UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DIVISIÓN DE ESTUDIOS DE POSGRADO

DISEÑO DE UN SINTETIZADOR DE VOZ DEL IDIOMA ESPAÑOL HABLADO EN MÉXICO

T E S I S

QUE PARA OBTENER EL GRADO DE

MAESTRÍA EN INGENIERÍA
PROCESAMIENTO DIGITAL
DE SEÑALES E IMÁGENES

PRESENTA
LILIA ELENA DE LA VEGA SEGURA

DIRECTOR DE TESIS
DR. ABEL HERRERA CAMACHO

CIUDAD UNIVERSITARIA **2007**

Dedicatorias

Agradezco infinitamente a **Dios**
por la vida y ser la luz
que me guía diario.

A mis papas Alejandro y Carmen,
por su gran amor, paciencia
y ejemplo de vida.

A mi hermana Care,
por su apoyo en todo momento
y enseñarme que
la tenacidad, constancia y alegría
llevan al éxito.

A mis hermanos Ale e Isaac,
porque siempre han sido
un ejemplo a seguir.

A mi hermano Edy y cuñada Dulce,
por sus consejos e insistir
que las metas
deben concluirse.

A mis sobrinos,
Ascan, Aldo, Dulcita,
José María, Armida,
Olaf y Patricio
por todo su cariño.

A mis tíos y primos.

A mis amigos Daniel y Toño Paniagua
por estar siempre ahí.

A mis amigos de la Escuela
de Ingeniería de la UP.

Agradecimientos

Agradezco al Dr. Abel Herrera Camacho por compartir sus conocimientos y sobre todo por aceptar pacientemente la dirección de esta tesis.

A los profesores de posgrado de la Facultad de Ingeniería de la UNAM, a quienes debo gran parte de mi formación profesional. En especial al Dr. Felipe Orduña Bustamante, Dr. Luis Alberto Pineda Cortés, Dr. Alfonso Medina Urrea y Dr. Jesús Savage Carmona, gracias por sus comentarios y correcciones a este trabajo.

A la Universidad Panamericana por su apoyo incondicional para concluir este trabajo.

A los Ingenieros Pedro Creuheras Vallcorba y Antonio Castro D´Franchis por su ejemplar calidad humana y profesional, por sus consejos y apoyo para poder concluir esta meta tan importante de crecimiento personal y profesional... Muchas Gracias.

A mis compañeros y amigos de la Escuela de Ingeniería, que a lo largo de más de trece años, con todas sus enseñanzas me han motivado a seguir creciendo sobre todo como persona: Ing. Alfredo González, M.C.C. Félix Martínez, Dr. Roberto González, M.C. Silvia Gómez, M.I. Paco Ortiz, M. I. Rodolfo Cobos, Ing. Paola Sánchez.

A Lupita, Irma, Connie, Aurora, Bety, , Marta y Mayte porque su apoyo siempre ha sido invaluable.

A los alumnos que apoyaron con tantas horas de trabajo, les agradezco mucho toda su ayuda para esta tesis: Ma. Lorena Siqueiros Fernández, Héctor Manuel Icaza Heredia y en especial a Jean Dirk De Rubens Von Sparr.

Índice

Introducción

Capítulo 1
Técnicas de procesamiento de voz

Capítulo 2
Fisiología de la voz humana

Capítulo 3
Características de la articulación en Español

Capítulo 4
Técnicas de síntesis de voz

INTRODUCCIÓN

Antecedentes

Debido a la evolución de los sistemas de comunicaciones hacia esquemas totalmente digitales, las tecnologías para el procesamiento digital de voz han experimentado un creciente desarrollo desde hace varias décadas. La necesidad de arquitecturas de comunicaciones más eficientes ha acelerado este crecimiento en diversas áreas: depuración constante de algoritmos de codificación; un rápido desarrollo en la electrónica que se ha especializado en este campo; asimismo las arquitecturas de redes de comunicaciones sobre las cuales se transmite este medio incrementan sus capacidades rápidamente.

El desarrollo de las técnicas de síntesis de voz se ha visto influenciado por diversos factores que han dado por resultado una gama muy extensa de esquemas y tecnologías. Estos factores han sido principalmente, la calidad deseada, la disponibilidad de electrónica capaz de ejecutar los algoritmos de síntesis de voz en el tiempo requerido y el costo de cada tecnología. La síntesis de voz está completamente ligada al avance de las computadoras. Es importante dividir el campo de investigación en dos áreas de estudio que se han desarrollado en forma separada.

La primera de éstas se refiere a la síntesis de bajo nivel, por ejemplo la producción real del sonido la cual simula la señal del lenguaje hablado. Dentro de esta área se están utilizando consideraciones como el tipo de modelo que se quiere adoptar ya sea por síntesis espectral, síntesis por articulación o síntesis en el dominio del tiempo. El tamaño de las unidades utilizadas para concatenar las palabras y las técnicas de procesamiento de la señal se usan en los diferentes métodos para generar la salida de la voz.

La segunda área, síntesis de alto nivel, maneja la conversión de textos escritos o una representación simbólica de conceptos dentro de una representación abstracta de señales acústicas eventuales.

El desarrollo tecnológico y de investigación que existe en procesamiento digital de voz se ha dado principalmente en Estados Unidos, por lo que en el área de síntesis de voz se ha dado impulso en el idioma inglés. Algunos países de habla diferente han tenido que ampliar sus conocimientos desarrollando en lenguaje propio como ha sido el finlandés, japonés, francés, entre otros. Cada lenguaje tiene sus propios fonemas, grafos, pronunciación, reglas ortográficas, por lo que hay que adecuar a éstos la teoría y técnicas diferentes que existen.

Objetivos

El objetivo principal de esta tesis es diseñar un sintetizador híbrido de palabras en español basado en la técnica de síntesis de voz por concatenación, y la utilizada es la PSOLA, por sus siglas en inglés "Pitch Synchronous Overlap and Add".

Este estudio pretende, además, mostrar y analizar las diferencias entre un programa de síntesis de voz de baja calidad y uno programado con algoritmo de síntesis de voz por concatenación.

Organización del trabajo

En el Capítulo 1 se estudia en forma general el procesamiento de voz y las técnicas de análisis que se han desarrollado en los últimos años. Se proporciona un panorama que abarca los distintos aspectos de técnicas espectrales aplicada al análisis de voz. Se han desarrollado varias técnicas y entre ellas se encuentra la síntesis de voz con sus diferentes métodos, algoritmos y técnicas, mismas que se estudiarán en capítulos subsecuentes.

Dentro del Capítulo 2, se analiza la fisiología de la voz humana, como la producción de la voz, percepción de la voz y descripción del aparato fono-articualdor, que permite comprender el funcionamiento del aparato vocal humano, mismo que trata de simularse mas adelante. Se realiza un estudio general de la generación de la voz humana en forma simulada.

Con estos dos capítulos como marco de referencia, en el Capítulo 3 se especifica la producción de la voz humana en idioma español; donde se hace la clasificación de los fonemas, puntos de articulación de los fonemas y reglas ortográficas de nuestro idioma.

En el Capítulo 4, se lleva a cabo un estudio de la evolución de los sintetizadores de voz, así como su clasificación y desarrollo, la síntesis por formantes, articulatoria y por concatenación que actualmente se han desarrollado en diferentes idiomas. Se analiza específicamente el método PSOLA, dentro de los que existen en síntesis de voz por concatenación.

Finalmente en el Capítulo 5 se lleva a cabo el diseño de un sintetizador de voz del español hablado en México. Se plantean dos programas desarrollados en lenguaje C y otro en C#; este primero es un algoritmo simple de síntesis de voz mientras que el segundo es una modificación aplicando técnicas de concatenación. Se plantean las diferencias básicas como calidad, naturalidad y la flexibilidad de reproducir cualquier mensaje llevando a cabo un procesamiento relativamente simple y rápido

CAPÍTULO 1
TÉCNICAS DE PROCESAMIENTO DE VOZ

Breve introducción histórica del procesamiento de voz.

El presente capítulo está basado en diferentes autores que se han dedicado al estudio del procesamiento digital de señales; Chris Rowden en los capítulos 1, 2 y 3 de su libro *Speech Processing*, F. A. Westall, capítulos 1, 10 y 11 del libro *Digital Signal Processing in Telecommunications*, Panos E. Papamichalis capítulos 1 y 4 *Practical Approaches to Speech Coding.*

Debido a la evolución de los sistemas de comunicaciones hacia esquemas totalmente digitales, las tecnologías para el procesamiento digital de voz han tenido un creciente desarrollo desde hace varias décadas. La necesidad de arquitecturas de comunicaciones más eficientes ha acelerado este crecimiento en diversas áreas; nuevos algoritmos en técnicas de procesamiento de voz como codificación, síntesis, reconocimiento, entre otras, así como un rápido desarrollo en la electrónica que se ha especializado en este campo.

Específicamente en el área de síntesis, uno de los retos que tuvo la persona humana en el siglo pasado fue el lograr que una máquina aprendiera a hablar, a leer y a escribir. Especialmente en la discapacidad visual, una máquina que lee textos es una ayuda de gran importancia para el acceso a la cultura de las personas con este tipo de minusvalías.

Hacia finales de los años 70, aparecieron las primeras máquinas capaces de convertir texto tecleado en voz (converso texto-voz), que junto con los programas de reconocimiento óptico de caracteres (Optical Carácter Recognition), en inglés, produjeron los primeros sistemas comerciales para leer libros en voz alta.

Posteriormente, a finales de los años 80, las principales operadoras telefónicas del mundo tomaron cartas en el asunto, y produjeron sus propios conversores texto a voz, en un conjunto de idiomas diversos. Se puede mencionar Bell Labs de ATT, más tarde Lucent Technologies y ATT Research, British Telecom, France Telecom, CSELT, NTT, por mencionar algunas. El interés de todas estas últimas, se centra sobre todo en la automatización de servicios de información telefónica, en los que los datos disponibles están sobre todo almacenados en el ordenador en modo texto. Precisamente los servicios de información y atención telefónica automática son uno de los pilares económicos importantes de todos los desarrollos actuales de la Tecnología del Habla.

1.1 Análisis del lenguaje hablado

La forma más primitiva de la que parte todo análisis es la de una señal continua en el tiempo y limitada en su ancho de banda. En cuanto a procesamiento digital, se refiere a la parte de archivos donde esta señal una vez digitalizada se ha almacenado. Los formatos más comunes para archivos de voz están formados por muestras de 12 ó 16 bits, cada cual con una tasa de muestreo entre los 8 y los 14 kHz. En muchas de las computadoras pequeñas es posible digitalizar la voz a 10 Khz. con un convertidor A/D y grabar directamente la secuencia en disco. Para tener una idea de la capacidad de almacenamiento requerida, por ejemplo, para un minuto de voz digitalizada con las características mencionadas anteriormente se requieren 1.2 Mbytes de espacio en disco.

Usualmente, se acostumbra accesar los archivos de voz en bloques de 512 ó 1024 muestras de voz, siendo el equivalente a 50 ms de señal, tiempo suficiente para acomodar la duración de un fonema.

Tradicionalmente, el nivel más bajo de análisis es el nivel de forma de onda, donde se prepara la señal de manera que pueda ser procesada más fácilmente en el siguiente nivel, denominado nivel de segmento o de bloque. Es en estos dos niveles donde el procesamiento digital, especialmente en lo que a codificación se refiere es más significativo. El siguiente nivel, llamado de sentencia y donde se conjuntan ideas e información para conformar un lenguaje, pertenece en mayor medida a otros campos de desarrollo. Asimismo se define un cuarto nivel –el nivel de aplicación -, el cual involucra la interacción entre los sistemas de lenguaje hablado y el ser humano.

1.2 Análisis a nivel de señal

Al análisis a nivel de forma de onda de una secuencia de muestras se le conoce también como *análisis a nivel de señal*. En este análisis, las características de la señal se estiman sobre un número de muestras utilizando un filtro promediador móvil. A continuación, se enuncian algunas de las principales operaciones útiles en este nivel de análisis, así como sus aplicaciones más comunes.

1.2.1 Promedio de magnitud

Esta operación es tal vez la forma más sencilla de detectar la presencia de señal. Consiste simplemente en sumar los valores de un conjunto de muestras y dividir la suma entre el número de ellas:

$$M(n) = \frac{1}{N} \sum_{m=n-N+1}^{n} |x(m)|$$

Ecuación 1.1

Una alternativa para calcular el promedio es efectuar la convolución de la magnitud de las muestras con una función de ventaneo *w(n-m).*

$$M(n) = \frac{1}{N} \sum_{m=-\infty 1}^{\infty} |x(m)| w(n-m)$$

Ecuación 1.2

Con el objeto de minimizar los cálculos necesarios, es posible obtener las muestras que se promediarán a una frecuencia más baja que la frecuencia de muestreo original. A este proceso se le llama comúnmente *down-sampling*.

El promedio de magnitud suele ser útil como parte de un control automático de ganancia (ACG) que asegure el acondicionamiento de la señal para etapas posteriores de procesamiento.

1.2.2 Función de densidad en amplitud

La función de densidad enunciada anteriormente, evalúa las características de la señal en un tiempo corto. Cuando se les requiere conocer en un tiempo más largo, se utiliza la función de densidad en amplitud, la cual evalúa la distribución estadística de las muestras. El conocimiento de esta función permite, por ejemplo, escoger el rango correcto de voltaje a la entrada de un convertidor A/D. Esta función puede ser estimada determinando la proporción *p(x)* de muestras x_i que están dentro del intervalo $x < x_i < x+dx$. Es obvio que cualquiera de estas muestras se encuentra dentro del intervalo (-∞,∞).

$$\int_{-\infty}^{\infty} p(x)dx = 1$$

Ecuación 1.3

Una aproximación usada muy comúnmente es la función de densidad de Laplace *p(x)*, la cual se normaliza también para hacer que el valor rms σ_x sea igual a la unidad:

$$p(x) = \frac{1}{\sqrt{2\sigma_x}} e^{\left(-\frac{\sqrt{2}|x|}{\sigma_x}\right)}$$

Ecuación 1.4

La función definida tiene la ventaja de ser sensible al tratamiento analítico, además de registrar asimetrías con respecto al cero en el eje del tiempo.

1.2.3 Varianza

La varianza, que se define como el cuadrado de la diferencia entre el valor de la muestra y el promedio, puede proporcionar una medida de la energía contenida en la señal. Para una señal promediada en cero, la energía en un promedio corto puede ser definida como el promedio de los cuadrados de las muestras.

$$E(n) = \frac{\sum_{m=-\infty}^{\infty} [x(m)w(n-m)]^2}{N}$$

Ecuación 1.5

Donde *w(n)* es una función de ventana aplicable a las muestras. Si se define un filtro *h(n),* el cual es teóricamente el cuadrado de la función de ventana, normalizándolo:

$$h(n) = \frac{1}{N} w^2(n)$$

Ecuación 1.6

la ecuación E(n) se redefine como:

$$E_n = \sum_{m=-\infty}^{\infty} x^2(m)h(n-m)$$

Ecuación 1.7

Sin embargo, en la práctica es común usar una ventana normalizada al tamaño *N* de la misma. La energía promedio de un conjunto de muestras denota gran diferencia entre las secuencias de sonidos vocales y no vocales. Normalmente, se le utiliza como detector de conversación pero en combinación con un detector de cruce por cero.

1.2.4 Tasa de cruce por cero

La frecuencia a la cual se presentan los cruces por cero es un indicador claro de que la señal es un sonido vocal o no vocal. Para los segmentos vocales dicha tasa es de aproximadamente 0.5 cruces/ms y para los segmentos no vocales se eleva hasta 3 cruces/ms. Usualmente, se calcula mediante el algoritmo siguiente:

$$Z(n) = \frac{1}{2N} \sum_{m=-\infty}^{\infty} |sign(x(m)) - sign(x(m-1))| \, w(n-m)$$

Ecuación 1.8

donde la función *sign(x(m))* se define como:

$$sign(x(m)) = \begin{cases} +1 & x(m) \geq 0 \\ -1 & otro\ caso \end{cases}$$

Ecuación 1.9

El cálculo de la función *Z(n)* se ve afectado por el ruido y por cualquier pequeño corrimiento de c.d. que pudiera presentarse. Este efecto indeseado puede corregirse fácilmente con la implementación de un filtro pasoaltas con frecuencia de corte cercana a los 70 Hz.

1.3 Análisis a nivel de segmento

Es en este nivel donde se realizan los procesos de mayor importancia utilizados por los *vocoders*[1]. Existen algunas características de la señal, las cuales tienen mayor ponderación cuando el análisis se realiza a nivel de bloques. Tal es el caso de los algoritmos que obtienen la respuesta en frecuencia como la DFT y la FFT.

Como un paso previo al procesamiento de un bloque, suele aplicarse a éste una función de ventana, la cual tiene por objeto conformarlo de manera que la siguiente etapa de procesamiento se efectúe de manera más eficiente. Una familia de ventanas cuya respuesta en frecuencia es especialmente apreciada es aquella que modela el bloque mediante una función senoidal. Casos especiales de esta familia son la ventana de Hamming y la ventana de Hann expresadas como:

$$w(n) = a - (1-a)\cos\left(\frac{2\pi n}{N-1}\right)$$

Ecuación 1.10

[1] La palabra vocoder proviene del anglicismo voice-coding, es decir, codificación de voz.

donde la función expresada anteriormente queda definida en el intervalo $0 \leq n \leq N-1$ y toma el valor de *cero* en otro caso. Para la ventana de Hamming *a=0.54* y para Hann *a=0.5*.

Por lo general, se prefiere la ventana de Hamming debido a que presenta una transición más abrupta en la magnitud a medida que se sopesan las muestras más alejadas del centro de la ventana.

Para asegurar que todas las muestras tengan el mismo peso en los cálculos, se acostumbra traslapar las ventanas adyacentes de manera que las muestras que tenían menor magnitud al evaluar un bloque sean mayormente consideradas en el siguiente.

El hecho de procesar digitalmente una señal que en su origen fue continua, acarrea necesariamente pérdidas de información, que se traducen en la aparición de lóbulos espectrales falsos o deformados, cuando se obtiene la respuesta de la señal a la frecuencia mediante algoritmos como la DFT. El uso de la ventana de Hamming disminuye en gran medida este efecto.

El multiplicar por cero todas aquellas muestras que están fuera de la ventana de longitud *N*, es un proceso que se conoce como *zero-padding* que se efectúa con el objeto de insertar la ventana inicial dentro de otra ventana más grande de longitud *N*. El uso de esta técnica tiene como consecuencia una disminución aún mayor en la magnitud de los lóbulos indeseables.

Una característica poco deseada en el espectro de la señal de la voz es la atenuación que crece a frecuencias altas. Para lograr un espectro uniforme se da una preamplificación a estas frecuencias utilizando un filtro de preénfasis, el cual tiene una función de transferencia H_{preenf} con un cero de baja frecuencia:

$$H_{preenf}(z) = 1 - az^{-1}$$

Ecuación 1.11

Una vez realizado el procesamiento requerido, la señal se convierte de nuevo a una señal auditiva canalizándola a través de un filtro de deénfasis que restaura el espectro original. El filtro de deénfasis tiene una función de transferencia dada por:

$$H_{deenf}(z) = \frac{1}{1 - az^{-1}}$$

Ecuación 1.12

usualmente el coeficiente "*a*" toma valores en el intervalo $0.95 \leq a \leq 0.98$.

1.4 Técnicas espectrales aplicadas al análisis de voz

La transformada de Fourier es una herramienta de gran utilidad y de uso generalizado para diversas aplicaciones. Sin embargo, para el caso del análisis automático de la voz existen técnicas de mayor eficiencia, como la predicción lineal y el análisis cepstral. Ambas técnicas se aplican cuando se analiza la señal vocal bajo el *modelo excitación/filtro*, figura 1.1

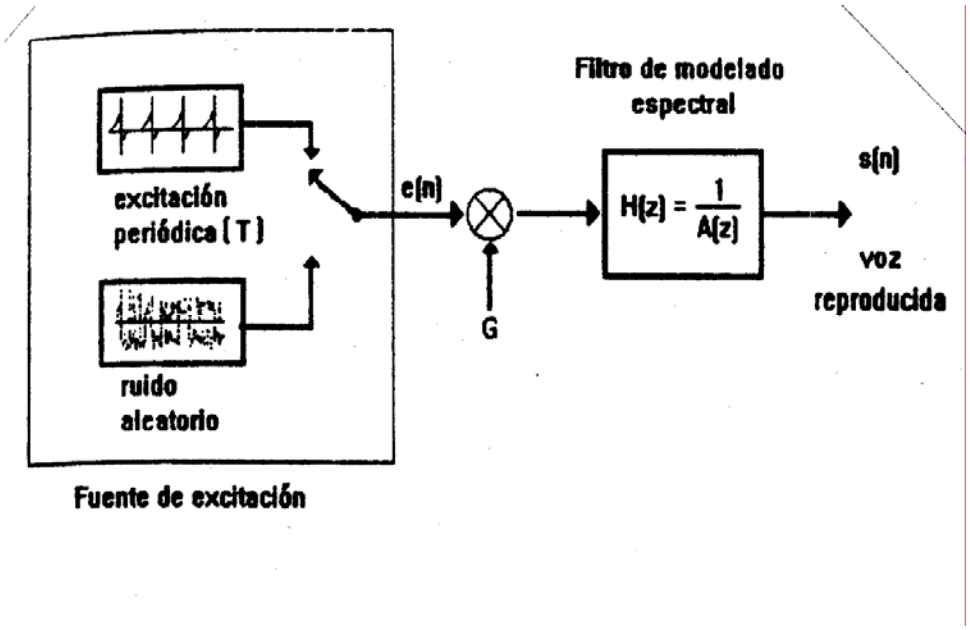


Figura 1.1 Modelo de excitación /filtro

Bajo este esquema, los pulsos que caracterizan el movimiento de la laringe se modelan como un tren de pulsos a una frecuencia fundamental (en el caso de sonidos *vocales*). En el caso de los sonidos *no vocales* como los fricativos en el idioma inglés, */th/, /s/, /f/,* la señal se asemeja mucho al ruido blanco. En el caso de algunos fricativos producidos en el fondo de la garganta como */h/* y */sh/* su espectro suele asemejarse al de un sonido vocal con frecuencias formantes bien definidas. Una vez obtenida la excitación (*filtro de análisis*) debe definirse también un filtro que modele espectralmente la excitación, de manera que puedan ser alineadas las frecuencias formantes para lograr que la señal obtenida sintéticamente tenga un espectro semejante a la señal original.
En este contexto, las tareas principales para realizar el modelado resultan ser:

- Determinar si el segmento en proceso es vocal o no vocal.
- Si el segmento es vocal, determinar la frecuencia fundamental.
- Determinar los parámetros del filtro modelador.

Para obtener de la señal eléctrica la información anterior, existen dos escuelas de análisis que se mencionan a continuación.

1.4.1 Predicción lineal

La predicción lineal se ha convertido en una de las técnicas más populares en el análisis del lenguaje hablado al enfocársele desde el modelo excitación/filtro. Existe una amplia variedad de aplicaciones de ella; su fácil entendimiento así como la eficiencia con la que se le puede implementar la hace una herramienta muy atractiva. El modelo general de la predicción lineal (LP) tiene la forma:

$$s_n = \sum_{k=1}^{p} a_k s_{n-k} + G \sum_{l=0}^{q} b_1 u_{n-1}, b_0 = 1$$

Ecuación 1.13

donde s_n representa las salidas del sistema, u_n son las entradas del sistema, y los coeficientes de cada una de ellas son a_k; k=1,…,p y b_l ; l=1,…q, además del factor de ganancia G. La idea entonces es que conociendo las últimas *p* salidas así como las últimas *q* entradas, adecuando los coeficientes a_k y b_l es posible predecir la siguiente salida del modelo mediante una combinación lineal de las

salidas y las entradas. Haciendo un reacomodo de la ecuación anterior y transformando al dominio de *z*, la función de transferencia del sistema se define:

$$H(z) = \frac{S(z)}{U(z)} = G\frac{1 + \sum_{l=1}^{q} b_l z^{-1}}{1 - \sum_{k=1}^{p} a_k z^{-k}} = \frac{B(z)}{A(z)}$$

Ecuación 1.14

donde los polos del sistema son las raíces de A(z) y los ceros son las raíces de B(z). Retomando la idea de la ecuación, en los sistemas de voz por lo general la entrada u_n no siempre se encuentra disponible, por lo que el algoritmo se limita entonces a calcular la posible salida en base a las previas salidas obtenidas, de donde la ecuación se redefine como:

$$\tilde{s}_n = \sum_{k=1}^{p} a_k s_{n-k}$$

Ecuación 1.15

El error de predicción, o residuo, suele ser un parámetro útil en el análisis automático de la voz, y se define como la diferencia que existe entre el valor predicho por el modelo y el valor real muestreado.

$$e_n = s_n - \sum_{k=1}^{p} a_k s_{n-k}$$

Ecuación 1.16

En una implantación eficiente se busca reducir el error cuadrático total, el cual está dado por la suma de los cuadrados de la última *n* errores:

$$E = \sum_{n} e^2{}_n$$

Ecuación 1.17

Existen a partir de este punto dos tendencias encaminadas a optimizar los coeficientes a_k: el método de autocorrelación y el método de covarianza .
Para un segmento de voz en el cual las características se consideran relativamente estacionarias, las secuencias de muestras de voz representadas por s_n pueden ser substituidas por *p* coeficientes a_k de un filtro predictor y por la secuencia e_n llamada señal de residuo.

La señal de residuo en segmentos vocales es por lo general de magnitud pequeña excepto por discontinuidades periódicas que coinciden precisamente con el inicio de los pulsos de la laringe.

Para los segmentos no vocales la señal de residuo (residuo LP) es semejante al ruido blanco con un nivel pequeño y uniforme. Los coeficientes del filtro predictor en un segmento de condiciones estacionarias cambian a un ritmo relativamente lento.

De hecho, esta característica de modelo excitación/filtro es aprovechada para economizar el número necesario de bits para codificar dicho segmento. La tasa de actualización de los coeficientes se le llama *tasa de segmentación* la cual está en el rango de 10 a 25 ms que para voz muestreada para 8khz equivale de 80 a 200 muestras. Por lo general, el número de muestras utilizadas por el predictor lineal está entre 10 y 20.

1.4.2 Análisis cepstral

Un método alternativo para separar la frecuencia fundamental omitida por la laringe de su envolvente espectral es el análisis cepstral. Haciendo referencia al aparato vocal humano como un sistema con una función de transferencia en el tiempo denotada por *v(t),* la cual modela el tracto vocal, con una entrada *e(t)* la cual se identifica con la excitación, se tiene a la salida del sistema una señal *s(t),* la cual representa la voz a la salida del tracto vocal.

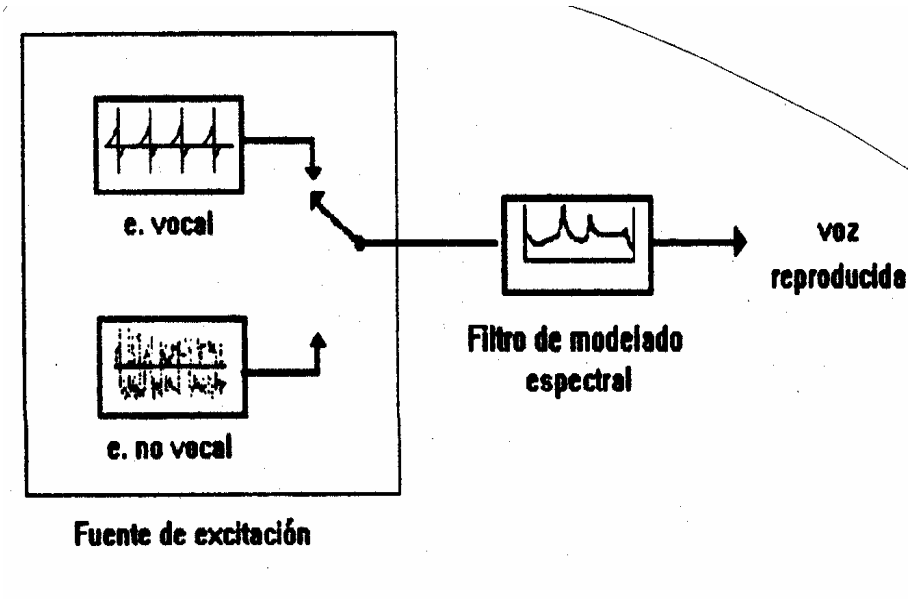


Figura 1.2 Modelo excitación/filtro en el dominio del tiempo

donde S(ω), E(ω) y V(ω) son las respectivas transformadas de Fourier de las señales ya conocidas. Ahora bien, ya en el dominio de la frecuencia es relativamente sencillo separar ambas señales aplicando la función logaritmo y sus propiedades

$$\log\{S(\omega)\} = \log\{E(\omega)\} + \log\{V(\omega)\}$$

Ecuación 1.18

Al graficar la función log S(ω), es posible identificar una serie de lóbulos igualmente espaciados, montados sobre una curva que delinea la envolvente espectral. El espaciamiento entre dichos módulos, dado en unidades de frecuencia, es el inverso de la componente fundamental que se busca (Figura 1.3a). El último paso en el procedimiento es regresar al dominio del tiempo aplicando la trasformada inversa de Fourier. En esta última etapa, se realiza la separación completa de la excitación y su envolvente (Figura 1.3b). La función obtenida tiene unidades inversas a la frecuencia, es decir, segundos, por lo que a la variable se le llama *quefrencia* y a la función en sí se le denomina *cepstrum.*

Al proceso de separación realizado, en muchas ocasiones se le conoce como *liftrado.*

En una gráfica de cepstrum para un segmento vocal, se destacan en su primera parte una serie de lóbulos que representan las componentes de baja frecuencia de la envolvente espectral así como un pico angosto y pronunciado en la parte derecha de la gráfica que identifica la frecuencia fundamental o *pitch* (Figura 1.3c).

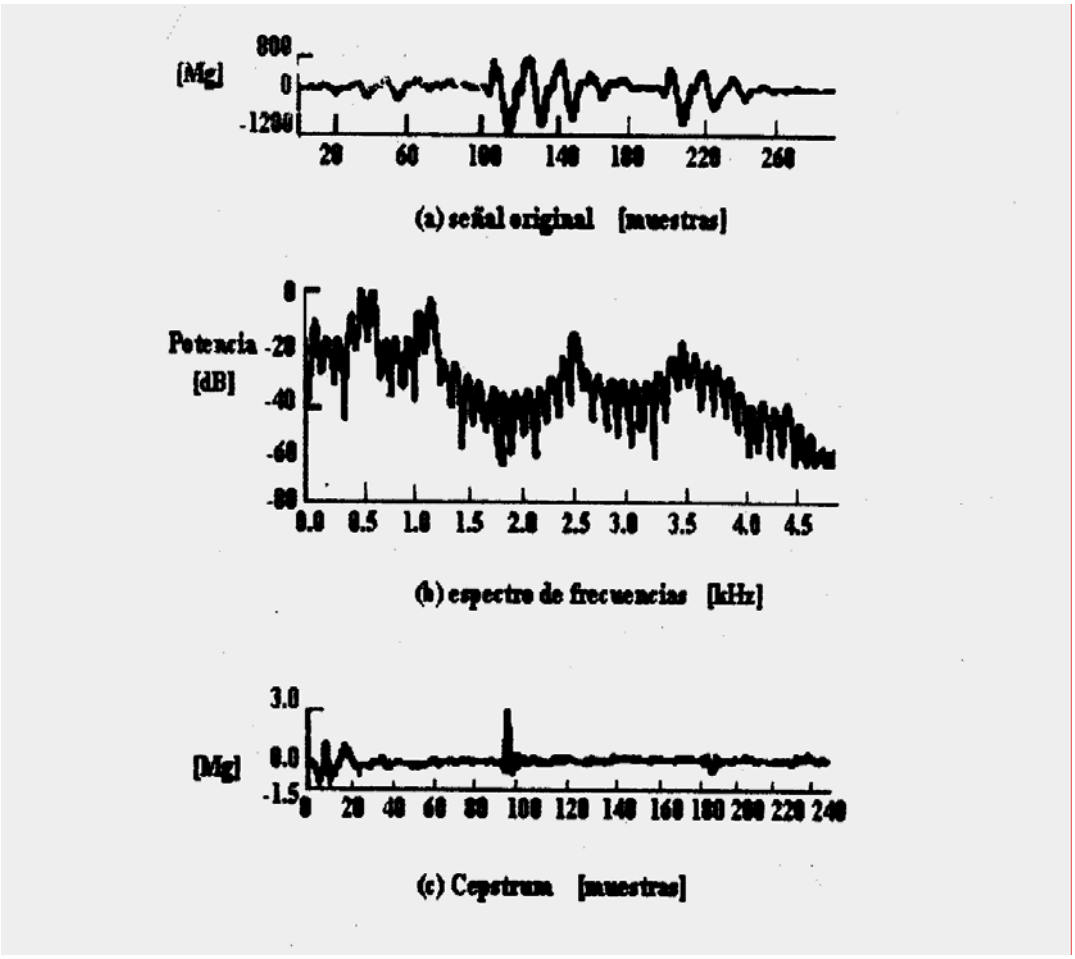


Figura 1.3 Aplicación de la función cepstrum.

1.5 Herramientas de análisis

El lenguaje hablado, al ser tratado como una señal eléctrica, es susceptible de ser analizado mediante técnicas que han evolucionado de acuerdo a los avances en electrónica y los métodos computacionales.

1.5.1 Análisis por métodos analógicos

Previo a la aparición de las herramientas de procesamiento digital de señales, el análisis de la señal de habla se lleva a cabo mediante técnicas puramente analógicas. En este sentido, la herramienta que sigue siendo uno de los pilares para el análisis de la señal eléctrica es el osciloscopio, el cual permite observar el desempeño de la forma de onda en el dominio del tiempo. Con un enfoque más especializado, han aparecido otras variantes en la instrumentación que dieron un gran impulso al desarrollo de la ciencia del análisis vocal.

ESPECTRÓGRAFO

Aunque es una herramienta que en la actualidad ha sido superada por las técnicas digitales, es conveniente mencionarla como un antecedente importante. Su funcionamiento se basa en los mismos principios que utiliza un analizador de espectros analógico, pero su banda de frecuencias de análisis se restringe a la banda ocupada por el espectro de la voz (4 Khz.). Básicamente, está constituido por filtros sintonizables dentro de la banda de frecuencias a ser exploradas.

Un elemento de control asigna la operación a cada filtro de manera que sea "barrida" la banda deseada. La respuesta en magnitud de cada filtro es traducida en una gráfica que muestra con bastante aproximación el espectro de frecuencias de la señal en análisis. Su uso forzaba al experimentador a repetir constantemente la reproducción de cada segmento de señal de manera que se obtuviesen gráficos más confiables.

LARINGÓGRAFO

El laringógrafo es un dispositivo que aporta información del movimiento de las cuerdas vocales. Mediante electrodos conectados a la garganta, se lleva un registro de los cambios en la conductancia a lo largo de la laringe. Dicha conductancia varía de acuerdo a las oscilaciones de las cuerdas vocales, incrementándose cuando se estrechan y decrementándose cuando se ensanchan.

En la figura 1.4, se muestra la curva obtenida por un laringógrafo para un segmento vocal emitido por una mujer. En dicha figura, puede apreciarse que el estrechamiento de la laringe se asocia con un rápido incremento en la conductancia, así como las aperturas de la laringe se identifican por un gradual decremento en magnitud.

Se observa también una tercera gráfica donde aparece el residuo LP para este segmento, donde sobresalen incrementos en el error cuando los pulsos de la laringe tienen un máximo –es decir, cuando el estrechamiento es máximo también -.

Cabe hacer notar que la relación entre la gráfica obtenida con el laringógrafo no sigue con precisión la forma de onda de los pulsos glotales, sino que su relación se identifica únicamente en cuanto a su periodicidad. Lo anterior se debe a que con este instrumento se obtiene la curva de estrechamiento de la laringe, mientras que la forma de onda de los pulsos se modela por la dinámica del flujo de aire a través del tracto vocal.

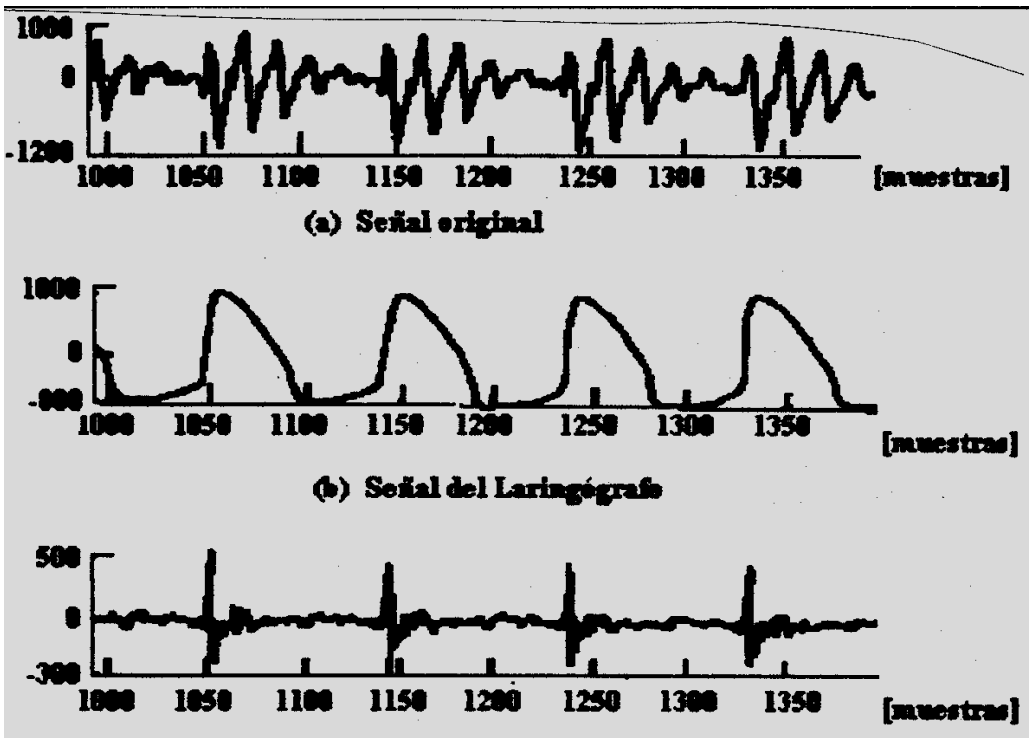


Figura 1.4 Formas de onda producidas por un laringógrafo

1.5.2 Análisis por métodos de procesamiento digital de señales

La aparición de los métodos de análisis de las señales discretas, aplicados al análisis del lenguaje hablado, marcó un punto de despegue en el desarrollo de las ciencias del lenguaje. La aplicación de algoritmos de análisis en la frecuencia tales como DFT, FFT o DCT se ha generalizado en la última década, al grado de formar parte integral de programas especializados en el análisis de señales como son MATLAB o Mathematica. Una de las herramientas de análisis más populares son los espectrogramas, los cuales son la aplicación directa de

métodos de respuesta en frecuencia como la FFT en el espacio del tiempo, generando gráficas frecuencia-tiempo que permiten observar la evolución espectral de la señal en un intervalo de tiempo dado.

Comúnmente, el tiempo (o el número de muestras), aparece como la abscisa, mientras que como ordenada se genera un patrón de color o de tonos de gris que representa los diferentes componentes espectrales (a un tono más elevado corresponde una magnitud mayor para una frecuencia dada).

Un espectrograma puede también ser representado por una superficie; en tal caso, el eje perpendicular al plano frecuencia-tiempo corresponde a la magnitud instantánea de una componente espectral. En este caso, la gama de colores que se asignaba a las diferentes magnitudes corresponde a la altura del relieve en el espectrograma de superficie.

