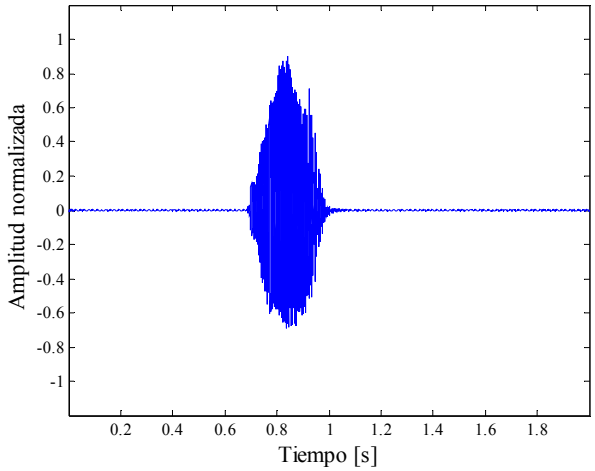


Figura 27. Representación en tiempo de la señal de voz con preénfasis

Es importante destacar que el filtro de preénfasis se usó en este proyecto con la finalidad única de reducir el nivel de ruido de la señal original para realizar la detección del inicio y fin de la información útil de la misma mediante el endpoint detection de una forma más precisa. Una vez obtenidos éstos, la señal que se truncó fue la original (figura 26a), debido a que la principal región de las formantes para las vocales se encuentra en las frecuencias bajas y no es conveniente eliminar esta información mediante el uso de un filtro pasaalto.

Obtenida la señal con el nivel de ruido reducido, se realiza el *endpoint detection* para detectar los extremos de la señal correspondientes al inicio y fin de la información útil. De esta forma se obtiene una señal libre de los silencios presentes en la señal original, tal y como se muestra en la figura 28

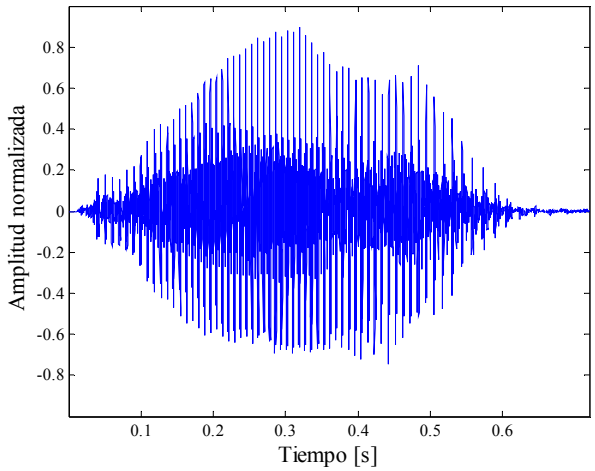
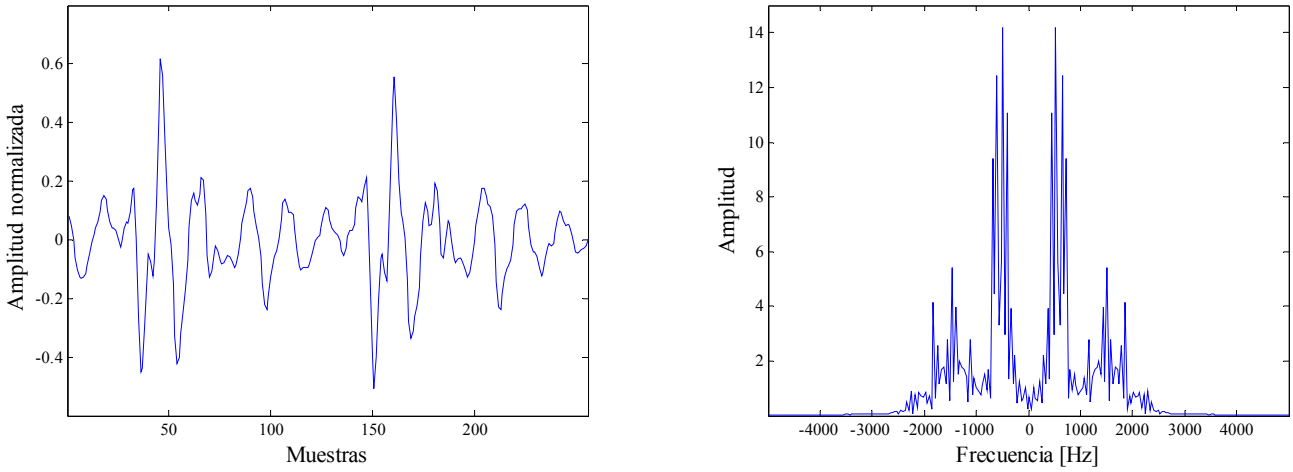


Figura 28. Señal de voz después de aplicada la técnica de endpoint detection

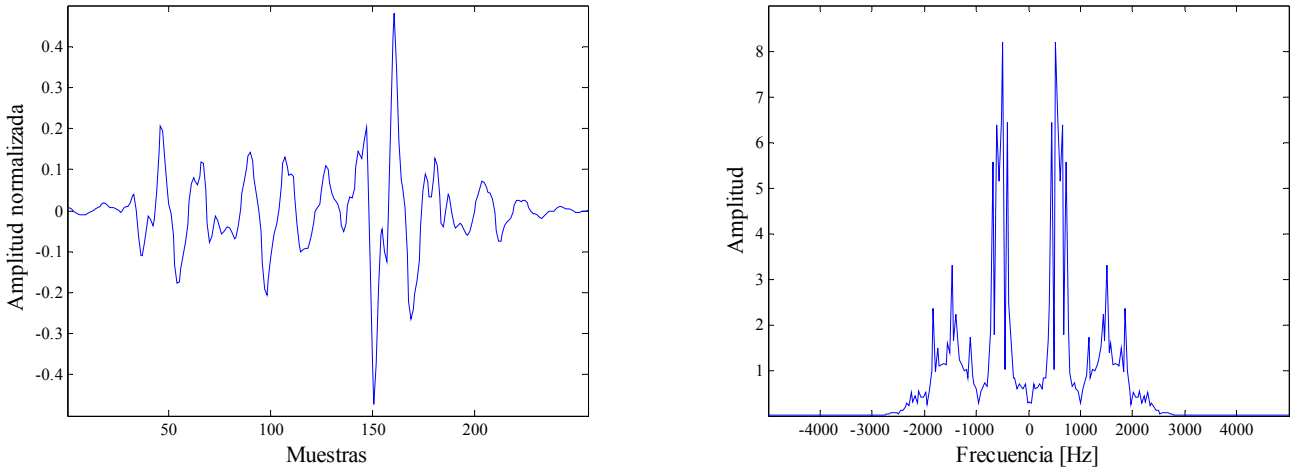
El paso siguiente a la obtención de la información útil de la señal es el de *segmentación*, la cual es importante para aprovechar que la señal es estacionaria en períodos cortos de tiempo, tal y como se explica en la sección II.3.3. Por su parte, la *escogencia y aplicación de la ventana* adecuada es muy importante, ya que esto define la resolución espectral que tendrá la señal, tal y como se puede apreciar en la figura 29 para el caso de la ventana rectangular.



(a) *(b)*

Figura 29. Representación de un segmento de la señal de voz con ventana rectangular: *(a)* En tiempo, *(b)* En frecuencia

Se puede observar de la figura 29 el patrón casi periódico que muestra este segmento de señal, ya que la misma corresponde a una señal sonora. Por su parte, la diferencia en cuanto a la resolución espectral de la señal se obtiene al comparar la figura anterior con la figura 30, en la que se muestra la representación de la ventana Hamming aplicada a la señal de voz.



(a) *(b)*

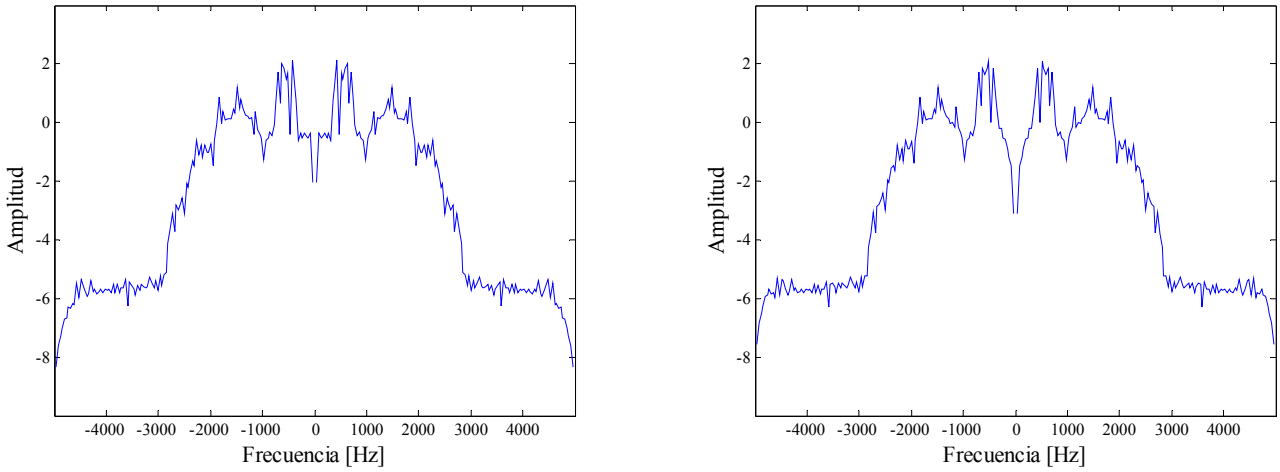
Figura 30. Representación de un segmento de la señal de voz con ventana Hamming: *(a)* En tiempo, *(b)* En frecuencia

Comparando las figuras 29 y 30 se puede observar que al aplicar la ventana Hamming se obtiene una mejor resolución espectral, ya que se logran capturar ciertas componentes espectrales de la señal que no se diferencian cuando se aplica la ventana rectangular. Esta mejora se debe al compromiso obtenido entre el ancho del lóbulo principal de la ventana y la atenuación de sus lóbulos secundarios, por lo que se escogió esta forma de ventana para la fase de pre-procesamiento de la señal de voz.

IV.1.2 Evaluación de las técnicas de parametrización estudiadas

Una vez obtenidas las tramas de voz mediante la segmentación y aplicación de la ventana, a cada una de ellas se le aplica el método de parametrización correspondiente. Los resultados que se muestran a continuación corresponden a pasar la trama que se muestra en la figura 30 por cada uno de los bloques correspondientes al esquema de parametrización de la señal de voz.

El primer paso para el cálculo de los MFCC's y LFCC's fue el de aplicar el banco de filtros triangular a la Transformada de Fourier de la trama de voz mostrada en la figura 30b. La diferencia entre los dos procedimientos radica sólo en la distribución en la escala de frecuencia de los filtros que conforman el banco. Luego se calcula el logaritmo del espectro obtenido, el cual se muestra en la figura 31

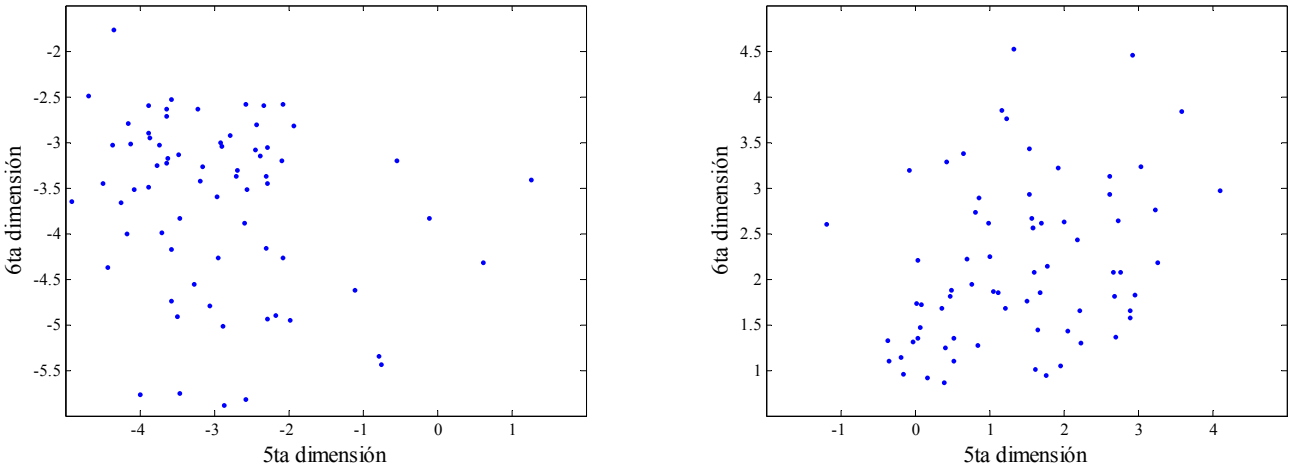


(a) *(b)*

Figura 31. Espectro logarítmico del banco de filtros aplicado a la señal: *(a)* Escala Mel, *(b)* Escala lineal

De la figura 31 se puede observar que entre ambos espectros existe semejanza, pero ellos a su vez difieren mucho del espectro de Fourier mostrado en la figura 30b. Esto es lo que permite obtener características que pueden usarse para establecer criterios de comparación para el reconocimiento de los vocablos.

Una vez obtenido el espectro logarítmico, para generar los MFCC's o LFCC's se le aplicó la Transformada Discreta de Coseno y se obtuvieron los resultados que se muestran en la figura 32



(a) *(b)*

Figura 32. Coeficientes cepstrales: *(a)* Escala Mel (MFCC), *(b)* Escala lineal (LFCC)

Las ubicación de los MFCC's y los LFCC's es la principal diferencia entre ellos, y el número de coeficientes depende de la cantidad *n* de tramas en las que se divide la señal de voz. Al realizar el procedimiento para el cálculo de los MFCC's o LFCC's se genera una matriz que tiene 20 filas y n columnas. Esto se puede decir que representa 20 "dimensiones" diferentes, donde cada dimensión representa el mismo coeficiente cepstral para las *n* tramas. Debido a que no se puede obtener una gráfica de 20 dimensiones, la figura 32 es la representación de dos de los coeficientes obtenidos para todas las tramas.

En la figura 33 se muestran los MFCC's calculados para la misma palabra dicha por la misma persona dos veces. Se puede observar que la ubicación de los puntos no es exactamente igual para las dos palabras parametrizadas, ya que la característica de aleatoriedad de la señal de voz hace que la primera señal difiera de la segunda en su representación temporal y frecuencial, a pesar de que fueron dichas por el mismo hablante. Pero, aunque los coeficientes de las dos palabras no son exactamente iguales, se aprecia que mantienen su ubicación en una zona cercana, lo que representa

esas características inherentes a la palabra que no se pueden obtener de una representación en tiempo o frecuencia

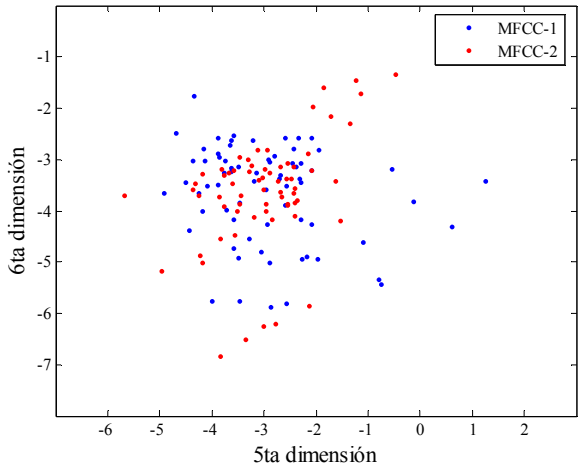


Figura 33. MFCC's de la vocal "a" dicha por el mismo hablante dos veces

Un comportamiento similar al que se observa en la figura 33 se presenta para el resto de las técnicas de parametrización, a pesar de las diferencias que existen en el cálculo de los coeficientes cepstrales para cada una de ellas. Esta diferencia es mayor para el caso del cálculo de los LPCC, ya que se calculan primero los coeficientes a_k del filtro de predicción lineal mediante la técnica del LPC y a partir de estos coeficientes se realiza la predicción del espectro de Fourier, la cual se muestra en la figura 34. Esto equivale al cálculo de la Transformada de Fourier de la trama con la que se está trabajando.

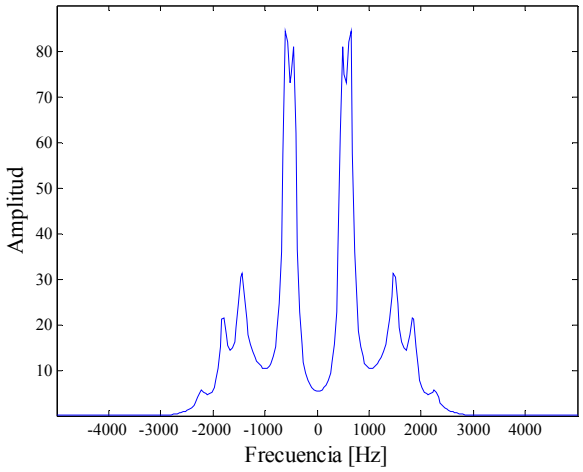


Figura 34. Predicción del espectro de Fourier de una trama de voz mediante la técnica del LPC

En este caso, el número de LPCC's depende directamente del número de polos que tiene el filtro de predicción lineal, que para el proyecto implementado es de 20. El espectro de Fourier obtenido representa la envolvente de las principales formantes que componen la señal de voz, tal y como se mencionó en la sección II.1.

Luego, como paso siguiente a la obtención del espectro de Fourier, se obtiene el espectro logarítmico de la señal, el cual se muestra en la figura 35. Se trabaja con la magnitud del espectro de Fourier para obtener el cepstrum real, ya que su implementación es más sencilla que la del cepstrum complejo, además de que la fase de la señal no es importante para la aplicación desarrollada por las razones que se exponen en la sección II.4.1

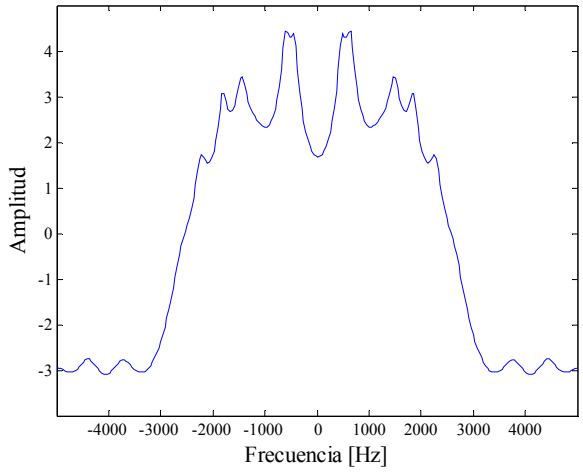


Figura 35. Espectro logarítmico de una trama de voz

Para obtener los LPCC's, se le aplica la Transformada Discreta de Coseno al espectro logarítmico de la señal en la figura 35. Estos cálculos se resumen en la ecuación 2.8, donde se obtienen los LPCC's a partir de los coeficientes a_k obtenidos de la aplicación de la técnica de predicción lineal a la señal de voz. La representación gráfica de dos de los coeficientes calculados para la señal de voz se muestra en la figura 36

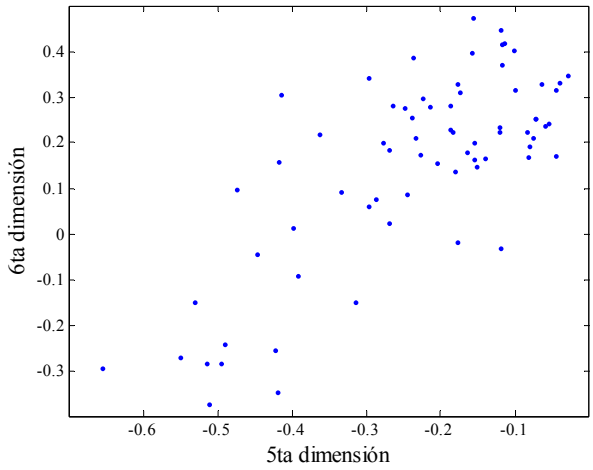


Figura 36. Representación de dos de los LPCC's calculados a la señal de voz

Se puede apreciar de la figura 36 que los LPCC's difieren de los MFCC's y los LFCC's por su ubicación y por los valores en magnitud que poseen. Esto no quiere decir nada sobre su desempeño en la etapa de reconocimiento, simplemente es una forma diferente de parametrizar la señal de voz.

De acuerdo a los métodos utilizados para el cálculo de los coeficientes cepstrales para cada una de las técnicas de parametrización, se obtienen representaciones diferentes de la misma señal de voz, tal y como se observa en la figura 37

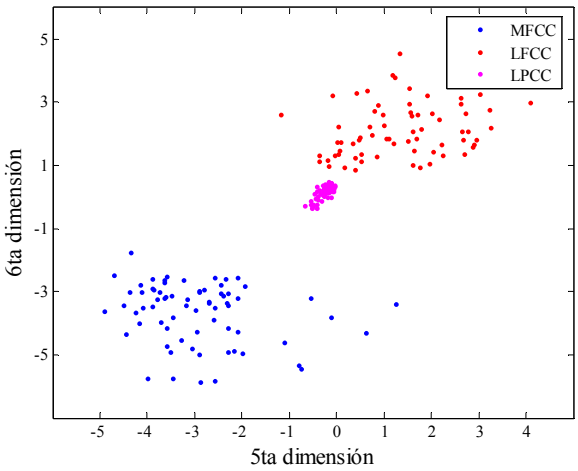


Figura 37. Representación paramétrica de la vocal "a" mediante MFCC, LFCC y LPCC

La ubicación de los puntos en la figura 37, representan la parametrización de la señal de voz mediante las diferentes técnicas y son diferentes entre sí porque cada uno extrae características diferentes de la señal de voz, que luego serán utilizadas en la fase de reconocimiento.

IV.1.3 Reconocimiento del habla

La señal que se obtiene del proceso de parametrización pasa por el proceso de cuantización vectorial ó cálculo de los VQ's, proceso que se lleva a cabo mediante el

algoritmo descrito en la sección III.3. La representación gráfica de los VQ's calculados a los MFCC's de la figura 32a se muestra en la figura 38

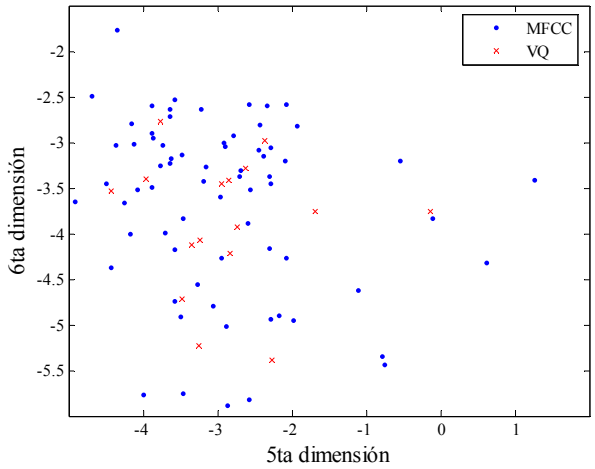


Figura 38. Cuantización vectorial de los MFCC's

Las cruces rojas que aparecen en la figura representan los 16 centroides calculados para la señal de voz de la figura 28, mientras que los puntos azules corresponden a los MFCC's calculados para esa misma señal. Estos centroides corresponden a un vector que contiene los 16 puntos ubicados a la menor distancia de todos los coeficientes cesptrales obtenidos del proceso de parametrización (bien sea mediante MFCC, LFCC o LPCC).

El conjunto de centroides obtenidos de la cuantización vectorial conforma un codebook, el cual representa las características inherentes a la señal de voz y que permite tener valores que participan en el cálculo de la distancia euclidiana entre el codebook de la palabra que dice el usuario y los codebooks guardados en la base de datos de entrenamiento del sistema. La ecuación 4.1 muestra el valor que arrojó la aplicación al calcular la menor distancia entre el codebook generado para la señal de voz de la figura 28 y una base de datos de entrenamiento con 200 codebooks correspondientes a 20 personas que grabaron las vocales a, e, i, o, u dos veces cada una.

$$d = 4,2367 \qquad (4.1)$$

Cabe destacar que el valor máximo de *d* para considerar que hay coincidencia entre la palabra dicha por el usuario y alguna de las palabras existentes en la base de datos de entrenamiento es de 6. De esta manera se pudo comprobar que el sistema encontró coincidencia entre la vocal "a" dicha por el usuario y una de las 40 vocales "a" guardadas en la base de datos de entrenamiento.

IV.2 Comparación entre los métodos de parametrización estudiados

Tal y como se describió en la sección III.4, se realizó la recolección de señales de voz para crear una base de datos de entrenamiento y una base de datos utilizada para simular el comportamiento "en vivo" del sistema. Usando esta última se realizaron las pruebas para comparar los métodos de parametrización evaluados.