ESPECTROGRAMA DE BANDA ANCHA

En el contexto de los espectrogramas generados, utilizando transformadas de Fourier, el término “banda ancha” implica que se utilizaron relativamente pocas muestras en la obtención de cada espectro instantáneo; ello acarrea una disminución en la resolución en el eje de la frecuencia, dando mayor preferencia a los detalles en el dominio del tiempo. En la figura 1.5, pueden apreciarse una serie de estrías verticales a través de los tonos oscuros de las frecuencias formantes.

Estas estrías son características de los segmentos de sonidos vocales, y representan las excitaciones individuales del tracto vocal cuando éste se cierra. Midiendo la distancia entre las estrías, puede hacerse un cálculo aproximado del periodo fundamental del tracto vocal *pitch*.

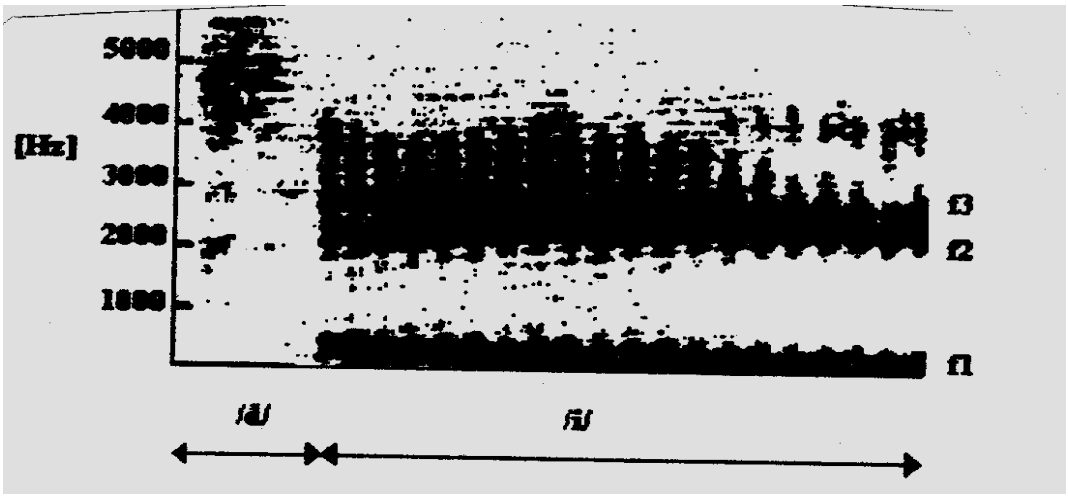


Figura 1.5 Ejemplo de espectrograma de banda ancha

ESPECTROGRAMA DE BANDA ANGOSTA

Una alternativa para la representación espectral de la señal vocal, puede ser generada aplicando la transformada de Fourier en un segmento más amplio de muestras. Los espectrogramas producidos de esta manera se denominan de "banda angosta" y contienen información más detallada en el dominio de la frecuencia a costa de un decremento en la resolución en el dominio del tiempo.

En la figura 1.6, se hace más evidente la presencia de las frecuencias formantes en los sonidos vocales. Así como en los espectrogramas de banda ancha era posible hacer un cálculo de la oscilación fundamental midiendo la distancia entre las estrías, en los espectrogramas de banda angosta es posible identificar las diferentes armónicas que conforman a una vocal y así hacer una aproximación de la frecuencia fundamental.

Figura 1.6 Ejemplo de espectrograma de banda angosta

Capítulo 2

FISIOLOGÍA DE LA VOZ HUMANA

2.1 Generalidades de la fonética

"Lo universal y lo diverso del habla llevan a una importante inferencia. Estamos obligados a creer que el lenguaje es una herencia inmensamente antigua de la raza humana… Es dudoso que cualquier otro recurso cultural del hombre, pudiera pretender una antigüedad mayor. Me inclino a creer que antecedió aún a las manifestaciones más humildes de la cultura material y a creer también que éstas, en efecto, no fueron posibles hasta que el lenguaje, el instrumento de la expresión significativa, se hubo formado".

Edward Sapir.

El estudio de la fisiología y fonética en este apartado se basa principalmente en los libros de Bolaño e Isla A., *Breve manual de fonética elemental*; y de Raúl Avila, *Aspectos fonéticos y léxicos del español*. Así como en los textos de procesamiento digital de voz, L. Rabiner y B. Juang. *Fundamentals of Speech Recognition.,* Chris Rowden, *Speech Processing*, F. A. Westall, *Digital Signal Processing in Telecommunications.*

Posiblemente, la intuición, la emoción, el instinto de conservación y poco a poco la imitación casual, explican los primeros intentos de emisión fónica que, por una interacción continua, dan origen al lenguaje, cuya creación no fue ningún acontecimiento repentino y no se ha detenido jamás, sino que prosigue su curso constantemente.

La doctrina evolucionista hace ver cómo en los prenombres surgen los primeros sonidos laríngeos de carácter expresivo, por una necesidad social de comunicación.

Tales sonidos se asociaban con ademanes o mímica. La etapa lingüística superior consiste en el paso de la expresión espontánea de las emociones a las voces inarticuladas que servían para designar intencionalmente los objetos.

El proceso de formación del lenguaje duró aproximadamente un millón de años; el lenguaje se va haciendo más independiente y surge como una necesidad la "gramática material", reflejada en el pensamiento del hombre. La circunstancia de que el pensamiento puede ser expresado e interpretado por medio de un vasto sistema de señales adaptadas y codificadas, permitió al hombre superar su pensamiento. Así se fueron elaborando procesos de abstracción, generalización y ***síntesis***; se adquirieron las nociones de los objetos, de las propiedades de las cosas y de sus relaciones y nacieron los conceptos, las ideas.

Así, el pensamiento humano, nacido junto con el lenguaje, permite la actividad cognoscitiva por medio de la palabra y facilita las relaciones del individuo con la sociedad. De ahí la importancia individual y social de la comunicación verbal para el hombre, único ser entre los de su género que posee este medio de expresión.

Las diferencias biológicas, raciales y culturales entre unos grupos humanos y otros marcaron las características especiales de cada idioma, tanto en su forma oral como en la escrita.

Es así, como la mecánica primitiva de los reflejos fónico-motores fue perfeccionándose hasta lograr la coordinación sensorio motriz tan precisa de toda la red de funcionamientos que intervienen en la producción del lenguaje.

Siguiendo el orden natural del proceso de elaboración del lenguaje desde un punto de vista fisiológico, se da desde la captación de los estímulos auditivos del medio externo, hasta la fase motriz del lenguaje oral o escrito, conforme a lo cual los aparatos y sistemas que intervienen son:

1) Aparatos sensoriales: audición y vista

2) Sistema nervioso central

3) Aparato fono-articulador (ver fig. 2.1)

La palabra debe ser oída; la audición es requisito indispensable en la captación de los estímulos sonoros verbales del medio externo. La vista en segundo lugar interviene en las asociaciones visuales de los objetos.

La palabra “gato” es oída y asociada
a su imagen visual

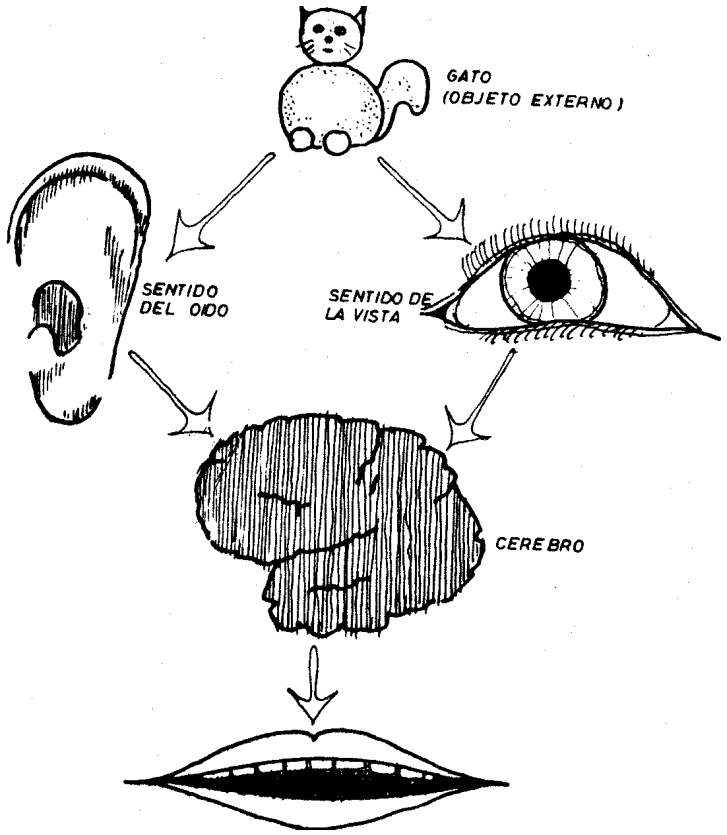


Figura 2.1 La elaboración del lenguaje

Enseguida, la palabra debe ser interpretada; la sensación auditiva se convierte en percepción. El sistema nervioso central controla una red de funcionamientos muy complejos que permite la elaboración de los conceptos mentales en relación con la palabra, dándole a ésta su significado propio y formando el lenguaje interior.

Posteriormente, el sistema nervioso central envía las órdenes motrices correspondientes a la ***emisión de la voz***, la palabra y la frase en su forma oral o escrita. Cuando la respuesta es oral, actúan una serie de sistemas que constituyen el aparato fono-articulador, el cual está controlado por el sistema nervioso central.

Es este sistema, el centro de la actividad lingüística, y forma con todos los órganos y aparatos que intervienen en la emisión de la palabra, una unidad que relaciona el medio externo con el interno y transforma el pensamiento en palabra.

2.2 Producción de la voz

2.2.1 Dinámica del aparato vocal

Con la finalidad de tener un mejor entendimiento de las técnicas utilizadas en el procesamiento digital de la voz, es necesario primero conocer la forma en que la voz se produce. Existen diferentes modelos para caracterizar la producción de sonidos artificialmente, y la mayoría de ellos se basan en una idealización del aparato vocal.

Los sonidos que percibimos como lenguaje hablado, llegan hasta nosotros como una onda de presión la cual toma su forma final en el aparato vocal. Inicialmente, estos sonidos son producidos al pasar el aire exhalado por los pulmones a través de las cuerdas vocales. Normalmente, el 60% del ciclo de respiración se utiliza en la exhalación y el 40% restante en la inhalación; sin embargo, durante los periodos de habla, el tiempo de exhalación se prolonga hasta un 90%. La respiración en reposo exige a los pulmones aproximadamente el 10% de su capacidad mientras que al hablar el esfuerzo se incrementa hasta en un 40%.

La siguiente etapa en la producción de sonidos es la laringe en la cual se ubican las cuerdas vocales, que consisten en dos segmentos de materia muscular y membranosa llamados cartílagos tiroides y cricoides. Es en este punto donde se producen los llamados sonidos vocales los cuales se caracterizan por poseer un espectro donde destacan dos o tres frecuencias dominantes.

Al pasar el aire a través de la laringe, la tensión en el interior varía de manera que se producen estrechamientos que se desplazan periódicamente a lo largo de ella. Esta oscilación del cartílago tiroides se traduce en la producción de una frecuencia vocal fundamental la cual depende principalmente de las dimensiones de las cuerdas y del tamaño de la laringe. Usualmente, en los hombres el tamaño de cada cuerda varía entre los 17 y los 24 mm, mientras que en las mujeres está entre 13 y los 17 mm. En promedio, esta frecuencia fundamental es de 125 Hz en los hombres, de 200 Hz en las mujeres y alcanza los 300 Hz en los niños. A este proceso, se le conoce como la teoría mioelástica-aerodinámica de la fonación.

El tracto vocal que recibe el sonido producido en la laringe, realiza el modelado espectral de la señal caracterizando de esta manera el tono de voz particular en cada persona. Es decir, en el tracto vocal algunas frecuencias obtienen mayor o menor ganancia que otras dependiendo de la morfología del aparato vocal.

Una aproximación que lo modela elegantemente es la siguiente, donde se supone inicialmente un tubo simple sin pérdidas a través del cual se desplaza una onda plana de manera axial. La variación de presión *p(x,t)*, función de la posición *x* y el tiempo *t*, y la variación de velocidad de flujo volumétrico *u(x,t)*, están dadas por:

$$-\frac{\partial p}{\partial x}=\rho\frac{\partial\left(u/A\right)}{\partial t}\qquad -\frac{\partial u}{\partial x}=\frac{1}{\rho c^{2}}\frac{\partial(pA)}{\partial t}+\frac{\partial A}{\partial t}$$

Ecuación 2.1

Donde además *c* es la velocidad del sonido (300 m/s) y *A(x,t)* es el área seccional del tubo, la cual se supone uniforme. Las resonancias en este tubo se dan cuando su longitud es ¼ de la longitud de onda y múltiplos impares de este valor. Así, por ejemplo, el tracto vocal típico en un hombre es de 17 cm, de donde la primera frecuencia de resonancia resulta ser

$$f_1=\frac{c}{\lambda}=\frac{340}{(4)(0.17)}=500\ \ \text{Hz}$$

Ecuación 2.2

De la misma manera, si se calculan las siguientes frecuencias de resonancia obtenemos para 3λ, f_2=1500 Hz y para 5λ, f_3=2500 Hz.

Las frecuencias obtenidas anteriormente resultan ser, notoriamente, las frecuencias que caracterizan el espectro de la llamada vocal neutra, que se emite al dejar salir el aire naturalmente a través del tracto sin efectuar ningún tipo de constricción en el mismo.

En la figura 2.2, se esquematizan los principales órganos que intervienen en la articulación del lenguaje. En la producción del sonido que asciende por la cavidad faringe (fig. 2.2), interviene también la cavidad oral la cual se encuentra separada de la cavidad nasal mediante el velum del paladar. La cavidad nasal esta siempre inmóvil y dependiendo de la posición del velum se acopla o no al tracto vocal para intervenir en algunos sonidos. La generación de los sonidos no vocales (nasales, fricativos, y constantes en general, los cuales se analizarán en el siguiente capítulo), se realiza mediante la manipulación de la lengua, labios, dientes y mandíbula dentro de estas cavidades.

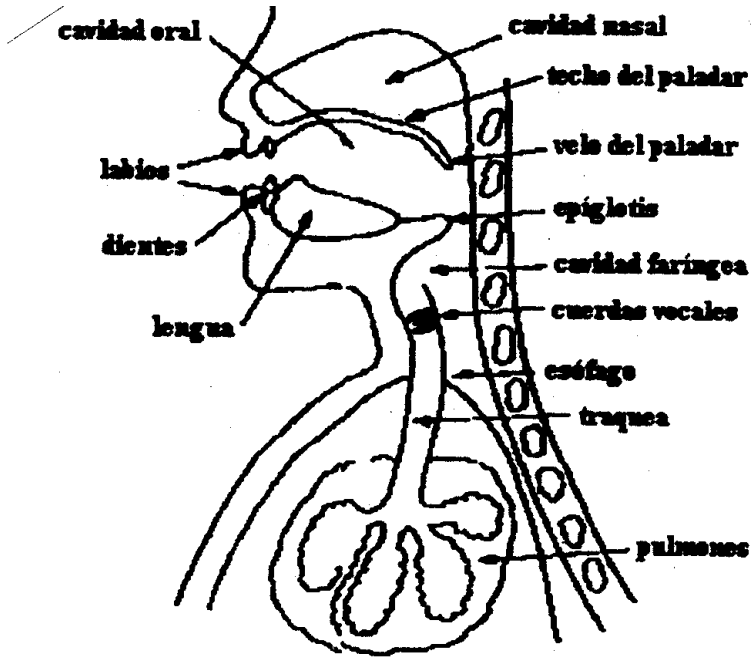


Figura 2.2 Órganos vocales del cuerpo humano

2.3 Percepción de la voz

La producción del lenguaje hablado es un proceso complicado; sin embargo, éste es aún la mitad de una conversación. En la percepción de la voz interviene un proceso fisiológico bastante complejo, el cual tiene sus bases en el funcionamiento del aparato auditivo. En el reconocimiento de los sonidos, esto es, el proceso cognoscitivo mediante el cual se extrae un significado, intervienen además de la habilidad del cerebro para reconocer patrones, factores de tipo cultural y la experiencia adquirida. Por esto, los intentos llevados a cabo en el campo del reconocimiento de la voz se caracterizan por utilizar algoritmos de elevada complejidad**.**

2.3.1 Aparato de la audición

"De todos los sentidos, el oído es el que más contribuye a la inteligencia y al conocimiento". Hipócrates

La audición controla y regula los procesos motores indispensables para la modulación adecuada de la ***voz*** e indispensable para la adquisición normal del lenguaje.

ANATOMÍA DEL APARATO

El aparato de la audición se compone de tres partes: oído externo, oído medio y oído interno. (ver fig. 2.3)

El oído externo está formado por el pabellón del oído y el conducto auditivo externo.

El pabellón del oído u oreja afecta la forma de una concha irregular unida por su parte anterior e interna a las partes laterales de la cabeza. Se compone de una armadura cartilaginosa y músculos, cubierto de piel y recibe vasos y nervios, provenientes de una rama del nervio facial.

El conducto auditivo externo es casi transversal y va del pabellón a la membrana del tímpano. Tiene 2 porciones: una ósea y otra cartilaginosa. La

cartilaginosa va desde el pabellón hasta la ósea y en su mayor parte está formada por un tejido fibroso. La piel que cubre la porción cartilaginosa presenta pelos ásperos y glándulas ceruminosas que secretan una sustancia grasa amarillenta: el cerumen.

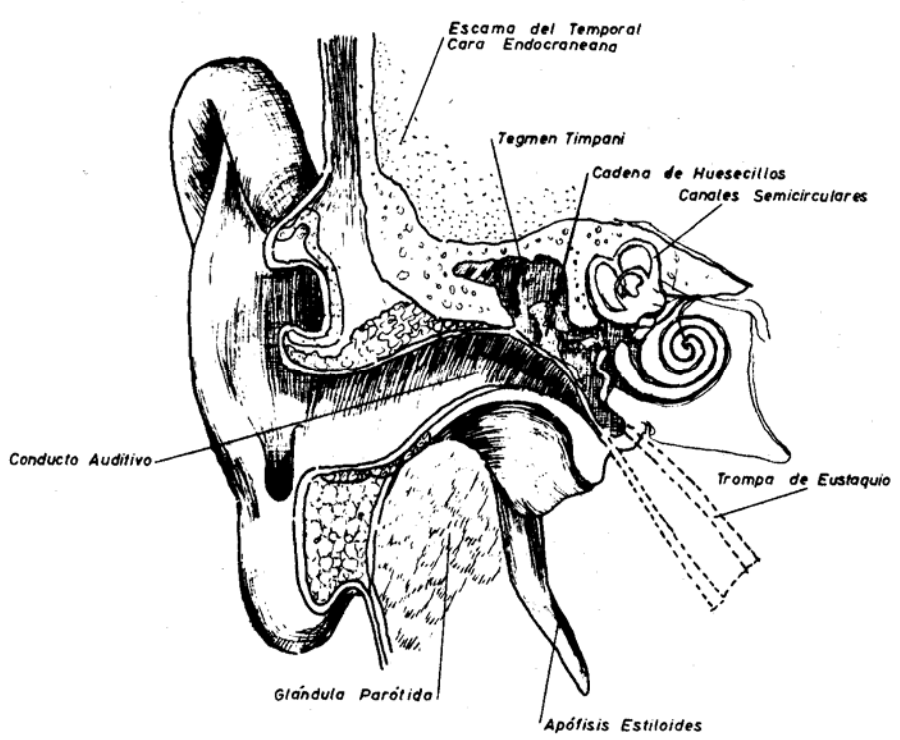


Figura 2.3 Sección transversal del oído externo, medio e interno

Los nervios vienen del plexo cervical y del pneumo-gástrico, ramas del aurículo-temporal y auricular, respectivamente.

La cara externa de la membrana del tímpano limita el conducto auditivo externo del medio.

El oído medio comprende una porción ósea y partes blandas. La porción ósea se compone de la caja del tímpano y los huesillos del oído. Las partes blandas son los ligamentos y músculos de los huesillos, la Trompa de Eustaquio y la membrana de la ventana redonda.

La caja del tímpano es una dilatación sobreañadida al conducto auditivo externo, así como el sombrero de un hongo está en relación con su pedículo. Presenta una pared interna, una externa y una circunferencia de donde parte hacia delante el conducto músculo-tubario y hacia atrás el orificio de comunicación de las células mastoides.

La pared externa es una abertura casi circular cerrada por la membrana del tímpano. La pared interna es convexa, desigual, presenta una eminencia en su parte media, que es el promontorio, cuya base corresponde al origen del caracol. Por encima del promontorio, se encuentra la ventana oval, abertura oblonga, en forma de riñón que conduce al vestíbulo. La ventana redonda está situada debajo de la ventana oval, es circular y estrecha y conduce al caracol. La circunferencia es muy irregular y está formada hacia arriba por una lámina ósea: el techo del tímpano; hacia atrás, presenta de arriba abajo la abertura de las células mastoides (que a veces comunican las células codíleas del occipital), y por dentro de la ranura timpánica está el conducto de la cuerda timpánica.

El conducto músculo-tubario va del ángulo entrante del temporal a la parte anterior de la caja y se divide en dos conductos separados por una lámina ósea muy delicada: el conducto superior o conducto del músculo del martillo y conducto inferior o porción ósea de la Trompa de Eustaquio.

Los huesillos del oído forman una cadena articulada desde la membrana del tímpano hasta la ventana oval y son: martillo, yunque, lenticular y estribo. El martillo presenta una cabeza, un cuello y tres apófisis. El yunque tiene un cuerpo y dos apófisis. El lenticular, hueso sumamente pequeño, tiene una cara interna convexa, soldada muchas veces a la apófisis vertical del yunque y una cara interna, convexa, que corresponde al estribo. El estribo está extendido horizontalmente entre el hueso lenticular y la ventana oval, tiene una cabeza, una base y dos ramas.

Los huesillos están unidos por ligamentos, que son dos articulaciones diartroidales (con líquido sinovial), y cuatro ligamentos que fijan los huesillos a las paredes de la caja; dos para el martillo y dos para el yunque.

Los músculos de los huesillos son dos: el del martillo, que está inervado por una rama del ganglio ótico y el del estribo que lo está por una rama del facial. Los movimientos de los huesillos hacen que la membrana del tímpano esté tensa o relajada.

La Trompa de Eustaquio se compone de dos porciones: una ósea y otra cartilaginosa. Estas dos porciones no se continúan en línea recta, sino que forman un ángulo obtuso abierto hacia abajo y van desde la abertura faríngea hasta la abertura timpánica.

La membrana del tímpano es muy delgada, transparente, de color gris perla o rosa pálido y refleja intensamente la luz. La membrana de la ventana redonda es un resto no osificado de la cápsula laberíntica membranosa. Los nervios sensitivos que llegan a esta membrana vienen del ramo de Jacobson y del gran simpático.

El oído interno comprende el laberinto óseo con el conducto auditivo interno y el laberinto membranoso.

El laberinto óseo está formado por una parte media que es la continuación de la caja del tímpano: el vestíbulo, una parte posterior que son los conductos semicirculares y una parte anterior denominada el caracol (Ver figuras 2.4 y 2.5).

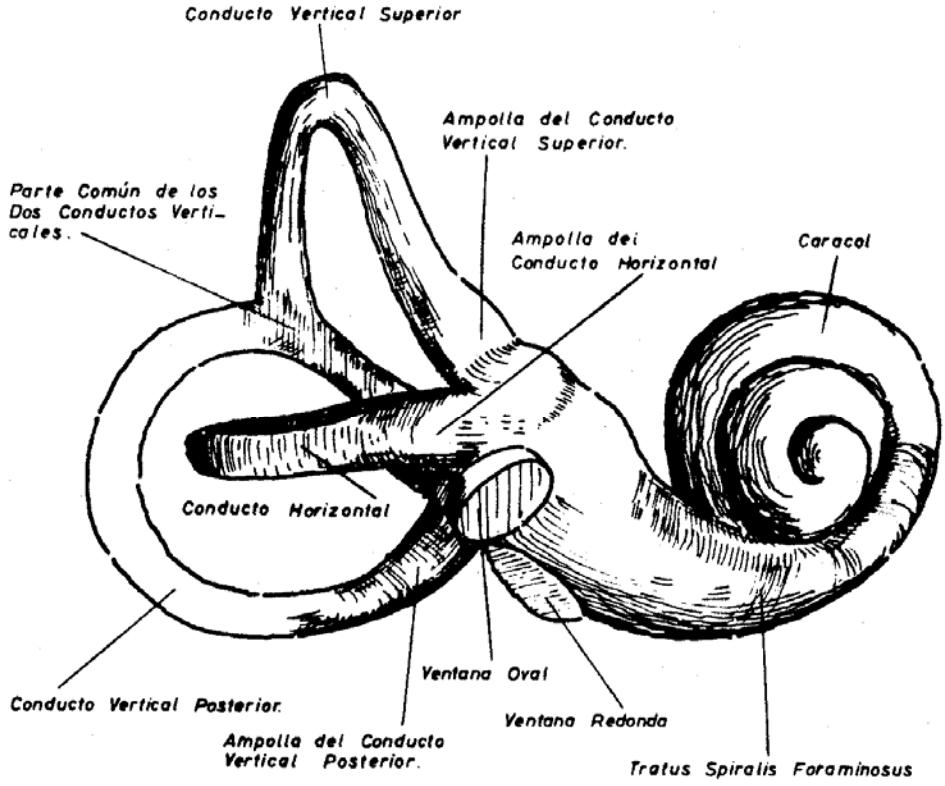


figura 2.4 Molde del laberinto visto por fuera

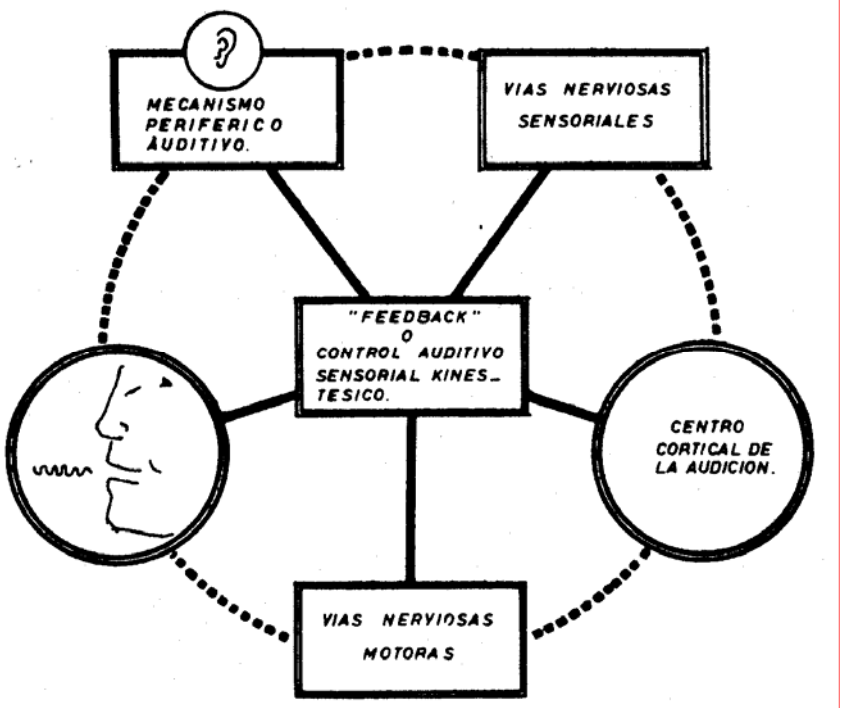


Figura 2.5 La mecánica del feedback (proceso de retroalimentación auditiva)

El conducto auditivo interno conduce el nervio acústico a las distintas partes del laberinto. Dicho nervio del caracol y otra posterior que es el vestibular.

El utrículo es la confluencia de los canales semicirculares y en su pared interna está la mancha acústica, que corresponde a la entrada del nervio utricular. El sáculo tiene la misma estructura del utrículo. El caracol membranoso tiene dos membranas: la basilar y la Corti. Así, se forman tres cavidades o rampas: una superior o vestibular, una inferior o timpánica y una media, que es la más estrecha y se le llama rampa auditiva por contener el órgano de Corti, que es el órgano auditivo por excelencia, ya que en sus arcos terminan las células ciliares del nervio auditivo. Éste proviene del octavo par y es el resultante de la unión de dos nervios: el acústico y el vestibular, los cuales se unen sólo en la parte

media de su recorrido y se separan en sus dos extremos: en su origen y en su terminación.

2.3.2 Fisiología de la audición

El oído externo recoge las ondas sonoras emanadas de los cuerpos vibrantes y las conduce a la membrana del tímpano poniéndola inmediatamente en vibración. Los pelos y el cerumen de este conducto impiden la entrada de cuerpos extraños.

El oído medio, que tiene el papel de resonador, refuerza el sonido. El músculo del martillo acomoda la tensión de la membrana timpánica de acuerdo con la amplitud de las vibraciones que recibe.

Las vibraciones del tímpano se transmiten a la ventana oval y a la redonda por la cadena de huesillos.

La Trompa de Eustaquio comunica el oído medio con el aire exterior y la parte posterior de las fosas nasales y mantiene la igualdad de presión entre las dos caras del tímpano.

El oído interno canaliza los sonidos de la ventana oval y redonda, y los propaga por los líquidos perilinfa y endolinfa, que los hace llegar a las células sensoriales del órgano de Corti. Las potencialidades microfónicas de estas células actúan sobre el sonido amplificándolo y haciéndolo oscilar.

Las manchas acústicas del utrículo y sáculo captan la sensación de intensidad del sonido. Mediante la vibración de las fibras de la membrana basilar del caracol, se percibe la altura y timbre del sonido. Las crestas auditivas de los canales semicirculares desempeñan un papel importante en el sentido del equilibrio.

Las vías ascendentes de conducción de la sensibilidad, transmiten las impresiones recibidas por el oído a la región del lóbulo temporal de la corteza cerebral, donde se hace consciente la sensación auditiva, transformándose en percepción.

El control auditivo de la voz se realiza a través de un proceso muy complejo denominado “feedback” o retroalimentación de la voz, el cual permite hacer los ajustes necesarios en los mecanismos fisiológicos que intervienen en la producción de la voz (hablada o cantada), de acuerdo con la impresión auditiva de la misma, todo lo cual se efectúa rápidamente, casi en forma instantánea.

Este fenómeno consiste en el establecimiento de un circuito en el cual una parte de la salida del sistema activo regresa para modular su actividad continua, pudiendo modificarla. Este sistema actúa automáticamente y es un estabilizador normal de la actividad nerviosa. Su mecanismo es complejo y constituye un control complementario de la organización nerviosa.

La técnica de “feedback ” puede compararse con una persona que habla, oye su propia voz y la modula adecuadamente según las características acústicas del lugar en que se encuentra.

2.3.3 Producción de los sonidos

Acústica es la parte de la Física que estudia el sonido; éste es el resultado de un movimiento vibratorio que se ha impuesto a la materia ponderable. Para que el sonido sea perceptible al oído, se requiere que las vibraciones hayan adquirido cierta velocidad.

Es preciso, distinguir el sonido del ruido: el primero produce una sensación continua, cuyo valor musical puede apreciarse, mientras que el ruido es una sensación instantánea o la mezcla de muchos sonidos discordantes.

La transmisión del sonido se hace a través de una serie de ondas sonoras, compuestas cada una de dos semiondas consecutivas, una condensada y otra dilatada. Las ondas sonoras perceptibles al oído humano varían desde 50 hasta 17000 Hz. Sin embargo, los sonidos relativos al lenguaje comprenden únicamente de 350 a 3500 Hz.

El sonido se propaga en el aire, en los líquidos y sólidos, pero no en el vacío. Su velocidad media en el aire es de 340 m/seg, en el agua de 2.432 metros y en los sólidos es mayor (se calcula que la velocidad del sonido a través del hierro colado es 10.5 veces mayor, 12 veces mayor en el cobre, 16 veces mayor en el

hierro y 18 veces mayor en la madera de pino, en comparación con su velocidad media en el aire).

La reflexión del sonido es el eco. Las ondas sonoras se reflejan formando un ángulo de reflexión igual al de incidencia, situado en un plano perpendicular a la superficie reflectora.

Las cualidades del sonido son: intensidad, tono, timbre y duración.

La ***intensidad*** es la mayor o menor fuerza con que se produce un sonido; depende de la amplitud de las vibraciones y es inversamente proporcional al cuadrado de la distancia de la fuente sonora al lugar en que se escucha el sonido. La intensidad normal de la voz varía entre 20 y 60 decibeles. Su potencia no es la misma en privado, en público o en un medio ruidoso; además, la conversación requiere ser matizada por fuertes y pianos.

El ***tono*** es la altura musical del sonido y depende del número de vibraciones por segundo; a mayor número de vibraciones (o frecuencia), corresponderá una mayor altura musical.

En la voz, hablada la altura habitual, se juzga por lo fundamental usual de la palabra: en el hombre, cualquiera que sea el tipo de su voz, su altura tonal está entre la1 y re2, en la mujer entre la2 y re3 y en el niño es la3.

En relación con el tono, está la entonación, que es la curva melódica que describe la voz al hablar. En fonética se marcan los niveles de entonación en cada grupo fónico y casi todos los fonetistas están de acuerdo en que en la voz hablada, habitualmente se usan tres niveles de entonación y que al final de cada frase el tono de la voz asciende, desciende, o permanece en un mismo nivel.

A la entonación de la voz al final de la frase se le llama “tonema”, y éste puede ser ascendente, descendente o de suspensión.

La ***entonación*** de la frase le da a ésta las características expresivas que denotan admiración, afirmación, duda, interrogación, corroboración, negación, alegría, regocijo, tristeza, indiferencia, entre otras; según el estado de ánimo del que habla y el contenido del mensaje oral.

El ***timbre*** es un complejo de movimientos vibratorios que permiten reconocer el instrumento que los produjo o la persona que los emitió. En el timbre se distingue un movimiento vibratorio fundamental y varios accesorios. El primero se produce por las vibraciones de las cuerdas y los segundos por la resonancia de dichas vibraciones en la cavidad resonadora.

Según su ***duración***, los sonidos pueden ser largos, breves, semilargos y semibreves.

Además de estas cualidades del sonido vocal, existen otras secundarias: el acento y la perceptibilidad.

El ***acento*** es el esfuerzo intencional que se realiza en determinada sílaba o sonido sobre otras de la misma palabra o frase. Aún dentro de la misma lengua, hay diferencias de acento y matices regionales y hasta individuales.

La ***perceptibilidad*** consiste en la mayor o menor posibilidad de un sonido de ser percibido a determinada distancia.

ACÚSTICA DEL LENGUAJE

La audición normal implica la capacidad para comprender el significado de los sonidos; la circunstancia de que un sonido sea audible no implica que sea comprensible.

La reacción al sonido no siempre significa audición, pudiendo ser en ocasiones un simple acto de reflejo. Para que ésta exista, el mecanismo fisiológico debe ser central y consciente. La audición normal requiere:

a) Un órgano auditivo normal.
b) Buen funcionamiento de los centros corticales auditivos.
c) Atención al estímulo auditivo.

Al proceso de reconocimiento y comprensión de los sonidos se le llama "discriminación auditiva" y depende de dos mecanismos: uno periférico, localizado en el órgano de Corti, y uno central ubicado en la corteza cerebral al nivel de los analizadores corticales.

En los centros corticales sólo pueden distinguirse los sonidos cuando se han repetido el número de veces necesario para ser memorizados.

Los sonidos del lenguaje son los más complejos y difíciles de aprender. Los sonidos vocálicos y consonánticos varían en intensidad. En los primeros, la intensidad es más elevada porque son sonidos continuos producidos con la boca abierta y con sólo pequeñas alteraciones en la forma de la cavidad bucal; son más fáciles de emitir y de escuchar que las consonantes, las cuales son producidas de una manera más complicada y tienen una intensidad variable y menor.
Las consonantes fricativas y silbantes generalmente son adquiridas al último, debido a las dificultades mecánicas para producirlas y para percibirlas. Este tema se analizará más adelante.

2.4 Aparato fono-articulador

Siendo el sistema nervioso el centro que rige y coordina toda la actividad lingüística, los centros motores primarios y secundarios envían los impulsos motores a través de las vías de motilidad hasta los órganos de ejecución: el aparato fono-articulador para la palabra oral.

Este aparato se encarga de la emisión de la palabra y está constituido por una gran variedad de órganos que se han agrupado en sistemas tomando en cuenta el papel fisiológico que desempeña durante la fonación. A saber:

(a) SISTEMA RESPIRATORIO

(b) SISTEMA DE FONACIÓN

(c) SISTEMA DE RESONANCIA

(d) SISTEMA DE ARTICULACIÓN

2.4.1 Sistema respiratorio

ANATOMÍA

El sistema respiratorio como se muestra en la figura 2.6, está formado por el aparato bronco-pulmonar y las paredes que al limitarlo, condicionan su movilidad; se integra por los pulmones, la caja torácica (ver fig. 2.7), el diafragma, un tronco cartilaginoso constituido por la tráquea y varias ramas progresivamente más pequeñas llamadas bronquios.

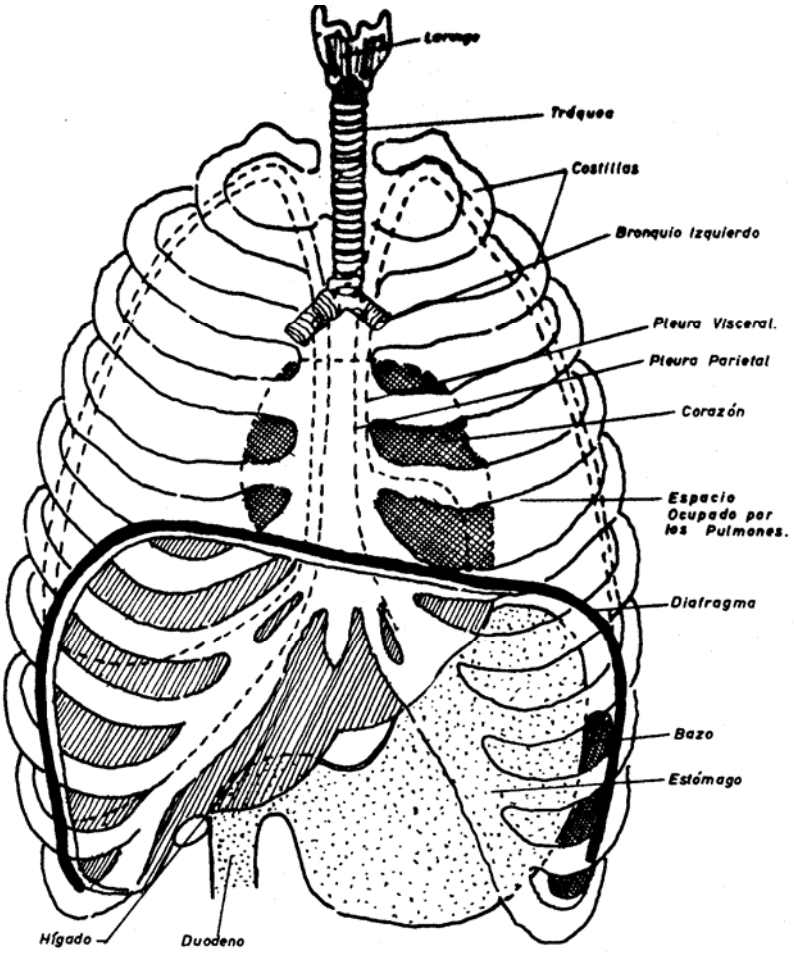


Figura 2.6 Órganos contenidos en el tórax y el abdomen, que tienen conexión con el órgano de la fonación.