Tal y como se menciona en la sección III.4, se recolectaron voces de 20 personas: 11 hombres y 9 mujeres, con edades comprendidas entre 20 y 24 años. De acuerdo a la metodología seguida para la recolección de señales de voz, se obtuvieron 200 muestras para la base de datos de entrenamiento y 100 muestras para la base de datos de prueba del sistema. El detalle de la participación de los hombres y mujeres en cada una de las bases de datos se puede encontrar en la tabla 3

	Muestras				
	a	**e**	**i**	**o**	**u**
Mujeres	9	9	9	9	9
Hombres	11	11	11	11	11

Tabla 3. Participación de mujeres y hombres en la recolección de voces

Se realizaron pruebas comparativas entre los tres métodos de parametrización estudiados: MFCC, LFCC y LPCC, usando como criterio de evaluación el porcentaje

de acierto en el reconocimiento de los vocablos. El resultado obtenido para las muestras de voz correspondientes a las mujeres se muestra en la tabla 4

	a		e		i		o		u	
	Muestras	%	Muestras	%	Muestras	%	Muestras	%	Muestras	%
MFCC	9	100	9	100	9	100	9	100	8	88,9
LFCC	9	100	9	100	9	100	6	66,7	7	77,8
LPCC	9	100	9	100	**8**	88,9	8	88,9	6	66,7

Tabla 4. Resultados de las pruebas para las mujeres

La celda sombreada en la tabla 4 representa el número de aciertos que tuvo el algoritmo del LPCC para la vocal "i". Según este resultado, de 9 muestras de voz correspondientes a la vocal "i" para las mujeres, este algoritmo identificó 8 de ellas correctamente, lo que corresponde a un 88,9% de acierto para esta palabra y así se puede observar para el resto de los métodos y las vocales. Por su parte, en la tabla 5 se muestran los resultados obtenidos de las pruebas para los hombres.

	a		e		i		o		u	
	Muestras	%	Muestras	%	Muestras	%	Muestras	%	Muestras	%
MFCC	**11**	100	10	90,9	10	90,9	8	72,7	10	90,9
LFCC	10	90,9	11	100	8	72,7	6	54,5	8	72,7
LPCC	11	100	10	90,9	11	100	10	90,9	9	81,8

Tabla 5. Resultados de las pruebas para los hombres

Según el ejemplo que se resalta en la tabla 5 se puede ver, de manera similar a la descrita para la tabla 4 que, para el caso de la vocal "a" y el método MFCC, de 11 muestras de voz recolectadas, el algoritmo acertó las 11 identificaciones correspondientes. Este mismo análisis se puede hacer para el resto de los métodos y vocales. En la tabla 6 se puede observar el comportamiento general de cada uno de los métodos, independientemente del sexo del hablante.

	a		e		i		o		u	
	Muestras	**%**	**Muestras**	**%**	**Muestras**	**%**	**Muestras**	**%**	**Muestras**	**%**
MFCC	20	100	19	95	19	95	17	85	18	90
LFCC	19	95	20	100	17	85	**12**	60	15	75
LPCC	20	100	19	95	19	95	18	90	15	75

Tabla 6. Resultados de las pruebas independiente del sexo del hablante

De forma similar a la que se menciona para las tablas anteriores, en la celda que se destaca en la tabla 6 se puede observar que de un total de 20 muestras de voz recolectadas para la letra "o" (9 de mujeres y 11 de hombres), el algoritmo LFCC identificó sólo 12, lo que corresponde a un 60% de acierto, bastante bajo en relación con los porcentajes de acierto del resto de los métodos.

Un aspecto importante que vale la pena destacar de los resultados mostrados en las tablas anteriores es que, independientemente del método o sexo del hablante, la mayor parte de los errores en el reconocimiento se dan para las vocales "o" y "u". Para comprender el por qué de esta situación, se muestra en la figura 39 la zonas donde se ubican los MFCC's de las vocales a, e, i, o, u

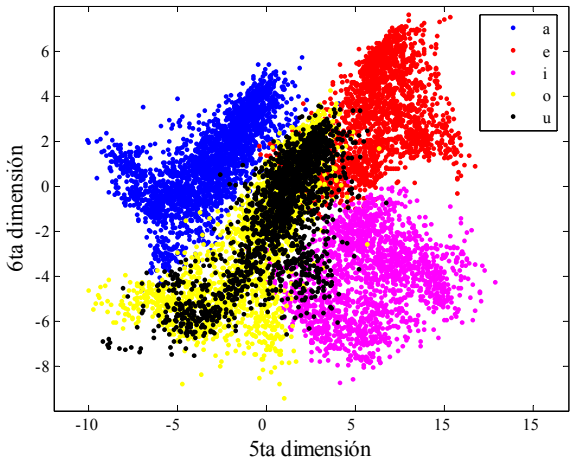


Figura 39. Principales formantes de las vocales a, e, i, o, u

La figura 39 corresponde al cálculo de los MFCC's de las 20 personas que participaron en la recolección de señales de voz realizada. Se puede notar que los MFCC's de las vocales "o" y "u" están ubicados en la misma zona, lo que hace muy probable que la aplicación encargada del reconocimiento al realizar el cálculo de la distancia euclidiana, tienda a confundir estas dos vocales. Por el contrario, se observa también cómo los MFCC's del resto de las vocales se encuentran claramente diferenciadas, por lo que se presentan escasos errores en el reconocimiento de éstas.

Así como se puede observar la cercanía entre las formantes de las vocales "o" y "u", cabe destacar que para las demás vocales no se producen casi errores, ya que la ubicación de sus formantes está claramente diferenciada del resto de las vocales. Tomando en cuenta estas observaciones, en la tabla 7 se muestra el porcentaje promedio de acierto para cada uno de los métodos para hombres y mujeres, independiente de la vocal en cuestión.

	MFCC	**LFCC**	**LPCC**
Mujeres	97,8 %	88,9 %	88,9 %
Hombres	89,1 %	78,2 %	92,7 %
Total	**93 %**	83 %	91 %

Tabla 7. Porcentaje promedio de acierto de cada uno de los métodos de parametrización

De la tabla 7 se puede observar que el método con el mayor porcentaje de acierto total es el MFCC, por lo que se escogió esta técnica de parametrización para la implementación de la aplicación encargada del reconocimiento del habla. Se observa también que este porcentaje de acierto está seguido de cerca por la técnica del LPCC, pero la información encontrada en la revisión bibliográfica sobre el gran uso de la técnica MFCC para la parametrización de señales de voz en aplicaciones para el

reconocimiento del habla, permitió hacer la escogencia de esta técnica como definitiva (Fernández, 2001).

IV.3 Diseño e implementación del hardware encargado del encendido y apagado de los dispositivos eléctricos

El hardware encargado del control de los dispositivos eléctricos conectados a él consta de varios módulos: alimentación, comunicación serial con el PC y de encendido y apagado como tal, los cuales fueron combinados en el mismo circuito impreso de la forma que se observa en la figura 40

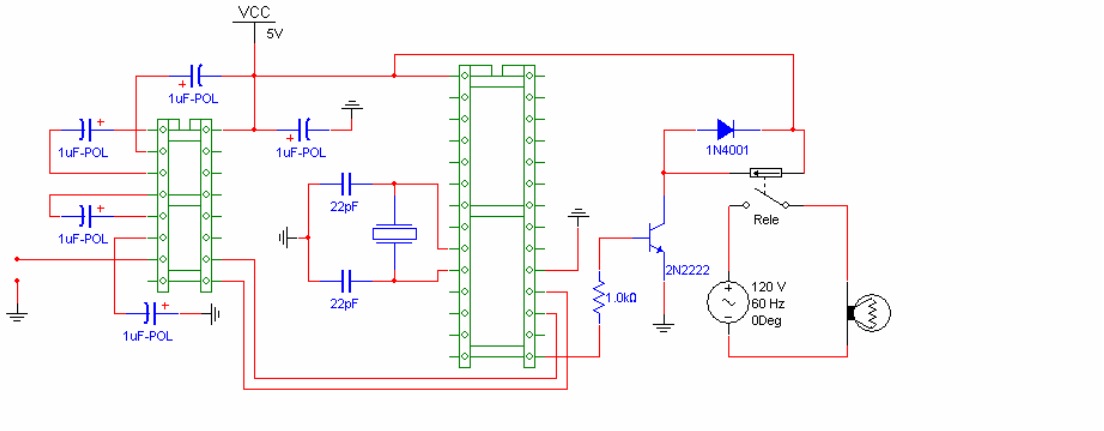


Figura 40. Diagrama circuital del hardware encargado del control de los dispositivos eléctricos

El módulo de comunicación serial con el PC representa la parte fundamental del hardware, ya que éste se basa en el uso del microcontrolador 16F873 como elemento de procesamiento de la información proveniente del PC como resultado de la interpretación del software encargado del reconocimiento del habla. El PIC recibe por unos de sus pines de comunicación serial una señal en formato TTL, provenientes del conversor HIN232 e interpreta la misma como un comando de encendido o apagado, verifica a cual dispositivo se hace referencia y su estado actual (encendido o apagado)

y luego niega el valor lógico presente a la salida de uno de sus 5 pines (dependiendo del dispositivo, se usa un pin diferente del puerto de salida c). Tal y como se explicó en la sección III.5.2, dependiendo de la salida que produce el microcontrolador (0 ó 1 lógico), el transistor se comporta como un suiche que cierra el circuito para polarizar la bobina del relé que enciende o apaga el dispositivo eléctrico involucrado (dependiendo del estado inicial del dispositivo) o, por el contrario, mantiene el estado de actividad del mismo.

Por su parte, el módulo de alimentación se incluyó como parte integral del hardware para prescindir de alimentación externa para polarizar la bobina del relé. Este módulo consiste en una fuente fija de 5 voltios que proporciona la corriente que el PIC no puede suministrar al circuito de polarización.

La unión de los módulos mencionados se realizó en una placa metálica mediante el diseño e implementación del circuito impreso que se muestra en la figura 41

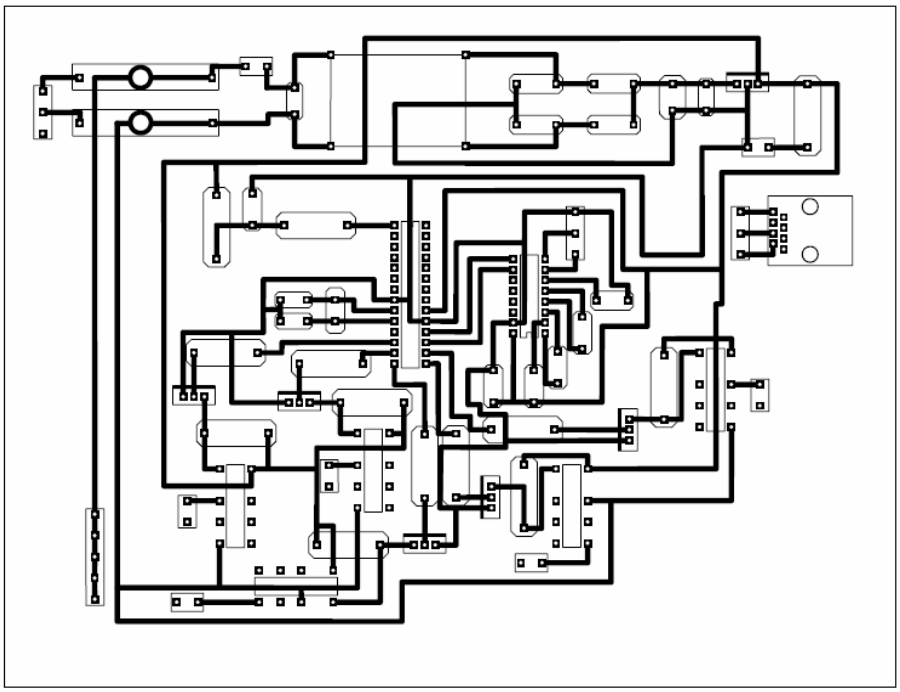
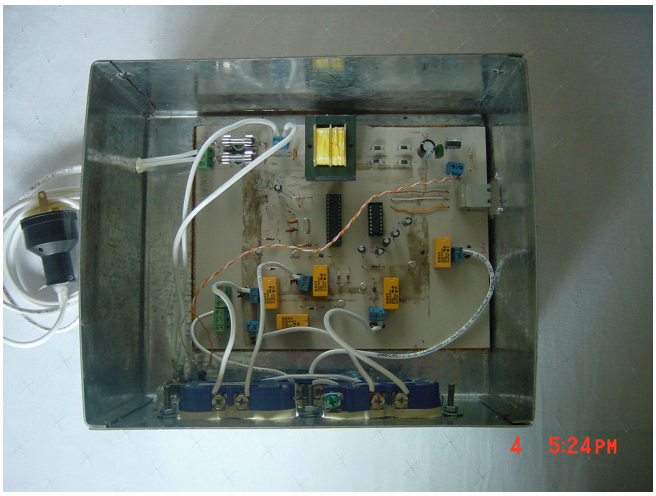


Figura 41. Circuito impreso para el montaje del hardware de control

Las siluetas que se observan en la figura 41 corresponden a la ubicación de cada uno de los componentes del hardware sobre el circuito impreso. En la figura 42 se muestra las fotos tomadas a la caja que se realizó para la implementación del hardware de control de los dispositivos eléctricos.

(a)



(b)

(c)

Figura 42. Implementación del hardware para el control de dispositivos eléctricos

Se puede observar de la figura 42b que la caja metálica tiene un zócalo correspondiente al conector RJ45 mediante el cual se realiza la comunicación serial

con el PC. El cable utilizado para tal fin es un cable de red de 8 hilos, del cual se utilizaron sólo 3: uno para transmisión, otro para recepción y el último para la tierra; dicho cable tiene en un extremo un conector RJ45 (conexión con el hardware) y en el otro un conector DB9 (conexión con el PC).

Además, la caja realizada tiene tres tomacorrientes con dos enchufes cada uno que se pueden ver en la figura 42a para conectar los dispositivos eléctricos que se deseen controlar y un suiche de encendido/apagado para indicar el estado de actividad del hardware. Los cuatro enchufes que están en la parte superior de la caja se controlan mediante las vocales a, e, i, o y los otros dos mediante la vocal u. Por último, el hardware posee un enchufe para alimentar a los dispositivos eléctricos con 110 voltios, el cual se puede observar en la figura 42c.

Capítulo V
Conclusiones y Recomendaciones

Se determinó mediante el estudio teórico y las evaluaciones realizadas de forma práctica que la etapa de pre-procesamiento es de gran importancia como preámbulo a la etapa de parametrización, ya que en ésta se resaltan y mejoran sustancialmente las características de la señal a parametrizar. Dentro de esta fase, la segmentación y aplicación de la ventana son de vital importancia para poder asumir que la señal de voz es estacionaria en esa ventana y aplicar técnicas que permitan el análisis de la misma, aprovechando las bondades que ofrece la estacionariedad de la voz. La ventana utilizada fue la Hamming, ya que ofrece el equilibrio entre la resolución en tiempo y frecuencia de la señal de voz requerido para la aplicación.

Del estudio teórico y práctico realizado para la etapa de parametrización y extracción de las características más importantes de las señales de voz, se encontró una gran variedad de técnicas. Se consideró desde un principio el uso de técnicas basadas en el cálculo de los coeficientes cepstrales por su gran uso en aplicaciones que involucran el procesamiento de señales de voz y por la gran variedad de combinaciones que entre ellos se pueden obtener. Como resultado del estudio comparativo realizado, se escogió la técnica del MFCC porque su porcentaje de acierto del 93% indica el excelente desempeño que este algoritmo tiene para la aplicación desarrollada en este Trabajo Especial de Grado. Además, la técnica MFCC permite agudizar las características de la señal e imitar el oído humano, basándose en sus anchos de banda críticos.

Como parte de la etapa de reconocimiento, se aplicó la cuantización vectorial para estandarizar el tamaño de las matrices que se generan del proceso de parametrización de la señal de voz. Aunque para este sistema no aplica, esta técnica es usualmente

empleada para compresión de datos, lo cual puede resultar útil si se plantea la posibilidad de enviar los datos por un canal y de esta manera reducir el ancho de banda necesario para la transmisión.

De las pruebas realizadas con la aplicación para determinar el desempeño de cada una de las técnicas de parametrización evaluadas, se observó que para el MFCC, el porcentaje de acierto en el reconocimiento de la vocal "a" fue de 100%, mientras que para las vocales "o" y "u" fue de 85% y 90% respectivamente. Este resultado se debe a la gran cercanía que tienen los MFCC's de las dos últimas vocales, lo que crea un margen de ambigüedad en la fase de reconocimiento, mientras que la vocal "a" tiene una zona muy bien definida en la cual se ubican sus MFCC's, disminuyendo cualquier posible margen de error en el reconocimiento.

En cuanto al lenguaje de programación utilizado en la aplicación encargada del reconocimiento, la elección fue MATLAB, ya que éste posee una amplia gama de módulos que pueden usarse para el desarrollo de aplicaciones relacionadas con procesamiento de señales de forma sencilla, en comparación con otros lenguajes de programación tales como C++, Visual Basic, entre otros.

A pesar de que en el sistema no se hace consideración alguna de la influencia de un ambiente ruidoso en el desempeño y porcentaje de acierto en el reconocimiento del habla, se realizó la recolección de las señales de voz que forman parte de la base de datos de entrenamiento y pruebas del sistema en ambientes con distintos niveles de ruido para tener una variedad de señales con las cuales comparar y simular el funcionamiento "en vivo" del sistema.

Una fase muy importante es el entrenamiento del sistema con las señales de voz guardadas en la base de datos. Dicho entrenamiento se realiza al inicio de la aplicación, ya que éste consiste en el cálculo de los codebooks de cada una de las

señales y esto lleva cierto tiempo que es preferible que el usuario espere al comienzo de la aplicación, y no cada vez que quiera realizar el reconocimiento de una palabra. De esta observación se desprende además la conclusión de que mientras más codebooks estén guardados en la base de datos de entrenamiento del sistema, más acertado es el reconocimiento, aunque esto implica un incremento en el tiempo de inicialización de la aplicación.

Entre algunas de las recomendaciones que se desprenden del desarrollo de este proyecto se encuentran: estudio de otras técnicas de parametrización de la señal de voz como Modelos Ocultos de Markov, redes neuronales, algoritmos genéticos, entre otros. Para la etapa de reconocimiento pueden considerarse otros métodos de compresión basados en cuantización vectorial, ya que ayudan a mejorar la comparación; además pueden evaluarse otros métodos de medición diferentes a la distancia euclidiana, como el Hitakura.

En cuanto al hardware empleado para controlar los dispositivos, en lugar de relés, se pueden emplear transistores de potencia y también ampliar el alcance de control de dicho hardware a dispositivos electrónicos y no solamente eléctricos

Bibliografía

- Davis, S. & Mermelstein, P. (Agosto de 1980). Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*. Volumen 28, Páginas: 357 – 366

- De la Torre, A. (n.d). *Procesamiento de señales de voz.* Extraído el 20 de diciembre de 2005 desde http://ceres.ugr.es/~atv/Documents/Docencia/voz.ppt

- Do, M. (n.d). *An Automatic Speaker Recognition System.* Extraído el 25 de noviembre de 2005 desde http://lcavwww.epfl.ch/~minhdo/asr_project/asr_project.pdf

- Enciclopedia Wikipedia (18 de abril de 2006). *Distancia euclidiana.* Extraído el 20 de junio de 2006 desde http://es.wikipedia.org/wiki/Distancia_euclidiana

- Enciclopedia Wikipedia (3 de julio de 2006). *Formante.* Extraído el 10 de junio de 2006 desde http://es.wikipedia.org/wiki/Distancia_euclidiana

- Faúndez, M y Rodríguez, D. (10 de febrero de 2005). *Estudio comparativo de diferentes distancias en sistemas basados en VQ para identificación automática de locutores*. Extraído el 27 de junio de 2006 desde http://veu.eupmt.es/publicacions/docs/ursi98-1.pdf

- Fernández, L. (2001). *Aportaciones a la Mejora de los Sistemas de Reconocimiento.* Extraído el 22 de febrero de 2006 desde http://www.gts.tsc.uvigo.es/~ldocio/PhdThesis.pdf

- Flores, A. (1 de octubre de 1998). *Reconocimiento de Palabras Aisladas*. Extraído el 20 de noviembre de 2005 desde http://alek.pucp.edu.pe/~dflores/procespre.html

- Grupo PAS – Universidad de Deusto (n.d). *Modelado de la señal de voz*. Extraído el 30 de abril de 2005 desde http://www.pas.deusto.es/recursos/Modelado de la señal de voz Grupo PAS.ppt

- GTAS – Universidad de Cantabria. *Predicción Lineal de la señal de voz "Linear Predictive Coding"(LPC)*. Extraído el 29 de diciembre de 2005 desde http://www.gtas.dicom.unican.es/tdv/Documentos/Apuntes/LPC.pdf.

- Herrera, C (n.d). *Fundamentos Básicos del Reconocimiento de Voz*. Extraído el 30 de enero de 2006 desde http://www.monografias.com/trabajos901/fundamentos-basicos-reconocimiento-voz/fundamentos-basicos-reconocimiento-voz.shtml

- Iosu, S. (24 de Junio de 1999). *Speech Recognition*. Extraído el 7 de enero de 2006 desde http://www.euskalnet.net/iosus/speech/recog2.html.

- Kornhauser, D. (4 de marzo de 1999). *Diseño de un reconocedor de voz de palabras aisladas público*. Extraído el 3 de febrero de 2006 desde http://bq.unam.mx/~daniel/gvoice/tesis/node22.html

- López, L. (Febrero de 1999). *Implementación del procesamiento de la señal de voz en la interfaz de radio del sistema de telefonía móvil GSM*. Extraído el 23 de mayo de 2006 desde http://ceres.ugr.es/~alumnos/alonso/p1.html

- Mahanad, R. (Julio 2005). *Fourier Transform Phase-Based features for speech recognition.* Extraído el 30 de noviembre de 2005 desde http://lantana.tenet.res.in/Publications/Speech/rajesh_thesis.pdf

- Martí, J, ed. 1-2, (1998). Situación actual de las tecnologías del habla. *Revista de Acústica*, Volumen 29, 5-12. Extraído el 7 de diciembre de 2005 desde http://www.ia.csic.es/sea/publicaciones/4372kb003.pdf

- Oppenheim, A & Schafer, R. *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice Hall, 1989.

- Reyes, J.A y Herrera, J.A (2005). *Reconocimiento de comandos de voz para Windows y Word.* Extraído el 10 de mayo de 2006 desde http://www.fip.unam.mx/ simposio_investigacion2005/ponencia29_ext.html

- San Martín, C y Carrillo, R. (3 de Mayo de 2004). *Implementación de un reconocedor de palabras aisladas dependiente del locutor.* Extraído el 7 de enero de 2006 desde http://www.scielo.cl/pdf/rfacing/v12n1/art02.pdf

- Slavinsky, J. (19 de diciembre de 1999). *Speaker Verification*. Extraído el 16 de febrero de 2006 desde http://www.owlnet.rice.edu/~elec301/Projects99/ wrcocee/endpt.htm

- Stefanelli, M. C. *Estimación espectral adaptiva del Electroencefalograma*. Tesis de maestría. USB. (Mayo 1985)

- Van, B. (21 de diciembre de 2004). *The Cepstrum Domain.* Extraído el 13 de abril de 2006 desde http://cnx.org/content/m12469/latest/

Anexo A

Demostración del cálculo de la Predicción Lineal de los Coeficientes Cepstrales (LPCC)

Recursive Calculation of Mel-Cepstrum from LP Coefficients

Keiichi Tokuda†, Takao Kobayashi††, and Satoshi Imai††,

†Department of Computer Science
Nagoya Institute of Technology, Nagoya, 466-8555 Japan
††Precision and Intelligence Laboratory
Tokyo Institute of Technology, Yokohama, 227 Japan

1 April 1994

Abstract

The mel-cepstral coefficients are often calculated from the linear prediction coefficients by using recursion formulas. However, the obtained mel-cepstral coefficients have errors caused by truncation in the quefrency domain. The purpose of this report is to point out that the mel-cepstral coefficients can be calculated from the LP (Linear Prediction) coefficients using the recursion formulas without the truncation error.

1 INTRODUCTION

The mel-cepstrum is useful parameter for speech recognition, and have widely been used in many speech recognition systems [1]. There exist several methods to obtain mel-cepstral coefficients. In the case using recursion formulas, the mel-cepstral coefficients, i.e., frequency-transformed cepstral coefficients, are calculated from the LP (Linear Prediction) coefficients as follows: the LP coefficients are first transformed to the cepstral coefficients by using the recursion formula of [2], then the cepstral coefficients transformed to the mel-cepstral coefficients by using the recursion formula of [3]. However, since the cepstrum calculated from the LP coefficients is an infinite sequence, it must be truncated in the quefrency domain. As a result, we cannot obtain exact values of mel-cepstral coefficients.

This report points out that the mel-cepstral coefficients can be calculated from the LP coefficients without the truncation error when we first apply the recursion formula for frequency-transformation, and then apply the recursion formula of LPC-cepstrum.

2 Definition of The Mel-Cepstrum

The cepstrum[1] $c(m)$ of a real sequence $x(n)$ is defined as

$$c(m) = \frac{1}{2\pi j} \oint_C \log X(z)\, z^{m-1}\, dz \tag{1}$$

[1]Strictly, the complex cepstrum defined by A. V. Oppenheim and R. W. Schafer [4].

$$\log X(z) = \sum_{m=-\infty}^{\infty} c(m)\, z^{-m} \tag{2}$$

where $X(z)$ is the z-transform of $x(n)$, and C is a counterclockwise closed contour in the region of convergence of $\log X(z)$ and encircling the origin of the z-plane. Frequency-transformed cepstrum, so-called mel-cepstrum [5], is defined as

$$\tilde{c}(m) = \frac{1}{2\pi j} \oint_C \log X(z)\, \tilde{z}^{m-1}\, d\tilde{z} \tag{3}$$

$$\log X(z) = \sum_{m=-\infty}^{\infty} \tilde{c}(m)\, \tilde{z}^{-m} \tag{4}$$

where

$$\tilde{z}^{-1} = \Psi(z) = \frac{z^{-1} - \alpha}{1 - \alpha\, z^{-1}}, \qquad |\,\alpha\,| < 1. \tag{5}$$

The phase response $\tilde{\omega}$ of the all-pass system $\Psi(e^{j\omega}) = e^{-j\tilde{\omega}}$ is given by

$$\tilde{\omega} = \beta(\omega) = \tan^{-1} \frac{(1-\alpha^2)\sin\omega}{(1+\alpha^2)\cos\omega - 2\alpha}. \tag{6}$$

Thus, evaluating (3) and (4) on the unit circle of the $\tilde{z}$-plane, we see that $\tilde{c}(m)$ is the inverse Fourier transform of $\log X(e^{j\omega})$ calculated on a warped frequency scale $\tilde{\omega}$:

$$\tilde{c}(m) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log \tilde{X}(e^{j\tilde{\omega}})\, e^{j\tilde{\omega}m}\, d\tilde{\omega} \tag{7}$$

$$\log \tilde{X}(e^{j\tilde{\omega}}) = \sum_{m=-\infty}^{\infty} \tilde{c}(m)\, e^{-j\tilde{\omega}m} \tag{8}$$

where

$$\tilde{X}(e^{j\tilde{\omega}}) = X(e^{j\beta^{-1}(\tilde{\omega})}). \tag{9}$$

The phase response $\tilde{\omega} = \beta(\omega)$ gives a good approximation to auditory frequency scales with an appropriate choice of α. An example of 16kHz sampling is shown in Fig. 1. In the figure, dashed lines show the mel [6] and Bark[2] frequency scales normalized by 8kHz. Table 1 shows examples of α for approximating the mel and Bark scales at several sampling frequencies.