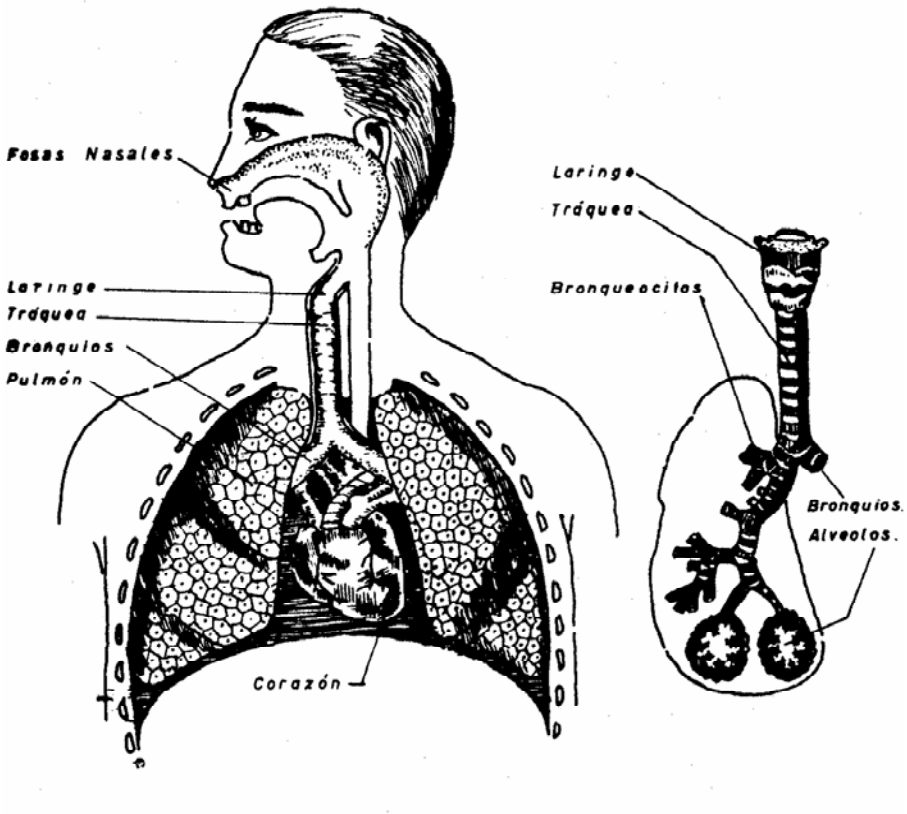


Figura 2.7 Aparato respiratorio

FISIOLOGíA DE LA RESPIRACIÓN

La respiración tiene tres tiempos: inspiración, pausa y espiración.

En la inspiración, el aire debe penetrar por las fosas nasales, faringe, laringe, tráquea, bronquios, pulmones y alvéolos pulmonares. En la inspiración, la caja torácica aumenta en su diámetro vertical, transversal y antero posterior. El esternón se aleja, la oblicuidad de las costillas disminuye y el diafragma se contrae y desciende. Los músculos que intervienen en la inspiración son el diafragma y los intercostales internos y externos, los cuales actúan en la respiración normal y con el elevador del omoplato, los elevadores de las costillas, el pectoral mayor, el pectoral menor, el romboideo mayor, el romboideo menor, el escaleno, el serrato mayor anterior, el serrato posterior superior, el esternocleido-mastoideo y el trapecio, que son los músculos accesorios de la inspiración y actúan cuando se refuerza este movimiento respiratorio, como acontece al emitir la voz.

Después de la inspiración, viene un momento de pausa y enseguida sucede la espiración, que es un acto pasivo: los órganos vuelven a su sitio porque los pulmones se contraen y expulsan el aire que contienen. Los músculos que intervienen en la espiración son: el interno oblicuo, el serrato posterior inferior, el torácico transverso y el transverso abdominal. El cuadrado lumbar actúa siempre como un estabilizador y los músculos accesorios que refuerzan la espiración son: el externo oblicuo, el recto abdominal y el gran dorsal.

De los movimientos respiratorios, la inspiración es la que tiene una función más activa, por lo que intervienen en ella mayor número de músculos, de los cuales el diafragma desempeña la función más importante, pues de este músculo va a depender esencialmente el dominio de las técnicas respiratorias.

En la espiración o exhalación, lo único que se necesita es relajarse. El tono normal de los músculos y la fuerza de gravedad hace que las costillas bajen, regresando a su posición normal y provocando la salida del aire debido a la disminución de volumen de la caja torácica. De este modo, en la espiración normal no se realiza ningún esfuerzo. Sólo en la práctica de los ejercicios respiratorios o en la emisión de la voz, se requiere de cierto esfuerzo muscular, para lo cual actúan los músculos accesorios.

2.4.2 Sistema de fonación

FISIOLOGÍA NORMAL DE LA LARINGE

Las funciones de la laringe son: respiratorias, circulatorias, fijativas, protectoras, deglutorias, tosivas, espectorativas, emocionales y fonatorias.

La voz es producida por la corriente de aire arrojada por los pulmones que llegando a la laringe con suficiente presión y encontrando tensas las cuerdas vocales, choca contra ellas y las hace vibrar dando lugar a un tono fundamental, al que se van a agregar posteriormente otros armónicos en las zonas de resonancia.

En general, la función de los músculos laríngeos se puede dividir en tres aspectos: unos provocan la aproximación de las cuerdas, otros su separación y otros más regulan su tensión. Por otra parte, unos actúan en la respiración y otros en la fonación.

El interaritenoide aproxima los dos aritenoides, estrechando la glotis. El crico-aritenoide posterior dilata la glotis durante la inspiración, mientras su porción lateral actúa como constrictor glótico en la fonación. La tensión de las cuerdas se controla mediante la acción de los pares crico-tiroideo y crico-aritenoide, principalmente. El tiro-aritenoideo también contribuye a la regulación de la tensión, rigidez, longitud y elasticidad de las cuerdas, al formar parte del cuerpo de las mismas. Y dependiendo de la longitud, tensión y grado de separación, será el tono que se va a emitir: mientras más agudo sea el sonido, más estrechamente se cierran. Por otra parte, la elasticidad propia del músculo vocal hace que pueda adelgazar o aumentar de espesor fácilmente de acuerdo con el tono que se emite.

La teoría de Perelló, denominada muco ondulatoria, analiza la relación existente entre la mucosa laringea y la acción de los músculos de este órgano. Explica cómo la mucosa de la laringe interviene en la calidad de la voz, toda vez que los cambios de la mucosa producen modificaciones en el funcionamiento motor de la laringe.

La teoría de Husson sustenta la tesis de que cada vibración de las cuerdas vocales está controlada por el impulso del nervio recurrente y por el centro acústico cerebral, que actúan presionando la columna aérea que hace vibrar las cuerdas vocales. Así se comprueba que la laringe no es el único órgano fonorio, sino que sólo es una parte de un sistema fisiológico complejo.

En el control normal de la calidad de la voz y de su adaptación fisiológica, interviene no sólo la laringe con todas las partes que la componen, sino además se requiere de un control cortical a cargo de los centros fonatorios del sistema nervioso central en relación con el sistema auditivo-cerebral y los procesos fisiológicos de las reacciones emocionales. Los centros fonatorios se localizan uno a cada lado del cerebro en la porción baja de la circunvolución pre-central, cerca del borde anterior. Estos centros corticales, por conducto del nervio recurrente, llevan a cabo el control de las cuerdas vocales cuando un individuo desea hacer una inspiración profunda y cuando desea hablar, envían los impulsos que producen la aproximación y tensión de las cuerdas vocales, para poder emitir la voz y controlan la respiración para graduar su fuerza e intensidad en el lenguaje oral, en el canto, en la voz cuchicheada, en la inhalación profunda y en los ejercicios respiratorios y fonatorios conscientes.

El sistema de fonación es sólo una parte de los procesos indispensables en el lenguaje; su misión es la de producir la voz y controlar su calidad, tono, modulación e inflexión durante un discurso oral o en el canto. En cambio, el lenguaje ya encierra un contenido mental traducido en palabras. La voz es adquirida al nacer, en tanto que el lenguaje necesita varios años para aprenderse y organizarse.

2.4.3 Sistema de resonancia

FISIOLOGÍA

Casi todos los instrumentos musicales están previstos de una caja de resonancia que permite agregar ciertos tonos armónicos al tono fundamental. Si esto no sucediera, los sonidos emitidos por tales instrumentos no tendrían una calidad musical. Lo mismo ocurre con la voz humana: la laringe es el órgano productor de la voz, pero el tono fundamental que elabora es ríspido, le falta armonía, musicalidad.

El sistema de resonancia, constituido por las cavidades faríngea, nasal y palatina, provee los tonos secundarios que le dan a la voz humana las cualidades armónicas individuales. No sólo hace agradable al oído la voz humana, sino que además imprime el timbre característico de la voz de cada persona y gradúa convenientemente la nasalidad, es decir, la cantidad de aire que debe ser arrojado por las fosas nasales en el momento de exhalar.

La faringe permite el paso del aire tanto en la inspiración como en la espiración. Cuando los sonidos son nasales, permite el paso del aire espirado por la rinofaringe y por la acción de sus músculos puede cambiar su forma ensanchándose o alargándose según la calidad del sonido que vaya a emitirse.

Las fosas nasales son otros órganos resonadores de gran importancia que tienen las siguientes funciones: respiratoria y olfativa; por lo que es importante su participación como parte del sistema de resonancia, interviene en la emisión de los fonemas nasales (m, n, ñ), al permitir el paso del aire por esta vía durante su emisión.

La cavidad bucal imprime características individuales a los sonidos permitiendo reconocer el timbre de la voz de la persona que habla aunque no se le vea.

El velo del paladar se eleva durante la deglución y la fonación (a excepción de la producción de los fonemas nasales m, n, ñ), impidiendo el pasaje del aire hacia la nariz. Se contrae en mayor o menor grado según la altura tonal del fonema y su forma de emisión.

La lengua desempeña numerosas funciones que puede realizar gracias a su extraordinaria movilidad tanto como parte del sistema de resonancia como dentro del sistema de articulación, además de la acción tan importante que realiza en la salivación, deglución y masticación.

Como parte del sistema de resonancia, la lengua adopta la forma y posición debidas a fin de darle a la cavidad bucal la forma y dimensiones convenientes según la calidad tonal del sonido que va a emitirse, lo que sólo se hace posible por la gran facilidad propia de este órgano para cambiar de forma y posición.

La lengua, en estado de reposo es ancha, blanda y ocupa completamente la cavidad bucal. Sus movimientos se clasifican en extrínsecos e intrínsecos. Los primeros son los cambios de lugar de la lengua, que se realizan debido a la contracción de los músculos que se insertan al hueso hioides y son cuatro: elevación, descenso, movimiento hacia delante y hacia atrás. Los intrínsecos son los cambios de forma de la lengua y son seis:

1. Prolongación, por contracción del músculo lingual transverso.
2. Acortamiento, por contracción de las fibras longitudinales.
3. Encogimiento transversal, al contraerse las fibras transversas.
4. Achatamiento y ensanche, cuando se contraen las fibras verticales.
5. Movimiento de lateralidad, por contracción del estilo-gloso y de las fibras longitudinales de un solo lado.
6. Encorvadura de la lengua a manera de canal, lo que se realiza al contraerse los genioglosos, estiloglosos, lingual superior y gloso-estafilinos, actuando todos en forma coordinada.

En esta forma, todos los órganos resonadores, dotados de la movilidad necesaria para poder adaptar la forma y tamaño de la caja de resonancia, actúan sobre el sonido producido por la laringe, imprimiéndole las modificaciones necesarias a fin de emitir una voz armoniosa y musical dotada de un timbre característico; tal mecanismo funciona tanto durante el discurso oral como en la voz cantada; sólo que ésta requiere un entrenamiento muy intenso.

2.4.4 Sistema de articulación

FISIOLOGÍA

El sistema de articulación tiene a su cargo el mecanismo final del aparato-articulador. Después que la voz es producida en la laringe, al pasar por las cavidades de resonancia adquiere los tonos armónicos que la hacen agradable al oído humano y finalmente estos sonidos se convierten en fonemas, palabras o frases mediante la acción conjunta de los órganos que constituyen el sistema de articulación.

Los órganos articulatorios se dividen en:

- Activos.- mandíbulas, labios, lengua y velo del paladar
- Pasivos.- dientes, alvéolos y paladar duro.

Se debe tomar en cuenta la secreción y deglución normal de la saliva, que sucede continua y mecánicamente, permitiendo la lubricación de los órganos de articulación y contribuyendo así a una clara y correcta pronunciación.

En la producción de las vocales, intervienen los órganos activos; la abertura de las mandíbulas tiene gran importancia en la articulación de las vocales. Los labios cambian la forma de la abertura bucal, y la lengua, situada en el piso de la boca, se hace hacia delante o atrás. El velo del paladar se eleva impidiendo la salida del aire por la vía nasal.

Las consonantes se forman por el choque de la corriente espiratoria en su canal de salida con los órganos de articulación que han tomado una posición determinada según el fonema que van a producir, para lo cual intervienen todos los órganos articulatorios, tanto los pasivos como los activos.

Capítulo 3

CARACTERÍSTICAS DE LA ARTICULACIÓN EN ESPAÑOL

3.1 Generalidades

Otro aspecto importante del estudio de la fonética y fonología es la articulación y las referencias bibliográficas son los libros de Bolaño e Isla A., *Breve manual de fonética elemental*; y de Raúl Avila, *Aspectos fonéticos y léxicos del español*; Chris Rowden, *Speech Processing*, F. A. Westall, *Digital Signal Processing in Telecommunications.*

La pronunciación de la lengua, que se refiere al modo especial de articular los sonidos para hablar, es un tema que, teniendo como bases científicas la fonética y la fonología, puede enfocarse desde distintos puntos de vista. La fonética estática analiza cómo se produce cada sonido que forma la palabra, describiendo la posición media de los órganos móviles que intervienen en su producción, lo que corresponde, en otros términos, a los puntos de articulación de los fonemas. La fonética dinámica, en cambio, analiza los movimientos necesarios en la articulación y las diferentes maneras en que se pueden producir las unidades sonoras que forman las palabras, lo que equivale al modo de articulación.

La fonología estudia los fonemas y la fonética los fonos, que son las realizaciones sonoras de los fonemas. Este concepto se refiere a que por sí solo, el fonema no tiene ningún significado, pero permite distinguir los significados de las palabras. El término “fonema” implica la idea de “unidad abstracta”, dado que abstrae la esencia del sonido. Los fonemas se pueden articular de diferente modo, según los sonidos que tienen en contacto y según la dinámica habitual de quien habla. Todas las variantes sonoras de un mismo fonema son los fonos, unidad fonética mínima.

La pronunciación varía en cada entidad geográfica, así también la norma fonética que predomina en una comunidad lingüística se deduce del uso habitual de la lengua por el grupo social que la habla. El modo de hablar refleja la

influencia de la sociedad en que vivimos y las circunstancias geográficas, raciales, históricas y culturales que la rodean.

3.2 Fonemas del español

Los fonemas de la modalidad del español hablado en México son:
Cinco vocálicos:

/i/, /e/, /a/, /o/, /u/

Dieciocho consonánticos:

/p/, /t/, /d/, /l/, /m/, /n/, /ñ/, /f/, /b/ (transcrito ortográficamente b, v), ***/ĉ/*** (transcrito ortográficamente como ch), **/x/** (transcrito ortográficamente j o g con sonido fuerte, cuando va unido a las vocales e, i), **/g/** (transcrito ortográficamente g, gu), **/s/** (transcrito ortográficamente s, z y c ante e, i), **/k/** (transcrito ortográficamente k, qu y c ante a, o, u), **/r/** (ere simple), **/^r/** (erre vibrante múltiple), **/y/** (transcrito ortográficamente ll, y, indistintamente), **/š/** (equivalente al sonido representado en inglés por la grafía sh). Este último fonema se usa en México en algunos aztequismos, como en la palabra Xola, que se pronuncia /shola/.

A continuación se presenta una tabla resumiendo el fonema con el grafo respectivo

Tabla 3.1

FONEMA	GRAFO
/p/	p
/t/	t
/d/	d
/l/	l
/m/	m
/n/	n
/ñ/	ñ
/f/	f
/b/	v, b
/ĉ/	ch
/x/	j ó g con sonido fuerte, cuando va unido a las vocales e, i
/g/	g, gu
/s/	s, z,c (ante e, i)
/k/	k, qu y c (ante a, o, u)
/r/ (vibrante simple)	r
/^r/ (vibrante múltiple)	rr
/y/	ll, y
/š/	sh

Atendiendo al lugar en que se producen los fonemas vocálicos, se pueden clasificar en tres categorías: anteriores o palatales **/i/, /e/**; central baja **/a/** y posteriores o velares **/o/, /u/.** Ver tabla 3.2

Tabla 3.2

Vocales	Anteriores	Central	Posterior
Cerradas	**/i/**		**/u/**
Medias	**/e/**		**/o/**
Abierta		**/a/**	

En la vocal central baja **/a/** la lengua descansa en la parte inferior de la boca, suavemente apoyada contra los dientes inferiores. En la articulación de las vocales palatales **/e/**, **/i/** el movimiento de la lengua se acentúa hacia delante y en las velares **/o/**, **/u/** la lengua se recoge cada vez más adentro, elevándose hacia el velo del paladar. El lingüista alemán Hellwag en 1781 ideó el "triángulo vocálico" para representar la articulación de las vocales, como se muestra en la figura 3.1

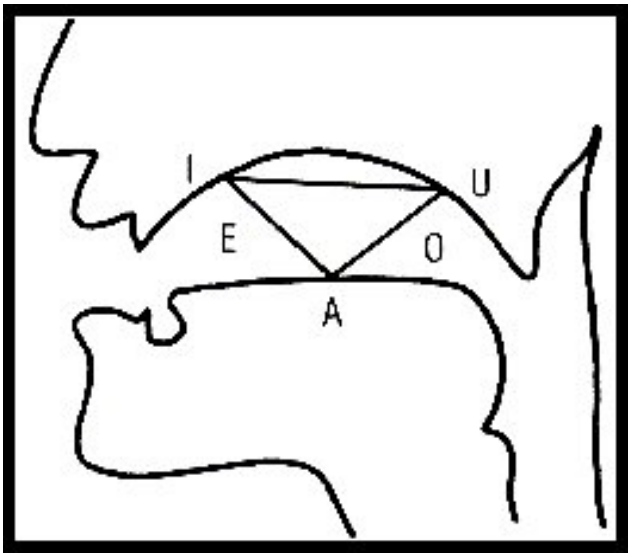


Figura 3.1 Triángulo Vocálico

La /i/ representa el vértice anterior de la posición de la lengua en el paladar; la /u/ el vértice posterior y la /a/ la base del piso de la boca.

En la articulación de las vocales intervienen también los labios, llevándose hacia delante en la articulación de los fonemas /o u/ en forma progresiva y retrayéndose en /e i/. Desde este punto de vista, los fonemas /o u/ son labializados; /e i/ deslabializados y el fonema /a/ es neutro.

La abertura que se forma entre lengua y paladar también es muy importante en la articulación de las vocales. La vocal más abierta es la /a/ y la abertura va disminuyendo en las demás vocales.

La glotis es sonora en todas las vocales, se puede percibir su vibración con el simple tacto en la región subhioidea.

El velo del paladar se eleva al articular las vocales orales, para impedir que el aire espirado salga por las vías nasales.

Las características propias de los fonemas consonánticos son el punto de articulación, el modo de articulación y la función de la glotis.

El punto de articulación es la posición que toman los órganos al articular los fonemas consonánticos y los puntos de apoyo de los órganos activos sobre los pasivos o partes duras de la cavidad bucal. El modo de articulación es la forma en que son producidos los fonemas con las modificaciones implícitas en la dinámica de la articulación. La función de la glotis hace que los fonemas sean sonoros o sordos según haya o no vibración laríngea.

3.3 Clasificación de los fonemas consonáticos

Los fonemas se clasifican en cuatro grupos tomando en cuenta el punto de articulación, por el modo de articulación, por la función de la glotis y por la posición del velo del paladar.

1. Según el punto de articulación, los fonemas se clasifican en:
   - BILABIALES /p b m/ Sonidos articulados con los labios, desempeñando una función activa el labio inferior y permaneciendo pasivo el superior.
   - LABIODENTAL /f/ Sonido articulado con el labio inferior (elemento activo) y el filo de los dientes superiores (elemento pasivo).
   - DENTALES /t d/ Sonido articulado con los dientes superiores e inferiores en contacto y la lengua en la base de la boca con la punta hacia abajo.
   - ALVEOLARES /s l n r ^r/ Sonidos articulados con el ápice de la lengua como órgano activo y los alveolos de los dientes superiores como órgano pasivo. El punto de articulación descrito corresponde a la /s/ mexicana, el cual se define como fonema predorso-dentoalveolar convexo.
   - PALATALES /ĉ ñ š/ Sonidos articulados con el dorso de la lengua (órgano activo) apoyado en el paladar duro (órgano pasivo).
   - VELARES /g k x/ Sonidos articulados con el postdorso de la lengua (órgano activo) y el velo del paladar (pasivo).

2. Por el modo de articulación, los fonemas se clasifican en:
   - OCLUSIVOS /p t k b d g/ Son los sonidos emitidos con los órganos cerrados, los cuales producen una pequeña explosión al permitir la salida del aire espiratorio.
   - FRICATIVOS /f s y š x/ Se articulan con órganos ligeramente entreabiertos; el aire espiratorio durante su salida produce una suave fricación.
   - AFRICADO ***/ĉ/*** Es un sonido oclusivo en su comienzo, pero al abrirse los órganos un poco, se convierte en fricativo.

3. Por la intervención de la glotis, los fonemas se clasifican en:
   - SORDOS /p f t s ***ĉ*** š x k/ Cuando en su articulación la glotis es sorda, es decir, basta para su fonación el aire contenido en la cavidad bucal.
   - SONOROS /b d y g l r ^r m n ñ / En la producción de estos fonemas es necesario utilizar el aire espirado por los pulmones, que al pasar por la laringe hace vibrar las cuerdas vocales, por lo que se dice que la glotis es sonora.

4. Por el movimiento del velo del paladar, los fonemas se clasifican en:
    - BUCALES u ORALES /p t ĉ k b d y g f s x l r ^r š/ Son los fonemas que se pronuncian con el velo del paladar elevado para impedir la salida del aire contenido en la boca por las fosas nasales.
    - NASALES /m n ñ/ Son los fonemas que se pronuncian manteniendo bajo el velo del paladar, lo cual permite que el aire contenido en la boca se escape por las fosas nasales.

Desde un punto de vista fonológico, los fonemas se distribuyen en órdenes y clases. Los órdenes son: labiales, dentoalveolares, palatales y velares, dentro de los cuales puede haber fonemas sonoros o sordos. Las clases de fonemas son: oclusivos, fricativos, laterales, vibrantes y nasales; de lo anterior resulta la tabla 3.3

Tabla 3.3

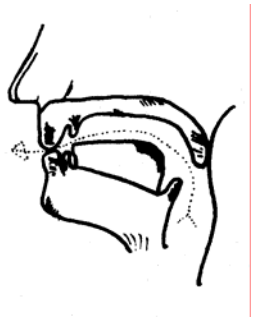
ORDEN → / CLASES ↓		LABIALES	LABIO-DENTALES	DENTALES	ALVEOLARES	PALATALES	VELARES
OCLUSIVAS	SORDAS	p		t			k
	SONORAS	b		d			g
AFRICADAS	SORDAS					ĉ	
FRICATIVA	SORDAS		f	s		š	x
	SONORAS					y	
NASALES		m		n	ñ		
VIBRANTES					r ^r		
LATERAL					l		

Los fonemas que están entre una categoría y otra en posición intermedia, participan de las órdenes o clases según su modo de articulación. Por ejemplo: los fonemas /b d y g/ pueden ser oclusivos o fricativos; el fonema /s/ puede ser sonoro y sordo. El fonema /**ĉ**/ se podría considerar oclusivo porque, siendo africado, aunque termina en fricación, empieza con una oclusión, que es lo distintivo fonológicamente. El fonema /**š**/ representa –como antes se ha dicho- un sonido palatal fricativo sordo.

3.4 Puntos de articulación de los fonemas del español hablado en México

Los puntos de articulación de los fonemas varían según el sonido que les antecede o les sucede, de acuerdo con la posición que tienen en la palabra, conforme a las normas sociales propias de cada entidad lingüística y según las normas individuales del hablante reforzadas por el hábito.

De una manera general, desde un punto de vista fonológico, esto es, sin considerar las variantes fónicas, los puntos de articulación de los fonemas del español son los que a continuación se enuncian:

/p/	
	**CARACTERÍSTICAS**.- Bilabial, oclusivo, oral y sordo. **Labios**.- Los labios están juntos y un poco contraídos. El aire se acumula en la boca, haciendo presión contra la pared labial tratando de separarlos, produciéndose la /p/ cuando se vence esta resistencia muscular. **Dientes**.- Algo separados sin ser visibles. **Lengua**.- No realiza ningún movimiento, la punta está colocada detrás de los incisivos inferiores y el resto extendido en el piso de la boca. **Velo del paladar**.- Se levanta contra la pared faríngea impidiendo el paso del aire por las fosas nasales, por lo que el aire sale totalmente por la boca. **Figura 3.2**

/t/	**CARACTERÍSTICAS**.- Dental, oclusivo, sordo y oral. **Labios**.- Los labios están entreabiertos. **Dientes**.- El espacio de separación de los dientes es muy pequeño. **Lengua**.- La punta de la lengua se levanta, apoyándose en la cara interna de los incisivos superiores. Sus bordes se apoyan en las coronas alveolares de los dientes, impidiendo la salida del aire. Cuando se pronuncia la /t/ la punta de la lengua se separa bruscamente, colocándose detrás de los incisivos inferiores. Esta retirada brusca produce una explosión (semejante a la /p/). **Velo del paladar**.- Se levanta, impidiendo la salida del aire por las fosas nasales. **Figura 3.3**

/ĉ/	**CARACTERÍSTICAS.-** Palatal, sordo, africado y oral. **Labios**.- Avanzan y se separan entre sí y de la cara anterior de los incisivos, permitiendo ver estos dientes. **Dientes**.- Se separan un poco. **Lengua**.- Las partes anterior y posterior de la lengua se apoyan en el paladar, la punta queda libre como suspendida entre los incisivos superiores e inferiores y sus bordes tocan los molares. La lengua toma una forma convexa formando un largo y estrecho canal que permite el paso del aire. . **Velo del paladar**.- Levantado, impidiendo el paso del aire a las fosas nasales. **Figura 3.4**

Haciendo referencia a la fig. 3.4, la articulación de este fonema tiene dos movimientos: en el primero, los órganos de articulación se colocan en la forma indicada y en el segundo, la parte anterior de la lengua se separa del paladar dejando un pequeño espacio frontal, por donde escapa el aire acumulado en la

boca, con fricación. La columna del aire es menos silbante que en la s, pero tiene más volumen que aquélla.

/g/ /k/	**Fonema /g/** **CARACTERÍSTICAS**.- Velar, oclusivo, sonoro y oral. **Labios y dientes.-** Medianamente separados, igual que el fonema /k/. **Lengua**.- La punta de la lengua se coloca detrás de los incisivos inferiores; el dorso se levanta y toca su parte posterior el velo del paladar. El contacto no es total y deja pasar el aire produciendo una suave fricación. El punto de contacto de la lengua y paladar se adelanta cuando va en sílabas con /a e i/; haciéndose más posterior con las vocales /o u/. **Velo del paladar**.- Levantado, impidiendo la salida del aire a las fosas nasales. **Fonema /k/** **CARACTERÍSTICAS.**- Velar, oclusivo, sordo y oral. **Labios.-** Separados, permitiendo ver los dientes y la lengua. **Dientes.-** Se alejan más de un centímetro. **Lengua.**- La punta de la lengua se coloca detrás de los incisivos inferiores tocando la encía; la parte posterior se levanta y se apoya con fuerza contra el velo del paladar, haciendo oclusión y cerrando el pasaje del aire. Igual que en todos los fonemas velares, la posición de la lengua varía según la vocal que le sigue, siendo post-palatal cuando va acompañada de /a e i/, y será velar cuando vaya acompañada de /o u/. **Velo del paladar**.- Se levanta. **Figura 3.5**

Respecto a la figura 3.5, el aire se acumula en la parte posterior de la boca. Cuando la lengua se desplaza separándose del velo del paladar, se produce la salida del aire y el ruido característico de este fonema. Para que esto suceda,

es necesario que la presión del aire sea superior a la tensión lingual y venza la resistencia de este órgano.

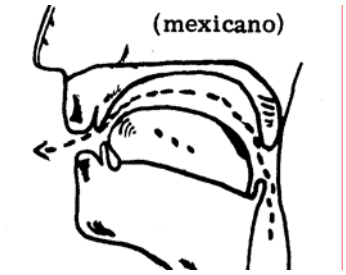
/b/	**CARACTERÍSTICAS.**- Bilabial, oclusivo, oral y sonoro. **Labios**.- Algo contraídos y ligeramente separados en el centro. La tensión muscular de los labios es débil. El aire espirado es sonoro y al pasar por la pequeña abertura central provoca un ligero temblor en los labios perceptibles al tacto. **Lengua**.- Ligeramente encorvada: su punta está colocada detrás de los incisivos inferiores y el resto extendido en el piso de la boca. **Velo del paladar**.- Levantado contra la pared faríngea. La corriente del aire sonoro sale por la boca. **Figura 3.6**

/d/	**CARACTERÍSTICAS.**- Interdental, oclusivo, sonoro y oral. **Labios**.- Los labios están entreabiertos permitiendo ver los dientes y la punta de la lengua. **Dientes**.- Están más separados que cuando se articula la /t/. La distancia entre ellos corresponde al espesor de la lengua. **Lengua.**- Se coloca entre los dientes haciendo una ligera presión contra las coronas de los dientes superiores. En las palabras que llevan d en posición inicial se saca menos la lengua que en la posición media. Cuando el fonema se prolonga por varios segundos la lengua vibra ligeramente. **Velo del paladar**.- Levantado; la corriente aérea, sonora, recorre el espacio que hay entre el dorso de la lengua y la arcada dental superior, saliendo totalmente por la boca. **Figura 3.7**

/y/	**CARACTERÍSTICAS**.- Palatal, fricativo, sonoro y bucal. **Labios**.- Entreabiertos, permitiendo ver los dientes. **Dientes**.- Ligeramente separados. **Lengua**.- El ápice de la lengua se coloca detrás de los incisivos superiores, su dorso se coloca ampliamente contra el paladar y sus bordes se separan de las coronas molares. **Velo del paladar**.- Levantado, impidiendo la salida del aire a las fosas nasales.- El aire sale por la parte lateral de la lengua y las coronas molares. **Figura 3.8**

/f/	**CARACTERÍSTICAS**.- Fricativo, labiodental, sordo y oral. **Labios y dientes**.- El labio inferior se repliega ligeramente, colocándose entre los incisivos superiores e inferiores y por debajo de los primeros; el labio superior se levanta un poco, dejando ver los incisivos superiores. Los incisivos inferiores quedan ocultos por el labio inferior; el aire sale por el centro, esto es, por entre el borde de los dientes superiores y el labio inferior. **Lengua**.- La punta se coloca detrás de los incisivos inferiores y se levanta un poco en sus bordes y en su base. **Velo del paladar**.- Se levanta contra la pared faríngea; la corriente de aire sale totalmente por la boca. **Figura 3.9**

/š/



CARACTERÍSTICAS.- Fricativo, predorso dentoalveolar convexo oral y sordo.

Labios.- Están entreabiertos, con las comisuras algo hacia atrás y dejando ver los dientes.

Dientes.- El maxilar inferior avanza un poco, colocándose los incisivos inferiores detrás de los superiores y casi juntos.

Lengua.- La lengua está arqueada; su punta se coloca detrás de los incisivos inferiores; la parte anterior de ella (predorso) se levanta, tocando los alveolos de los molares superiores; su dorso toca el paladar formándose un surco central lingual, por donde pasa la corriente de aire que choca contra los dientes superiores, desciende y sale produciendo un silbido característico, por lo que a este fonema se le llama "silbante".

Velo del paladar.- Levantado, impidiendo el pasaje del aire a las fosas nasales.

Figura 3.10

/x/

CARACTERÍSTICAS.- Velar, fricativo, sordo y oral.

Labios.- Entreabiertos, dejando ver los dientes y la lengua.

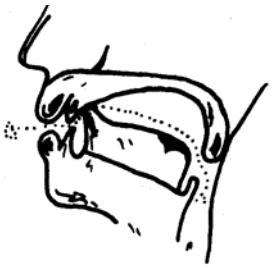
Dientes.- Separados un poco más de medio centímetro.

Lengua.- La punta de la lengua detrás de los incisivos inferiores más abajo del nivel de sus bordes libres. La lengua se ensancha y se arquea; sus bordes tocan los molares superiores y el dorso del velo del paladar formando un canal en su parte céntrica para permitir el paso del aire; su posición varía según la vocal que le siga. Si va acompañada de la /u/ es uvular y si le sigue la /i/ es casi palatal.

Velo del paladar.- Levantado, impide el paso del aire a las fosas nasales.

Figura 3.11

/l/



CARACTERÍSTICAS.- Alveolar, lateral, sonoro y oral.

Labios.- Están entreabiertos, sin contracción, permitiendo ver los dientes de ambos maxilares.

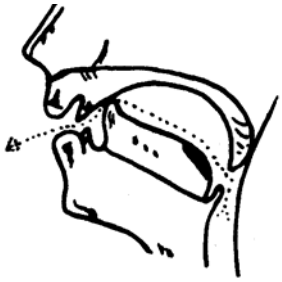
Dientes.- Están un poco separados; la posición de los labios y dientes permite ver la cara inferior de la lengua levantada hacia el paladar.

Lengua.- La punta de la lengua se levanta, apoyándose en las protuberancias alveolares de los incisivos superiores. Entre el borde de la lengua y los molares queda una abertura que permite el paso del aire, que choca contra la cara interna de las mejillas, haciéndolas vibrar.

Velo del paladar.- Levantado, impidiendo el paso del aire a las fosas nasales.

Figura 3.12

/r/



CARACTERÍSTICAS.- Apicoalveolar, vibrante simple, oral y sonoro.

Labios.- Los labios están entreabiertos, permitiendo ver los incisivos inferiores y superiores.

Dientes.- Están separados; la posición de los labios y los dientes permite ver la cara inferior de la lengua que está levantada hasta el paladar.

Lengua.- La punta de la lengua (ápice) se coloca en la protuberancia alveolar de los incisivos superiores; sus bordes tocan los molares, la encía y parte del paladar, impidiendo la salida lateral del aire.

Velo del paladar.- Levantado; el aire sale por la boca. El aire se acumula en la boca y al pronunciar el fonema la punta de la lengua se separa rápidamente de su posición, saliendo el aire en forma de pequeña explosión. Este fonema se presenta en sílabas trabadas, en las cuales aumenta ligeramente su vibración o fricación, como en: [kára], [tóro], [períko], [séro], y en sílabas trabadas en las cuales aumenta ligeramente su vibración o fricación, como en: [árte], [mar], [pwérta].

Figura 3.13

/^r/	**CARACTERÍSTICAS**.- Apicoalveolar, vibrante múltiple, oral y sonoro. **Labios**.- Están entreabiertos, permitiendo ver los incisivos superiores e inferiores. **Dientes**.- Separados; permiten ver la cara inferior de la lengua, que se levanta hasta el paladar. **Lengua**.- La punta de la lengua (ápice) se apoya con cierta fuerza en las protuberancias alveolares de los incisivos superiores; sus bordes tocan la cara interna de los molares, la encía y parte del paladar, impidiendo la salida lateral del aire; el dorso de la lengua, en su centro, toma una forma cóncava. **Velo del paladar**.- Levantado, impidiendo la salida del aire por las fosas nasales. **Figura 3.14**

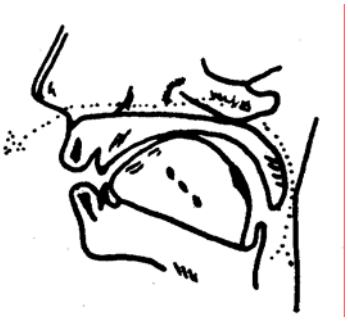
Cabe mencionar que en la figura 3.14 que el aire se acumula en la boca; la presión de la lengua sobre la protuberancia alveolar es vencida por la presión del aire, permitiendo su expulsión; al vencer el aire, la lengua muestra resistencia para impedir su salida y vibra. La articulación de este fonema exige gran agilidad en la punta de la lengua. La vibración de la lengua en este fonema se presenta en su grado máximo y se observa en las siguientes posiciones:

1.- Posición media: [péro], [búro], [tóre], etc.

2.- Posición inicial de la palabra: [rósa], [ratón], etc.