The system function obtained by the LP method has the form

$$H(z) = \frac{K}{1 + \sum_{m=1}^{M} a(m)\, z^{-m}} \tag{10}$$

where we assume that $H(z)$ is a minimum phase system[3] . The problem is to calculate the mel-cepstral coefficients $\tilde{c}(m)$, $m = 0, 1, \ldots, N$ given by

$$\log \frac{K}{1 + \sum_{m=1}^{M} a(m)\, z^{-m}} = \sum_{m=0}^{\infty} \tilde{c}(m) \tilde{z}^{-m} \tag{11}$$

[2]A piecewise function shown in [7] is used for plotting the Bark scale.

[3]The system $H(z)$ obtained by the autocorrelation method is always minimum phase.

Table 1: Examples of α.

Sampling Frequency	8kHz	10kHz	12kHz	16kHz	20kHz	22.05kHz
mel scale	0.31	0.35	0.37	0.42	0.44	0.45
Bark scale	0.42	0.47	0.50	0.55	—	—

from the LP coefficients $a(m)$, $m = 1, 2, \ldots, M$ and K. It is noted that when the system $H(z)$ is of minimum phase, the mel-cepstrum becomes a causal and stable sequence.

In the conventional recursive calculation method, the mel-cepstral coefficients are calculated as follows:

1. The cepstral coefficients $c(m)$'s given by

$$\log \frac{K}{1 + \sum_{m=1}^{M} a(m)\, z^{-m}} = \sum_{m=0}^{\infty} c(m)\, z^{-m} \tag{12}$$

   are calculated by using the recursion formula of [2].

2. The mel-cepstral coefficients $\tilde{c}(m)$, $m = 1, 2, \ldots, N$ given by

$$\sum_{m=0}^{\infty} c(m)\, z^{-m} = \sum_{m=0}^{\infty} \tilde{c}(m) \tilde{z}^{-m} \tag{13}$$

   are calculated by using the recursion formula for frequency transformation [3].

Unfortunately, the cepstrum obtained by (12) is an infinite sequence and we require the infinite sequence to calculate finite number of the mel-cepstral coefficients given by (13). Practically, the mel-cepstrum is calculated from truncated cepstrum by an approximation:

$$\sum_{m=0}^{L} c(m)\, z^{-m} = \sum_{m=0}^{\infty} \tilde{c}(m) \tilde{z}^{-m}. \tag{14}$$

It is noted that to reduce the truncation error sufficiently, L should be chosen in such a way that $L >> M$. The complexity to calculate $\tilde{c}(m)$, $m = 0, 1, \ldots, N$ from $a(m)$, $m = 1, 2, \ldots, M$ and K is $O(ML) + O(LN)$.

3 Recursive Calculation of The Mel-Cepstrum

The following procedure can avoid the trancation error:

1. The frequency transformed LP coefficients are first calculated based on the relation:

$$\frac{K}{1 + \sum_{m=1}^{M} a(m)\, z^{-m}} = \frac{\tilde{K}}{1 + \sum_{m=1}^{\infty} \tilde{a}(k) \tilde{z}^{-m}} \tag{15}$$

   by the recursion formula for frequency transformation [3].

2. The mel-cepstrum given by

$$\log \frac{\tilde{K}}{1+\sum_{m=1}^{\infty} \tilde{a}(k)\tilde{z}^{-m}} = \sum_{m=0}^{\infty} \tilde{c}(m)\tilde{z}^{-m} \tag{16}$$

are calculated by the recursion formula of [2].

Although $\tilde{a}(m)$ is an infinite sequence, the mel-cepstral coefficients $\tilde{c}(m)$, $m = 0, 1, \ldots, N$ given by (16) can be calculated from the finite sequence $\tilde{a}(m)$, $m = 1, 2, \ldots, N$ and $\tilde{K}$. Therefore, it is not necessary to calculate infinite number of coefficients $\tilde{a}(m)$, $m = 1, 2, \ldots, \infty$.

The recursion formulas for calculation of mel-cepstrum from LP coefficients are written as follows:

$$\tilde{a}^{(i)}(m) = \begin{cases} a(-i) + \alpha\,\tilde{a}^{(i-1)}(0), & m = 0 \\ (1-\alpha^2)\,\tilde{a}^{(i-1)}(0) + \alpha\,\tilde{a}^{(i-1)}(1), & m = 1 \\ \tilde{a}^{(i-1)}(m-1) + \alpha\,(\tilde{a}^{(i-1)}(m) - \tilde{a}^{(i)}(m-1)), & m = 2, 3, \ldots, N \end{cases}$$
$$i = -M, \ldots, -1, 0 \tag{17}$$

$$\tilde{K} = K/\tilde{a}^{(0)}(0), \qquad \tilde{a}(m) = \tilde{a}^{(0)}(m)/\tilde{a}^{(0)}(0), \quad 1 \le m \le N \tag{18}$$

$$\tilde{c}(m) = \begin{cases} \log \tilde{K}, & m = 0 \\ -\tilde{a}(m) - \sum_{k=1}^{m-1} \frac{k}{m}\tilde{c}(k)\,\tilde{a}(m-k), & 1 \le m \le N \end{cases} \tag{19}$$

where $a(0) = 1$.

The above algorithm calculates $\tilde{c}(m)$, $m = 1, 2, \ldots, N$ from $a(m)$, $m = 1, 2, \ldots, M$ and K with the complexity of $O(MN)+O(N^2)$. Note that the above algorithm is a special case of the recursion formula of [8], [9].

4 Example

Fig. 2(a) shows an example of the spectrum obtained by the LP method. Fig. 2(b) and (c) show the spectra represented by the mel-cepstra which obtained the conventional and proposed methods, respectively. The spectrum represented by the mel-cepstral coefficients $\tilde{c}(m)$, $m = 0, 1, \ldots, N$ is given by

$$H_N(e^{j\omega}) = \exp \sum_{m=0}^{N} \tilde{c}(m)\tilde{z}^{-m}. \tag{20}$$

From the figures, it is seen that the spectrum obtained by the conventional method loses the feature at the low frequencies because of the truncation in the quefrency domain. On the other hand, the proposed method is independent from the truncation in the quefrency domain. In this simulation, $L = 15$ $(= N = M)$ is used for the conventional calculation method. Although the truncation error can be reduced sufficiently with L greater than about $2M$, we cannot obtain exact values of mel-cepstral coefficients unless $L \to \infty$.

Anexo B

Especificaciones técnicas del HIN232

+5V Powered RS-232 Transmitters/Receivers

The HIN232-HIN241 family of RS-232 transmitters/receivers interface circuits meet all EIA RS-232E and V.28 specifications, and are particularly suited for those applications where ±12V is not available. They require a single +5V power supply (except HIN239) and feature onboard charge pump voltage converters which generate +10V and -10V supplies from the 5V supply. The family of devices offer a wide variety of RS-232 transmitter/receiver combinations to accommodate various applications (see Selection Table).

The drivers feature true TTL/CMOS input compatibility, slew-rate-limited output, and 300Ω power-off source impedance. The receivers can handle up to ±30V, and have a 3kΩ to 7kΩ input impedance. The receivers also feature hysteresis to greatly improve noise rejection.

Features

- Meets All RS-232E and V.28 Specifications
- Requires Only Single +5V Power Supply
  - (+5V and +12V - HIN239)
- High Data Rate. 120kbps
- Onboard Voltage Doubler/Inverter
- Low Power Consumption
- Low Power Shutdown Function
- Three-State TTL/CMOS Receiver Outputs
- Multiple Drivers
  - ±10V Output Swing for 5V Input
  - 300Ω Power-Off Source Impedance
  - Output Current Limiting
  - TTL/CMOS Compatible
  - 30V/μs Maximum Slew Rate
- Multiple Receivers
  - ±30V Input Voltage Range
  - 3kΩ to 7kΩ Input Impedance
  - 0.5V Hysteresis to Improve Noise Rejection
- Pb-Free Plus Anneal Available (RoHS Compliant)

Applications

- Any System Requiring RS-232 Communication Ports
  - Computer - Portable, Mainframe, Laptop
  - Peripheral - Printers and Terminals
  - Instrumentation
  - Modems

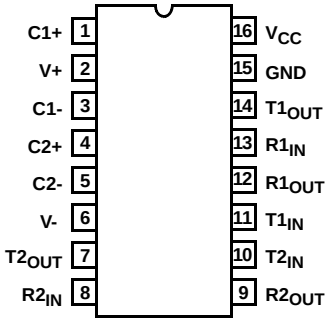
Selection Table

PART NUMBER	POWER SUPPLY VOLTAGE	NUMBER OF RS-232 DRIVERS	NUMBER OF RS-232 RECEIVERS	EXTERNAL COMPONENTS	LOW POWER SHUTDOWN/TTL THREE-STATE	NUMBER OF LEADS
HIN232	+5V	2	2	4 Capacitors	No/No	16
HIN236	+5V	4	3	4 Capacitors	Yes/Yes	24
HIN237	+5V	5	3	4 Capacitors	No/No	24
HIN238	+5V	4	4	4 Capacitors	No/No	24
HIN239	+5V and +7.5V to 13.2V	3	5	2 Capacitors	No/Yes	24
HIN240	+5V	5	5	4 Capacitors	Yes/Yes	44
HIN241	+5V	4	5	4 Capacitors	Yes/Yes	28

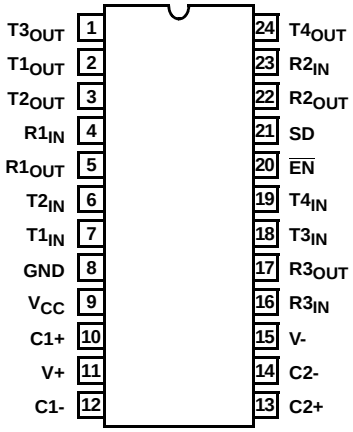
CAUTION: These devices are sensitive to electrostatic discharge; follow proper IC Handling Procedures.
1-888-INTERSIL or 1-888-468-3774 | Intersil (and design) is a registered trademark of Intersil Americas Inc.
Copyright © Intersil Americas Inc. 2004, 2005. All Rights Reserved.
All other trademarks mentioned are the property of their respective owners.

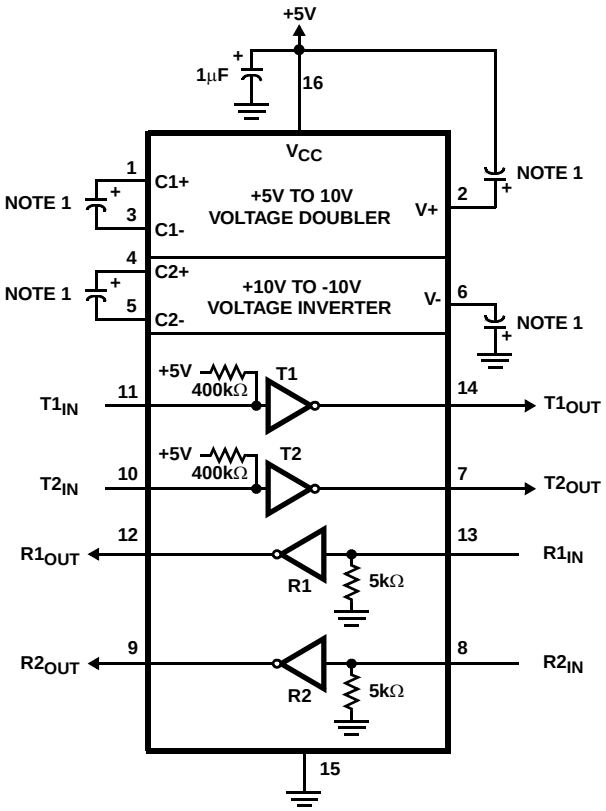
Pinouts

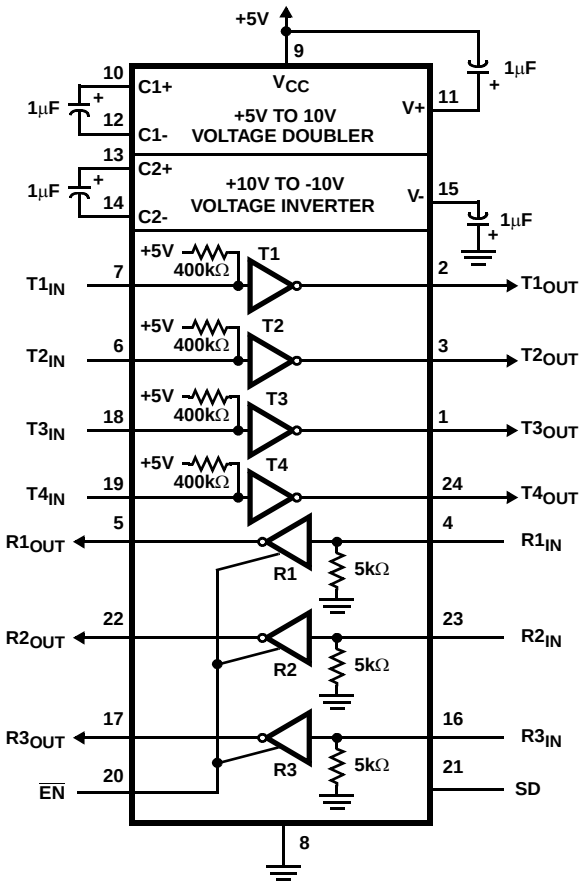
HIN232 (PDIP, SOIC)
TOP VIEW



HIN236 (SOIC)
TOP VIEW







NOTE:

1. Either 0.1µF or 1µF capacitors may be used. The V+ capacitor may be terminated to V_{CC} or to GND.

Anexo C

Especificaciones técnicas del PIC 16F873

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

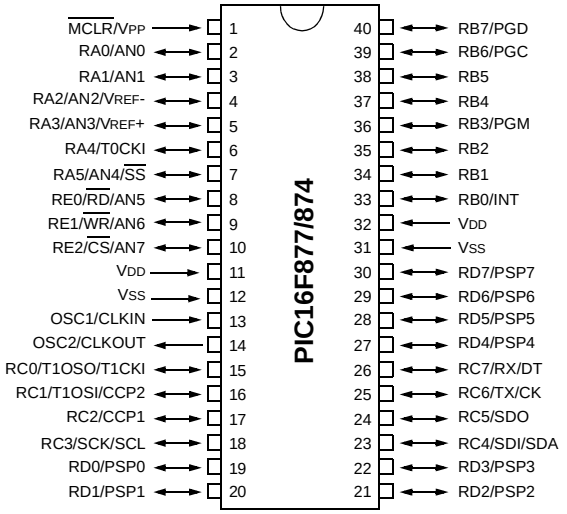
- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and
  Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 μA typical @ 3V, 32 kHz
  - < 1 μA typical standby current

Pin Diagram

PDIP

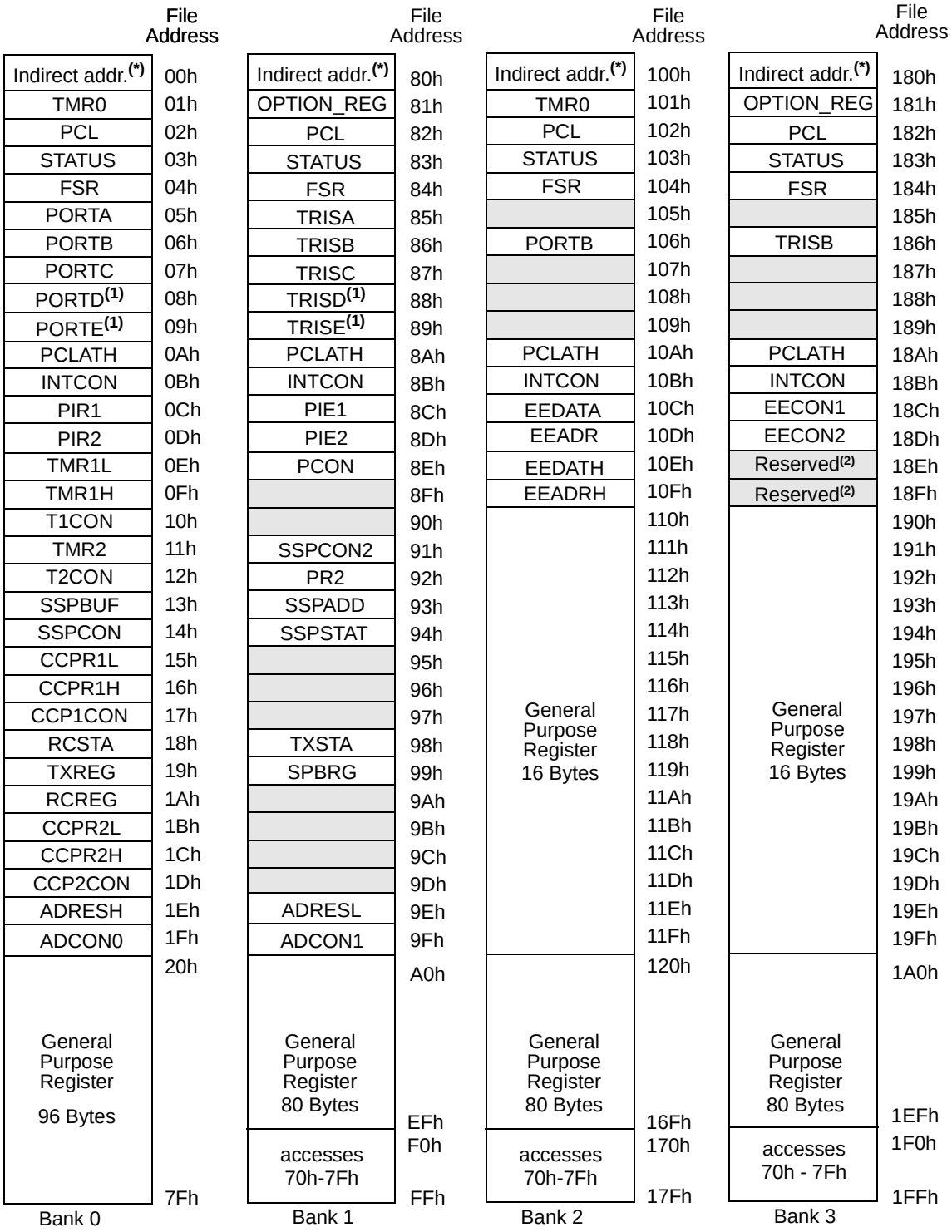


Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^2C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

© 2001 Microchip Technology Inc.

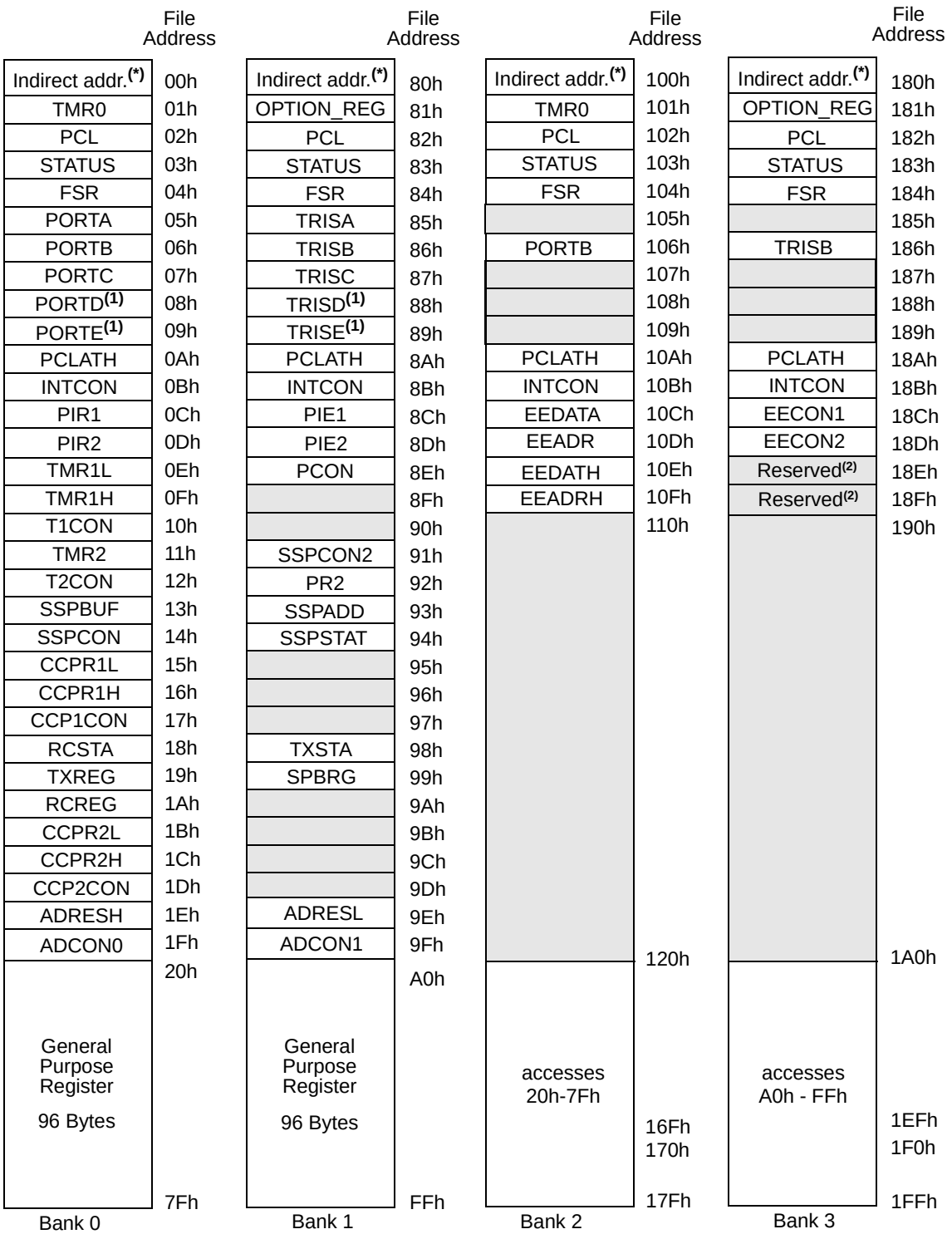
FIGURE 2-3: PIC16F877/876 REGISTER FILE MAP



Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876.
2: These registers are reserved, maintain these registers clear.

FIGURE 2-4: PIC16F874/873 REGISTER FILE MAP



Unimplemented data memory locations, read as '0'.

* Not a physical register.

Note 1: These registers are not implemented on the PIC16F873.

2: These registers are reserved, maintain these registers clear.

© 2001 Microchip Technology Inc.

Apéndice A

Código de programación del PIC 16F873

```
            LIST   P=16F873
            RADIX  HEX
            INCLUDE      <P16F873.inc>

CONV_HIGH   EQU    20h
CONV_LOW    EQU    21h
CounterA    EQU    22h
CounterB    EQU    23h
BUFFER      EQU    24h
OnOFF_A            EQU    25h
OnOFF_E            EQU    26h
OnOFF_I            EQU    27h
OnOFF_O            EQU    28h
OnOFF_U            EQU    29h

            ORG    0x00
            goto   RS232

            ORG    0x04
            goto   INTERR       ;Rutina de interrupción

RS232
            call   BANCO0       ;paso  al  banco  0  para  cambiar  ciertas
variables
            clrf   PORTC        ;Limpia salidas y entradas
            call   BANCO1       ;paso al banco 1
            movlw  b'10000000'  ;RC7/Rx y RC6/Tx: RS232
            movwf  TRISC        ;RC5-RC0:Entradas digitales adicionales
            movlw  b'00100100'  ;OJO: El bit 2 hay q cambiarlo a 0 cuando
la
                                ;velocidad=300 bps
            movwf  TXSTA        ;Configuración    USART:    activación    de
                   transmisión,
                                  ;baja velocidad, 8 bits de tx
            movlw  .129         ;OJO: 9600 bps
            movwf  SPBRG
            movlw  b'00100000'
            movwf  PIE1         ;sin interrupción para la transmisión, con
                                ;interrupción para la recepción
            call   BANCO0
            movlw  b'10010000'
            movwf  RCSTA        ;Habilita  el  puerto  serial  para  recepción
continua,
                                ;8 bits para la rx
            clrf   RCREG
            movlw  b'11000000'
```

```
            movwf  INTCON       ;habilita  interrupciones  globales  y  de
periféricos
            movlw  D'1'
            movwf  OnOFF_A
            movlw  D'1'
            movwf  OnOFF_E
            movlw  D'1'
            movwf  OnOFF_I
            movlw  D'1'
            movwf  OnOFF_O
            movlw  D'1'
            movwf  OnOFF_U
            goto   ESPERA

BANCO1
            bsf          STATUS,RP0
            bcf          STATUS,RP1
            return

BANCO0
            bcf          STATUS,RP0
            bcf          STATUS,RP1
            return

ESPERA
            goto   ESPERA

INTERR
            call   BANCO0
            btfss  PIR1,RCIF    ;Chequea la bandera de Rx
            goto   EXIT_INTERR
            clrf   BUFFER
            movf   RCREG,W
            movwf  BUFFER
            goto   COMPARAR_A

EXIT_INTERR
            retfie

COMPARAR_A
            movf   BUFFER,W
            xorlw  'A'
            btfss  STATUS,Z
            goto   COMPARAR_E
            goto   PRENDER_A

COMPARAR_E
            movf   BUFFER,W
            xorlw  'E'
            btfss  STATUS,Z
            goto   COMPARAR_I
            goto   PRENDER_E

COMPARAR_I
            movf   BUFFER,W
            xorlw  'I'
            btfss  STATUS,Z
            goto   COMPARAR_O
            goto   PRENDER_I
```

```
COMPARAR_O
            movf   BUFFER,W
            xorlw  'O'
            btfss  STATUS,Z
            goto   COMPARAR_U
            goto   PRENDER_O

COMPARAR_U
            movf   BUFFER,W
            xorlw  'U'
            btfss  STATUS,Z
            goto   EXIT_INTERR
            goto   PRENDER_U

PRENDER_A
            decfsz OnOFF_A,1
            goto   OFF_A
            goto   On_A
            goto   EXIT_INTERR

On_A
            bsf    PORTC,1
            goto   EXIT_INTERR

OFF_A
            movlw  D'1'
            movwf  OnOFF_A
            bcf    PORTC,1
            goto   EXIT_INTERR

PRENDER_E
            decfsz OnOFF_E,1
            goto   OFF_E
            goto   On_E
            goto   EXIT_INTERR

On_E
            bsf    PORTC,2
            goto   EXIT_INTERR

OFF_E
            movlw  D'1'
            movwf  OnOFF_E
            bcf    PORTC,2
            goto   EXIT_INTERR

PRENDER_I
            decfsz OnOFF_I,1
            goto   OFF_I
            goto   On_I
            goto   EXIT_INTERR

On_I
            bsf    PORTC,3
            goto   EXIT_INTERR

OFF_I
            movlw  D'1'
            movwf  OnOFF_I
            bcf    PORTC,3
```

```
            goto   EXIT_INTERR

PRENDER_O
            decfsz OnOFF_O,1
            goto   OFF_O
            goto   On_O
            goto   EXIT_INTERR

On_O
            bsf    PORTC,4
            goto   EXIT_INTERR

OFF_O
            movlw  D'1'
            movwf  OnOFF_O
            bcf    PORTC,4
            goto   EXIT_INTERR

PRENDER_U
            decfsz OnOFF_U,1
            goto   OFF_U
            goto   On_U
            goto   EXIT_INTERR

On_U
            bsf    PORTC,5
            goto   EXIT_INTERR

OFF_U
            movlw  D'1'
            movwf  OnOFF_U
            bcf    PORTC,5
            goto   EXIT_INTERR

            END
```