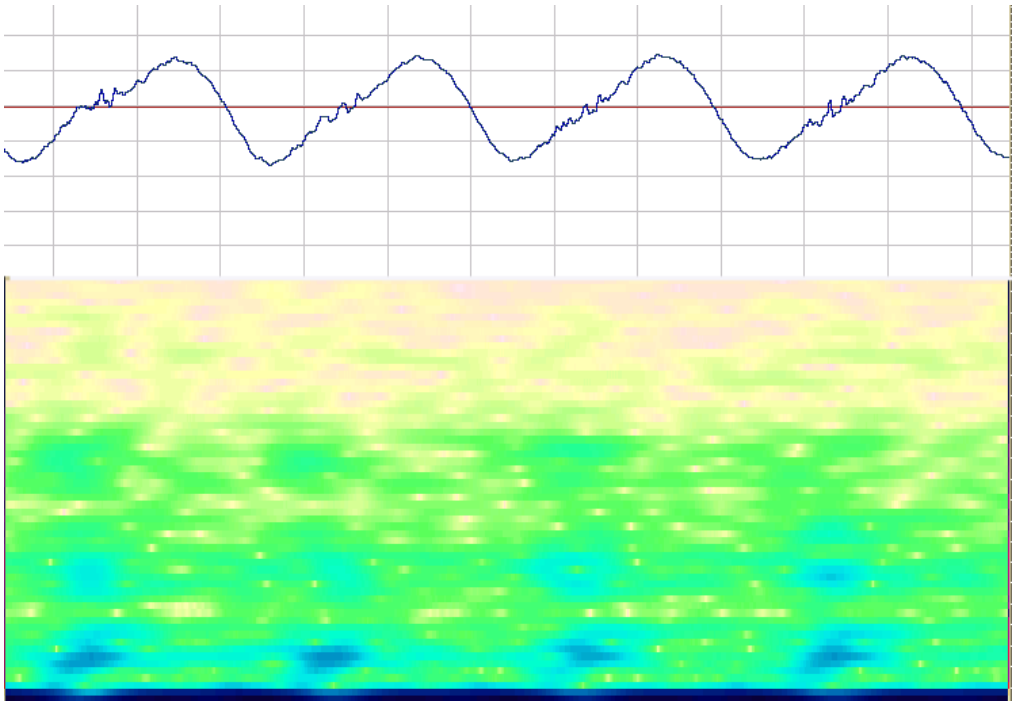
/m/	**CARACTERÍSTICAS**.- Bilabial, sonoro y nasal. **Labios**.- Los labios unidos sin llegar a contraerse; la presión labial es media. **Dientes**.- Casi juntos, estando los incisivos detrás de los superiores. **Lengua**.- La punta de la lengua se encuentra colocada detrás de los incisivos inferiores y el resto de ella en el piso de la boca. **Velo del paladar**.- El velo desciende, dejando libre el paso del aire a la cavidad nasal. **Figura 3.15**

/n/ 	**CARACTERÍSTICAS**.- Alveolar, sonoro y nasal. **Labios**.- Entreabiertos, permitiendo ver los incisivos y la cara inferior de la lengua. **Dientes**.- Separados aproximadamente 5 mm. **Lengua**.- La punta de la lengua se levanta, apoyándose en la protuberancia alveolar de los incisivos superiores; sus bordes tocan la cara interna de los molares y encías y su dorso está en contacto con una pequeña parte del paladar. **Velo del paladar**.- Desciende, poniendo en comunicación la cavidad bucal y sale por las fosas nasales. **Figura 3.16**

/ñ/ 	**CARACTERÍSTICAS**.- Palatal, sonoro y nasal. **Labios**.- Están entreabiertos, permitiendo ver los incisivos superiores o inferiores. **Dientes**.- Casi juntos (unos 2 ó 3 mm de separación). **Lengua**.- El dorso de la lengua se coloca ampliamente contra la palabra óseo, empezando este contacto desde las protuberancias alveolares de los incisivos superiores hasta la parte posterior. Los bordes de la lengua tocan la cara interna de los molares. La punta de la lengua queda libre detrás de los incisivos sin tocarlos. Por la posición de la lengua, se impide la salida frontal o lateral del aire por la boca. **Velo del paladar**.- Desciende; el aire sale totalmente por las fosas nasales. **Figura 3.17**

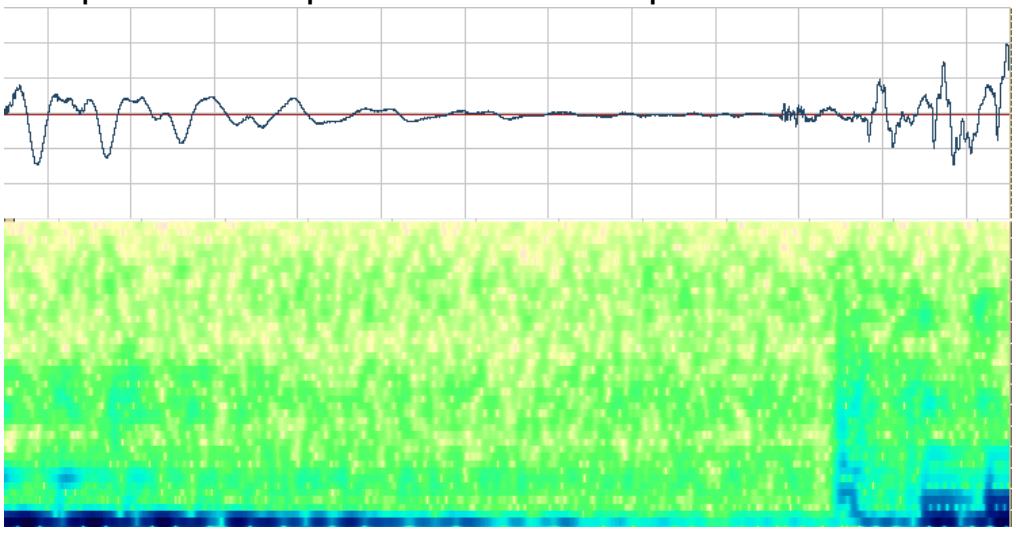
Formas de onda y espectrogramas de algunos fonemas

1.- Fonema /i/



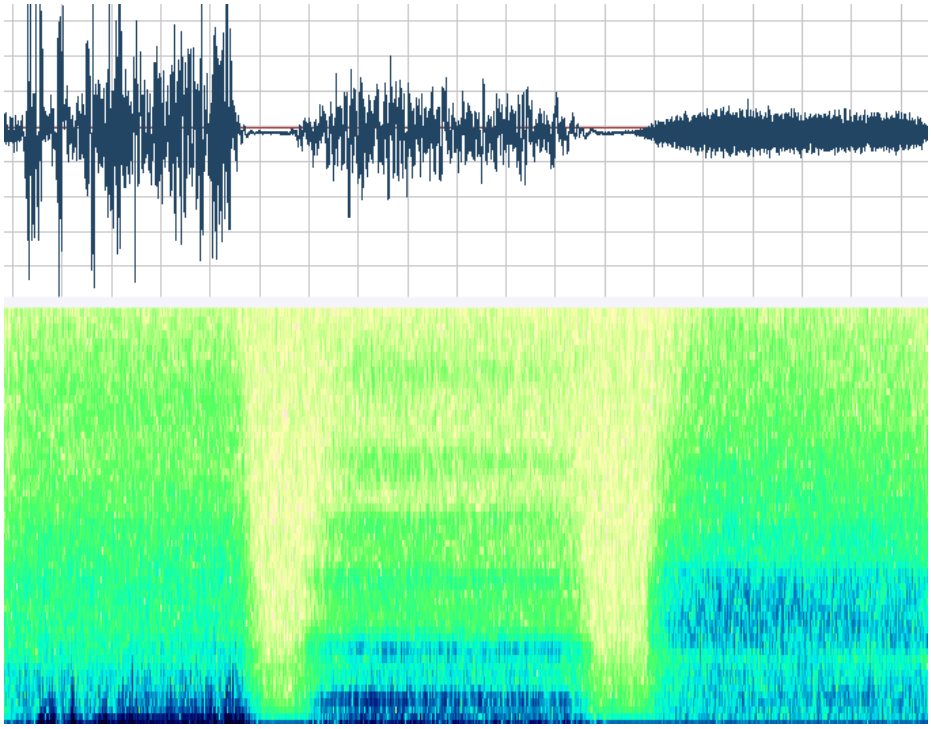
Gráfica 3.1

2.- Fonema /t/ dentro de una palabra. Antes de un fonema plosivo, existe una pequeña pausa en lo que se obtiene la presión de aire suficiente para efectuar la exhalación.



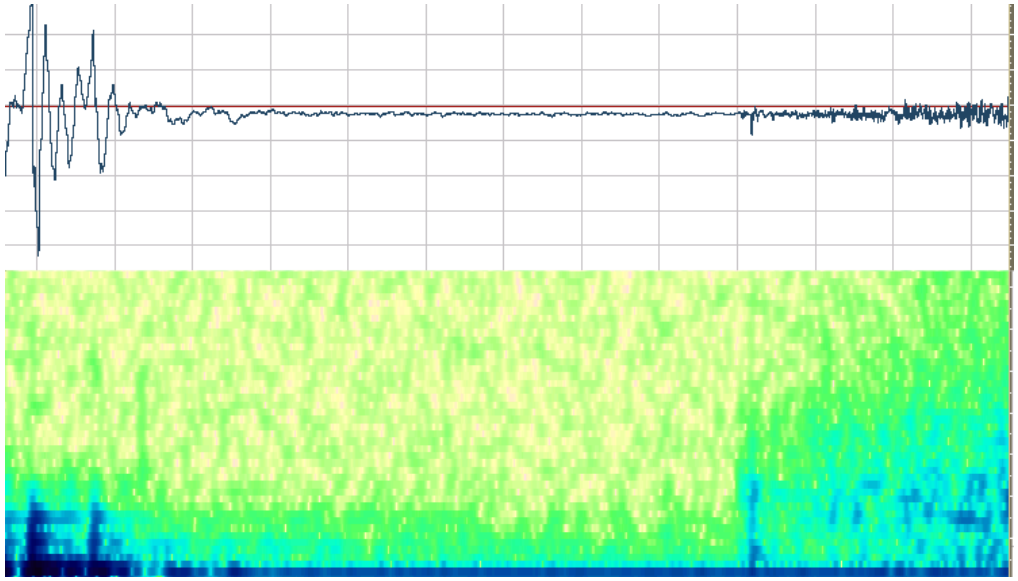
Gráfica 3.2

3.- Fonemas /f/,/j/ y /s/. Los fonemas fricativos están compuestos principalmente por ruido



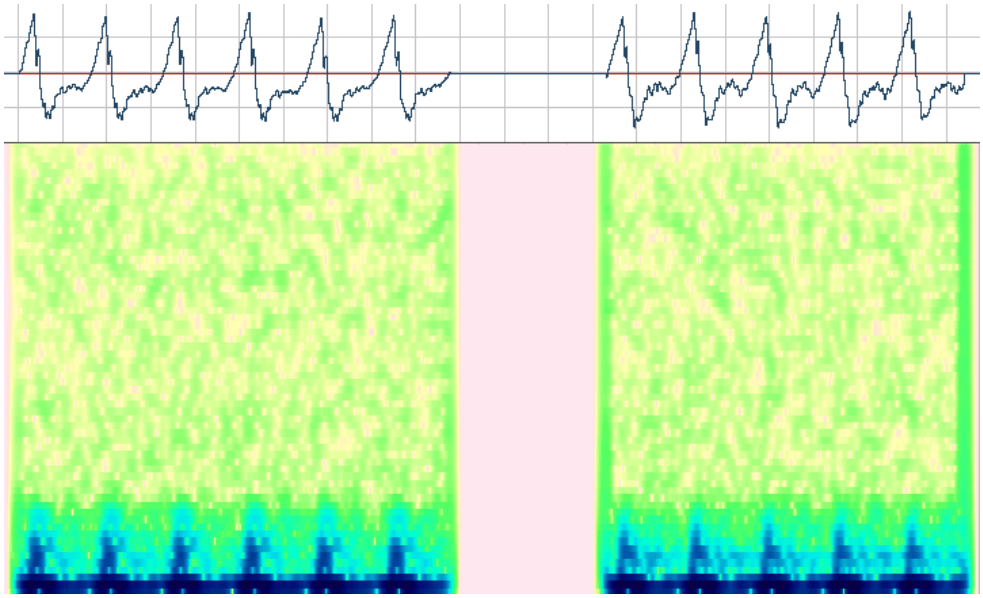
Gráfica 3.3

4.- Fonema /ĉ/. Nótese la pausa que existe al inicio del fonema al igual que en las plosivas



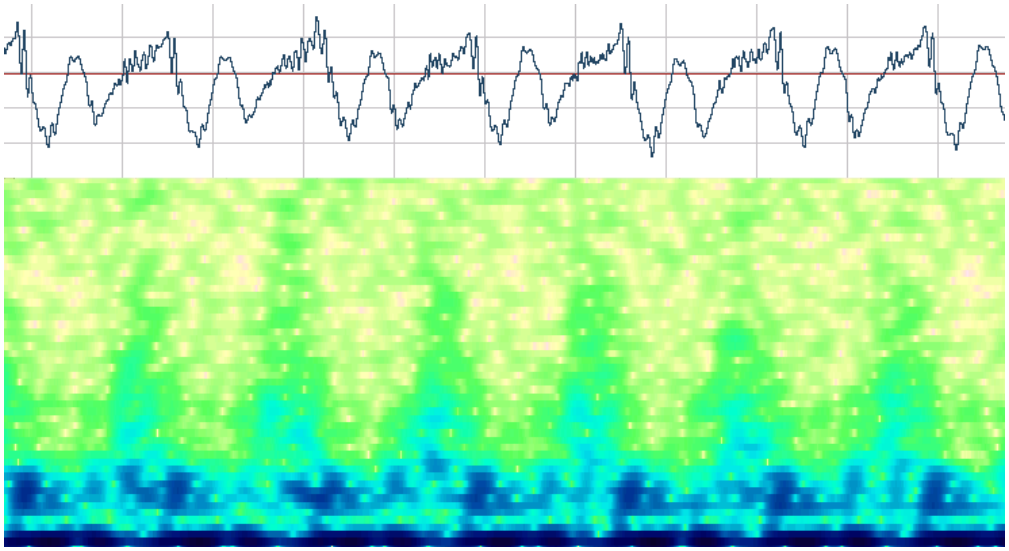
Gráfica 3.4

5.- Fonema /y/. Nótese que las semivocales son periódicas al igual que las vocales



Gráfica 3.5

6.- Fonemas /m/ y /n/



Gráfica 3.6

3.5 Reglas de transcripción grafema a fonema

3.5.1 Reglas básicas para obtener los fonemas a partir de los grafos

La tabla 3.4 muestra las reglas de transcripción grafema a fonema del idioma español.

Tabla 3.4

El grafo:	toma el sonido (fonema):	si...
c	/s/	está seguida de una vocal débil (e,i)
	/k/	está seguida de una vocal fuerte (a,o,u)
	/ĉ/	está seguida de la letra h
s	/s/	excepto si está seguida por una 'h'
	/š/	está seguida la letra h
l	/l/	excepto si forma ´ll´ en cuyo caso forma el sonido /y/
r	/r/	está en medio de una palabra, después de una vocal y la letra siguiente es diferente de 'r'
	/^r/	la letra siguiente es r
qu	/k/	después de la 'q' no existe una 'u' se mantiene el sonido /k/ para poder leer palabras mal escritas ya que este caso no existe en el español.
y	/y/	a menos que sea la última letra de una palabra o esté seguida por una consonante en cuyo caso tiene el sonido /i/.
g	/x/	está seguida de 'e' o 'i'
	/g/	es otro caso si la letra siguiente es 'u' seguida de una vocal débil ('e' o 'i'; ésta no se pronuncia a menos que tenga un símbolo de diéresis (ü)
X	/s/	aunque en algunos casos tiene pronunciación 'ks'

3.5.2 Sílaba tónica

Una vez que se ha obtenido la secuencia de fonemas, se debe encontrar la sílaba que lleva el énfasis dentro de la palabra llamada sílaba tónica, la cual puede ser representada por un acento escrito (´). El acento será escrito en los siguientes casos de acuerdo a la sílaba que lleva el acento tónico:

- Palabras Agudas: la sílaba que lleva el énfasis es la última; llevan acento escrito sólo si terminan en n, s o vocal.

- Palabras Graves: la sílaba que lleva el énfasis es la penúltima; llevan acento escrito sólo si no terminan en n, s o vocal.

- Palabras Esdrújulas: la sílaba que lleva el énfasis es la antepenúltima y siempre llevan acento escrito.

- Palabras Sobre-esdrújulas: llevan el énfasis antes de la penúltima sílaba y siempre llevan acento escrito.

Si la palabra lleva acento escrito ya no se requiere hacer nada para encontrar la sílaba tónica. En caso contrario se debe obtener la última o las dos últimas sílabas de la palabra para encontrar el punto donde se colocará el acento tónico.

3.5.3 Separación de sílabas

Para la separación de sílabas, existen 10 reglas básicas:

REGLA 1
En las sílabas, por lo menos, siempre tiene que haber una vocal. Sin vocal, no hay sílaba.

REGLA 2
Existen conjuntos de consonantes que deben ser mantenidos juntos y pertenecen siempre a la misma sílaba:

br, bl, cr, cl, dr, fr, fl, gr, gl, kr, ll, pr, pl, tr, rr, ch.

REGLA 3

Cuando una consonante se encuentra entre dos vocales, se une a la segunda vocal.

Ejemplo: une -> u-ne

REGLA 4

Cuando hay dos consonantes entre dos vocales, cada vocal se une a una consonante excepto si son consonantes consideradas inseparables (ver regla 2)

Ejemplo: componer -> com-po-ner
Aprender -> a-pren-der

REGLA 5

Si son tres las consonantes colocadas entre dos vocales, las dos primeras consonantes se asociarán con la primera vocal y la tercera consonante con la segunda vocal excepto si la segunda y tercera consonantes están dentro del grupo de inseparables.

Ejemplo: transporte -> trans-por-te
Cumple -> cum-ple

REGLA 6

Las palabras que contienen un grafo 'h' precedida o seguida de otra consonante, se dividen separando ambas letras.

Ejemplo. Anhelo -> an-he-lo

REGLA 7

El diptongo es la unión inseparable de dos vocales. Se pueden presentar tres tipos de diptongos posibles:

1. Una vocal abierta + una vocal cerrada
   ai, au, ay, oy, ey, ei, eu, ou, oi
2. Una vocal cerrada + una vocal abierta
   io, ua, ua, ie, ue, uo
3. Una vocal cerrada + una vocal cerrada
   ui, iu

Entonces podemos resumir que son diptongos sólo las siguientes parejas de vocales:

ai, au, ei, eu, io, ou, ia, ua, ie, ue, oi, uo, ui, iu, ay, ey, oy.

Ejemplo: jaula -> jau-la

La unión de dos vocales abiertas o semiabiertas no forma diptongo, es decir, deben separarse en la segmentación silábica. Pueden quedar solas o unidas a una consonante.

Ejemplo: aéreo -> a-é-re-o

REGLA 8

El grtafo 'h' entre dos vocales, no destruye un diptongo.

Ejemplo: ahuyentar -> ahu-yen-tar

REGLA 9

La acentuación sobre la vocal cerrada de un diptongo provoca su destrucción.

Ejemplo: María -> Ma-rí-a

REGLA 10

La unión de tres vocales forma un triptongo. La única disposición posible para la formación de triptongos es la que indica el esquema:

Vocal cerrada + vocal abierta o semiabierta + vocal cerrada

Sólo las siguientes combinaciones de vocales, forman un triptongo:

iai, iei, uai, uei, uau, iau, uay, uey.

De acuerdo a estas reglas existen 5 tipos de sílabas:

- V → vocal (1 ó 2)
- VC → vocal (1 ó 2) + consonante (1)
- CV → consonante (1 ó 2) + vocal (1,2 ó 3)
- CVC → consonante (1 ó 2) + vocal (1,2 ó 3) + consonante (1 ó 2)
- CCVCC → consonante (1 ó 2) + consonante (1 ó 2) + vocal (1,2 ó 3) + consonante (1 ó 2) + consonante (1 ó 2)

Capítulo 4

TÉCNICAS DE SÍNTESIS DE VOZ

4.1 Generalidades

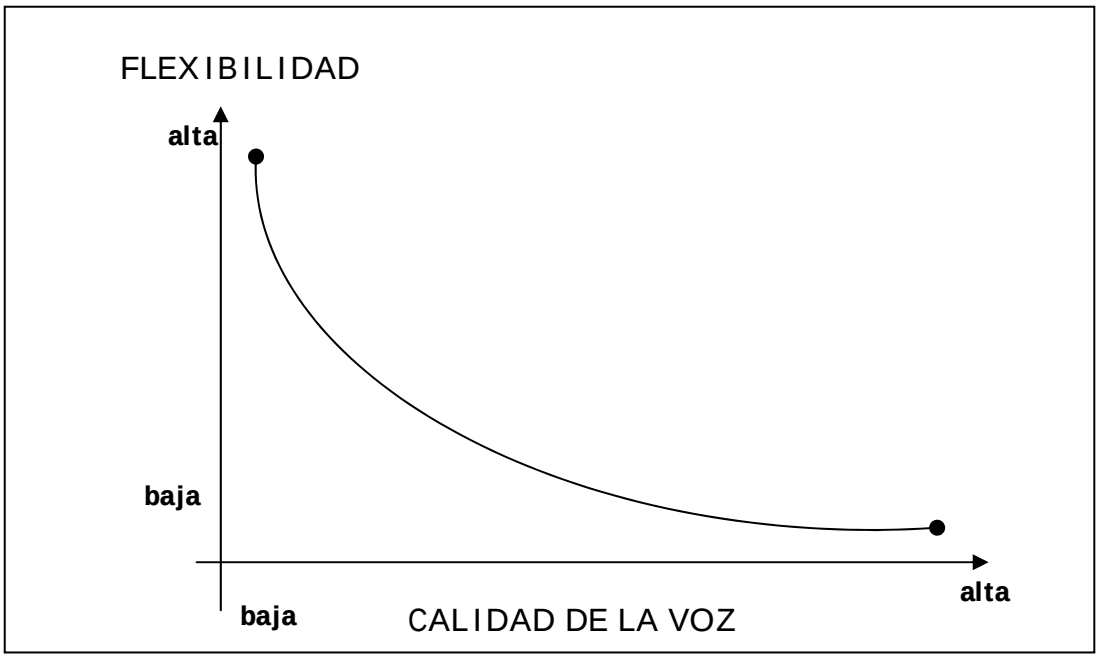
La síntesis de voz, respecto del proceso de generar voz a partir de texto, es una emulación del proceso del habla producido por el humano a través de las cuerdas vocales y tiene gran importancia porque apoya en el desempeño de un sin fin de tareas. La investigación de este capítulo está basada en textos y consulta de páginas WEB referentes al procesamiento digital de voz así como de artículos especializados en sistemas de conversión de texto en habla: Sammi Lemmetty, *Review of Speech Synthesis Technology;* Abe, M.: *Speaking Styles: Statistical Analysis and Synthesis by a Text-to-Speech System*; Santen, J.P.H. - Sproat, R.W.- Olive, J.P.- Hirschberg. J. (Eds.) *Progress in Speech Synthesis*. New York: Springer. pp. 495-510., 1997, entre otros mencionados en la bibliografía.

Los sistemas de conversión de texto en habla o TTS, por sus siglas en inglés, (Text-to-Speech), es el proceso de transformación de un texto a una señal auditiva, también conocida como un tipo de síntesis de voz. Un sistema TTS tiene como finalidad la naturalidad, calidad e inteligibilidad del habla sintetizada.

- Síntesis del habla como técnica de generación automática de una señal vocal para la producción artificial de mensajes orales:
    a. Con vocabulario restringido a partir de la decodificación de señales sonoras previamente almacenadas y codificadas.
    b. Síntesis a partir de un texto escrito (TTS Text to Speech Síntesis, CYH Conversión de texto al habla).
- La síntesis es una técnica complementaria del reconocimiento en la comunicación persona-máquina.

Actualmente, existen diversos sintetizadores de diferentes niveles de desempeño. Al desarrollar un sistema de texto al habla, es importante tener claro cinco factores:

1. Determinar cuál va a ser el contexto del sistema.
2. El tipo de unidades que se va a manejar (fonemas, difonemas, trifonemas, sílabas, entre otros).
3. El costo/beneficio entre la flexibilidad y calidad de voz que se requiere. Ver gráfica 4.1



Gráfica 4.1 Costo/Beneficio entre flexibilidad y calidad de voz

4. La metodología a utilizar como mecanismo para generar la voz.
5. La arquitectura del sistema de texto al habla.

Existen diferentes tipos de enfoques para la producción de voz a través de los sistemas de texto al habla; entre ellos están los que manejan síntesis paramétrica, síntesis articulatoria y la síntesis concatenativa.

En la síntesis concatenativa, se genera voz sintética uniendo unidades de voz digitalizadas, por ejemplo, fonemas, difonemas, sílabas, entre otros. Este tipo de síntesis se usa en los sistemas “TTS Festival” y es el tipo de síntesis con el que se desarrolló este proyecto, y actualmente ésta ofrece los mejores

resultados junto con la técnica llamada “Unit Selection”, propia de los sintetizadores que implementan esta tecnología.

Los segmentos de voz que se utilizan en este tipo de síntesis se almacenan a partir de grabaciones hechas por alguna persona con el objetivo de conservar las propiedades fonológicas de los segmentos. Las unidades que participan en la concatenación pueden ser de diferentes tipos:

FONEMAS

Son unidades naturales que dan la gran flexibilidad a los sistemas de voz y que resultan económicas desde el punto de vista del número de unidades, en nuestro idioma Español. Como ya se mencionó en el capítulo anterior, son 23; sin embargo, constituyen una unidad abstracta que está sometida a muchas variaciones contextuales que originan una baja calidad en la voz sintetizada.

DIFONEMAS

Estos consisten en la unión de la parte estable de un fonema (mitad del fonema) con la parte estable del siguiente fonema. Existen 23^2 posibles difonemas y a pesar de los métodos para suavizar las fronteras, este tipo de síntesis todavía no se escucha natural.

SÍLABAS

La calidad que se obtiene a partir de las sílabas es mucho mejor que las anteriores ya que a través de las sílabas podemos manejar mejor la coarticulación, ya que las unidades pueden ser más grandes y por lo tanto más completas. El problema es el número de sílabas tan grande.

PALABRAS

Es el tipo de concatenación de más alto nivel y donde se puede obtener la mayor naturalidad de voz posible, pero es necesario contar con un conjunto completo de palabras para un dominio predeterminado.

Las UNIDADES DE LONGITUD VARIABLE se concatenan en partes de frases, palabras o longitudes menores, aprovechando las frases ya contenidas en el corpus, como se explicará a continuación.

La Selección de Unidades (UNIT SELECTION) en inglés, es una metodología de síntesis de voz mediante la cual podemos concatenar las formas sonoras de

diferentes estructuras gramaticales, tales como los fonemas, difonemas, trifonemas, incluso palabras y frases completas, tanto como se tengan en el corpus.

4.2 Parámetros en el diseño y evaluación de un sistema de síntesis

Los parámetros más importantes para el diseño de un sintetizador de voz son los siguientes:

- La ***naturalidad, calidad*** e ***inteligibilidad*** del habla sintetizada.
- La ***versatilidad del sistema:*** Vocabulario limitado vs. Generación de textos sin restricciones.
- Condicionantes ***lingüísticos*** y condicionantes ***tecnológicos***; considerar la utilidad del producto y realización de pruebas con una interfaz de evaluación que permite verificar si el concepto del diseño es adecuado para el entorno de aplicación.
- Relación entre el ***sistema*** y la ***aplicación:*** buen diseño del programa.
- La concepción de la síntesis como ***modelo de producción del habla*** y como aplicación tecnológica.
- ***Evaluación***

Se debe considerar, además de los puntos anteriores, el tipo de elementos que se utilizarán: la información lingüística en la síntesis

Elementos segmentales

Definición de las mejores unidades de análisis y síntesis.
Extracción automática de las unidades a partir de un corpus.
Mejora de los modelos de transición entre unidades.

Elementos suprasegmentales (duración, frecuencia fundamental –F_0 e intensidad).

Modelos prosódicos.
Modelos de fuente para mejorar y variar la calidad de la voz.
Flexibilidad en la elección de estilos.

4.3 Sintetizadores de voz

4.3.1 Historia y evolución de los sintetizadores de voz

SINTETIZADORES DE VOZ MECÁNICOS

Desde la segunda mitad del siglo XVIII, se han construido diversos sistemas: primero mecánicos, luego electrónicos y actualmente digitales, que sean capaces de generar una salida de voz de forma artificial.

El primer intento registrado fue realizado en 1773 donde Ch. G. Kratzenstein, un profesor de fisiología de Copenhague logró producir sonidos vocálicos a partir de tubos de resonancia conectados a tubos de órgano.

Simultáneamente a este invento, Wolfgang von Kempelen Hungría 1734 – Viena 1804, trabajaba en la construcción de una máquina que pudiera generar sonidos que simularan a la voz humana. En 1791, publicó sus descubrimientos y la forma en que se podía construir su máquina de forma que otras personas pudieran continuar su investigación. Esta máquina funcionaba por medio de un fuelle que generaba una salida de aire. Éste después era enviado través de varias tuberías y aberturas que podían ser modificadas manualmente para producir palabras o frases cortas.

Durante el siglo XIX, se construyeron otros aparatos similares que mejoraron levemente el modelo de Kempelen al introducir elementos adicionales que simularan la lengua o los cambios en la forma de la boca, que se realizan al hablar.

SINTETIZADORES DE VOZ ELÉCTRICOS

En 1922, Stewart construyó el primer sistema que intentaba simular la voz mecánica por medios eléctricos. Este dispositivo constaba de dos circuitos resonantes activados por un buzzer. Esto permitía aproximarse a vocales estáticas al ajustar dos de las frecuencias fundamentales (resonantes) de cada una de las vocales.

En 1939, se diseñó el VODER en los laboratorios Bell. Este aparato estaba diseñado para analizar la voz humana y obtener de ésta parámetros acústicos. Después, a partir de estos parámetros, se podía reconstruir una salida similar a la onda de voz original.

A partir de este sistema, se construyó una segunda versión que fue mostrada en la feria mundial de Nueva York (1939). Este sistema llamado VODER y desarrollado por Homer Dudley generaba una salida de ruido o una salida de audio senoidal, de acuerdo a un selector. La frecuencia de esta salida podía ser controlada por un pedal. Esta salida era después filtrada por 10 filtros pasobanda cuya amplitud se modificaba por medio de los dedos. Aunque la inteligibilidad de este sistema es mínima, fue el primer sistema en mostrar la posibilidad de generar voz por medios eléctricos. El sistema de filtros fijos que utiliza este sistema es demasiado limitado para generar las diferentes salidas requeridas, por lo que en los sistemas más modernos este sistema no es utilizado.

En 1950, se presentó el reproductor de patrones desarrollado en los laboratorios Haskins. Éste utilizaba espectrogramas los cuales se iluminaban y eran enviados a un conjunto de celdas fotovoltaicas, cada una de las cuales controlaba la intensidad de una onda fundamental de diferentes frecuencias en saltos de 120 Hz que podían reconstruir aproximadamente la señal del espectrograma. Franklin Cooper, Alvin Liberman, Pierre Delattre, y otros asistentes, experimentaron con espectrogramas reales y con adaptaciones dibujadas a mano para experimentar con la importancia de diferentes factores. Este sistema generó una inteligibilidad mucho más alta (de más del 90% con espectrogramas reales y de poco más del 80% con espectrogramas estilizados).

Estos dos sistemas funcionaban copiando los patrones espectrales de la voz. Poco después de estos sistemas, se le dio un nuevo enfoque a la teoría de síntesis de voz. Este nuevo enfoque fue la generación de una teoría acústica de la forma en que se produce la voz y no sólo en los resultados del proceso. Esta teoría es la base de la síntesis por formantes. Según esta teoría, la voz se puede considerar como la salida de un filtro lineal excitado por una o más fuentes, principalmente las cuerdas vocales y por ruido turbulento debido a diferencia de presiones a través de un estrangulamiento. El filtro en este caso es una simulación de los efectos del conducto acústico (faringe, cavidad oral y

labios). Este tracto vocal es simulado por una función de transferencia con pares de polos complejos conjugados que producen picos en el espectro de salida, llamados formantes. Además de los polos, se requiere la introducción de ceros (antiresonadores) para modelar la absorción de las ramas laterales en algunas articulaciones como nasales y fricativas. Ver figura 4.1

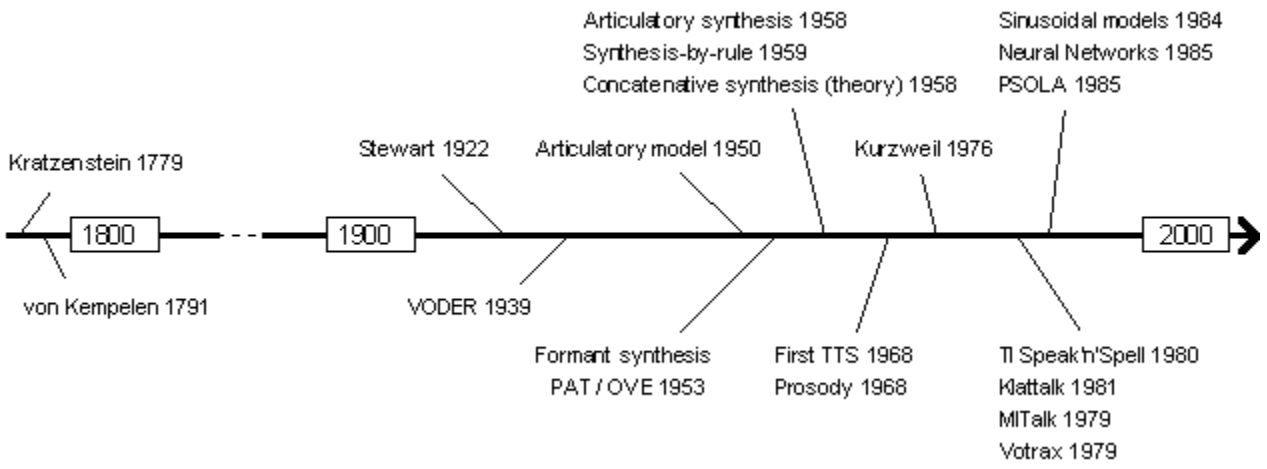


Figura 4.1 Evolución de los sintetizadores de voz

En 1953, se crearon los primeros sintetizadores de formantes (Parametric Artificial Talker (PAT)) construido por Walter Lawrence y el orador Verbis Electris (OVE I), que construyó Gunnar Fant. Estos sistemas fueron enfrentados en una conversación en 1956 en el MIT.

El PAT tenía tres resonadores en paralelo. Se tenía una señal de entrada de ruido o periódica, y de ahí, a través de patrones dibujados sobre un vidrio que se deslizaba, se controlaban las 3 frecuencias del formante, amplitud del ruido, amplitud de fraseo y fundamental.

El OVE utilizaba filtros en cascada en lugar de en paralelo. Los dos más bajos eran controlados por movimientos en dos dimensiones de un brazo mecánico, mientras que la amplitud y la fundamental eran controladas por potenciómetros. Sin embargo, este sistema sólo podía generar vocales.

Es interesante notar que aunque ambos sistemas parten de la misma teoría y usan los mismos principios, utilizaron diferentes métodos para llevarla a la práctica y hasta la fecha aún existe controversia sobre cuál de los dos métodos es mejor o si la mejor opción es utilizar una mezcla de ambos, teoría propuesta en 1972 por Klatt.

Estos sistemas sufrieron varias modificaciones. A PAT se le introdujeron controles individuales para los formantes y un circuito independiente para fricativas, llegando a ser un sistema en cascada. A OVE I se le agregó otra rama estática para simular murmullos nasales y una cascada de dos formantes y un antiformante para simular mejor la función de transferencia del tracto vocal y la excitación de los sonidos fricativos, transformándose en OVE II. Estos sistemas mejorados fueron enfrentados una vez más, en 1962 en una conferencia en Estocolmo.

Con algunas modificaciones como modulando la amplitud del ruido en fricativas sonoras o agregando nuevos parámetros, estos dos sistemas continúan siendo la base de los modernos de síntesis por formantes. Uno de los más grandes cambios a esta clase de sistemas fue la introducción de sistemas híbridos (cascada y paralelos).

En el sistema propuesto por Klatt, cada sistema se utilizaba para modelar diferentes tipos de sonidos, en paralelo para sonidos sonoros y en cascada para sonidos sordos. Este sistema propuesto por Klatt fue además presentado como un listado en Fortran, en 1980, lo que permitió su uso más extendido.

Es importante señalar que en la historia de los sintetizadores por formantes, en una conferencia en Boston, en 1972, cuando John Holmes presentó una salida de voz generada, resultó que era prácticamente indistinguible de una voz natural. Desafortunadamente, esta señal fue generada de forma manual y basada en un proceso de prueba y error de varios meses de duración. Aunque este experimento demostró varios factores importantes en la generación de

oraciones y otros factores que podían ser despreciados, su método no ha podido ser llevado a métodos automáticos.

El siguiente avance importante en este tipo de sintetizadores es cambiar el tipo de señal de entrada de una señal monótona (triangular, tren de pulsos), en una señal que más se asemeje a la señal que entra al tracto vocal.

El primer avance de este tipo se dio en 1975 (Rothenberg), donde se utilizó un sistema de tres parámetros de acuerdo a la apertura de la glotis y la respiración. Se han creado métodos más avanzados que simulen más parámetros, pero hasta la fecha el resultado aún no es completamente natural, debido posiblemente a la falta de conocimiento del modelo real.

Adicional a este tipo de sintetizadores, está otra línea paralela de investigación que es la generación de líneas de transmisión simulando un tubo similar al tracto vocal. Sin embargo, debido a restricciones en el conocimiento del tracto vocal y en cantidad de cálculos, se ha avanzado poco en esta área, aunque existen algunos modelos de sintetizadores de este tipo.

Una vez que se obtuvieron sistemas que pudieran simular la voz humana, una aplicación muy importante que sólo se hizo posible con el advenimiento de computadoras y circuitos integrados es la generación de una señal de voz a partir de una entrada fonética o de texto.

El primer programa de este tipo se desarrolló en 1961 (Kelly y Gerstman), con un sintetizador en cascada de tres formantes, cuyos parámetros posteriormente se modificaban a mano.

En 1964, apareció otro sistema (Holmes) que funcionaba a partir de síntesis por formantes en paralelo y un conjunto de tablas que permitía generar resultados más complejos como coarticulación. En 1966, (Matt-ingly) modificó el programa para dar transiciones más realistas, pero con poca mejora perceptiva y el uso de alófonos.

El primer uso práctico que se le intentó dar a este sistema, fue una adaptación como parte de una máquina de lectura para ciegos, pero nunca se concretizó por falta de recursos.

A finales de los 60 y principio de los 70, se continuó la investigación de los sistemas de síntesis por regla ajustando diferentes parámetros para hacerlos más similares a la voz natural, principalmente por Klatt, dando como resultado el sistema de síntesis del M.I.T. MITalk (1976), el cual fue vendido y cambió de manos varias veces durante los siguientes años. Después de este sistema, se desarrolló el Klattalk que continuó siendo mejorado hasta finales de los 80.

El Votrax SC-01 (1976), fue el primer sistema de síntesis por formantes en estar incorporado dentro de un circuito integrado dentro de sistemas de síntesis de bajo costo y el TMS-5520 de Texas Instruments, que es la base del Echo, un circuito de síntesis por concatenación de segmentos pregrabados. También, se ha desarrollado la investigación de tomar segmentos de voz pregrabados como bloques para construir una frase cualquiera.

Debido a las características de la voz, no se pueden usar palabras o sílabas (son demasiadas); ni fonemas (aunque son pocos, no toman en consideración los efectos de coarticulación ni la transición entre fonemas). Debido a esto, en 1958, Peterson propuso una unidad denominada difonema, que corresponde al segmento entre el centro de un fonema al centro del siguiente, debido a que así, sí se toma en cuenta la transición y los efectos de la coarticulación, aunque sean pocos en el centro de un fonema.

En teoría, se requiere sólo el cuadrado del número de fonemas de una lengua de difonemas, pero hay combinaciones que no pueden existir con lo que se reduce el número, pero se pueden agregar algunos difonemas para grabar diferencias entre sílabas acentuadas y no alófonos. Peterson estimó que se requieren unos 8000 difonemas para el idioma inglés, aunque el número normal en un sistema es más cercano a 1000.

En 1961, (Sivertsen) propuso mezclar difonemas y unidades más largas llamadas diadas, que contienen la mitad final de un fonema, un fonema completo y la mitad inicial del siguiente (VCV) para conservar algunos fenómenos que pueden no estar previstos en un difonema.

Aunque tienen algunos problemas como discontinuidades en algunas uniones, estos sistemas son muy utilizados debido a su relativa sencillez y alta inteligibilidad. El primer sistema de este tipo fue mostrado en 1967 en el MIT, pero este sistema se canceló por falta de recursos.

En 1976, Olive y Spickenagle intentaron extraer las características de los fonemas para crear un sistema que generara un cátalogo de difonemas de forma automatizada.

Este sistema de síntesis es el más utilizado actualmente y la investigación actual es sobre métodos para mejorar la naturalidad de la voz, disminuyendo o eliminando discontinuidades y desarrollando métodos de generación de contornos para mejorar la entonación, así como análisis sintácticos y semánticos que permitan mejorar los contornos de acuerdo al contenido de una oración.

Las tablas siguientes describen brevemente los diferentes tipos de sintetizadores que se han tenido a lo largo del tiempo, de acuerdo al desarrollo.

TABLAS RESUMEN
HISTORIA Y EVOLUCIÓN DE LOS SINTETIZADORES DE VOZ

Desarrollo de sintetizadores de voz en el periodo 1939-1982
• El VODER desarrollado por Homer Dudley, 1939.
• El “pattern playback” diseñado poor Franklin Cooper, 1951.
• PAT, por sus siglas en inglés "Parametric Artificial Talker" de Walter Lawrence, 1953
• El "OVE" es un sintetizador de formantes en cascada creado por Gunnar Fant
• En 1962, se copia oración usando el PAT, sintetizador de formantes de Walter Lawrence
• En 1962, utilizando la misma oración del ejemplo anterior se realiza con la segunda generación de sintetizadores de formantes en cascada OVE de Gunar Fant's
• En 1961, comparación de oraciones sintetizadas y en lenguaje natural usando OVE II de John Holmes.
• Comparación de síntesis usando formantes paralelos, 1973.
• John Holmes compara oraciones sintetizadas y en lenguaje natural usando el sintetizador de formantes en paralelo en 1973
• Primeros intentos para que el sonido de voz masculina se escuche como femenina con el DECtalk
• Dennis Klatt en 1986 compara oraciones sintetizadas y en lenguaje natural en voz femenina.
• En 1958, George Rosen del M.I.T, desarrolló DAVO un sintetizador articulatorio
• James Flanagan and Kenzo Ishizaka, através de un modelo articulatorio hacen pruebas para reproducir oraciones, en 1976.
• Con el análisis de predicción lineal y resíntiesis de voz a baja tasa de muestreo de desarrolla en Texas Instruments un juguete llamado Speak'n'Spell en 1980.
• Comparación de síntesis y grabación natural utilizando análisis-síntesis por predicción lineal por multipulsos, 1982.
• Bishnu Atal en 1982, compara una grabación de síntesis y voz natural con un sintetizador automático utilizando predicción lineal multi-pulsos.