Apéndice B

Código de programación de la aplicación encargada del reconocimiento del habla

1.- PocoNE.m

```
clear; clc; close all; fs=10000; i=1;

newobjs = instrfind;
if size(newobjs,1)~=0
   fclose(newobjs);
end
com1 = serial('COM1','BaudRate',9600,'DataBits',8);
fopen(com1);
load secudb;
nd=secudb-1;
clear secudb;
save('nd','nd');
if nd>0
    code = train(' ', 'db_train\', nd, 256, 100, 20, 'MFCC', 'tria', 'no');
    save('code','code');
end
clc;

while 1
    clc; cabecera;
    disp('1.- Agregar');
    disp('2.- Reconocer');
    disp('3.- Salir');
    while 1
        ch=getkey;
        if ch==49, break, end
        if nd>0
            if ch==50, break, end
        end
        if ch==51, break, end
    end
    if ch==49
        clc; cabecera;
        disp('Instrucciones');
        disp('* Se grabara en forma secuecial cada una de las vocales');
        disp('* Teclee ESPACIO para continuar');
        getkey2;
        vuel=1;

        while 1
            while 1
                clc; cabecera;
                disp('Instrucciones');
```

```
            disp('- Teclee ESPACIO para iniciar la grabación de la VOCAL
''A''');
            disp('- Teclee ESPACIO para finalizar la grabación');
            getkey2;
            addnew;
            if er==1
                disp(' ');
                disp('* Teclee ESPACIO para continuar');
                getkey2;
            else
                disp('* Grabacion correcta');
                disp('* Teclee ESPACIO para continuar');
                getkey2;
                break;
            end
        end

        while 1
            clc; cabecera;
            disp('Instrucciones');
            disp('- Teclee ESPACIO para iniciar la grabación de la VOCAL
''E''');
            disp('- Teclee ESPACIO para finalizar la grabación');
            getkey2;
            addnew;
            if er==1
                disp(' ');
                disp('* Teclee ESPACIO para continuar');
                getkey2;
            else
                disp('* Grabacion correcta');
                disp('* Teclee ESPACIO para continuar');
                getkey2;
                break;
            end
        end

        while 1
            clc; cabecera;
            disp('Instrucciones');
            disp('- Teclee ESPACIO para iniciar la grabación de la VOCAL
''I''');
            disp('- Teclee ESPACIO para finalizar la grabación');
            getkey2;
            addnew;
            if er==1
                disp(' ');
                disp('* Teclee ESPACIO para continuar');
                getkey2;
            else
                disp('* Grabacion correcta');
                disp('* Teclee ESPACIO para continuar');
                getkey2;
                break;
            end
        end

        while 1
            clc; cabecera;
            disp('Instrucciones');
```

```
                disp('- Teclee ESPACIO para iniciar la grabación de la VOCAL
''O''');
                disp('- Teclee ESPACIO para finalizar la grabación');
                getkey2;
                addnew;
                if er==1
                    disp(' ');
                    disp('* Teclee ESPACIO para continuar');
                    getkey2;
                else
                    disp('* Grabacion correcta');
                    disp('* Teclee ESPACIO para continuar');
                    getkey2;
                    break;
                end
            end

            while 1
                clc; cabecera;
                disp('Instrucciones');
                disp('- Teclee ESPACIO para iniciar la grabación de la VOCAL
''U''');
                disp('- Teclee ESPACIO para finalizar la grabación');
                getkey2;
                addnew;
                if er==1
                    disp(' ');
                    disp('* Teclee ESPACIO para continuar');
                    getkey2;
                else
                    disp('* Grabacion correcta');
                    disp('* Teclee ESPACIO para continuar');
                    getkey2;
                    break;
                end
            end

            if vuel==2
                break;
            end

            if vuel==1
                vuel=2;
            end
        end

        clc; cabecera;
        load secudb;
        secudb=secudb-1;
        load nd;
        nd=nd+1;
        clear code;
        code = train2(nd, 'db_train\', secudb, 256, 100, 20, 'MFCC', 'tria',
'no');
        save('code','code');
    end

    if ch==50
        while 1
            clc; cabecera;
```

```
            disp('Instrucciones');
            disp('- Teclee ESPACIO para iniciar la grabación');
            disp('- Menciona la VOCAL que identifique el dispositivo que
desea manejar');
            disp('- Teclee ESPACIO para finalizar la grabación');
            getkey2;
            clc; cabecera;
            disp('Grabando ...');
            micrecorder=audiorecorder(10000,16,1);
            record(micrecorder);
            getkey2;
            stop(micrecorder);
            s=getaudiodata(micrecorder);
            s=s-mean(s);
            s2=s;
            s=prenfasis(s);
            s=s-mean(s);
            e=calenergia(s2);
            er=0;
            clc; cabecera;
            if e<40
                disp('Vuelva a intentar --> Energia muy baja');
                en=0;
                er=1;
            else
                [M,nbFrame] = framing2(s,100);
                [en,E,cv,ptv] = energia(nbFrame,M,s,100,0,7,5,s2);
            end
            if en==0
                if er==0
                    disp('Vuelva a intentar --> No es posible tratar la
señal');
                end
            else
                wavwrite(en,fs,'aeiou_test\s1');
                clc; cabecera;
                disp('Reconociendo ...');disp(' ');
                TEG(2);
                load letra;
                load distmin;
                if distmin>6
                    disp('Vuelva a intentar --> El reconocimiento no es
preciso');
                else
                    load pal;
                    disp(pal);
                    fprintf(com1,letra);
                end
            end
            salir;
            while 1
                ch=getkey;
                if ch==50
                    clc;
                    break;
                end
                if ch==49
                    clc;
                    break;
                end
```

```
            end
            if ch==50
                break;
            end
        end
    end

    if ch==51
        newobjs = instrfind;
        fclose(newobjs);
        clc; clear; close all;
        break;
    end

end
```

1.1.- train.m

```
function code = train(names, traindir, n, n2, m, p, cual, MuWind, Modi)

error(nargchk(7,9,nargin));
if nargin < 8
    MuWind=0.1;
    Modi='No';
elseif nargin < 9
    Modi='No';
end

k = 16; %El original es 16 % Numero de Centroids (caso VQ)
for i = 1:n
    file = sprintf('%ss%d.wav', traindir, i);
    disp(file);
    [s, fs] = wavread(file);
    v = parametrizar(s, fs, n2, m, p, cual, MuWind, Modi);
    code{i} = VQ(v, k); %Forma 1
end
```

1.1.1.- parametrizar.m

```
function v = parametrizar(s, fs, n, m, p, cual, MuWind, Modi)

% s --> Señal de voz arreglada
% fs --> Frecuencia de muestreo
% n --> Tamaño del frame
% m --> Tamaño del solapamiento
% p --> Numero de filtros o polos (depende del caso).
% cual --> Tipo de parametrización
% MuWind --> Constante de aprendizaje / Tipo de ventana (en caso de MFCC o
LPCC).
% Modi --> Opcion para modificar la forma de los filtros

error(nargchk(6,8,nargin));
if nargin < 7
    MuWind=0.1;
    Modi='No';
elseif nargin < 8
    Modi='No';
end

if cual=='MFCC'
```

```
    [M,nbFrame] = framing(s,n,m);
    M2 = windowing(M,n);
    fframe = fftFrame(nbFrame,M2);
    fb = filterBank(p, n, fs, 'MFCC', MuWind, Modi);
    n2 = 1 + floor(n / 2);
    z = fb * abs(fframe(1:n2, :)).^2;
    v = dct(log(z));
end

if cual=='LFCC'
    [M,nbFrame] = framing(s,n,m);
    M2 = windowing(M,n);
    fframe = fftFrame(nbFrame,M2);
    fb = filterBank(p, n, fs, 'LFCC', MuWind, Modi);
    n2 = 1 + floor(n / 2);
    z = fb * abs(fframe(1:n2, :)).^2;
    v = dct(log(z));
end

if cual=='LPLM'
    [M,nbFrame] = framing(s,n,m);
    M2 = windowing(M,n);
    M2=M2';
    y=s';
    for j=1:nbFrame
        a(p,n)=0;
        y=M2(j,:);
        for i=1:(n-p+1)-1
            yp=fliplr(y(i:i+p-1))*a(:,i);
            e=y(i+p)+yp;
            a(:,i+1)=a(:,i)-2*MuWind*e*(fliplr(y(i:i+p-1)))';
        end
        aa(:,j)=mean(a,2);
    end
    b(1:nbFrame)=1; aa=[b;aa]; clear a; a=aa;
    [p1,nf]=size(a);
    cc(p1-1,nf)=0;
    cm=(1:p1-1).^(-1);
    cm=cm*(-1);
    xm=-(1:p1-1);
    for k=1:nf
        ar=a(:,k)';
        cc(:,k)=filter(1,ar,ar(2:p1).*xm).*cm;
    end
    v=cc;
end

if cual=='LPCC'
    [M,nbFrame] = framing(s,n,m);
    M2 = windowing(M,n);
    a(p+1,nbFrame)=0;
    for i = 1:nbFrame
        a(:,i) = lpc(M2(:,i),p);
    end
    [p1,nf]=size(a);
    cc(p1-1,nf)=0;
    cm=(1:p1-1).^(-1);
    cm=cm*(-1);
    xm=-(1:p1-1);
    for k=1:nf
```

```
        ar=a(:,k)';
        cc(:,k)=filter(1,ar,ar(2:p1).*xm).*cm;
    end
    v=cc;
end
```

1.1.1.1.- framing.m

```
function [M,nbFrame] = framing(s,n,m)

% n --> Tamaño del frame
% m --> Tamaños del solapamiento

l = length(s);
nbFrame = floor((l - n) / m) + 1; %Redondea hacia abajo
for i = 1:n
    for j = 1:nbFrame
        M(i, j) = s(((j - 1) * m) + i); %Matriz n x nbFrame
    end
end
```

1.1.1.2.- windowing.m

```
function M2 = windowing(M,n)

% M --> Matriz n x nbFrame, cada columna es un frame de n muestras
% n --> Tamaño del frame

h = hamming(n); %Ventana Hamming
M2 = diag(h) * M; %Se aplica la ventana a cada Frame
```

1.1.1.3.- fftFrame.m

```
function frame = fftFrame(nbFrame,M2)

% nbFrame --> Numero de frames totales en la matriz M2
% M2 --> Matriz Transformada

for i = 1:nbFrame
    frame(:,i) = fft(M2(:, i)); %Se calcula la FFT de cada frame
end
```

1.1.1.4.- filterBank.m

```
function fb = filterBank(p,n,fs,tipo,wind,Modi)

% p --> Numero de filtros
% n --> Tamaño del frame
% fs --> Frecuecnia de muestreo
% tipo --> Tipo de eje de los filtros. Ej.: espaciado MEL o lineal
% wind --> Tipo de ventana. Ej.: triangular o otra


error(nargchk(5,6,nargin));

if tipo=='MFCC'
    f0 = 700 / fs;
    fn2 = floor(n/2); %Porque la transformada es par
```

```
    lr = log(1 + 0.5/f0) / (p+1); %Equivalente de la frecuencia maxima (eje
Mel)
    ale= (700 * (exp([0:(p+1)] * lr) - 1)); %Hace la correspondencia
logaritmica
    ale2bl2 = n*ale/fs;
    bl2=floor(ale2bl2)+1;
    %Hacer que todos sean impares
    for i=1:(p+2)
        impar=round(bl2(i)/2);
        if impar==(bl2(i)/2)
            %disp(bl2(i));
            bl2(i)=bl2(i)+1;
        end
    end
    %Verificar impares
    for i=1:p
        poi(i)=[bl2(i+2)-bl2(i)]+1;
    end
    %Hcer los filtros triangulares, convolucion de dos cuadrados
    fb(p,1 + floor(n / 2))=0;
    co=filterMod;
    for i=1:p
        imp=([bl2(i+2)-bl2(i)+2])/2;
        u=ones(1,imp);
        w=conv(u,u)-1;
        if wind=='tria'
            w=w/max(w);
            if Modi=='si'
                w=w*co(i);
                fb(i,bl2(i):bl2(i)+length(w)-1)=w;
            else
                fb(i,bl2(i):bl2(i)+length(w)-1)=w;
            end
        else
            wk=kaiser(length(w),2.5);
            wk=wk-min(wk);
            wk=wk/max(wk);
            if Modi=='si'
                wk=wk*co(i);
                fb(i,bl2(i):bl2(i)+length(w)-1)=wk;
            else
                fb(i,bl2(i):bl2(i)+length(w)-1)=wk;
            end
        end
    end
    %plot(linspace(0,5000,129),fb);pause;close;
end

if tipo=='LFCC'
    f0 = 700 / fs;
    fn2 = floor(n/2);
    lr = log(1 + 0.5/f0) / (p+1);
    ale=[0:(p+1)]*(fs/2)/(p+1);
    ale2bl2 = n*ale/fs;
    bl2=floor(ale2bl2)+1;
    %Hacer que todos sean impares
    for i=1:(p+2)
        impar=floor(bl2(i)/2);
        if impar==(bl2(i)/2)
            %disp(bl2(i));
```

```
            bl2(i)=bl2(i)+1;
        end
    end
    %Verificar impares - Se puede obviar este bloque --- Prueba
    for i=1:p
        poi(i)=[bl2(i+2)-bl2(i)]+1;
    end
    %Hacer los filtros triangulares, convolucion de dos cuadrados
    fb(p,1 + floor(n / 2))=0;
    co=filterMod;
    for i=1:p
        imp=([bl2(i+2)-bl2(i)+2])/2;
        u=ones(1,imp);
        w=conv(u,u)-1;
        if wind=='tria'
            w=w/max(w);
            if Modi=='si'
                w=w*co(i);
                fb(i,bl2(i):bl2(i)+length(w)-1)=w;
            else
                fb(i,bl2(i):bl2(i)+length(w)-1)=w;
            end
        else
            wk=kaiser(length(w),2.5);
            wk=wk-min(wk);
            wk=wk/max(wk);
            if Modi=='si'
                wk=wk*co(i);
                fb(i,bl2(i):bl2(i)+length(w)-1)=wk;
            else
                fb(i,bl2(i):bl2(i)+length(w)-1)=wk;
            end
        end
    end
    %plot(linspace(0,5000,129),fb);pause;close;
end
```