Desarrollo de sistemas de síntesis de voz por regla en el periodo 1959-1968
• Generación de una oración por regla usando el sistema "Haskins Pattern Playback", hecho por Pierre Delattre en 1959.
• En 1961, John Kelly y Louis Gerstman, crearon el primer programa de computadora que sintetizaba voz basado en reglas de fonética.
• Un programa más sofisticado diseñaron John Holmes, Ignatius Mattingly, y John Shearme, 1964.
• En 1968, se diseña un sistema de síntesis de formantes usando concatenación de difonemas, Rex Dixon y David Maxey
• Se diseñan reglas para controlar el modelo articulatorio de baja dimensionalidad, Cecil Coker, 1968

Desarrollo de sistemas de síntesis de voz por regla y prosodia en el periodo 1968-1980
• Primer análisis de síntesis por reglas prosódicas, por Ignatius Mattingly, 1968
• En 1976, Dennis Klatt incorporó un sistema de oraciones bajo reglas de fonología
• Se concatenan difonemas por predicción lineal, Joe Olive en 1977
• Se concatenan demisílabas por predicción lineal, Catherine Browman, 1980

Conversión texto-voz 1968-1985
• El primer sistema de conversión de texto-voz se desarrolla en Japón por Noriko Umeda en 1968
• En 1973, se desarrolla en los laboratorios Bell un primer, sistema-voz desarrollado por Cecil Coker, Noriko Umeda, y Catherine Browman
• Se desarrolla el sistema texto-voz en los laboratorios Haskins en 1973
• Raymond Kurzweil en 1976 diseña una máquina lectora para ciegos.
• El sistema Votrax Votrax Type-n-Talk System por Richard Gagnon en 1978
• Se crea el sistema de concatenación de difonemas a bajo costo en 1982
• En 1979 se desarrolla el sistema MITalk por Jonathan Allen, Sheri Hunnicut, y Dennis Klatt
• Creación del sistema multi lenguaje Infovox por Rolf Carlson, Bjorn Granstrom, y Sheri Hunnicut, 1982
• La compañía Speech Plus Inc. crea el sistema comercial "Prose-2000" en 1982
• El sistema Klattalk de Dennis Klatt del M.I.T formó las bases para el sistema comercial DECtalk, en 1983
• Los laboratorios Bell AT&T crean un sistema texto-voz en 1985
• DECtalk crea algunas voces
• DECtalk habla más de 300 palabras por minuto

4.3.2 Sintetizadores de voz en la actualidad

En la actualidad, la mayoría de los sistemas de síntesis están basados en la unión de segmentos pregrabados; esto porque, aunque los otros tipos de sistemas no pueden generar diferentes tipos de voces, y en teoría pueden generar audio de gran calidad y generar muchos tipos de variantes, son de muy alta complejidad por lo que aún no se han podido generar sonidos de alta calidad.

En cambio, los sistemas de concatenación tienen menos variabilidad, como por ejemplo, sólo pueden tener un tipo de voz y para tener un tipo de voz diferente se requiere grabar toda una nueva base de datos y tienen menor rango de flexibilidad, además de requerir un espacio mucho mayor de almacenamiento, pero al ser de una mayor sencillez pueden generar audio de mejor calidad.

Otro de los avances actuales es la mejora en los modelos prosódicos (entonación, ritmo), y la utilización de métodos empíricos (estocásticos); en vez de métodos lingüísticos. Esto permite que reglas que no estén bien definidas puedan inferir estadísticamente del estudio de grabaciones en vez de tratar de generarlas a partir de reglas.

Estos métodos generan el ritmo y la variación de entonación y frecuencia fundamental en el tiempo a partir de análisis sintáctico y gramático e información estadística por varios medios (sistemas lineales, redes neuronales, sumas de productos, árboles sintácticos)

El problema más grande de estos sistemas estadísticos, es que requieren grandes cantidades de información y algunos modelos poco frecuentes pueden ser totalmente ignorados en el proceso o tipos de oraciones que varían de forma muy drástica. En el proceso pueden quedar totalmente fuera de rango.

Para resolver este problema, se están generando también sistemas que sirvan para analizar grandes cantidades de audio pre-grabados de forma automática, a diferencia de tener que analizarlos a mano, como se hace normalmente.

Sin embargo, hay tipos de diálogos, principalmente coloquiales o regionales que no tienen parámetros fácilmente reconocibles o que tienen demasiadas variantes posibles.

Es importante mencionar el avance que se ha tenido entre el contorno de la frecuencia fundamental y el ritmo, ya que son interdependientes.

Dados estos avances, la calidad de la voz sintética ha mejorado considerablemente, pero aún quedan fuertes problemas por resolver, principalmente, en la generación de mejores modelos prosódicos y una mayor variabilidad de parámetros y tipos de voz, posiblemente generando nuevos sistemas de síntesis por formantes, ya que las restricciones de velocidad de los sistemas son mucho menores que antes con la llegada de equipos de mucho mayor velocidad.

4.4 Tipos de sintetizadores de voz

Actualmente, la mayoría de los sistemas de síntesis de voz son creados por métodos electrónicos como computadoras, circuitos integrados, entre otros. Los métodos utilizados para estos sistemas pueden dividirse principalmente en tres grupos:

1. Articulatorios: tratan de modelar directamente el sistema generador de voz.

2. Por formantes: modelan la función de transferencia o de polos de frecuencias del tracto vocal.

3. Por concatenación: utilizan segmentos pregrabados que son unidos (concatenados).

Los dos tipos más usados son por formantes y por concatenación. Aunque el primero fue más usado inicialmente debido a limitaciones en la capacidad de almacenamiento, actualmente el segundo es más usado debido a que son posibles mayores capacidades de almacenamiento.

El sistema articulatorio es poco usado debido a que aún no se tiene un conocimiento suficiente y hay demasiados factores a considerar para ser prácticamente posible; aunque es teóricamente el más flexible y con mayores avances, podría ser una buena opción.

4.4.1 Síntesis articulatoria

Este tipo de síntesis trata de modelar los órganos vocales lo más perfectamente posible, por lo que en teoría es el sistema que podría generar síntesis de más alta calidad, pero a la vez es el más complicado y de más alta carga computacional.

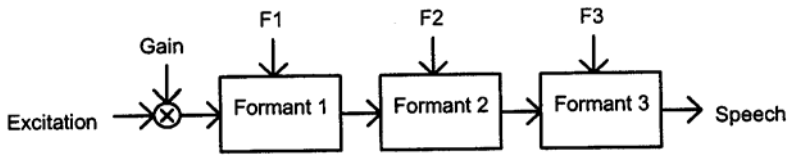
4.4.2 Síntesis por formantes

Este tipo de síntesis involucra normalmente modelos de las cuerdas humanas, de la lengua (posición, altura), apertura del velo, presión de los pulmones, apertura glotal. El modelo de articulación se genera normalmente a partir de radiografías pero éstas no proporcionan información suficiente para conocer todos los parámetros necesarios. La ventaja de éste es que puede utilizar efectos que difícilmente se podrían llevar a cabo en otros sistemas.

Este es uno de los métodos de síntesis más usados. Se basa en un modelo de entrada-filtro-salida del cual existen básicamente dos tipos, filtros en cascada y filtros en paralelo así como las combinaciones de ambos, lo cual produce un mejor resultado. Además, este sistema proporciona mayor flexibilidad que la síntesis por concatenación y una menor dificultad que la síntesis articulatoria.

Un sintetizador de formantes en cascada consta de filtros paso banda conectados en serie y la salida de cada uno se aplica como la entrada del siguiente. Esta estructura necesita sólo frecuencias formantes para controlar la información. Una ventaja es que las amplitudes de los formantes para las vocales no necesitan controles individuales. Ver figura 4.2

a)



b)

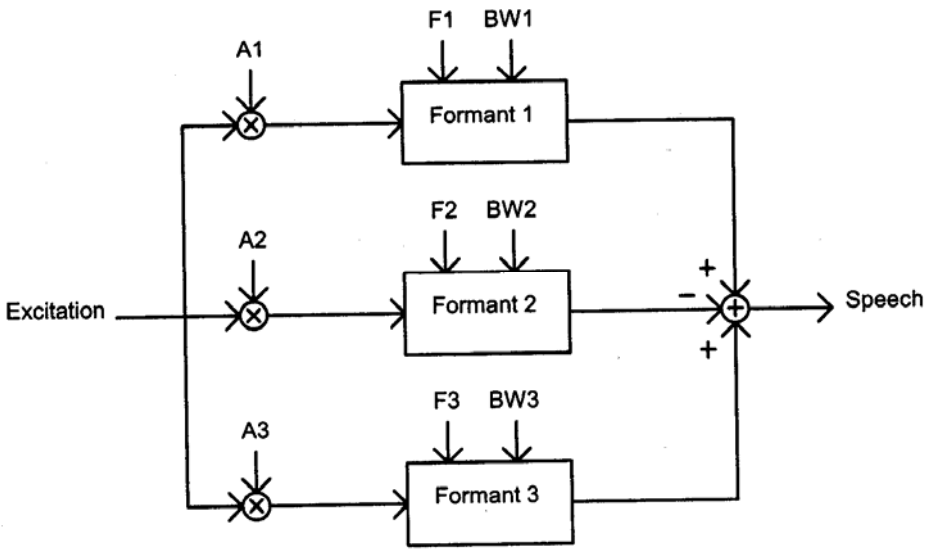


Figura 4.2 a) Estructura básica de sintetizador de cascada
b) Estructura básica de sintetizador en paralelo

Los sintetizadores de formantes en paralelo constan de filtros conectados en paralelo. A veces se usan para los sonidos nasales. La señal de excitación se aplica simultáneamente a todos los formantes y las salidas se suman. La estructura en paralelo es capaz de controlar el ancho de banda y la ganancia de cada uno en forma individual pero requiere de mayor control de información.

Normalmente, se usan al menos tres formantes para producir la señal de voz, aunque a veces se usan hasta cinco para mejorar la calidad. Cada formante se modela por medio de resonador de dos polos donde se indica la frecuencia del formante y su ancho de banda.
La síntesis por formantes utiliza cierto conjunto de reglas que determinan los parámetros necesarios para cada sonido. Algunos de estos parámetros pueden ser: Frecuencia fundamental (F_0), Grado de excitación (V_O), Frecuencias y amplitudes formantes (F_1,F_2,F_3,A_1,A_2,A_3).

Los resonadores de cascada funcionan mejor con los sonidos no nasales pero tienen problemas con las fricativas y plosivas. Los resonadores en paralelo tienen problemas con las vocales pero funcionan bien para nasales, fricativas y plosivas. Debido a esto el modelo mixto ideado por Klatt (1980), que consiste en un sistema mixto con 6 formantes, filtros extras, la adición de un ruido de alta frecuencia y con un sistema de excitación compleja es el más utilizado en los sistemas comerciales actuales como el MITalk, DECtalk y Prose-2000.

4.4.3 Síntesis por concatenación

Este es el tipo de sintetizador que se desarrolla como aplicación práctica del presente trabajo (Capítulo 5). Este sistema se basa en conectar segmentos de voz previamente grabados y almacenados. Es el sistema que presenta menor complejidad aunque tiene la limitación de que sólo se puede usar un tipo de voz.

Este tipo de síntesis se analizará en los siguientes apartados con mayor profundidad.

4.4.4 Síntesis derivados de las técnicas de predicción lineal

Se basan en la posibilidad de modelar el tracto vocal como una serie de cilindros huecos de diámetro variable. Tras pasar la onda sonora a través de ellos, sus propiedades pueden predecirse tomando en cuenta que cada uno de los cilindros condiciona la forma de la onda al entrar en el siguiente. Este cálculo se simplifica si se usan la predicción lineal y la periodicidad de la onda.

Los hay de varios tipos, sin embargo, los que más se usan actualmente son los sintetizadores multipulso (MLPC), caracterizados por su sencillez y buena calidad

4.5 Métodos, técnicas y algoritmos por concatenación

Uno de los aspectos más importantes de la síntesis por concatenación es el de seleccionar un tipo y tamaño de segmento de acuerdo al tipo de sistema que se quiere desarrollar. En unidades largas, se requieren menores puntos de concatenación, por lo que se obtiene un mejor control de coarticulación pero se requiere un número muy grande de unidades. En unidades pequeñas, se tienen más puntos de concatenación pero se reduce el número de unidades requeridas.

Las unidades más grandes son frases y palabras. Éstas funcionan bien para sistemas que tienen un vocabulario limitado, por ejemplo un sistema que dé la hora del día. Pero para sistemas de entrada libre, hay demasiadas unidades para ser prácticos, además de que al ser libre la entrada, el usuario tiene la libertad de, por ejemplo, insertar palabras inexistentes o frases mal construidas.

Otra unidad que se podría considerar son las sílabas. Su número es considerablemente menor al de palabras o frases pero aún tiende a ser muy grande (10,000+). Además, en un sistema basado en sílabas, no se pueden almacenar los efectos de coarticulación entre sílabas ni la prosodia.

Debido a estos problemas, no existe al momento ningún sistema de conversión texto-habla que utilice estas unidades.

La siguiente unidad que se puede considerar son los fonemas. Su número es bastante reducido (normalmente entre 25 y 50 según el idioma). Los fonemas presentan el problema de la falta de información de coarticulación por lo que son poco usados, aunque en muchos sistemas se utilizan los fonemas como unidades lógicas que son transformadas a la unidad correspondiente después de su análisis en el proceso de concatenación.

Otra unidad son las demisílabas, que representan la parte inicial y final de las sílabas. Su número es grande pero aceptable (aprox. 1000). Éstas cubren un buen número de problemas como la coarticulación, algunos alófonos y requieren menos puntos de concatenación que los fonemas. Su número es grande, pero aún es aceptable. Desafortunadamente su número no puede ser determinado fácilmente y hay algunas combinaciones que no pueden ser generadas con demisílabas, por lo que normalmente se usan sólo en sistemas mixtos.

La siguiente unidad a considerar son los difonemas. Un difonema se define como el segmento que inicia del punto central del estado estable de un fonema, al punto central del estado estable del siguiente fonema. Lo cual ayuda a disminuir la distorsión al estar el punto de concatenación en una zona de relativa estabilidad. Además, permite evitar los problemas de coarticulación al estar ésta presente explícitamente en los segmentos. El número existente de difonemas es el cuadrado de los fonemas existentes. De éstos existen algunos que pueden ser eliminados al no presentarse en una lengua. Su número está cerca de los 1000. El número es grande, pero aún aceptable, y debido a las ventajas que presenta es un tipo de unidad muy utilizado. Estas son las unidades que serán utilizadas en el sistema a desarrollar.

Un sistema difonémico que requiere mención es el sistema MBROLA, al ser uno de los pocos sistemas de síntesis de voz de distribución libre y sin costo,

además de ser de los pocos sistemas de síntesis que ofrecen síntesis de la lengua española (con voces española y mexicana).

Existen otros tipos de unidades como los trifonemas (contienen un fonema entero entre dos medios fonemas y son poco usados), aunque actualmente se encuentran en investigación métodos para optimizar los sistemas de síntesis usando unidades de diferentes longitudes, donde estas unidades podrían formar una parte importante.

Otro tipo de unidad que está en investigación actualmente es el uso de microfonemas que son pequeños segmentos que al concatenarse forman fonemas completos. Este sistema promete buenos resultados. Aunque es un sistema bastante flexible, es muy complejo generar los segmentos necesarios.

Otro problema básico en estos sistemas es la distorsión que se presenta en los puntos de unión; lo cual se ha minimizado un poco con el uso de difonemas donde la unión se lleva a cabo en un punto más favorable. Otro proceso para eliminar esta distorsión es el uso de filtros para suavizar la unión. Este es otro punto de investigación actual para obtener un buen suavizado sin alterar demasiado la señal original.

Se pueden distinguir entre los sintetizadores basados en el método PSOLA (Pitch Synchronous Overlap and Add, en inglés), los de codificación armónica, codificadores multibanda y sintetizadores por selección de unidad.

4.5.1 Métodos de síntesis por concatenación

MODELO DE SELECCIÓN DE UNIDAD

En este sistema, cada caso de una unidad en la base de datos (típicamente un segmento fonético-clasificado), se etiqueta con un vector de características. Las características pueden ser discretas o continuas. Las características típicas de etiqueta del fonema son: duración, energía, y F_0. También, las características acústicas tales como inclinación espectral que se incluyen en las bases de datos. Otras características describen el contexto de la unidad:

etiquetas del fonema de unidades vecinas, la posición en la frase, o de la dirección del cambio de “pitch/power”

Los vectores pueden incluir las características sobre el contexto de una unidad así como la misma unidad. Cuando es posible, las características se describen en una forma normalizada por ej. distancia en unidades de la desviación estándar alrededor de un cero medio). Otro requisito es que tenga una medida de la distancia entre dos valores de la característica del mismo tipo. Para las características continuas esto es más fácil, pero para las características discretas (por ej. fonemas), una distancia necesita ser definida explícitamente. Las distancias entre los valores de las características se normalizan para medir en la gama 0 (buena) a 1 (malo).

Para la selección, los segmentos objetivo, predichos por componentes anteriores del sintetizador, o para los propósitos de prueba tomados del discurso natural, también se etiquetan con un subconjunto de estas características-específicas, excepto cualquier característica solamente disponible de medidas acústicas. Esto especifica explícitamente las características segmentarias y prosódicas previstas de la elocución.

Para medir cómo está un sistema de unidades seleccionadas comparando con un sistema de segmentos objetivo, dos tipos de **distorsión** pueden ser definidos.
Se define la **unidad de distorsión** $D_u(u_i, t_i)$ como la distancia entre una unidad seleccionada y un segmento objetivo, es decir, la diferencia entre el vector seleccionado de la característica de la unidad $\{ u_{f1}, u_{f2}..., u_{fn} \}$ y el vector del segmento objetivo $\{ t_{f1}, t_{f2}..., t_{fn} \}$ multiplicado por un vector de pesos: $W_u \{ w_1, w_2,..., w_n \}$.

La **distorsión de la continuidad** $D_c(u_i, u_{i-1})$ es la distancia entre una unidad seleccionada y su previa adjunción inmediata de la unidad seleccionada, definida como la diferencia de distancia entre el vector de la característica de una unidad seleccionada y su anterior, multiplicada por un vector de pesos W_c.

Esta distancia representa el costo de ensamblar dos unidades. Este vector incluye las distorsiones del contexto de una unidad seleccionada con el contexto de otra seleccionada anterior que debe ser concatenada.

Variando los valores de W_u, y W_c, permite la importancia relativa de que las características cambien, por ejemplo permitir que F_o desempeñe un mayor

papel en la selección de unidad, que la duración. Los valores pueden también ser cero, así se elimina una característica de los criterios de selección. Los vectores de peso: W_c y W_u serán diferentes.

La **mejor secuencia de la unidad** se define como la trayectoria de unidades de la base de datos que reduce al mínimo

$$\sum_{i=1}^{n} = (D_c(u_i, u_{i+1}) * W_C + D_u(u_i, t_i) * W_U)$$

Donde n es el número de segmentos en la elocución objetivo; y W_C, W_U son pesos. Maximizando W_C con respecto a W_U, se reduce al mínimo la distorsión entre las unidades seleccionadas a expensas de distancia de los segmentos objetivo.

MBROLA:

Es una herramienta de conversión de texto en habla basada en la concatenación de difonemas desarrollada en la Faculté Polytechnique de Mons (Bélgica) que funciona en más de 20 lenguajes, entre otras, portugués de Brasil, bretón, inglés británico, holandés, francés, alemán, español y sueco.

Este programa permite dar información sobre los valores de duración y de frecuencia fundamental de cada alófono considerado, y acepta hasta un máximo de 20 valores de F_0 para cada alófono. Sin embargo, aunque estos valores de F_0 se pueden utilizar para dibujar una curva melódica, esto implica la existencia de un modelo prosódico. MBROLA ha sido usado, por ejemplo, con fines de evaluación de resultados obtenidos en trabajos sobre generación automática de prosodia para la conversión de texto en habla.

PSOLA: PITCH SYNCHRONOUS OVERLAP AND ADD

Es un sistema de Codificación en el Dominio Temporal. Este método fue desarrollado en France Telecom CNET. Actualmente, no es un método de síntesis solamente sino que pregrabando muestras de voz concatenadas se controla el tono y duración de la señal. Se ha utilizado comercialmente por Pro Verbe y HADIFIX.

La síntesis TD-PSOLA (Time Domain Pitch-Synchronous Overlap and Add), en inglés, forma parte de las técnicas de síntesis por concatenación de forma de onda. Es necesario, además del proceso de adquisición y grabado de unidades, un procesado previo de las mismas para obtener la información prosódica necesaria para el algoritmo.

La concatenación directa de las unidades no es posible, por lo que el principal problema tiene que ver con las modificaciones necesarias para adaptar la prosodia de las unidades pregrabadas a la prosodia del texto en donde se quiere utilizar, sin que se produzcan pérdidas graves apreciables de la calidad. El algoritmo TD-PSOLA trata de hacer dicha adaptación a través de modificaciones en la transformada de Fourier, afectando a la frecuencia fundamental y a la duración de las unidades.

En el Capítulo 5 se tratará este método con mayor profundidad ya que es la técnica en la que está basado el diseño del algorimto de aplicación.

4.6 Aplicaciones de los sintetizadores de voz

- **SISTEMA DE LECTURA PARA CIEGOS**

Hoy día, gran cantidad de publicaciones se desarrollan en sistemas de habla con soporte informático, de tal modo que el texto de libros y periódicos se encuentra almacenado en computadoras antes de ser impreso en papel. Un Converso Texto-Voz podría "leer" directamente estos textos, o podría integrarse con un reconocedor óptico de caracteres para leer textos ya impresos en papel.

Hay que resaltar que en un sistema de este tipo los requisitos que se piden al conversor cambian, pues suele ser más apreciado por los potenciales usuarios que el conversor hable muy rápido y pueda controlarse para avanzar y retroceder en el texto al modo de una cinta magnetofónica, a que tenga una

alta calidad de voz, pues el usuario puede acostumbrarse con rapidez a una voz degradada que resultaría poco inteligible para otra persona.

SISTEMA DE HABLA PARA PERSONAS MUDAS

Un Conversor Texto-Voz puede servir de ayuda a personas que no pueden hablar, mejorando en muchos casos la integración de estas personas con el resto (especialmente en el caso de los niños), y permitiéndoles el uso de un instrumento de comunicación tan básico hoy día como es el teléfono.

AYUDA A LA ENSEÑANZA

Algunos niños presentan problemas de aprendizaje en el lenguaje hablado y escrito (dislexia, problemas de fonación), a cuya solución puede colaborar un Conversor Texto-Voz, empleando como herramienta de ayuda a la enseñanza que facilita cierta autonomía en el aprendizaje. También puede ser útil en la enseñanza de otros idiomas, si se dispone de un conversor en el idioma que se pretende aprender, para comprobar dudas de pronunciación y corregir errores.

VERIFICACIÓN DE TEXTO

Muchas veces, al repasar un texto una y otra vez, no se detectan los fallos que pueda haber en la escritura. Incluso los correctores ortográficos no detectan el uso inadecuado de: una palabra ortográficamente válida (p. ej., "pera" por "pero", o "depósito" por "depositó"). Al oír el texto "leído" por el conversor, esos errores se hacen evidentes y pueden ser corregidos.

ALARMAS HABLADAS

Para favorecer una rápida reacción de una persona a cargo de varios controles, es ventajoso no sólo recibir una alarma acústica o visual, sino un mensaje que describa mejor la causa del aviso. Por ejemplo, pensemos en una enfermera de una Unidad de Vigilancia Intensiva, que en lugar de recibir el sonido de un timbre o zumbador, recibiese un mensaje hablado que dijese: "el enfermo de la cama 3 presenta una fuerte subida de la tensión arterial".

INSTRUCCIONES DE MONTAJE HABLADAS

Así se permite el uso de las manos y la vista sin necesidad de apartar la atención para consultar un folleto o una pantalla de ordenador.

CAPÍTULO 5
SINTETIZADOR DE VOZ DE PALABRAS EN ESPAÑOL POR CONCATENACIÓN

5.1 Análisis de un sintetizador de voz

Como se analizó en el capítulo anterior, la síntesis de voz del proceso de conversión texto-voz dota a las máquinas de la capacidad de producir mensajes orales no grabados previamente, como es el caso de los sistemas de respuesta oral. Tomando como entrada un texto, los sistemas de conversión texto-voz realizan el proceso de lectura de forma clara e inteligible y con una voz lo más natural posible. La síntesis de voz conforma la interfaz oral de comunicación entre una máquina y el usuario de la misma.

Así también, el desarrollo de los primeros capítulos fue necesario como marco de referencia para poder llevar a cabo el simulador. Es indispensable entonces comprender los principios básicos sobre los que se asientan los sistemas de síntesis de voz, estudiando primero el proceso de generación de un mensaje oral desde el punto de vista acústico y lingüístico.

Es necesario entender el comportamiento físico del aparato fonador del ser humano y cómo son procesados por el sistema auditivo humano para desarrollar un modelo matemático del mismo. A la vez, hay que saber cómo extraer del texto, en base a su estructura lingüística, la información necesaria para controlar el modelo matemático y de este modo convertir el texto en voz.

Este capítulo se basa en diferentes estudios que han realizado H. Valbret, E. Monlines, *Voice transformation using PSOLA technique*; Campbell y Black *Prosody and the selection of sources units for concatenative synthesis.; Allen J, Overview of Text-to-Speech Systems";* por mencionar algunos (ver bibliografía).

Es importante mencionar que también está sustentado en estudios y desarrollos llevados a cabo en el Posgrado de Ingeniería, laboratorio de señales de voz por el Dr. Abel Herrera y estudiantes.

En el aspecto lingüístico, el primer problema que se encuentra en un sistema de conversión texto-voz es que debe inferir el contenido real de la representación escrita del mensaje. Para ello, se debería realizar un procesado lingüístico del texto a partir de un análisis fonético-morfológico para derivar la pronunciación, un análisis sintáctico para dar la estructura gramatical del texto y poder inferir rasgos prosódicos, un análisis semántico para dar una representación del significado del mensaje y un análisis pragmático para dar una relación entre frases e ideas de la conversación global.

Este procesado lingüístico es muy ambicioso y los sistemas actuales simplemente realizan un análisis fonético-morfológico y sintáctico para de este modo determinar los rasgos segmentales y prosódicos de los sonidos que componen el mensaje oral. Un aspecto importante en la inteligibilidad y naturalidad de la señal sintetizada son las reglas prosódicas, que aunque en cierta medida pueden ser inferidas de la estructura sintáctica de la frase, la mejor forma de generar una entonación adecuada a una frase, es que la máquina entienda lo que está diciendo.

5.1.1 Sintetizador de voz por concatenación

La forma más sencilla de generar voz consiste simplemente en grabar la voz de una persona pronunciando las frases deseadas. Este sistema sólo es viable cuando el número de frases que es necesario sintetizar es pequeño. Por ejemplo, un número concreto de mensajes que se emiten en una estación de tren. En casos como éste, la calidad del sistema depende de la calidad de grabación en las frases.

Sin embargo, en el caso de un sistema conversor texto-voz, se necesita un sistema que permita sintetizar cualquier texto que se introduzca por teclado. La solución consiste en dividir la voz en segmentos, los cuales van a constituir una base de sonidos con la que trabajará el módulo de síntesis.

El diseño del primer simulador conversor texto-voz emplea para producir voz el método de "Síntesis de voz por Concatenación", que, como ya se comentó, consiste en concatenar uno tras otro todos los sonidos que constituyen el texto. Estos sonidos han sido previamente almacenados y constituyen la base de sonidos.

Para crear una buena base de sonidos, se debe decidir el tipo de unidades acústicas o segmentos fónicos adecuados para formar parte de dicha base. Existe un compromiso entre la calidad de voz conseguida y el tamaño de la base de datos que se necesita para almacenar los segmentos. Cuanto más pequeñas sean las unidades en que se descompone la voz, menor es la base de sonidos utilizada, pero la calidad de la voz también decrece.

Una solución a este problema es el empleo de difonemas, los cuales están compuestos por la porción final de un fonema y la inicial del fonema que le sigue. Como el corte está hecho en el centro del fonema, las transiciones entre ellos permanecen intactas. Para el caso del español, el número de difonemas necesario para constituir una base es de, al menos, 550. Para aumentar la calidad de la síntesis, se puede utilizar un número limitado de trifonemas para representar a sonidos en los que los tres fonemas se coarticulan a elevada velocidad (pla, ple, pli, plo, plu, tra, tre,...)

Una forma de obtener la base de difonemas y trifonemas consiste en la grabación de logo-átomos. Los logo-átomos son palabras carentes de significado, compuestas por tres sílabas, que permiten que el segmento a tratar esté aislado sin coarticular con los sonidos anterior y posterior. De las tres sílabas que componen el logo-átomo nos interesa la sílaba central, que es donde se encuentra el segmento a extraer. Su estructura general es la siguiente:

Como ya se mencionó en el capítulo anterior, los métodos de síntesis de voz se pueden dividir en dos grupos: la síntesis por reglas y la basada en la concatenación de unidades previamente almacenadas. En el primer método, se intenta conseguir una reproducción de la voz humana a partir de un estudio de determinadas características de ésta y mediante una serie de reglas que se aplican a ciertas fuentes de señal sonora a lo largo del tiempo.

En cuanto a las técnicas de síntesis por concatenación de unidades fonéticas, surgen históricamente como un intento de reducir la complejidad, sobre todo en los cálculos necesarios, que presentan los sistemas de síntesis por reglas. Se basan en el almacenamiento de segmentos de voz que posteriormente son concatenados para producir frases de cualquier longitud. Los factores a tener en cuenta con este tipo de síntesis son:

- ✓ Elección de unidades.

Para seleccionar el tipo de unidad hay que buscar minimizar la cantidad de memoria para almacenar y/o reducir al máximo el problema de coarticulación. Las posibilidades existentes abarcan desde el empleo de los fonemas, en cuyo caso es necesario disponer de un completo conjunto de reglas, al de las frases completas, cuya utilidad se reduce a aquellos casos en que el conjunto de mensajes a emitir es limitado. Por ejemplo, las unidades muy utilizadas son los difonemas y las semisílabas.

- ✓ Selección de técnica de codificación.

Otro aspecto a considerar es la elección de la técnica de codificación que nos permita la reconstrucción de la onda sonora a partir de las unidades almacenadas. Suelen emplearse, entre otras, técnicas en el dominio de la frecuencia, técnicas en el dominio del tiempo y codificación predictiva lineal (LPC)

- ✓ Empleo de un método que permita modificar los parámetros prosódicos de los segmentos sin que se degrade la calidad de la voz.

Y por último, es necesario disponer de un mecanismo que permita modificar la frecuencia fundamental para poder dar al mensaje la curva prosódica adecuada. Este proceso resulta sencillo si la técnica de codificación empleada es LPC o basada en el dominio de la frecuencia. La modificación de la frecuencia fundamental de una señal en el dominio del tiempo, sin embargo, presenta dificultades si se pretende conservar su envolvente espectral.
Una alternativa, es la de tomar muestras de voz a diferentes frecuencias y emitir una u otra según el tono deseado de esa muestra dentro de la frase. Este método presenta el inconveniente de que, al ser mayor el número de unidades a almacenar, se requiere una cantidad de memoria más elevada. Sin embargo, esto se resuelve tomando de distintas frecuencias únicamente las muestras correspondientes a las vocales y controlando con ellas el tono de toda la frase. Aunque el resultado obtenido en cuanto a la relación entre calidad y

memoria requerida es más que aceptable, el sistema construido presenta ciertas deficiencias en cuanto a su naturalidad, que convendría mejorar.

5.1.2 Técnica PSOLA "Pitch Synchronous Overlap and Add"

DESCRIPCION DEL ALGORITMO

TD-PSOLA es un algoritmo que actúa en el dominio del tiempo y que combina la rapidez propia de este tipo de técnicas con la calidad de otras más sofisticadas.

El esquema general de funcionamiento del PSOLA se puede resumir en la ejecución de tres etapas:

- Un análisis de la onda original para conseguir una representación no paramétrica de la misma. La señal original se descompone en una serie de unidades de corta duración superpuestas denominadas "bloques" o "unidades" y en inglés se conoce como Short-term signal, ST.

En lo sucesivo se hará referencia a estas como unidades siendo una porción corta de la señal.

- La modificación prosódica a partir de esta representación. Esta señales son modificadas con lo que se convierte en lo que denominamos bloques de síntesis.

- La producción de la señal sintética construida a partir de la representación intermedia modificada.

Mediante la superposición y suma de esta últimas se genera la onda sintetizada, siendo este proceso el que da lugar a la denominación "Pitch Synchronous Overlap and Add" PSOLA, una traducción al español cercana a este concepto es: solapamiento y suma sincronizada con la frecuencia fundamental. La forma en que se opere sobre las unidades de análisis para obtener las de síntesis ha dado lugar a diferentes variantes del método general.

PSOLA/FFT → las unidades de síntesis se obtienen a partir de las modificaciones en el dominio de la frecuencia de las de análisis. La complejidad de los cálculos que requiere este método sigue siendo demasiado elevada para poder implementarse directamente en sistemas sencillos.

PSOLA/MLPC → aunque actúa en el dominio de la excitación mediante impulsos y es por tanto computacionalmente más simple que el anterior, sigue requiriendo la descomposición de la señal sonora.

TD-PSOLA → Time Domain o dominio del tiempo, esta es la que requiere menor esfuerzo computacional; la idea básica de funcionamiento consiste en variar el grado de solapamiento de las unidades de síntesis con lo que se consigue modificar la frecuencia fundamental de la onda resultante sin cambiar su envolvente espectral. Paralelamente a este ajuste de solapamiento entre las señales, resulta sumamente sencillo controlar la longitud o duración de la onda producida mediante la duplicación o eliminación de alguna de las unidades.

ANALISIS Y SINTESIS PSOLA

La señal de voz digitalizada ***s(n)*** se descompone en una serie de unidades superpuestas ***$s_m(n)$*** denominada unidades de análisis. Estas se obtienen multiplicando la señal por una secuencia de "ventanas" ***$h_m(n)$*** según la expresión

$$s_m(n) = h_m(t_m - n)s(n)$$

Ecuación 5.1

En esta ecuación (5.1), ***$h_m(n)$*** representa una ventana simétrica y centrada en ***n=0***. Los sucesivos instantes ***t_m*** se seleccionan síncronamente con la frecuencia fundamental de la señal.

La secuencia de unidades que se obtiene es procesada para producir otro conjunto de unidades, ***$s'_q(n)$***, que denominamos de síntesis y que se sincronizan con un nuevo conjunto de marcas temporales ***t_q***. La relación entre las unidades de análisis y las de síntesis viene de este modo determinada por una función de alineamiento temporal: ***$t_q \rightarrow t_m$*** que implícitamente expresa la frecuencia de la onda sintetizada respecto a la de la original.

Una vez realizados estos cálculos, la señal de voz sintética s´(n) puede obtenerse mediante un proceso de solapamiento y suma de unidades de síntesis. Esto se puede hacer según la expresión *5.2*

$$s'(n) = \frac{\sum_q \alpha_q s'_q(n) h'_q(t'_q - n)}{\sum_q h'^2_q(t'_q - n)}$$

Ecuación 5.2

En la que ***h*** representa las ventanas de síntesis, y α_q un factor de compensación debido a las variaciones de energía que se producen.

Conviene utilizar una ventana de síntesis constante sin una pérdida significativa de prestaciones, reduciéndose la expresión anterior a:

$$s'(n) = \sum_q \alpha_q s'_q(n)$$

Ecuación 5.3

El factor α_q se mantiene, en este caso, para compensar las modificaciones de energía que se pueden producir debido a la suma de valores de las ventanas en las zonas de solapamiento de las mismas.

ALGORITMO DE MODIFICACIÓN PROSÓDICA

El control de la frecuencia y duración en la señal sintética se lleva a cabo mediante la selección de las marcas t_q y la definición de la función de alineamiento temporal $t_q \rightarrow t_m$, que relaciona las marcas de síntesis con las de análisis. Esta función asocia cada unidad de síntesis ***s(n)***, con la de análisis que debe ser copiada en su lugar, y los valores de ***tq*** determinan los retardos que deben ser introducidos entre unidades sucesivas.

Esto puede representarse mediante la siguiente expresión, que define las unidades de síntesis a partir de las de análisis:

$$s'_q(n) = s_m(n - t_m + t'_q)$$

Ecuación 5.4

Si la duración y la frecuencia de la señal deben ser modificadas por un mismo factor β, la relación entre las unidades de análisis y las de síntesis será de uno a uno. En este caso, el algoritmo debe copiar las unidades de análisis en el eje de tiempo de las de síntesis, ajustando el retardo entre ellas según el factor β. En el caso general en el que la duración y la frecuencia requieran factores de ajuste diferentes, la relación ya no será de uno a uno, y el algoritmo deberá ajustar el retardo y eliminar o duplicar algunas de las unidades de análisis.

En cuanto a la ventana a utilizar, es la denominada ventana de Hamming, definida en el Capítulo 1 (1.3 Análisis a nivel de segmento), y que se nombrará ecuación 5.5

$$h(t) = 0.54 - 0.46\cos\left(\frac{2t}{L}\right)$$

Ecuación 5.5

Siendo ***L*** la longitud de la ventana. La función se define para valores de ***t*** en el intervalo ***0 ≤ t ≤ L y cero en otro caso.*** En cuanto a la longitud que debe tener, es conveniente sea proporcional al periodo de la señal en ese punto, según la expresión siguiente:

$$h'_q(n) = h\left(\frac{n}{\mu P}\right)$$

Ecuación 5.6

h(t) es la ventana definida en el intervalo unidad, P es el periodo de la onda en el punto y μ un factor de proporcionalidad que indica el número de periodos que abarca la ventana; se recomienda un tamaño de ventana de dos periodos completos de onda μ=2, es decir, definirla de una longitud doble al periodo de la señal en ese punto. Otro factor importante a tener en cuenta a la hora de

definir las ventanas es el hecho de que deben estar rigurosamente sincronizadas con los instantes de mayor amplitud de la onda dentro de cada periodo, ya que en caso contrario la calidad de la voz se ve sensiblemente afectada.

GENERACIÓN DE LA PROSODIA

La parte inicial del tratamiento que recibe el texto introducido en el sistema consiste en descomponer cada grupo fónico en una serie de unidades fonéticas pertenecientes al conjunto definido. Dentro de este proceso, se determina la acentuación y la duración de cada vocal, de modo que, la serie de unidades fonéticas se almacenan en una estructura que contiene, para cada unidad, su nombre, su tono y su duración.

El tono se desdobla en dos componentes: inicial y final, que serán utilizadas posteriormente y que de momento reciben el mismo valor. Además, en este nivel todas las unidades adoptan únicamente dos posibles tonos, según se trate de una vocal acentuada o no.

Para formar la entonación de la frase se asigna un tono a cada unidad fonética, en función de su posición relativa dentro del grupo fónico y del tipo de curva prosódica que tenga este asociado.

El procedimiento consiste en asignar valores en primer lugar a las cuatro unidades fonéticas que definen la curva melódica, a saber, la primera y última unidad acentuada del grupo fónico, y las que se encuentran en la primera y última posiciones absolutas del mismo. Al resto de las unidades se les asigna un tono calculado mediante una interpolación lineal entre estas cuatro referencias.

Lo anterior, se hace de forma que el tono final de una unidad coincide siempre con el inicial de la siguiente, con la finalidad de que no haya variaciones bruscas de frecuencia a lo largo de la frase. Y con esto, se llega a una serie de unidades fonéticas pertenecientes a tres valores numéricos asociados a cada una de ellas: tono inicial, tono final y duración.

MÉTODO DE PROCESO Y ORGANIZACIÓN DE LAS ESTRUCTURAS DE DATOS UTILIZADAS

Cada muestra de voz se almacena en un fichero, codificada mediante técnica PCM a una frecuencia de muestreo de 11 Khz. Se define, para cada unidad, un conjunto de marcas sincronizadas con la frecuencia fundamental, y coincidentes con máximos relativos de la misma.