1.1.1.4.1.- fiterMod.m

```
function c=fiterMod()
x=linspace(0,1,40);
y=-log(x(2:40).*x(2:40));
y=y(1:20);
y=y/max(y);
c=y;
```

1.1.2- VQ.m

```
function r = VQ(d,k)
e   = .01;
r    = mean(d, 2); % Media de cada fila colocada en una posicion de la
columna resultante.
dpr = 10000; %Puede ser cualquier grande ((dpr-t)/t es el valor que se tiene
que minimizar)
for i = 1:log2(k) %Es igual a decir hasta 4, pero me ayuda por lo de r (se
duplica)
                  %16 se detiene (crea los 16 centroids)
    r = [r*(1+e), r*(1-e)]; %Aplica el paso 2 de LGB

    while (1 == 1)
```

```
        z = disteu(d, r);  %Se calcula la distancia euclideana de cada "d"
con todos los "x"
                           %Cada columna es la dimension de un punto. Ej.:
dimension 20
        [m,ind] = min(z, [], 2); % Buscar el "min value" / m --> Es la
columna de los menores
                                 %(entre las 2) / ind --> es el indice del
                           menor z
        t = 0;
        for j = 1:2^i
            si=size(d(:, find(ind == j)));
            if si(2)==0 %Corregido / El simbolo de distinto es: ~=
            else
                r(:, j) = mean(d(:, find(ind == j)), 2);
            end
            x = disteu(d(:, find(ind == j)), r(:, j)); %Calcula la distancia
de los menores
                                                       %(d) al nuevo r (solo
la cloumna j)
            %t2=t2+sum(x); %Da igual que el el FOR
            for q = 1:length(x)
                t = t + x(q);
            end
        end
        if t==0
            r(20,16)=0;
            return;
        end
        if (((dpr - t)/t) < e)
            break;
        else
            dpr = t;
        end
   end
end
```

1.1.2.1.- disteu.m

```
function d = disteu(x, y)

[M, N] = size(x);
[M2, P] = size(y);

if (M ~= M2)
    error('Matrix dimensions do not match.')
end

d = zeros(N, P);

if (N < P)
    copies = zeros(1,P);
    for n = 1:N
        d(n,:) = sum((x(:, n+copies) - y) .^2, 1);
    end
else
    copies = zeros(1,N);
    for p = 1:P
        d(:,p) = sum((x - y(:, p+copies)) .^2, 1)';
    end
end
```

```
d = d.^0.5;
```

1.2.- addnew.m

```
clc; cabecera;
disp('Grabando ...');
micrecorder=audiorecorder(10000,16,1);
record(micrecorder);
getkey2;
stop(micrecorder);
s=getaudiodata(micrecorder);
s=s-mean(s);
s2=s;
s=prenfasis(s);
s=s-mean(s);
e=calenergia(s2);
disp(e);
er=0;
clc; cabecera;
if e<40
    disp('Vuelva a intentar --> Energia muy baja');
    er=1;
else
    [M,nbFrame] = framing2(s,100);
    [en,E,cv,ptv] = energia(nbFrame,M,s,100,0,5,7,s2);
    if en==0
        if er==0
            disp('Vuelva a intentar --> No es posible tratar la señal');
        end
        er=1;
    else
        load secudb;
        resp=num2str(secudb);
        resp=strcat('s',resp);
        wavwrite(en,fs,['db_train\' resp]);
        clc; cabecera;
        secudb=secudb+1;
        save('secudb','secudb');
    end
end
```

1.2.1.- prenfasis.m

```
function s = prenfasis(s)

% p --> Señal de voz arreglada (despues de haber calculado su energia)

s = filter([1 -0.95], 1, s);
```

1.2.2.- calenergia.m

```
function e=calenergia(s)
e=sum(s.*s);
```

1.2.3.- framing2.m

```
function [M,nbFrame] = framing2(s,n)
```

```
% n --> Tamaño del frame

l = length(s);
nbFrame = l/n; %Numero de Frames
for i = 1:n
    for j = 1:nbFrame
        M(i, j) = s((j - 1)*n + i); %Matriz n x nbFrame
    end
end
```

1.2.4.- energia.m

```
function [en,E,cv,ptv] = energia(nbFrame,M,s,n,m,por,comp,s2)

% n --> Tamaño del frame
% m --> No se usa!!


for j = 1:nbFrame
    E(j)=sum(M(:,j).*M(:,j));
end

%z=find(s>0.8) % ---> Tipo un comparador con OPAM
cv=ones(1,nbFrame)*max(E)*por/100;
c=cv(1);
pt=find(((E>(c)) & (E<(comp*c))));
%disp(pt);pause;

if size(pt)==[1 0]
    en=0;
    E=0;
    cv=0;
    ptv=0;
    %disp('Return1');
    return;
end

for i = 1:length(pt)
    ptv((pt(i)))=E((pt(i)));
end

if size(ptv)==[1 0]
    en=0;
    E=0;
    cv=0;
    ptv=0;
    %disp('Return2');
    return;
end

l1=(pt(1)*n)-5*n+1;
l2=(pt(length(pt))*n)+8*n+1;
if l1<0
    en=0;
    E=0;
    cv=0;
    ptv=0;
    %disp('Return3');
    return;
```

```
end
if l2>length(s)
    en=0;
    E=0;
    cv=0;
    ptv=0;
    %disp('Return4');
    return;
elseif (l2-l1)<256*4
    en=0;
    E=0;
    cv=0;
    ptv=0;
    %disp('Return5');
    return;
else
    en=s2(l1:l2);
    en=en-mean(en);
    en=en*0.9/max(en);
    en=resample(en,2,1);
    en=en-mean(en);
end
```

1.3.- TEG

```
function TEG(tipo)

if tipo == 1
    load espacio;
    nd=6;
    np=22;
    n=256;
    m=100;
    p=20;
    cual='MFCC';
    MuWind='tria';
    Modi='no';
    code = train(espacio, 'train\', nd, n, m, p, cual, MuWind, Modi);
    test(espacio, 'test\', np, code, n, m, p, cual, MuWind, Modi);
end

if tipo == 2
    load espacio;
    load secu;
    nd=secu-1;
    clear secu;
    np=1;
    n=256;
    m=100;
    p=20;
    cual='MFCC';
    MuWind='tria';
    Modi='no';
    %code = train(espacio, 'aeiou_train\', nd, n, m, p, cual, MuWind, Modi);
    %test(espacio, 'aeiou_test\', np, code, n, m, p, cual, MuWind, Modi);
    load code;
    test(espacio, 'aeiou_test\', 1, code, n, m, p, cual, MuWind, Modi);

end
```

```
if tipo == 3 %Para pruebas estadisticas
    clc; clear all; close all;
    numError=0;
    save('numError','numError');
    numErrorA=0;
    save('numErrorA','numErrorA');
    numErrorE=0;
    save('numErrorE','numErrorE');
    numErrorI=0;
    save('numErrorI','numErrorI');
    numErrorO=0;
    save('numErrorO','numErrorO');
    numErrorU=0;
    save('numErrorU','numErrorU');
    arrA=0;
    save('arrA','arrA');
    arrE=0;
    save('arrE','arrE');
    arrI=0;
    save('arrI','arrI');
    arrO=0;
    save('arrO','arrO');
    arrU=0;
    save('arrU','arrU');
    load names;
    nd=210;
    np=105;
    n=256;
    m=100;
    p=20;
    cual='MFCC';
    MuWind='tria';
    Modi='no';
    code = trainPr(names, 'db_trainPr\', 200, n, m, p, cual, MuWind, Modi);
    testPr(names, 'match_test\', 100, code, n, m, p, cual, MuWind, Modi);
    disp(' ');
    load numError;
    load numErrorA;
    load numErrorE;
    load numErrorI;
    load numErrorO;
    load numErrorU;
    load arrA;
    load arrE;
    load arrI;
    load arrO;
    load arrU;
    disp(strcat('Errores A --> ',num2str(numErrorA),' -->',MH(arrA)));
    disp(strcat('Errores E --> ',num2str(numErrorE),' -->',MH(arrE)));
    disp(strcat('Errores I --> ',num2str(numErrorI),' -->',MH(arrI)));
    disp(strcat('Errores O --> ',num2str(numErrorO),' -->',MH(arrO)));
    disp(strcat('Errores U --> ',num2str(numErrorU),' -->',MH(arrU)));
    disp(strcat('Total ------> ',num2str(numError)));
end
```

1.3.1-. test.m

```
function test(names, testdir, n, code, n2, m, p, cual, MuWind, Modi)

error(nargchk(8,10,nargin));
```

```
if nargin < 9
    MuWind=0.1;
    Modi='No';
elseif nargin < 10
    Modi='No';
end

for k = 1:n
    %reco=dispo(k); %Solo para prueba de estadisticas
    %if reco==1 %Solo para prueba de estadisticas
        file = sprintf('%ss%d.wav', testdir, k);
        [s, fs] = wavread(file);
        v = parametrizar(s, fs, n2, m, p, cual, MuWind, Modi);
        cod = VQ(v,16); %Caso 1
        distmin = inf;
        k1 = 0;
        for l = 1:length(code)      % Calculo de la distorsion --> Menor
            dista = disteu2(cod, code{l}); %Caso 1
            %d = disteu(v, code{l}); %Caso 2
            %dist = sum(min(d,[],2)) / size(d,1);
            %disp(dist);
            if dista < distmin
                distmin = dista;
                k1 = l;
            end
        end
        k1=dispo(k1);
        letra=num2letra(k1);
        %msg = sprintf('Speaker %d matches with speaker %d', k, k1);
        %res=[' ' num2str(distmin)];
        %disp(strcat(msg,' --> ',res));
        dijis='Dijiste ---->';
        vocal=dispo(k);
        vocalO=num2letra(vocal);
        if vocalO==letra
            errorVo='';
        else
            errorVo=' -----> Se pelo AQUI!!';
        end
        dmin=num2str(distmin);
        raya=' ----- ';
        %disp(strcat(vocalO,dijis,letra,raya,dmin,errorVo)); %Para pruebas
        %disp(strcat(dijis,letra,raya,dmin)); %Para reconocer en vivo
        %Aqui hay que hacer lo de la conversion de 1,6,11 ...
        %disp(strcat(dijis,letra)); %Para reconocer en vivo
        pal=strcat(dijis,letra);
        save('pal','pal');
        save('letra','letra');
        save('distmin','distmin');
    %end %Solo para prueba de estadisticas
End
```

1.3.1.1.- disteu2.m

```
function d = disteu2(x, y)

[M, N] = size(x);
[M2, P] = size(y);

if (M ~= M2)
```

```
    error('Matrix dimensions do not match.')
end

d = zeros(N, P);

if (N < P)
    copies = zeros(1,P);
    for n = 1:N
        d(n,:) = sum((x(:, n+copies) - y) .^2, 1);
        %disp(d(n,:));pause;
    end
    %disp(d);pause;
else
    copies = zeros(1,N);
    for p = 1:P
        d(:,p) = sum((x - y(:, p+copies)) .^2, 1);
        %disp(d(:,p));pause;
    end
    %disp(d);pause;
    d=d';
    %disp(d);pause;
end

d = d.^0.5;
d = sum(min(d,[],2)) / size(d,1);
```

1.3.1.2.- dispo.m

```
function k=dispo(k1)
while k1>5
    %disp(num2str(k1));
    k1=k1-5;
    %pause;
end
k=k1;
```

1.3.1.3.- num2letra.m

```
function letra=num2letra(k1)
if k1==1
    letra=' A';
end

if k1==2
    letra=' E';
end

if k1==3
    letra=' I';
end

if k1==4
    letra=' O';
end

if k1==5
    letra=' U';
end
```