Pese a que existen multitud de métodos automáticos de detección de la frecuencia fundamental, se omite su utilización puesto que, al ser el tono de las muestras conocido y estable, se puede implementar un programa relativamente simple que permita realizarlo manualmente de forma cómoda, obteniendo una mayor precisión.

Una vez definido el conjunto completo de marcas para la totalidad de los ficheros, se calcula la ventana de Hamming centrada en cada marca y se multiplica por los valores del fragmento de señal que comprende. Pese a que las ventanas se solapan entre ellas, son almacenadas de forma contigua en una estructura tipo matriz en la que cada fila contiene una unidad, es decir, el producto de una ventana por un fragmento de señal, habiendo tantas filas como unidades definidas para esa muestra.

Las modificaciones del tono y la duración en la señal de voz sintetizada se producen variando el grado de solapamiento de las unidades de la muestra. Para llevar a cabo un control simultáneo de estos dos factores, se puede adoptar una simplificación siempre que tanto el tono inicial como el final de la muestra a emitir sean superiores al tono definido como medio o normal, se duplicará la unidad con la finalidad de compensar la variación en la longitud; si, por el contrario, ambos tonos son inferiores al normal, se eliminará una ventana.

Si cada uno de los dos tonos queda a un lado del tono medio, se asume que se compensan el efecto alargador que se produce por un extremo de la muestra y el recortador que se produce por el opuesto, por lo que no se efectuará ningún control de tiempo.

De un modo independiente a este proceso, la duración de cada unidad viene representada por un valor numérico que indica la cantidad de ventanas a

eliminar o duplicar, en función de su diferencia respecto a un valor constante considerado como duración normal. La selección de las filas de la matriz a eliminar o replicar se hace de forma lineal, de modo que se hallen equidistantes entre sí y respecto a las filas de los extremos.

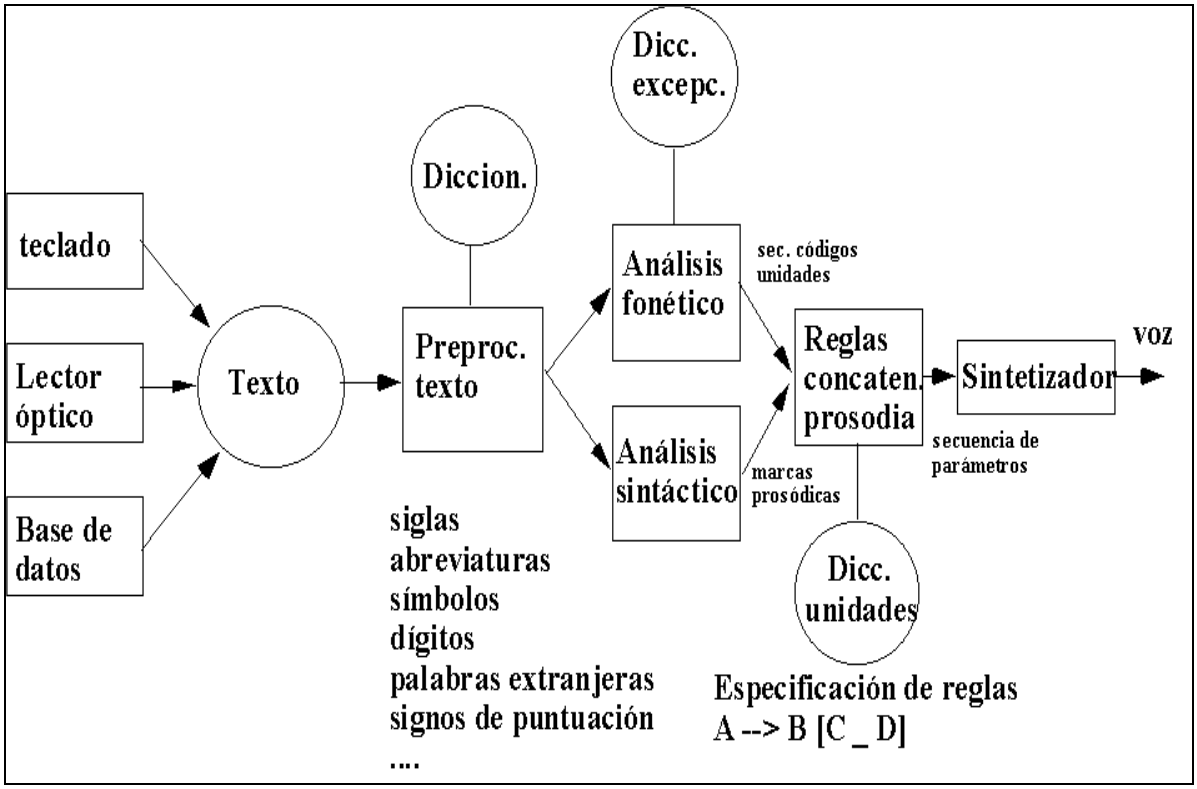
La modificación de la frecuencia se lleva a cabo en forma paralela a la emisión de los valores de la muestra. Para ello, se calculan, a partir de los tonos inicial y final asignados a la unidad, un punto de solapamiento asociado a cada fila de la matriz. La emisión propiamente dicha consiste en recorrer cada ventana, teniendo en cuenta que, si se alcanza el punto de solapamiento, debe sumarse el valor que corresponda de la ventana siguiente.

En la ecaución 5.3, si se considera el factor α una constante de valor 1 entonces las zonas de mayor frecuencia de la señal resultan asimismo ligeramente aumentadas en su amplitud.

Esto da lugar a un ligero acento de intensidad, que se superpone al acento melódico, mejorando la naturalidad de la voz producida. Este fenómeno se aprecia en las figuras siguientes, que representan respectivamente, un fragmento de la primera y segunda vocal de la palabra CAZAR.

5.2 Diseño del sintetizador de voz por concatenación

La gráfica que aparece a continuación resume los pasos que se deben llevar a cabo para realizar un sintetizador de voz en forma general.



En base a lo descrito en los apartados anteriores, el simulador está dividido en tres partes:

Separación de palabras

- Reemplazar las palabras escritas por la representación fonética correspondiente.
- Encontrar la letra acentuada. Se asume que la palabra ha sido bien escrita porque el módulo obtiene la pronunciación de acuerdo a la forma en que está escrita la palabra.
- Analizar la palabra y agregar un símbolo de acento si no lleva acento escrito.
- La palabra es enviada al módulo de generación de voz que la transforma a una salida de audio.

Generación de voz

Para la salida de audio, se requiere tomar en cuenta las reglas ortográficas, así como la clasificación de los fonemas analizados en el Capítulo 3.

- Generar la base de datos de segmentos haciendo una grabación. Éstos son difonemas grabados que corresponden a la sección desde la mitad de un fonema hasta la mitad del siguiente. Como cada difonema consta de 2 medios fonemas, el número total debe ser el número de combinaciones existentes de todos los fonemas en grupos de 2. Al haber 24 fonemas en el español mexicano, se requieren 24^2 (576) difonemas diferentes.

- Algunas combinaciones no existen comúnmente en nuestro idioma pero hay que dejarlas por si se presentan en palabras inventadas que pueden ser introducidas también en el sistema. Para reducir el procesamiento, cada fonema vocálico se consideró para fines prácticos como dos: una versión llana y una acentuada. Cabe aclarar que se debería utilizar el número total de alófonos y no nada más el número de fonemas, pero esto es poco utilizado debido a la cantidad de espacio de almacenamiento requerido.

- Cortar y clasificar los segmentos. Para los sonidos donde la zona de estabilidad es un ciclo (vocales, semivocales y nasales), se tomó un punto donde existiera un cruce con el origen (para disminuir las discontinuidades), como inicio del ciclo. Para las consonantes fricativas que consisten principalmente en ruido, se tomó un cruce con cero cercano al centro. Para las consonantes plosivas y africativas, se utilizó la zona de silencio que existe en el momento en que se está generando la presión necesaria para la explosión de sonido. Cada uno de los difonemas, una vez cortados, se almacenan en un archivo con el nombre de los dos medios fonemas que contiene.

- Se analizan los fonemas que se obtuvieron de revisar el texto de entrada para separar en grupos de dos y unir los difonemas requeridos.

Reproducción del archivo de salida

- Almacenar los segmentos en un archivo nuevo al cual se le agrega un encabezado o etiqueta para que pueda ser reproducido con cualquier reproductor de audio, por ejemplo archivos WAV.

En el laboratorio de señales de voz del posgrado de Ingeniería, fue implementado un sintetizador de voz desarrollado por el estudiante Fernando del Rio Avila, ver referencias bibliográficas [20] y [21]; este fue programado en lenguaje C++ y tiene por objeto sintetizar la voz por concatenación de difonemas.

Sobre este se realizaron diferentes pruebas de análisis y cambios mismas que se muestran en los apartados siguientes, así como un rediseño para mejorar la síntesis de voz.

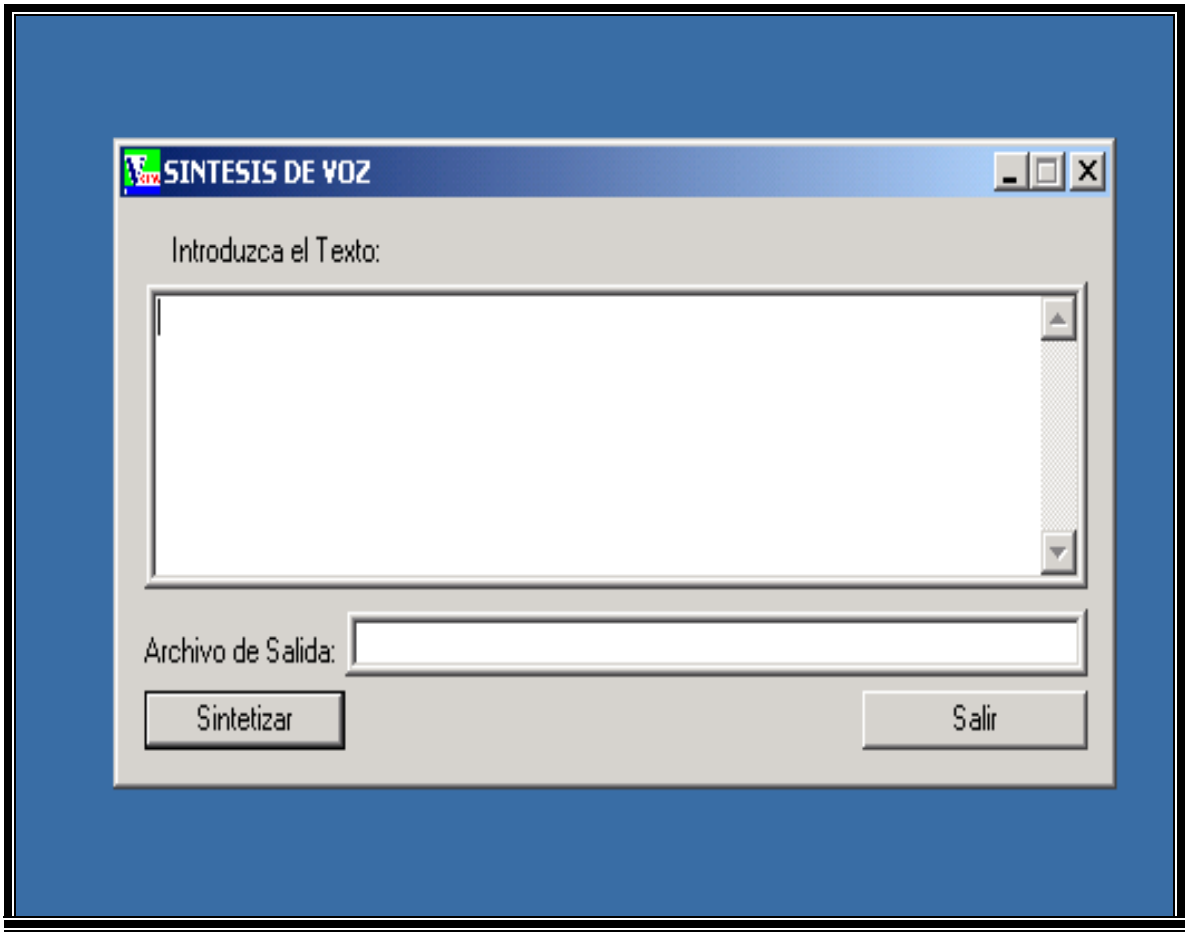
A continuación, se explican los cambios realizados:

1.- Se agregó la opción de realizar la síntesis de voz con opciones de hombre y mujer; el original sólo lo realizaba con voz de hombre.

2.- Se le agregó abreviaturas a la base de datos para que identificara la palabra completa.

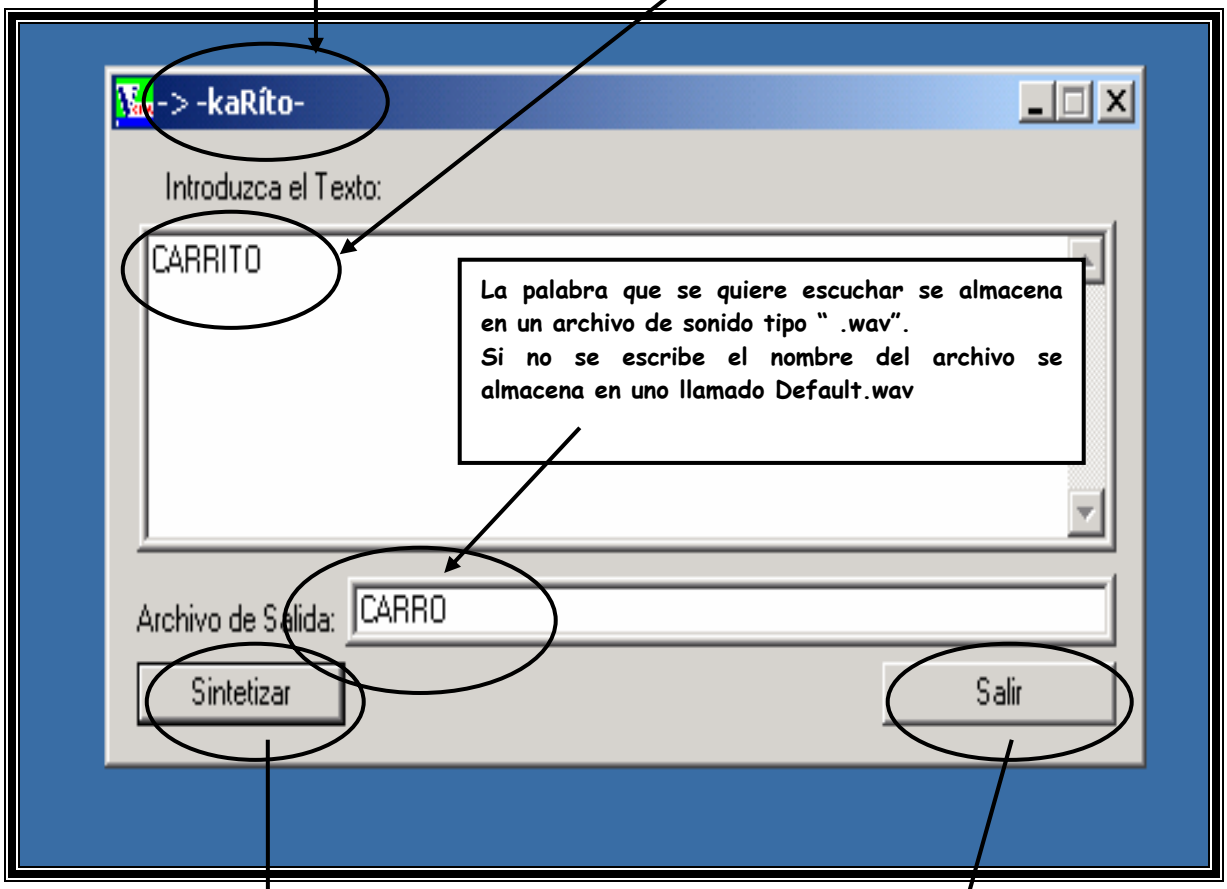
3.- Opción de escuchar el número escrito. El programa original daba como salida dígito por dígito.

Pantalla Principal



Al procesar la palabra, se visualizan los fonemas de la(s) palabra(s)

Se escriben la(s) palabra(s) que se quieren escuchar.



La palabra que se quiere escuchar se almacena en un archivo de sonido tipo " .wav".
Si no se escribe el nombre del archivo se almacena en uno llamado Default.wav

Se oprime la tecla "sintetizar" para que se escuche la palabra escrita inicialmente

Se oprime la tecla "salir" para terminar de utilizar el programa.

Con este simulador, se llevaron a cabo varias pruebas, específicamente cambios al programa principal, que nos llevan a realizar varios análisis espectrales de las diferentes señales generadas.

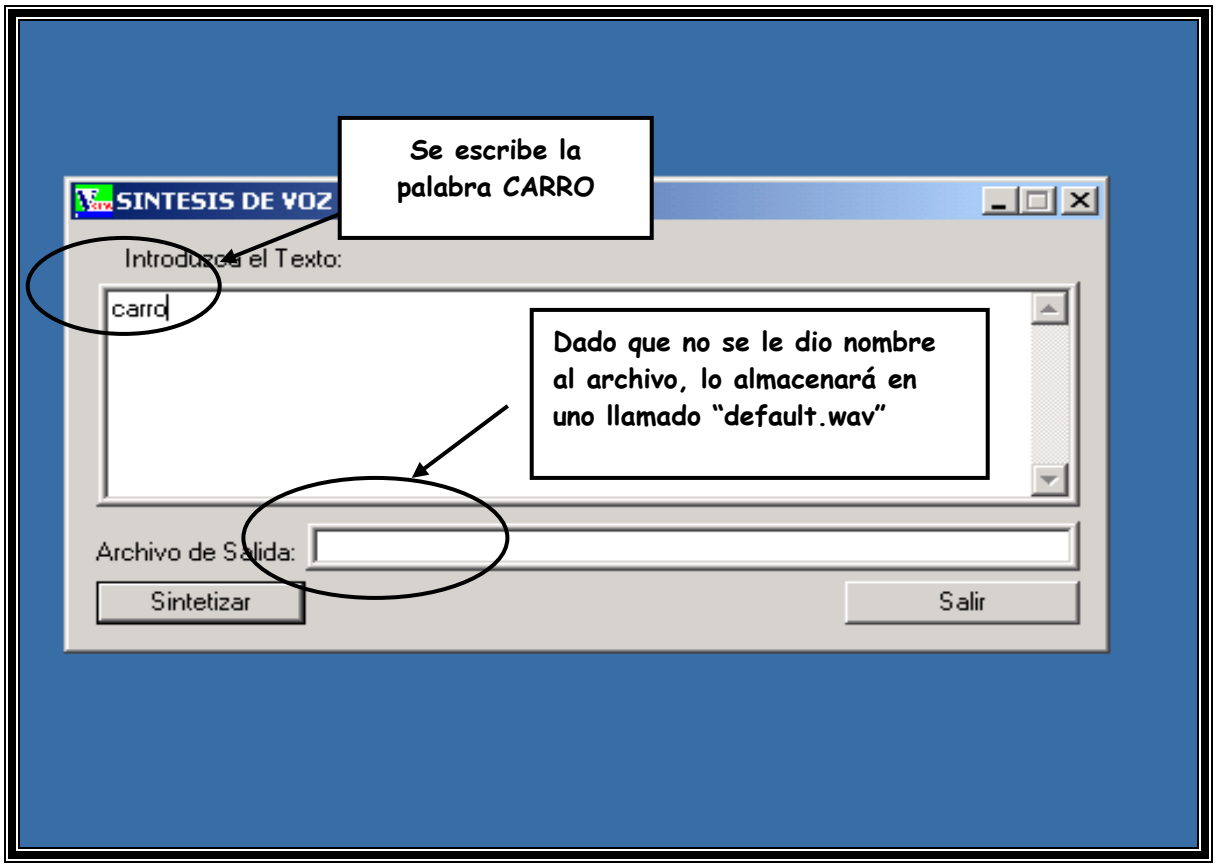
PRUEBA No. 1
CONVERSION TEXTO-VOZ
LA LETRA "R" SE ESCUCHA COMO "L", SIMULANDO LA PRONUNCIACIÓN DE UN NIÑO.

CAMBIOS AL PROGRAMA PRINCIPAL (MARCADOS EN AMARILLO)

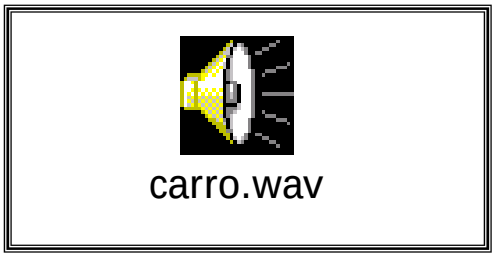
```
...
CString palabra(CString palabra) {
CString salida="";
int a,b,look,remember,found;
TCHAR p1,p2,l1,l2,l3;
b=palabra.GetLength();
b=b-1;

if (b==0) {
l1=palabra[0];
        switch(l1){
        case 'b':
                palabra="be";
        break;
        ...
case 'p':
                palabra="pe";
        break;
        case 'q':
                palabra="ku";
        break;
        case 'r':
                palabra="ele";
        break;
...
```

```
/* escribir header */
SetWindowText("->"+salida);
salida=salida+"-";
cad1 = (CEdit*) GetDlgItem(IDC_EDIT2);
cad1->GetWindowText(archson);
if (archson=="") {archson="default";}
archson=archson+".wav";
...
switch (l1){
case 'C':
ctemp="ch";
break;
...
case 'R':
ctemp="l";
break;
case 'S':
ctemp="sh";
break;}
switch (l2){
...
case 'R':
ctemp2="l";
break;
case 'S':
ctemp2="sh";
break;}

...
```

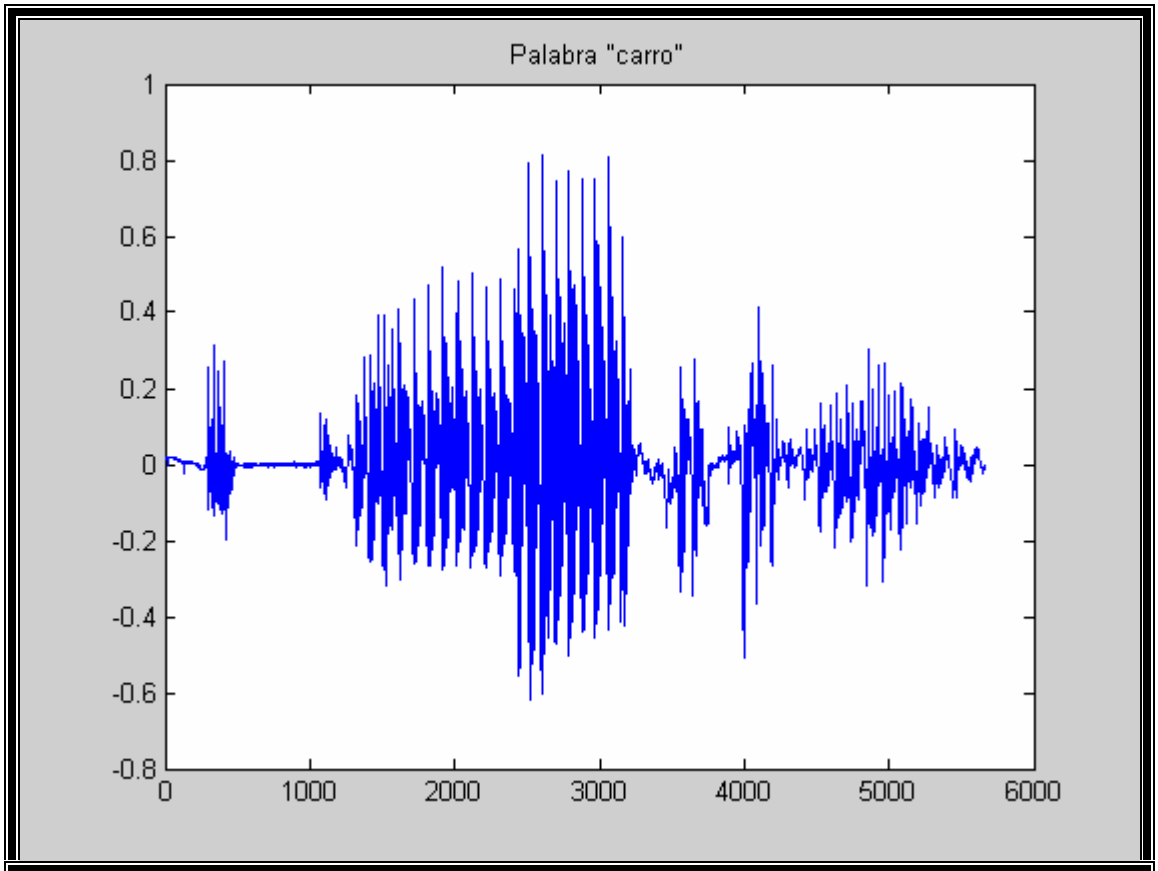
Se escribe la palabra CARRO antes de modificar el programa:

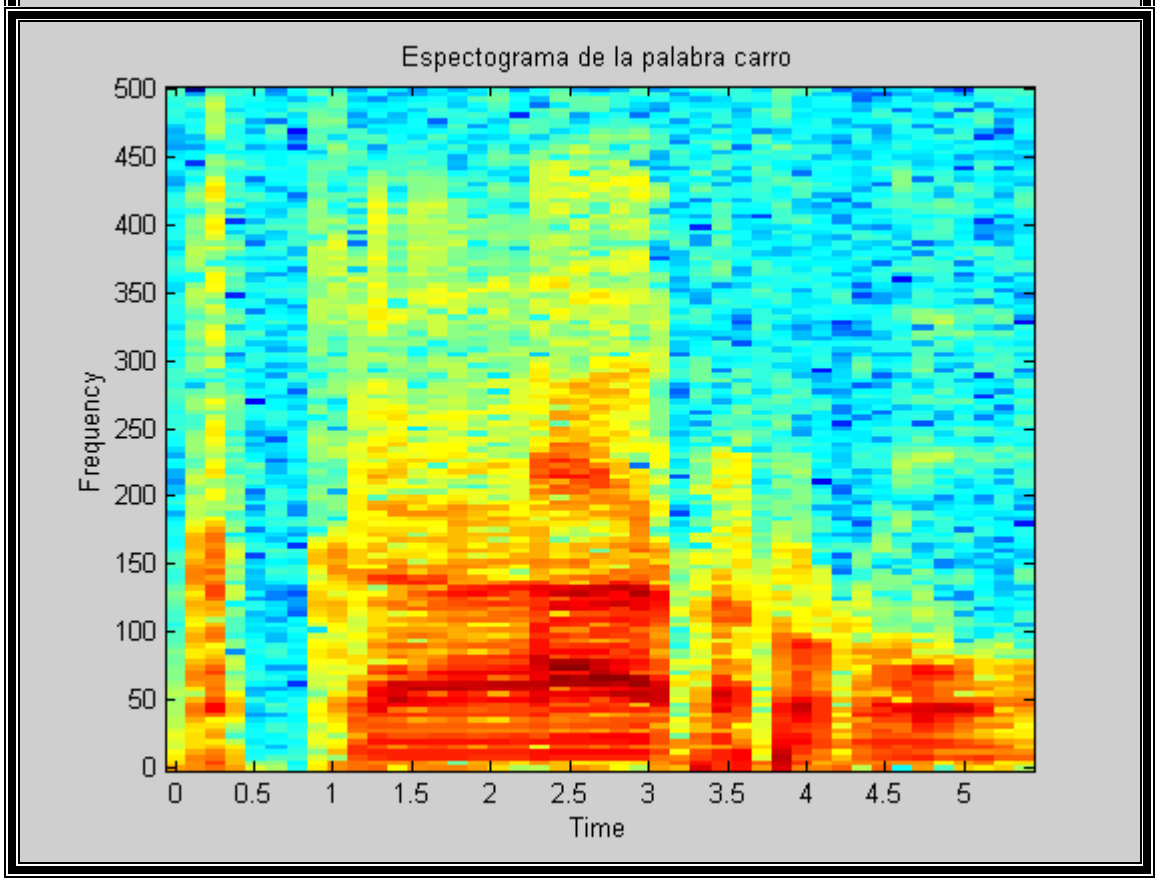


Se crea un archivo de sonido ".wav"

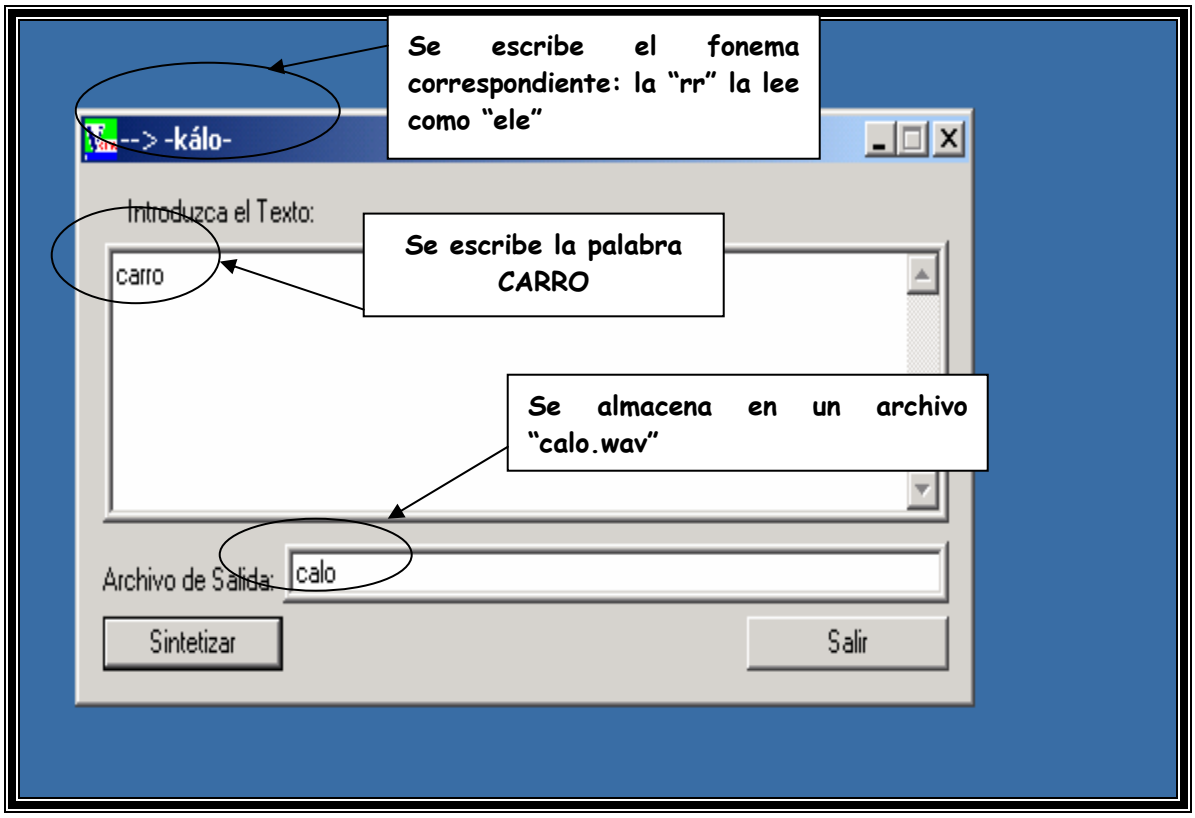


Se muestra la gráfica generada en Matlab de la palabra "carro"

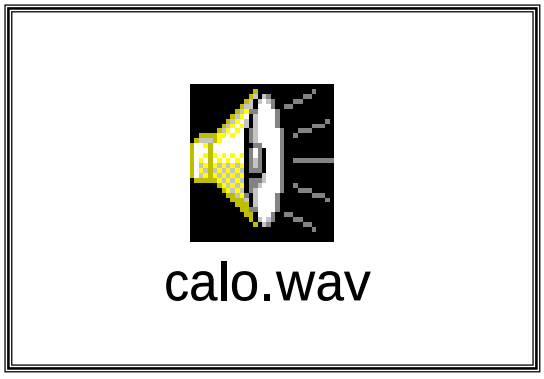

Palabra "carro"
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1000
2000
3000
4000
5000
6000


Espectograma de la palabra carro
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time

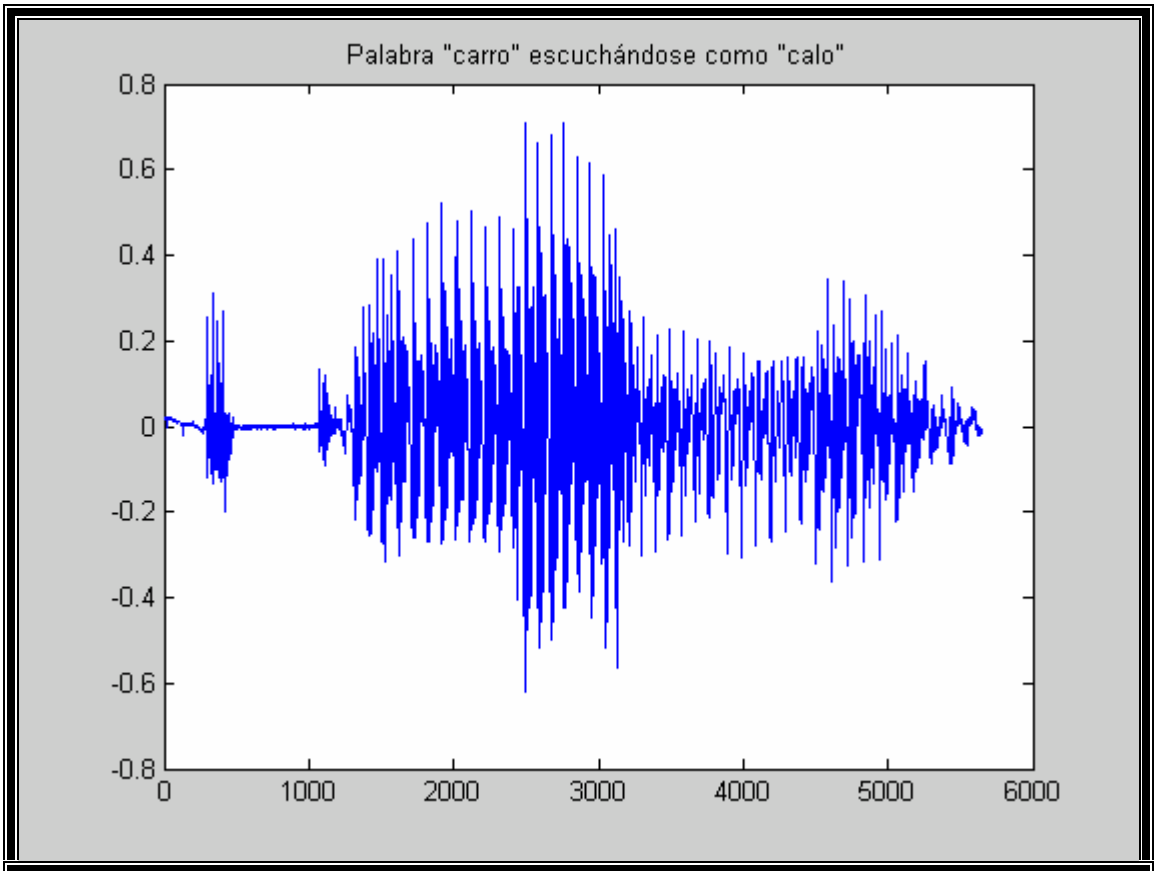
De la misma forma, se escribe la palabra CARRO la cual se escucha como CALO en el programa modificado, creando el archivo de sonido ".wav"

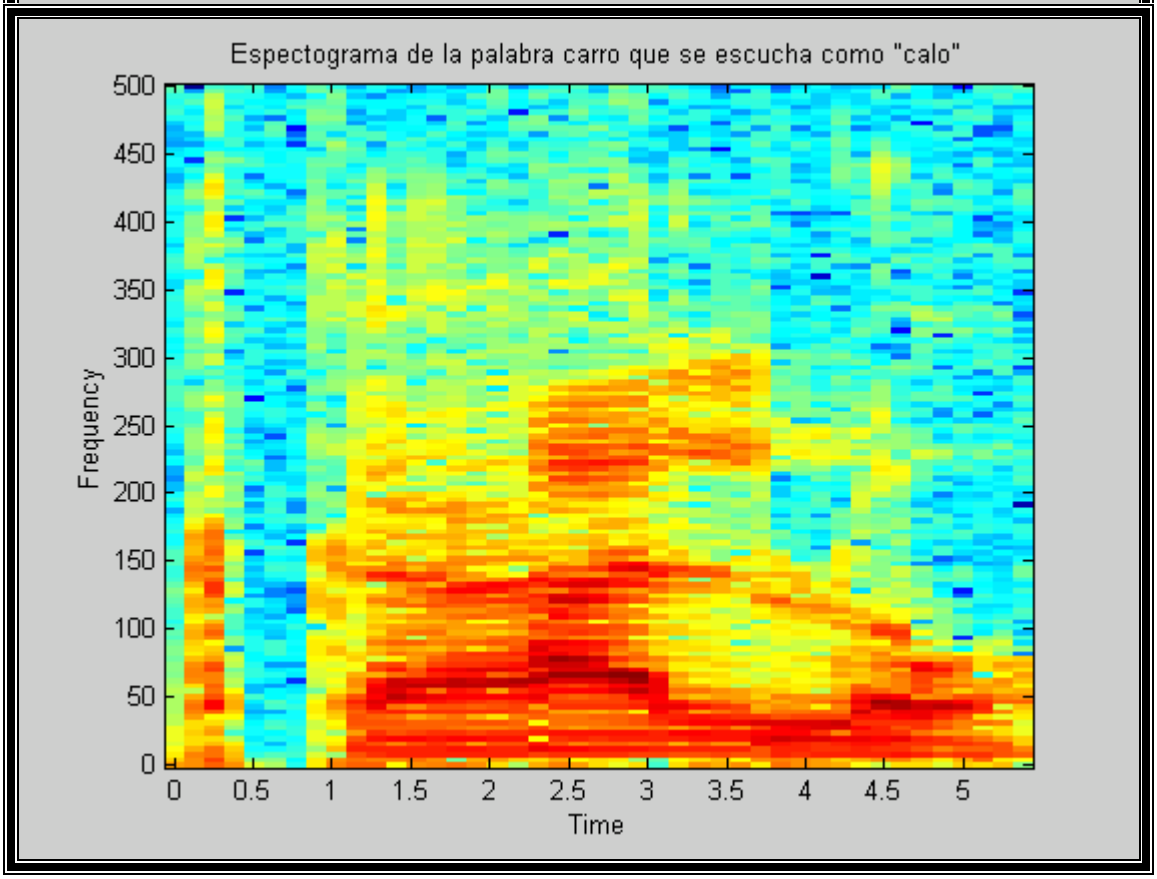


Archivo de sonido generado:



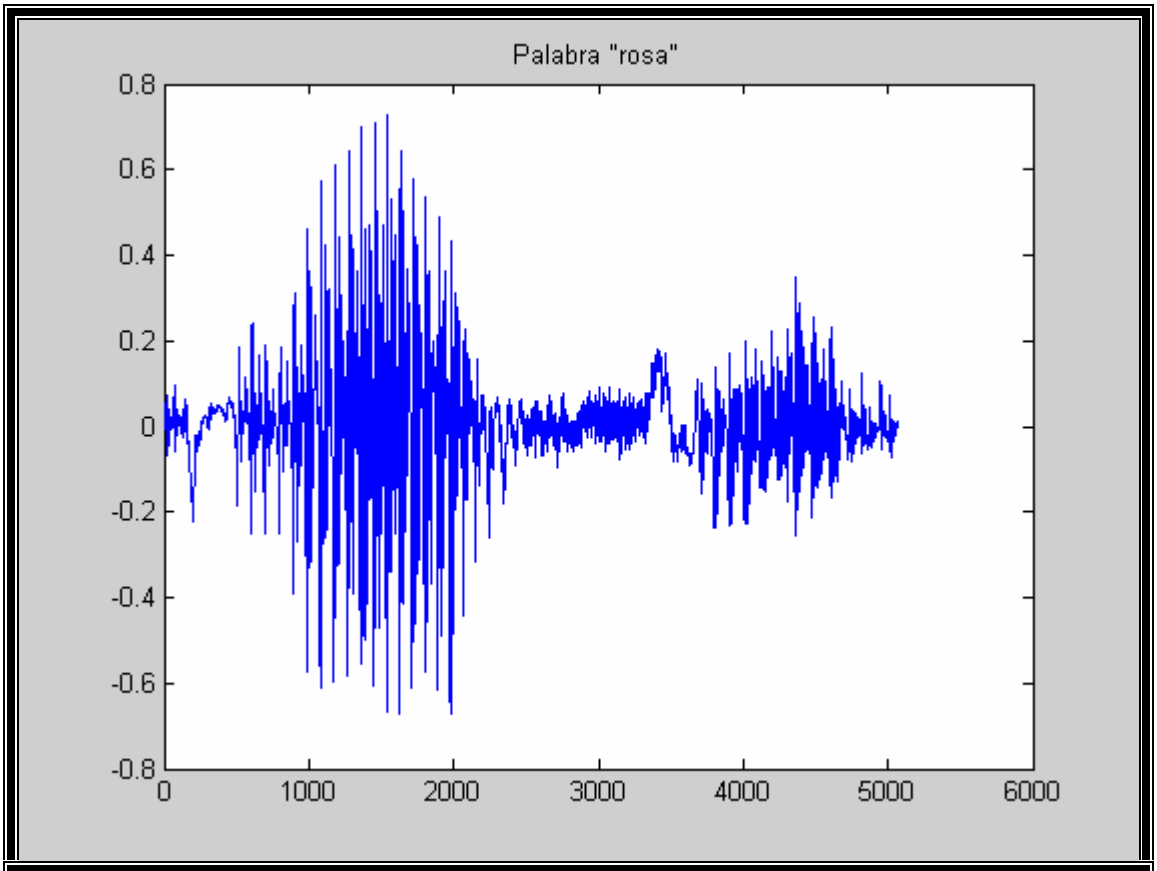
Se representa la gráfica en Matlab de la palabra "carro" escuchándose como "calo

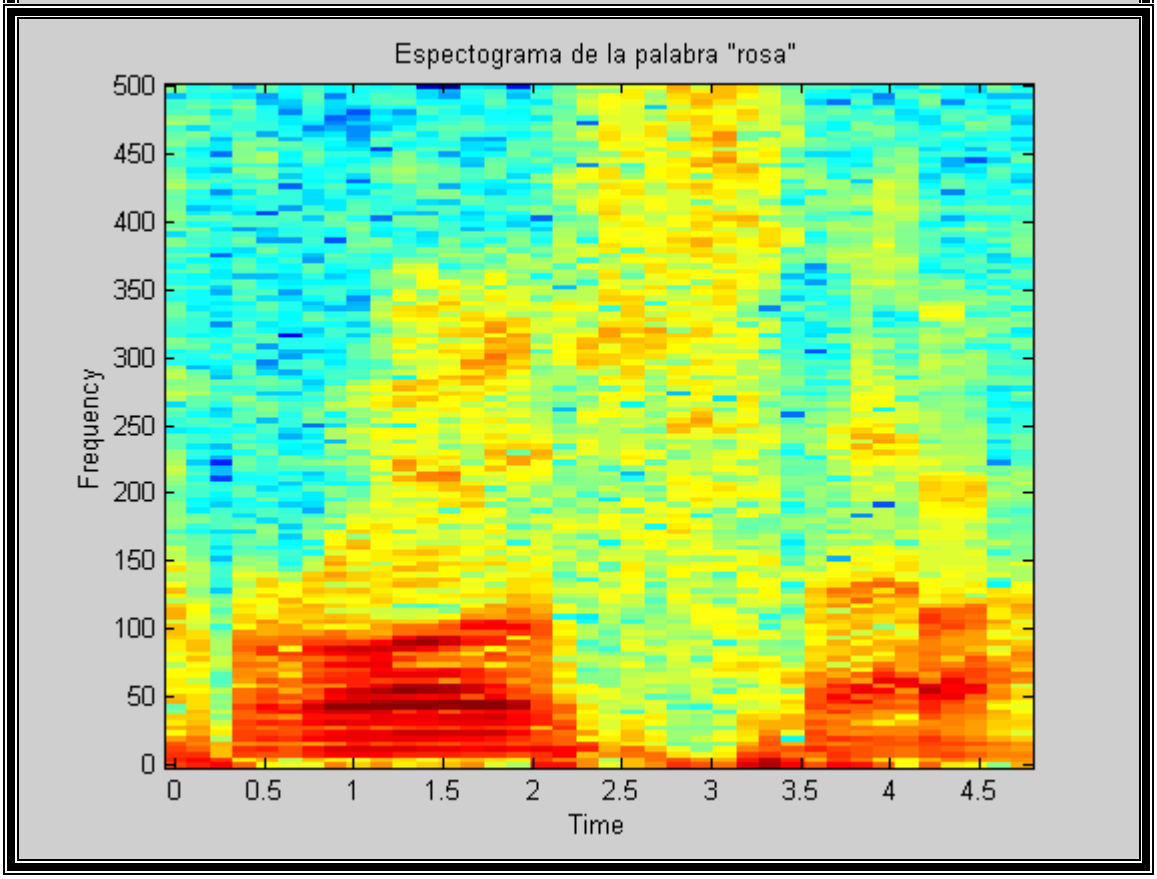

Palabra "carro" escuchándose como "calo"
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1000
2000
3000
4000
5000
6000

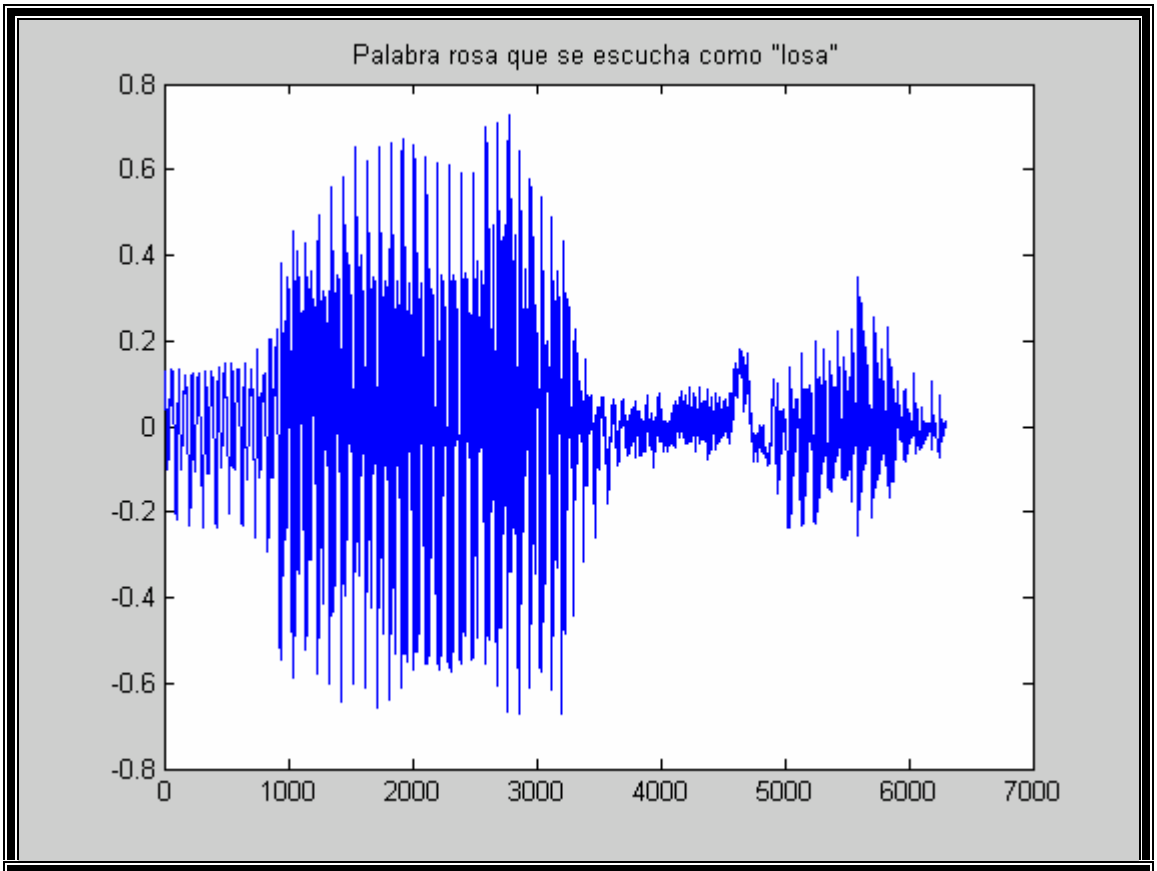

Espectograma de la palabra carro que se escucha como "calo"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time

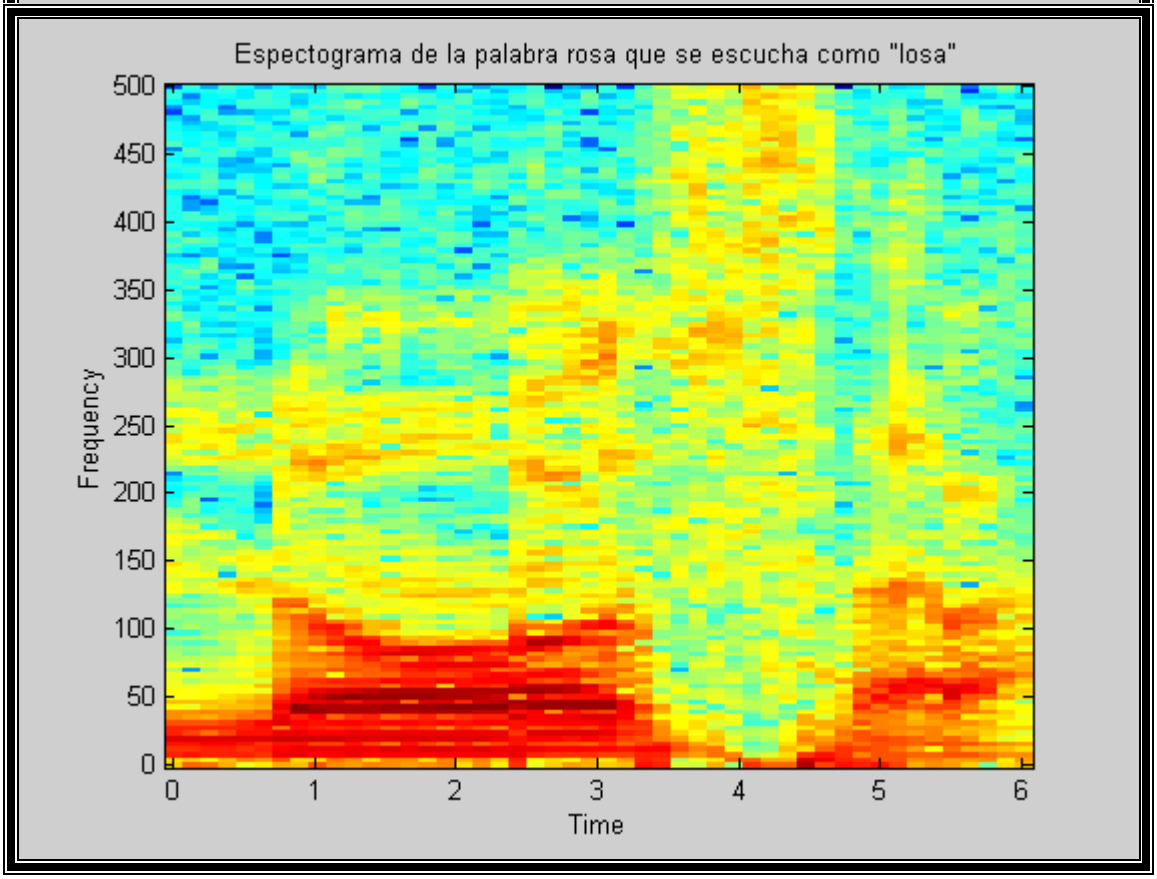
Otros ejemplos

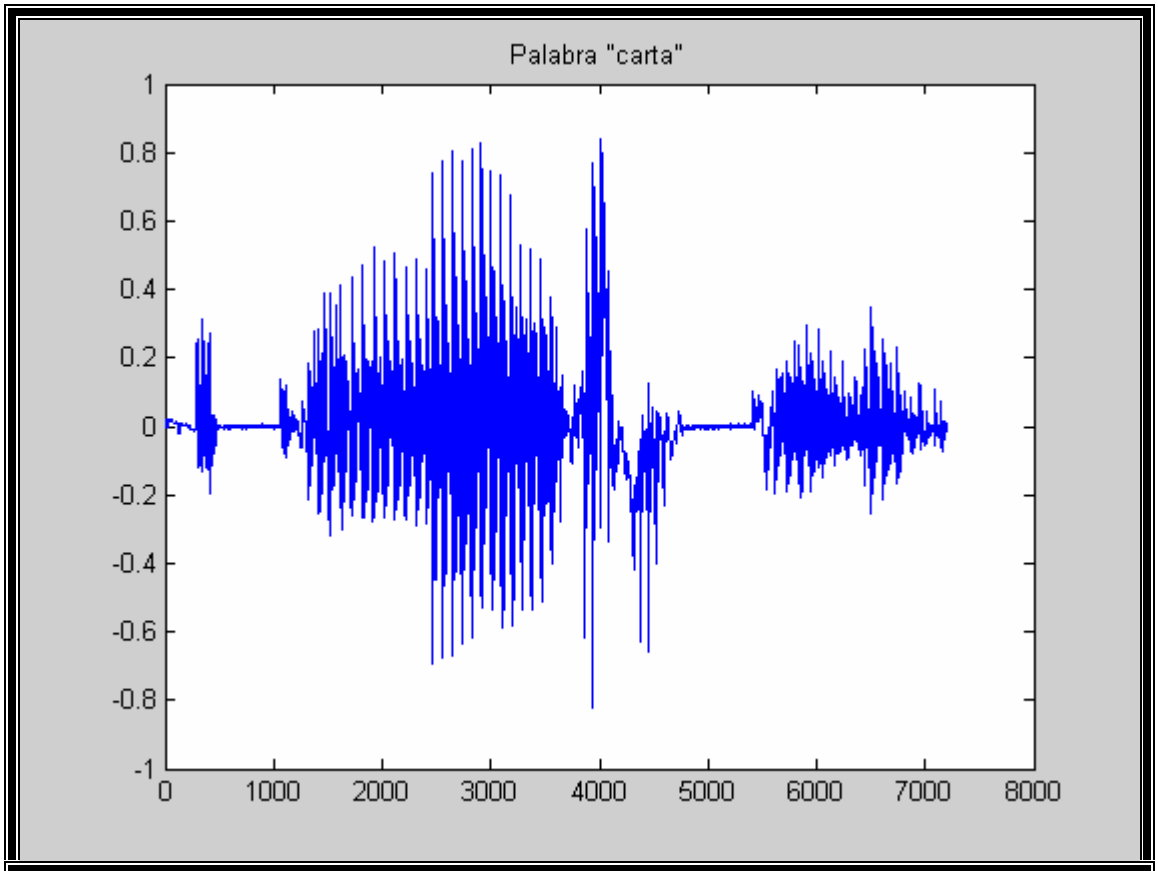
PALABRA	ARCHIVO DE SONIDO creado con el programa original	PALABRA (se escucha la letra "r" como "l")	ARCHIVO DE SONIDO creado con el programa modificado
Rosa	 rosa.wav	Losa	 losa.wav
Carta	 carta.wav	Calta	 calta.wav
Risa	 risa.wav	Lisa	 lisa.wav

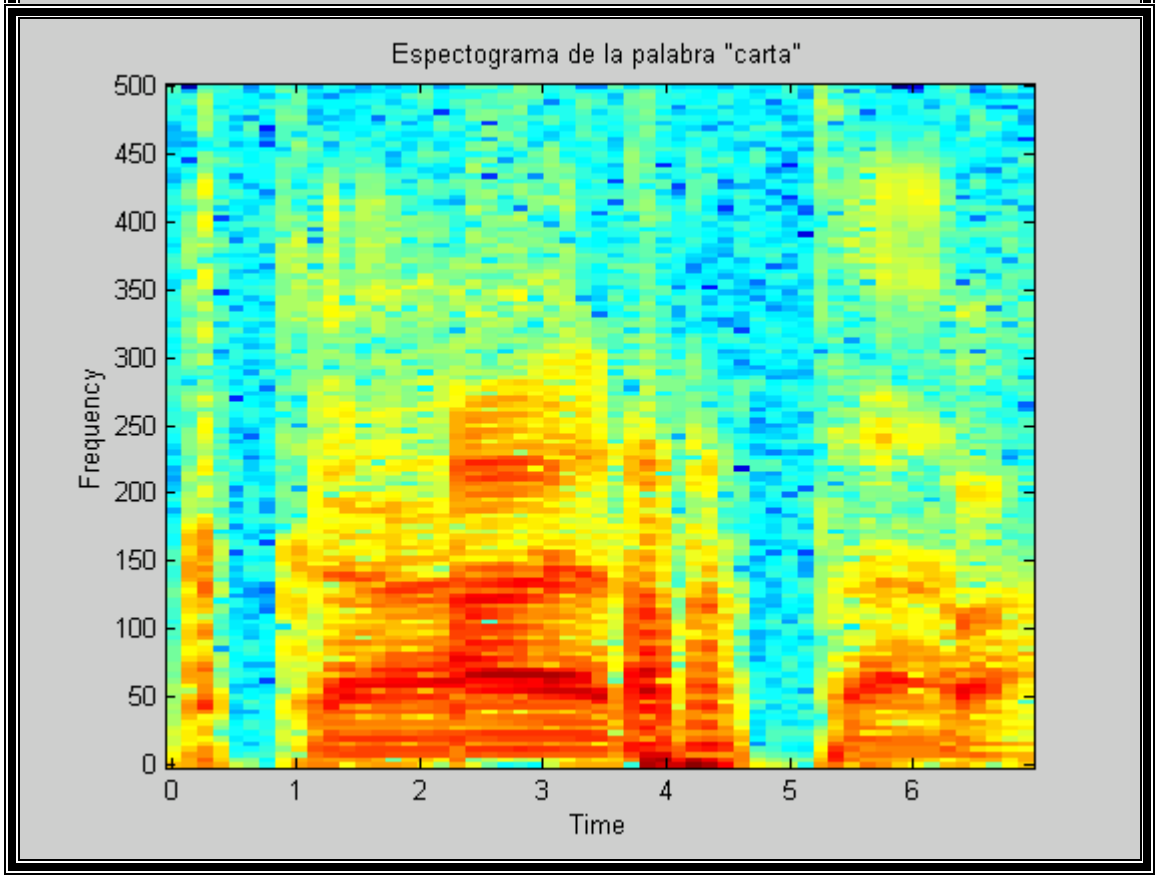

Palabra "rosa"
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1000
2000
3000
4000
5000
6000

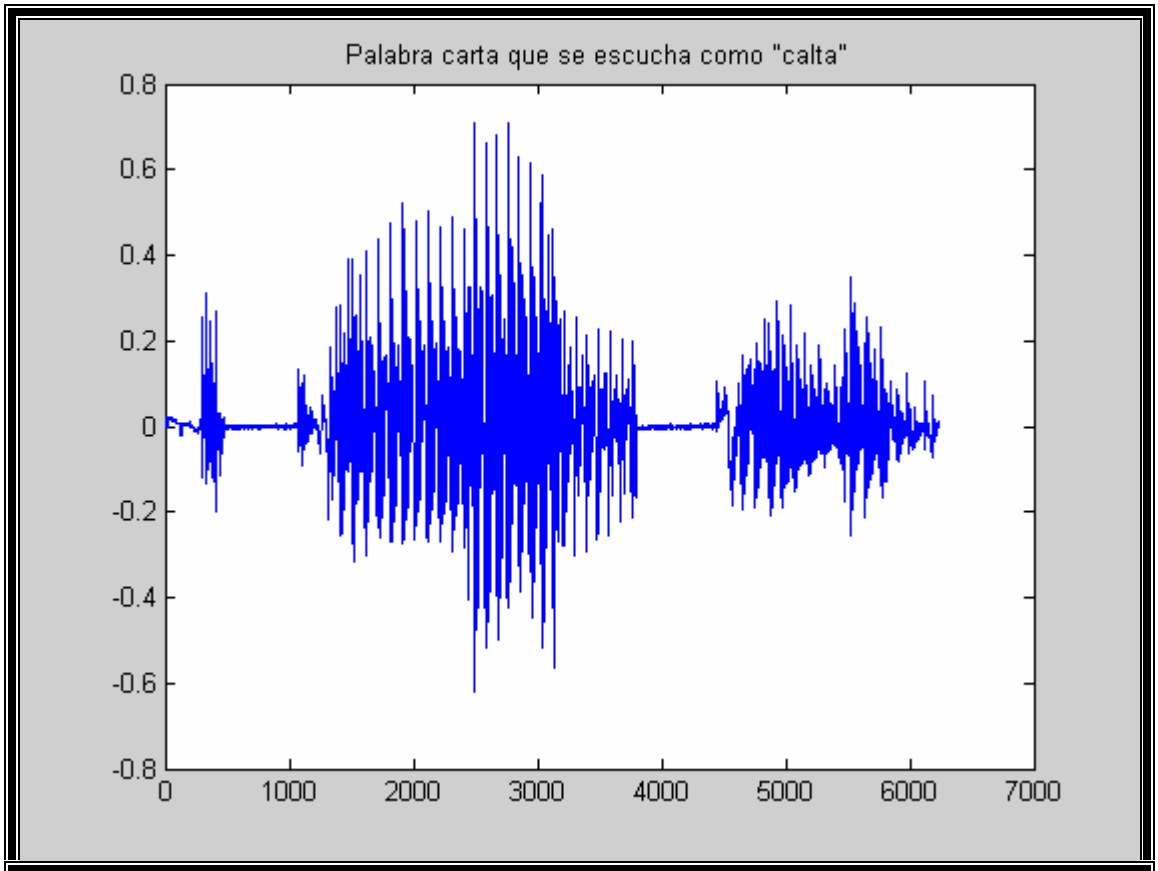

Espectograma de la palabra "rosa"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
Time

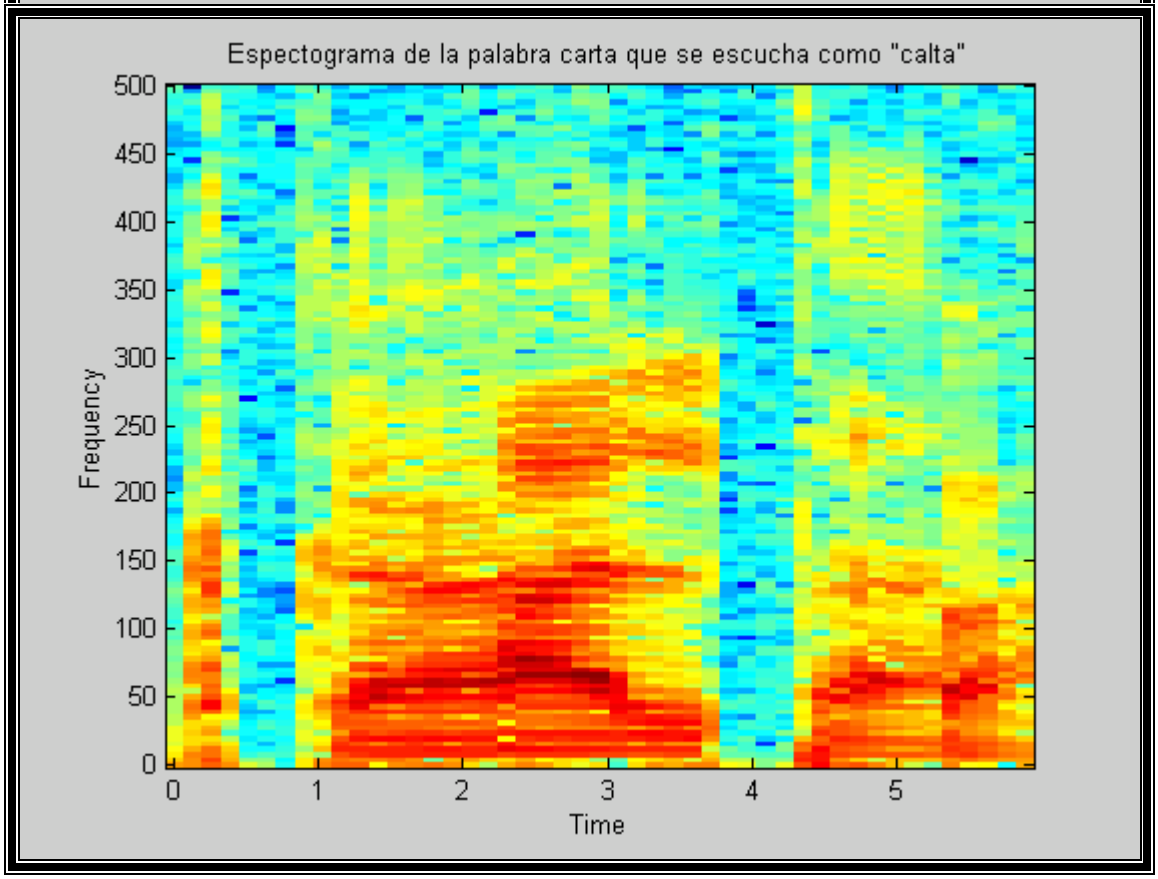

Palabra rosa que se escucha como "losa"
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1000
2000
3000
4000
5000
6000
7000

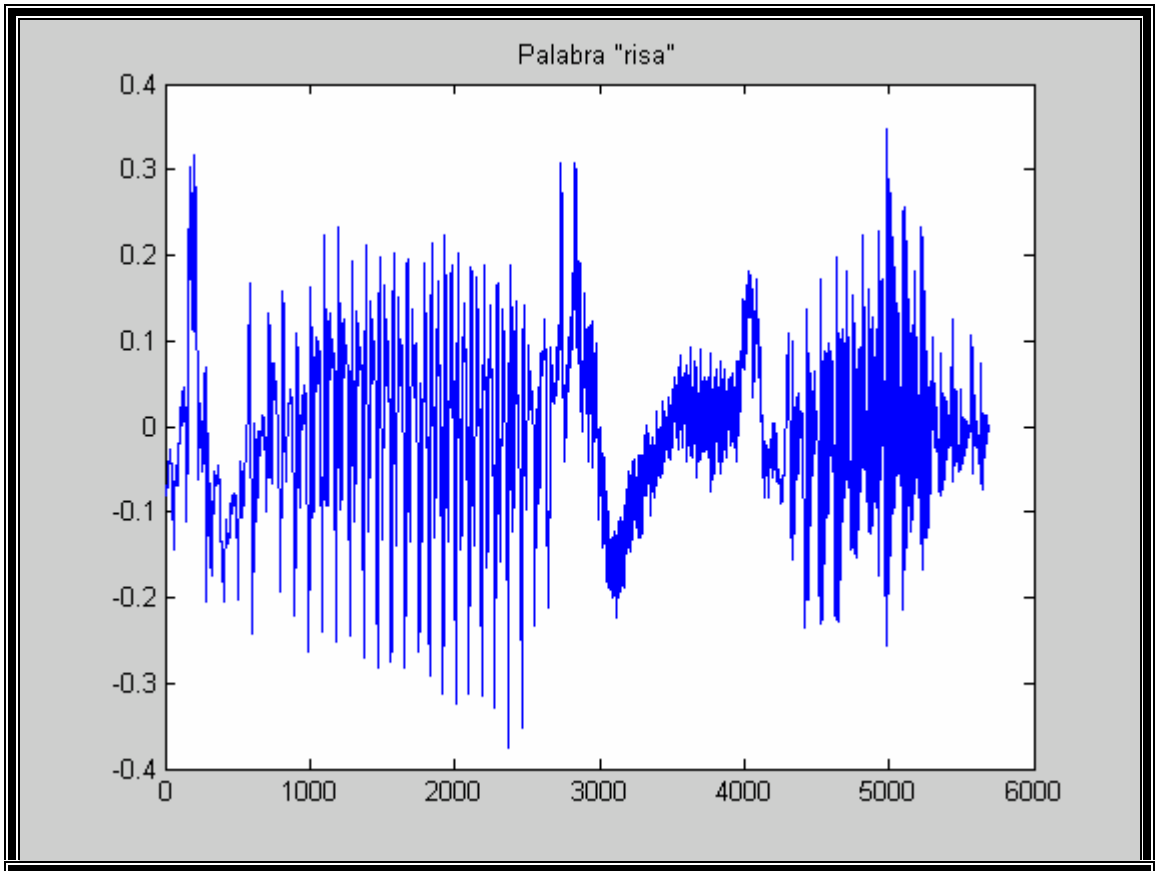

Espectograma de la palabra rosa que se escucha como "losa"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
1
2
3
4
5
6
Time

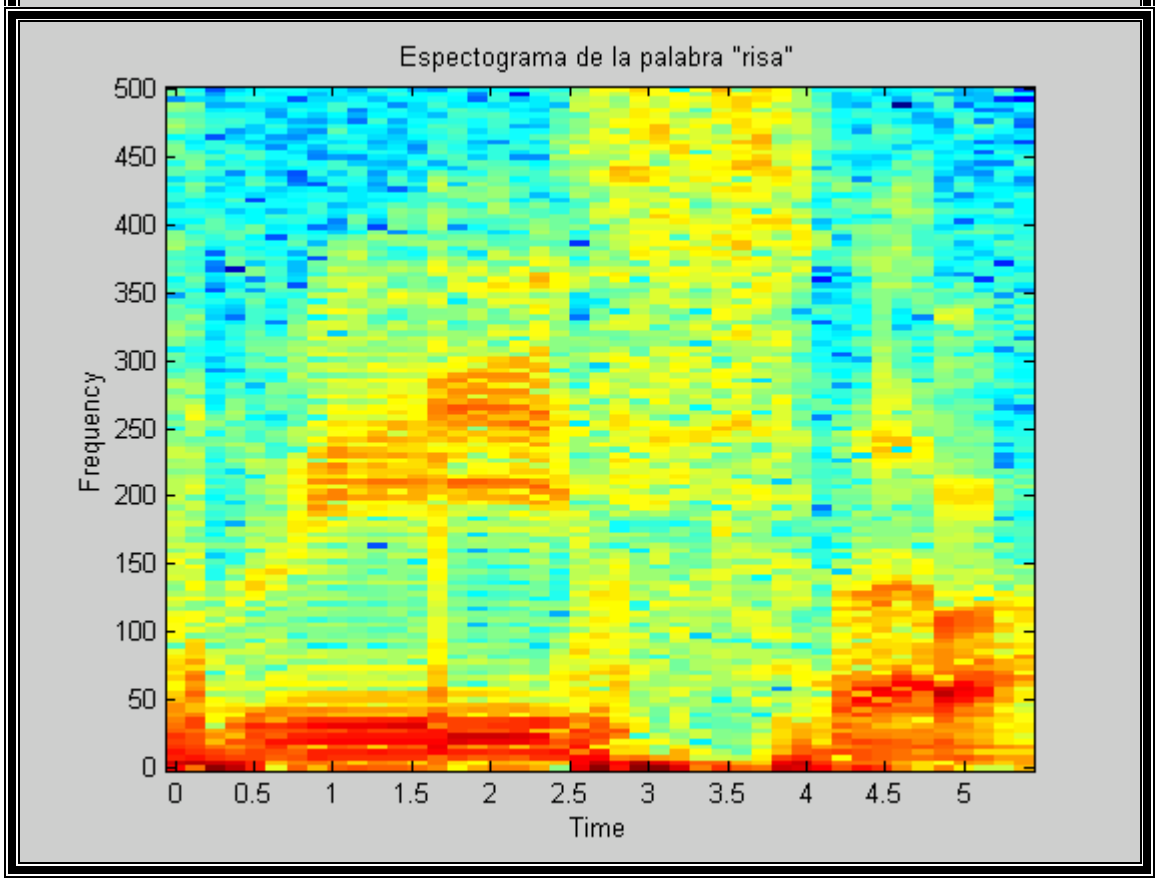

Palabra "carta"
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1000
2000
3000
4000
5000
6000
7000
8000

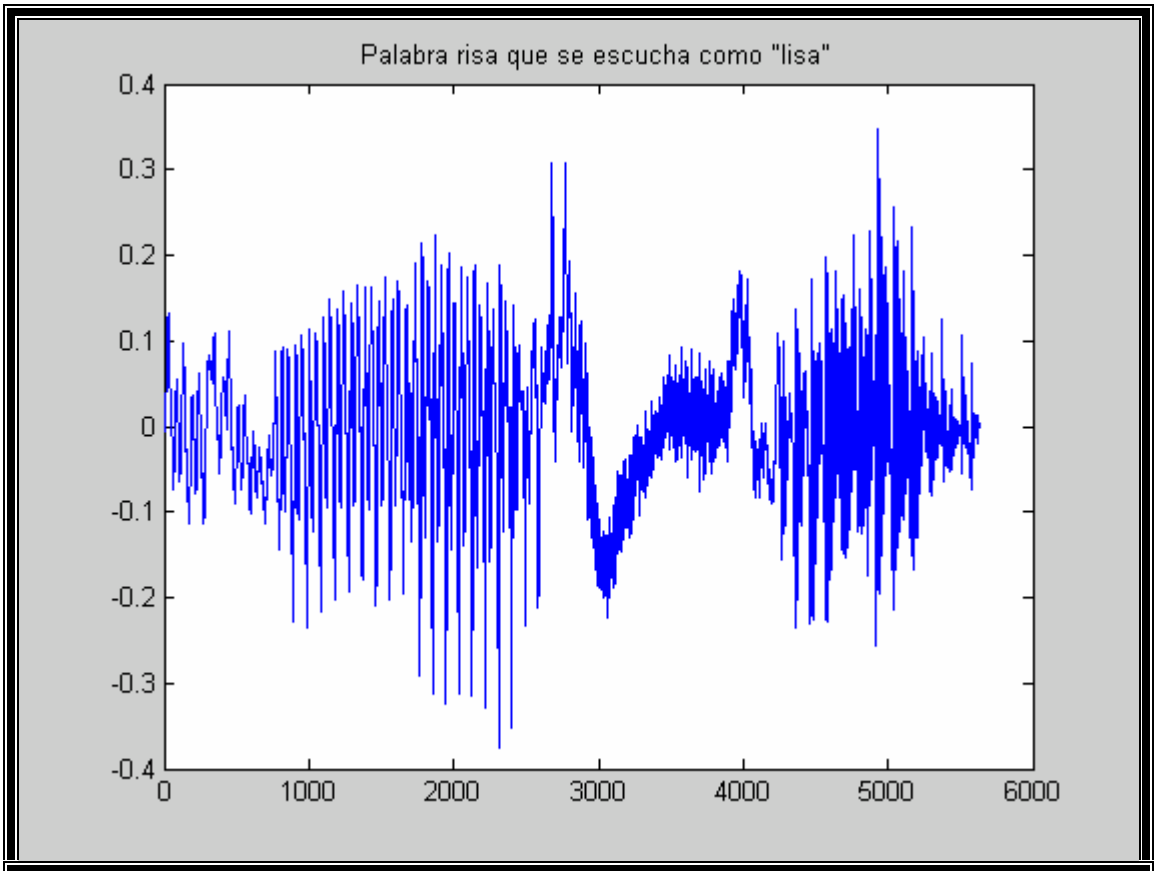

Espectograma de la palabra "carta"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
1
2
3
4
5
6
Time

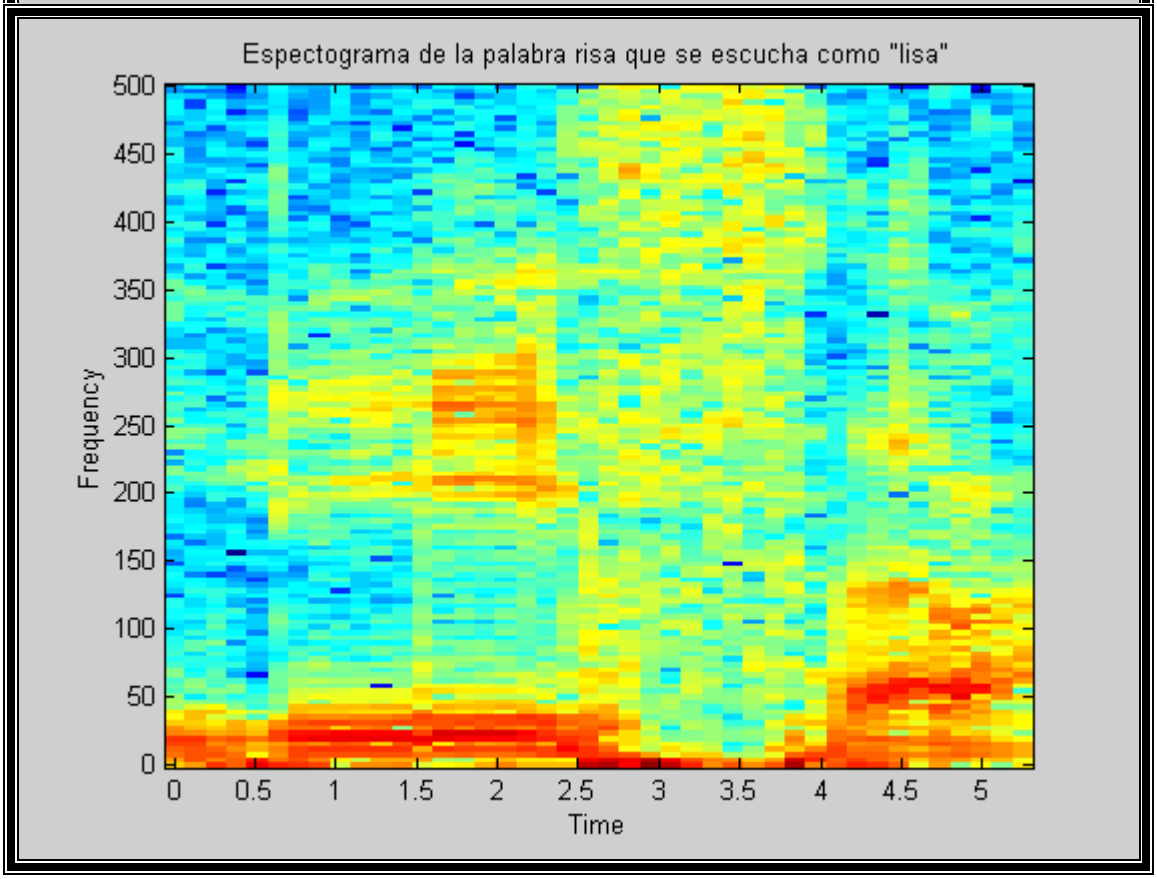

Palabra carta que se escucha como "calta"
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1000
2000
3000
4000
5000
6000
7000


Espectograma de la palabra carta que se escucha como "calta"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
1
2
3
4
5
Time


Palabra "risa"
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
1000
2000
3000
4000
5000
6000


Espectograma de la palabra "risa"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time


Palabra risa que se escucha como "lisa"
0.4
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
0
1000
2000
3000
4000
5000
6000


Espectograma de la palabra risa que se escucha como "lisa"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
Time

PRUEBA No. 2
CONVERSIÓN TEXTO-VOZ
DIPTONGOS DONDE SÓLO SE ESCUCHA LA ÚLTIMA SÍLABA

CAMBIOS AL PROGRAMA (MARCADOS EN AMARILLO)

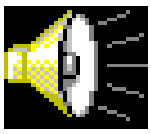
```
...

for (a=0;a<=b;a++) {

        l1=palabra[a];
        if(a<b)   l2=palabra[a+1];
        else l2=' ';
        if(a<b-1) l3=palabra[a+2];
        else l3=' ';
        if(a>0)   p1=palabra[a-1];
        else p1=' ';
        switch (l1){
        case 'a':
                if (l2=='ú') {l1='ú';a=a+1;}
                if (l2=='é') {l1='é';a=a+1;}
                if (l2=='ó') {l1='ó';a=a+1;}
                if (l2=='í') {l1='í';a=a+1;}
                if (l2=='u') {l1='u';a=a+1;}
                if (l2=='e') {l1='e';a=a+1;}
                if (l2=='o') {l1='o';a=a+1;}
                if (l2=='i') {l1='i';a=a+1;}
        break;
        case 'e':
                if (l2=='ú') {l1='ú';a=a+1;}
                if (l2=='á') {l1='á';a=a+1;}
                if (l2=='ó') {l1='ó';a=a+1;}
                if (l2=='í') {l1='í';a=a+1;}
                if (l2=='u') {l1='u';a=a+1;}
                if (l2=='a') {l1='a';a=a+1;}
                if (l2=='o') {l1='o';a=a+1;}
                if (l2=='i') {l1='i';a=a+1;}
        break;
        case 'i':
                if (l2=='ú') {l1='ú';a=a+1;}
                if (l2=='á') {l1='á';a=a+1;}
                if (l2=='ó') {l1='ó';a=a+1;}
                if (l2=='é') {l1='é';a=a+1;}
                if (l2=='u') {l1='u';a=a+1;}
                if (l2=='a') {l1='a';a=a+1;}
                if (l2=='o') {l1='o';a=a+1;}
                if (l2=='e') {l1='e';a=a+1;}
        case 'á':
                if (l2=='ú') {l1='ú';a=a+1;}
                if (l2=='é') {l1='é';a=a+1;}
                if (l2=='ó') {l1='ó';a=a+1;}
                if (l2=='í') {l1='í';a=a+1;}
                if   (l2=='u')   {l1='u';a=a+1;}
                        if          (l2=='e')
{l1='e';a=a+1;}
                if (l2=='o') {l1='o';a=a+1;}
                if (l2=='i') {l1='i';a=a+1;}
        break;
        case 'é':
                if (l2=='ú') {l1='ú';a=a+1;}
                if (l2=='á') {l1='á';a=a+1;}
                if (l2=='ó') {l1='ó';a=a+1;}
                if (l2=='í') {l1='í';a=a+1;}
                if (l2=='u') {l1='u';a=a+1;}
                if (l2=='a') {l1='a';a=a+1;}
                if (l2=='o') {l1='o';a=a+1;}
                if (l2=='i') {l1='i';a=a+1;}
        break;
        case 'í':
                if (l2=='ú') {l1='ú';a=a+1;}
                if (l2=='á') {l1='á';a=a+1;}
                if (l2=='ó') {l1='ó';a=a+1;}
                if (l2=='é') {l1='é';a=a+1;}
                if (l2=='u') {l1='u';a=a+1;}
                if (l2=='a') {l1='a';a=a+1;}
                if (l2=='o') {l1='o';a=a+1;}
                if (l2=='e') {l1='e';a=a+1;}
        break;
        case 'ó':
                if (l2=='ú') {l1='ú';a=a+1;}
                if (l2=='á') {l1='á';a=a+1;}
                if (l2=='é') {l1='é';a=a+1;}
                if (l2=='í') {l1='í';a=a+1;}
                if (l2=='u') {l1='u';a=a+1;}
                 if (l2=='a') {l1='a';a=a+1;}
                if (l2=='e') {l1='e';a=a+1;}
                if (l2=='i') {l1='i';a=a+1;}
        break;
        case 'ú':
                if (l2=='é') {l1='é';a=a+1;}
```

```
        break;
        case 'o':
                if (l2=='ú') {l1='ú';a=a+1;}
                if (l2=='á') {l1='á';a=a+1;}
                if (l2=='é') {l1='é';a=a+1;}

                if (l2=='í') {l1='í';a=a+1;}
                if (l2=='u') {l1='u';a=a+1;}
                 if (l2=='a') {l1='a';a=a+1;}
                if (l2=='e') {l1='e';a=a+1;}
                if (l2=='i') {l1='i';a=a+1;}
        break;
case 'u':
                if (l2=='é') {l1='é';a=a+1;}
                if (l2=='á') {l1='á';a=a+1;}
                if (l2=='ó') {l1='ó';a=a+1;}
                if (l2=='í') {l1='í';a=a+1;}
                if (l2=='e') {l1='e';a=a+1;}
                if (l2=='a') {l1='a';a=a+1;}
                if (l2=='o') {l1='o';a=a+1;}
                if (l2=='i') {l1='i';a=a+1;}
        break;
```

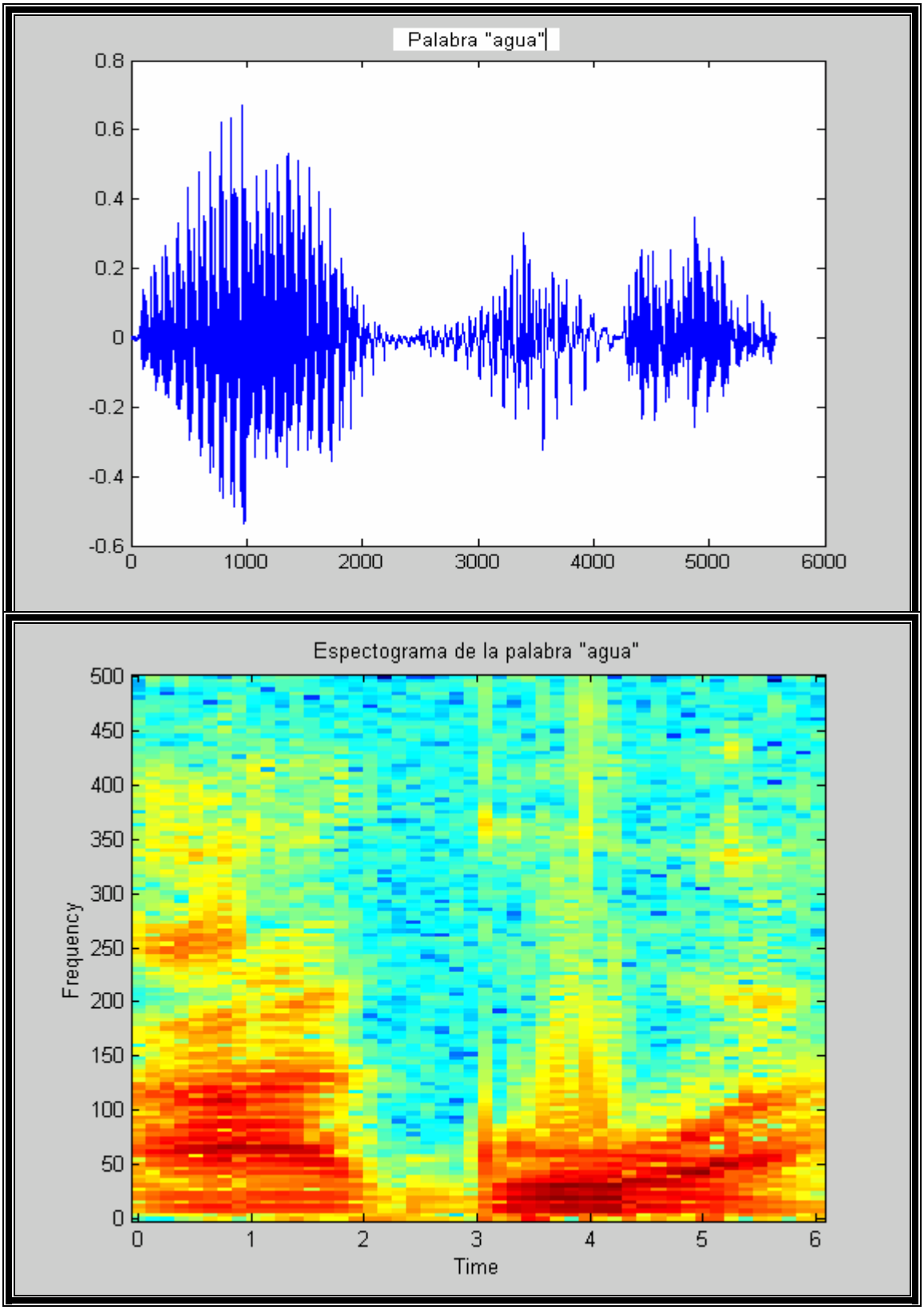
```
                if (l2=='á') {l1='á';a=a+1;}
                if (l2=='ó') {l1='ó';a=a+1;}
                if (l2=='í') {l1='í';a=a+1;}
                if (l2=='e') {l1='e';a=a+1;}
                if (l2=='a') {l1='a';a=a+1;}
                if (l2=='o') {l1='o';a=a+1;}
                if (l2=='i') {l1='i';a=a+1;}
        break;


case 'c':
                l1='k';
                if (l2=='h') {l1='C';a=a+1;}
                if (vdebil(l2)) {l1='s';}
        break;
        case 's':
                if (l2=='h') { l1='S';a=a+1;}
        break;

if(l1!='h') salida = salida + l1; }
...
```

Ejemplo:

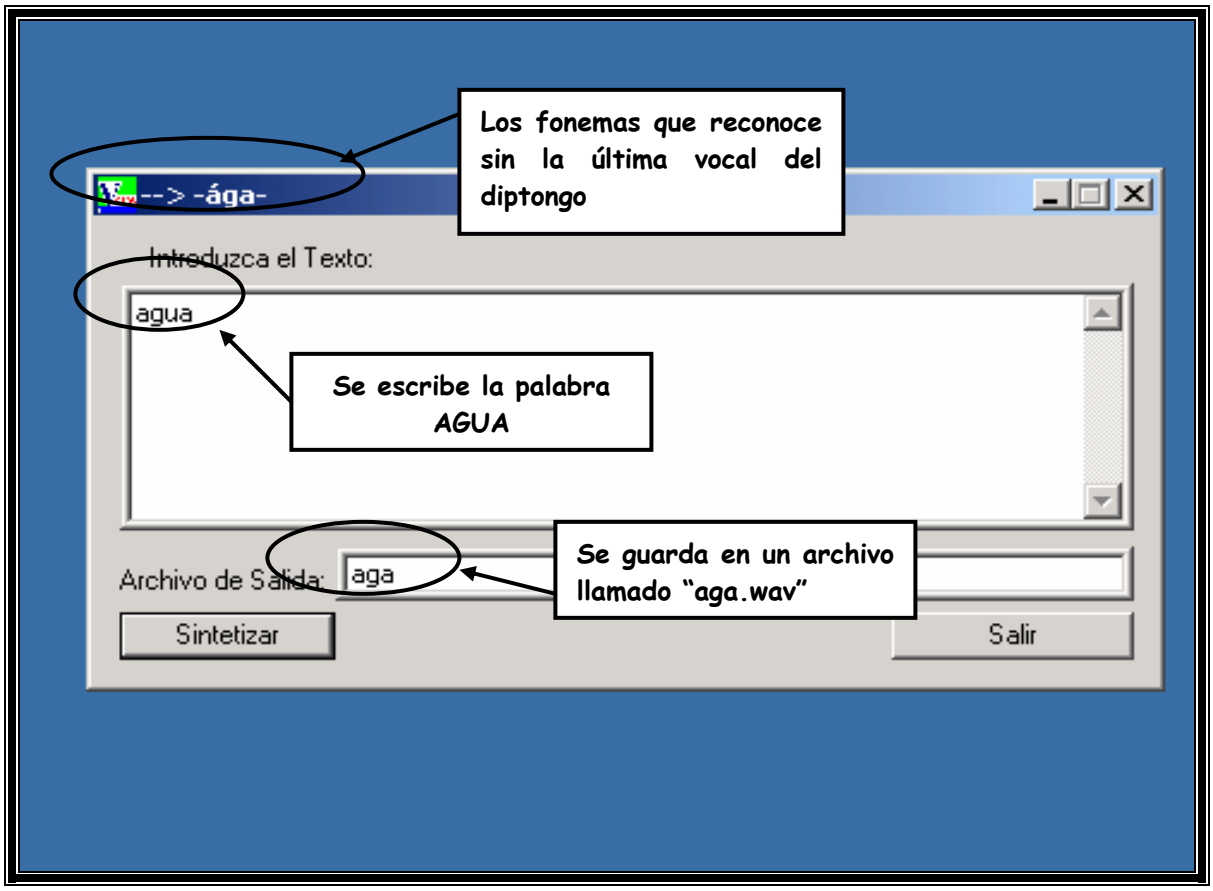
Palabra "agua"



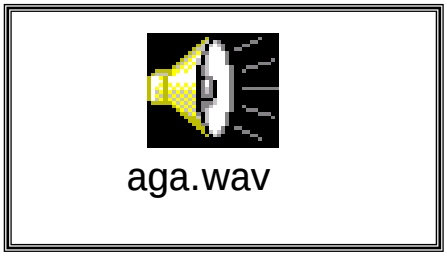
agua.wav


Palabra "agua"
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
1000
2000
3000
4000
5000
6000
Espectograma de la palabra "agua"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
1
2
3
4
5
6
Time

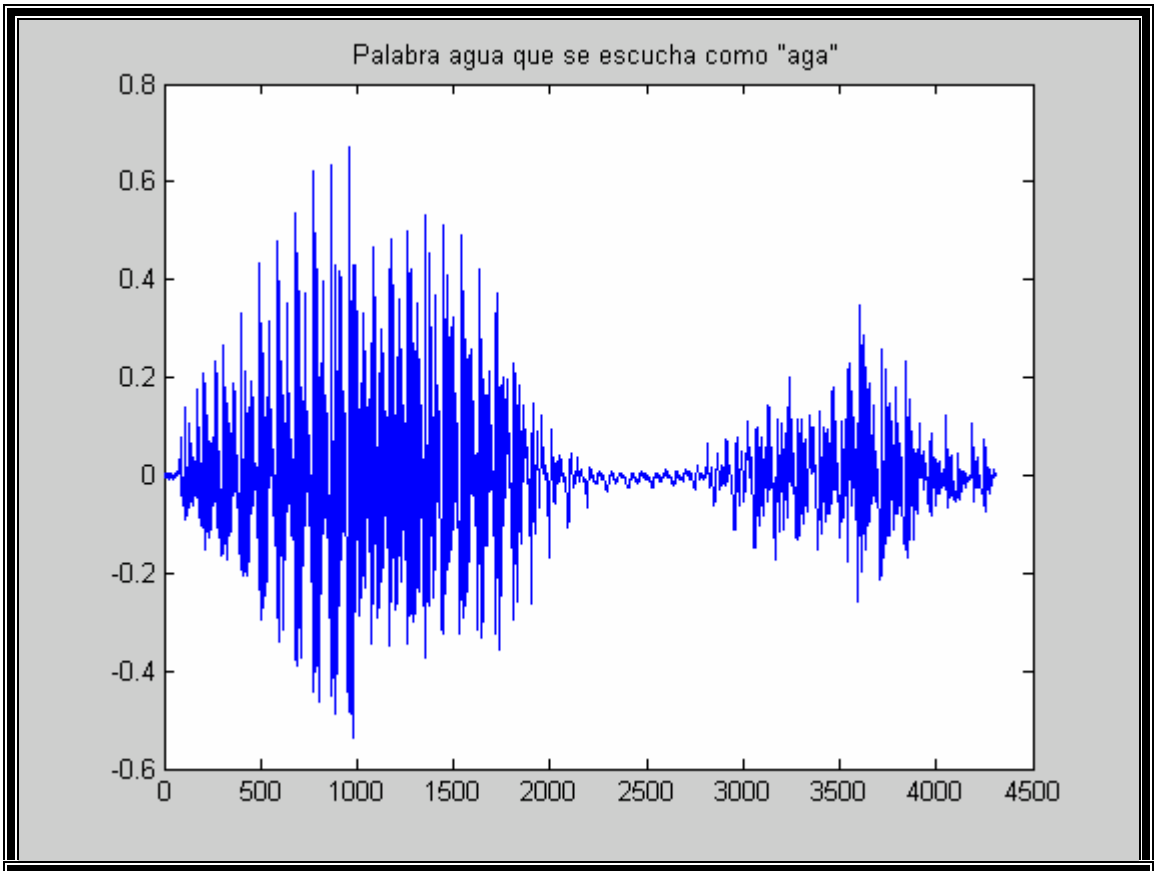
Palabra agua que se escucha como "aga"

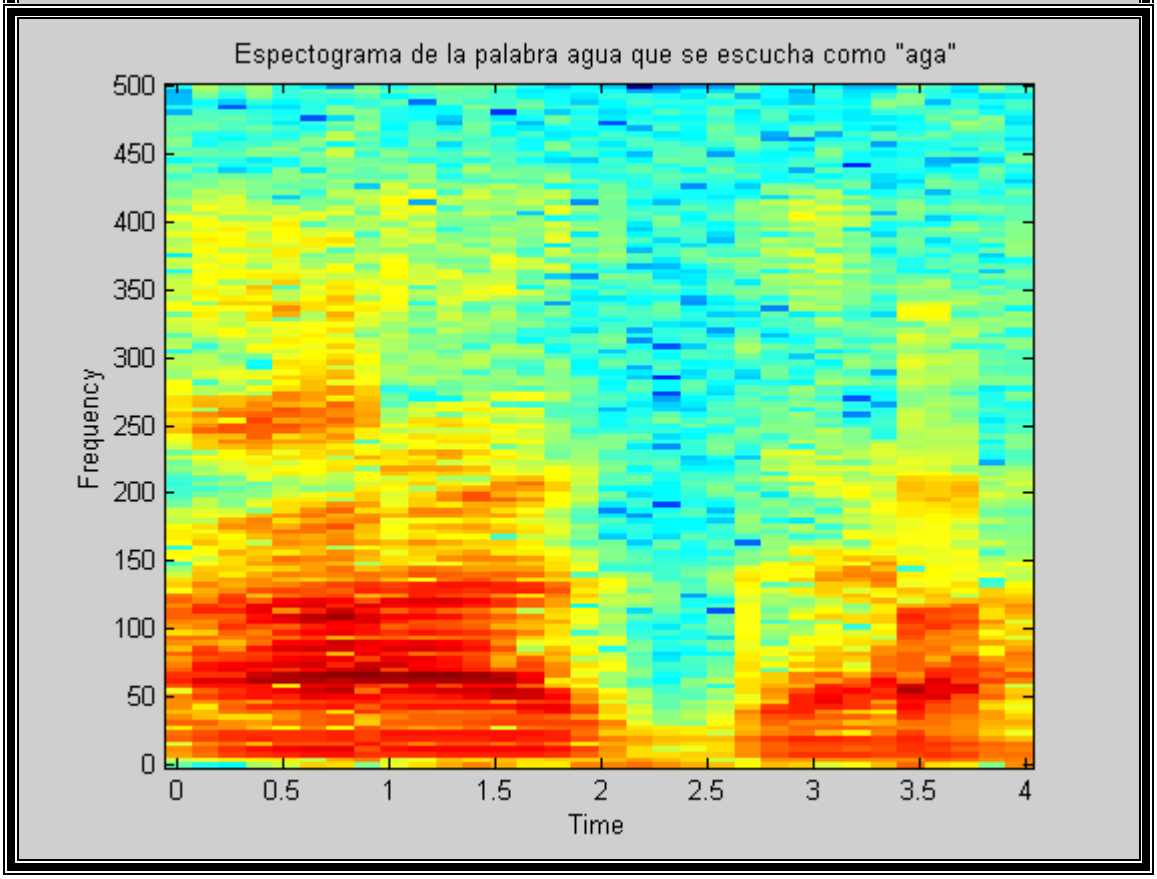


Archivo de sonido generado



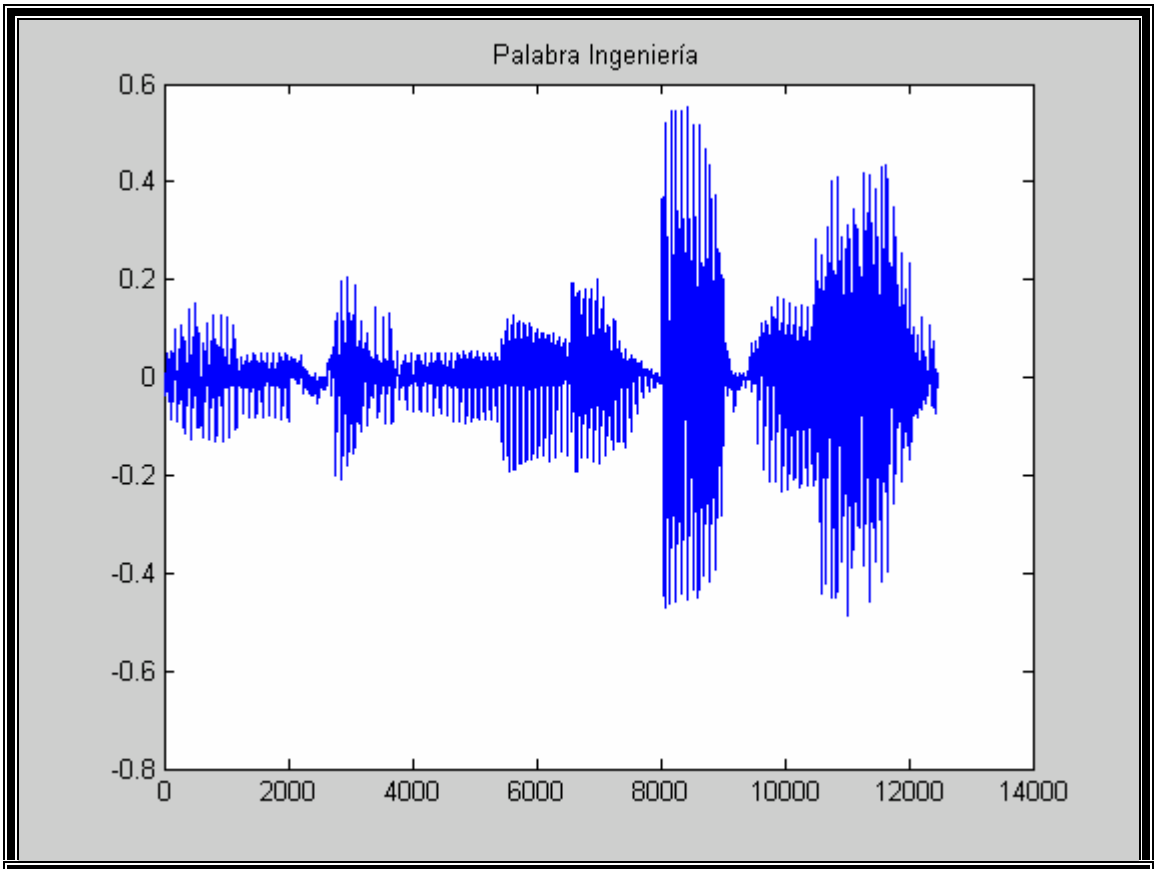
La gráfica en matlab,

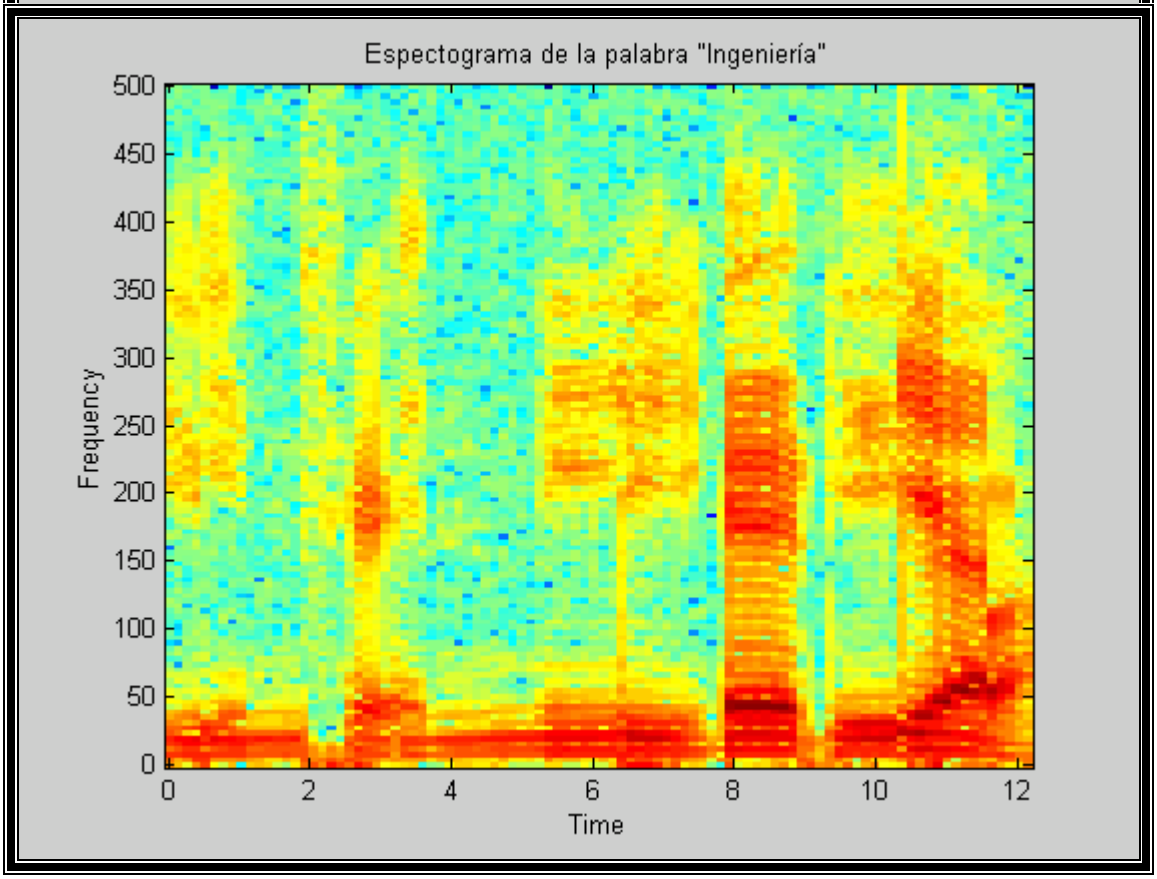

Palabra agua que se escucha como "aga"
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
0
500
1000
1500
2000
2500
3000
3500
4000
4500

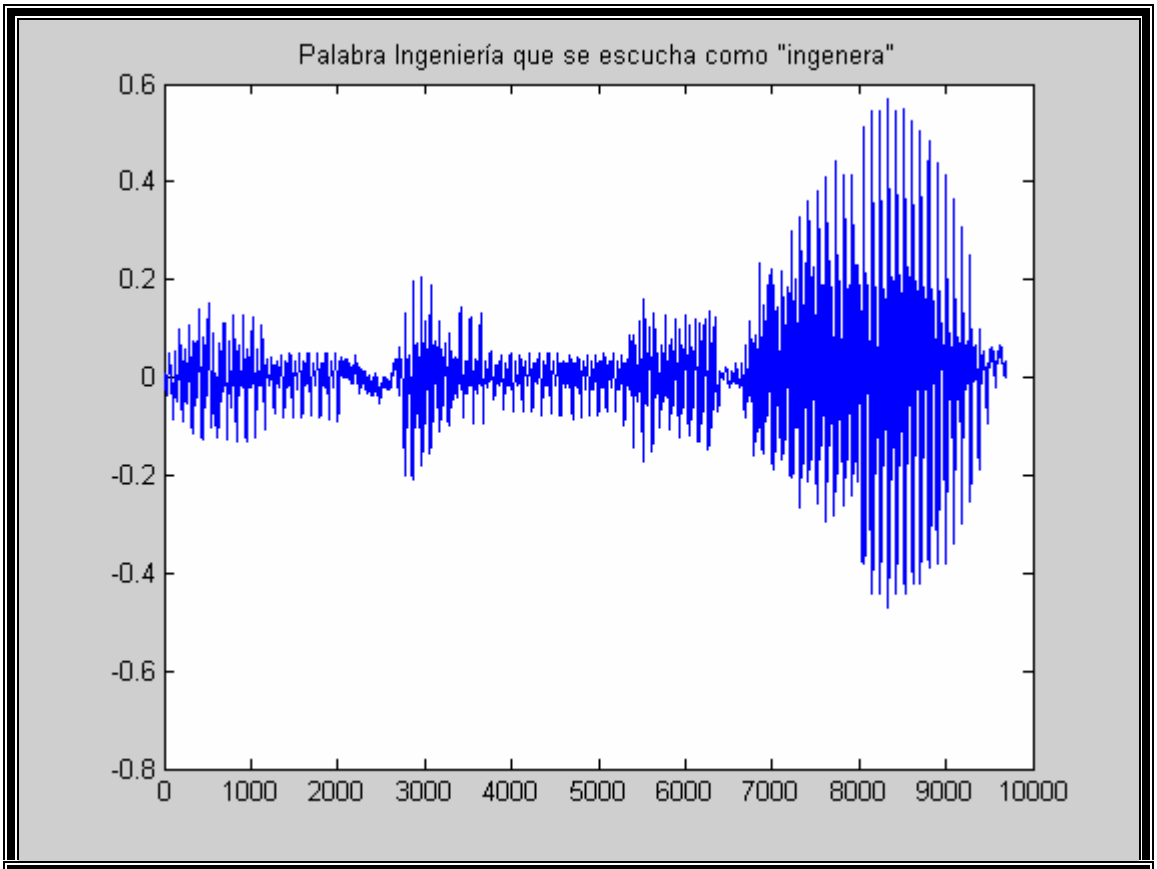

Espectograma de la palabra agua que se escucha como "aga"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5
4
Time

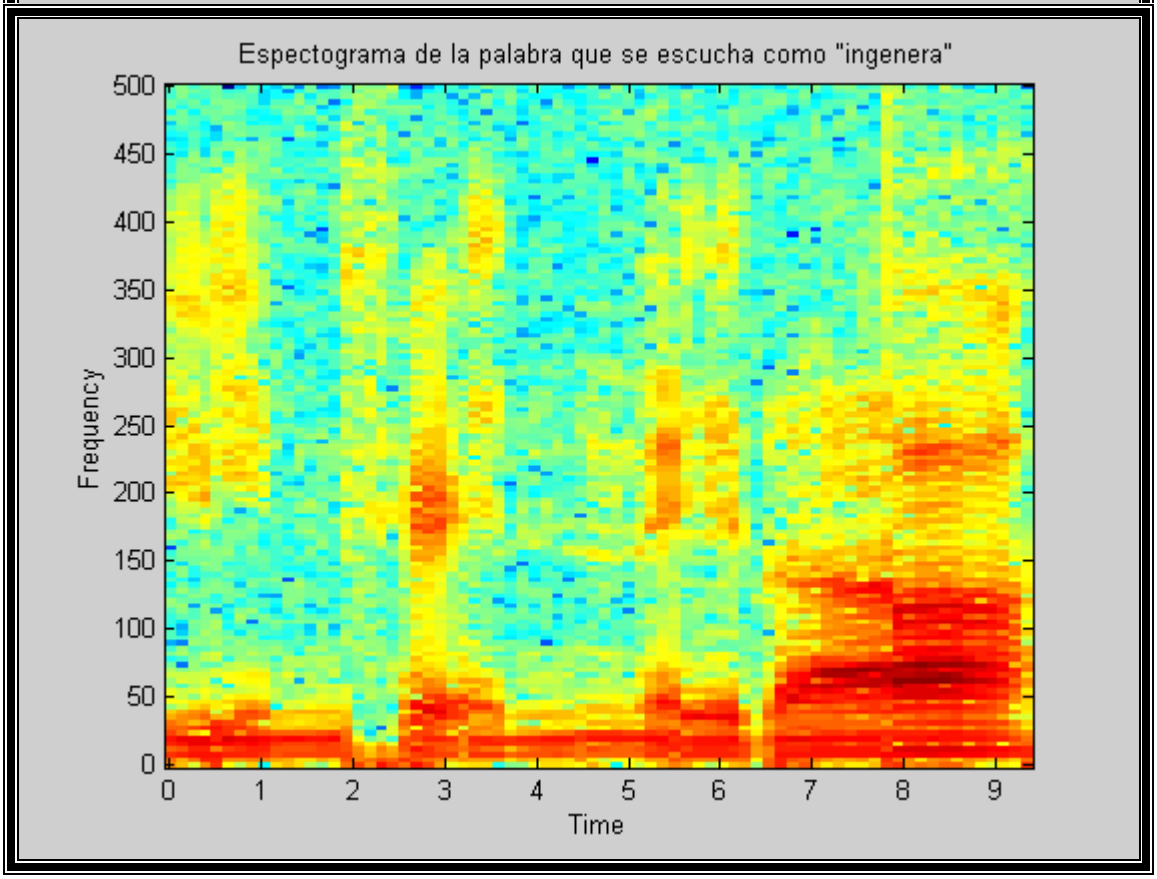
Otros ejemplos, archivos de voz, gráficas y espectrogramas

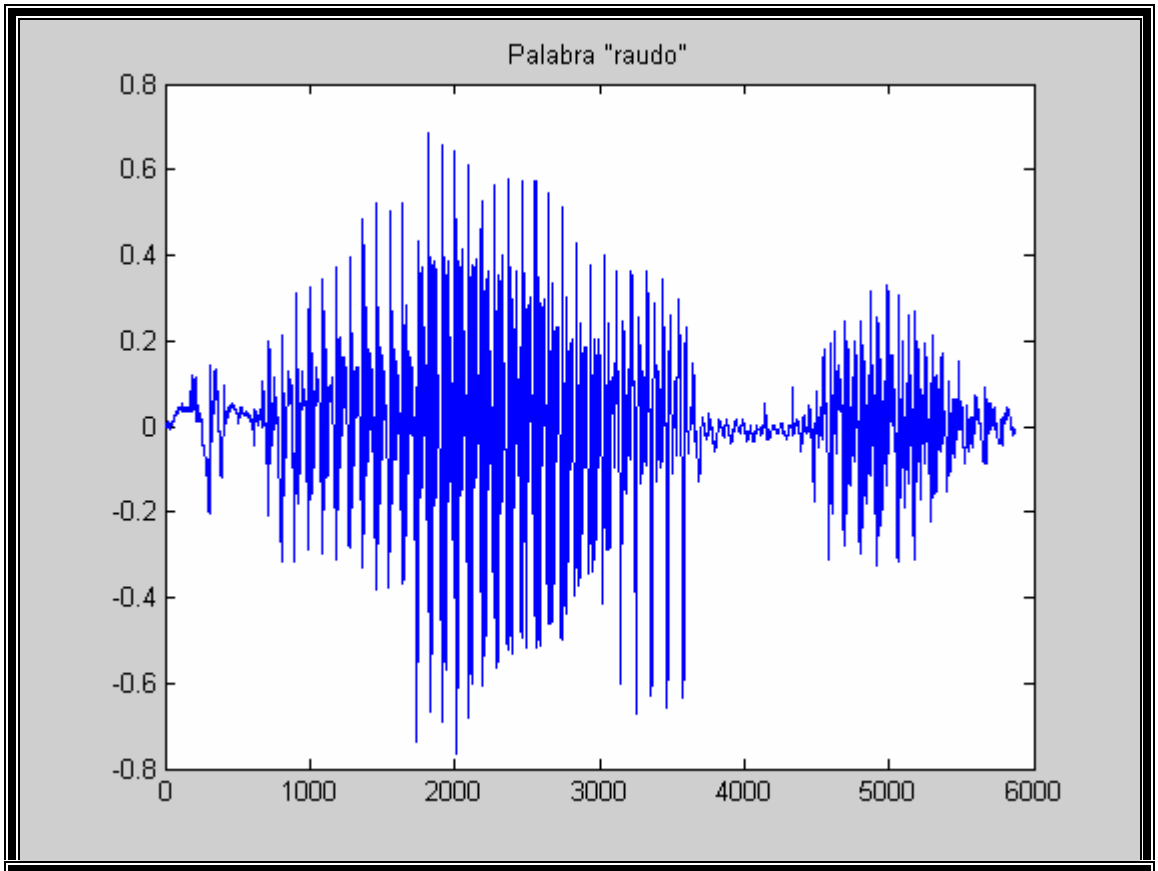
PALABRA	**ARCHIVO DE SONIDO** creado con el programa original	**PALABRA** (se escucha sólo la última sílaba del diptongo)	**ARCHIVO DE SONIDO** creado con el programa modificado
Ingeniería	 ingeniería.wav	**Ingenera**	 ingenera.wav
Raudo	 raudo.wav	**Rudo**	 rudo.wav
Prueba	 prueba.wav	**Preba**	 preba.wav

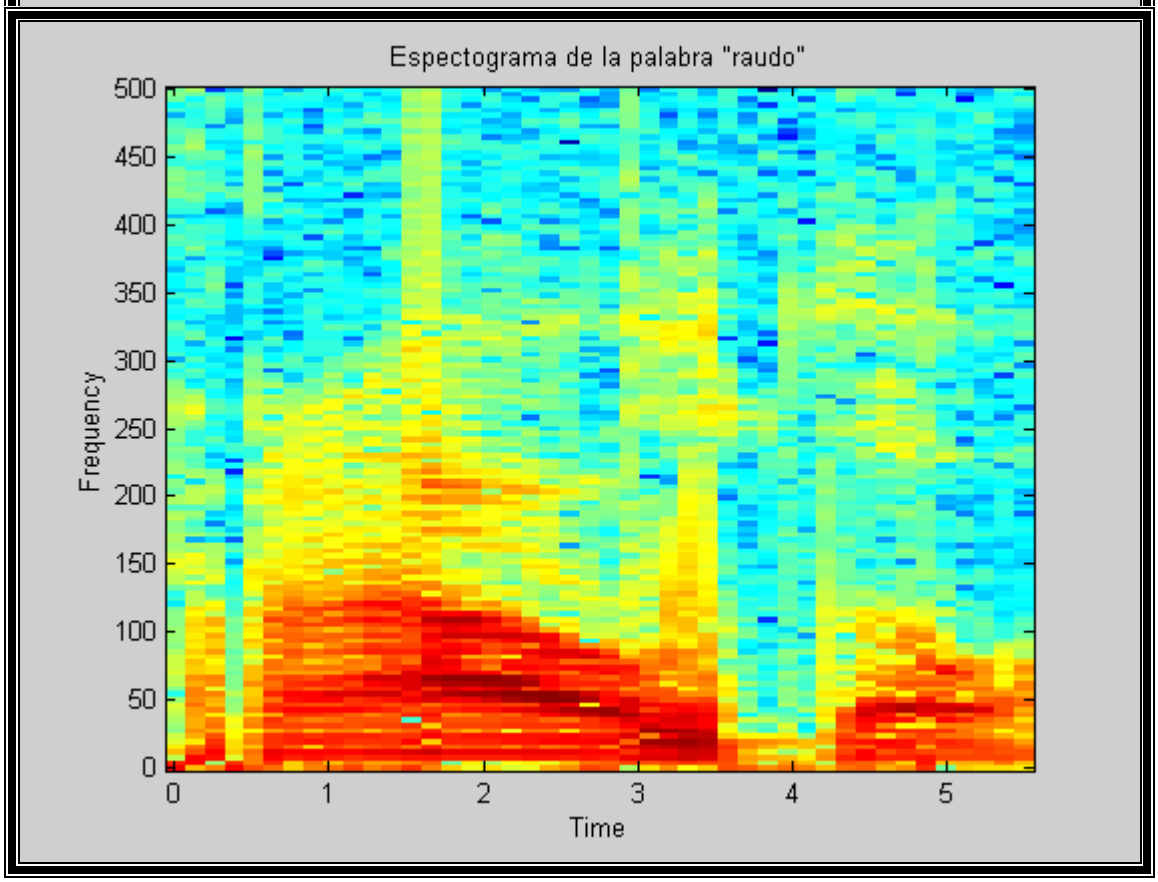

Palabra Ingeniería
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
2000
4000
6000
8000
10000
12000
14000

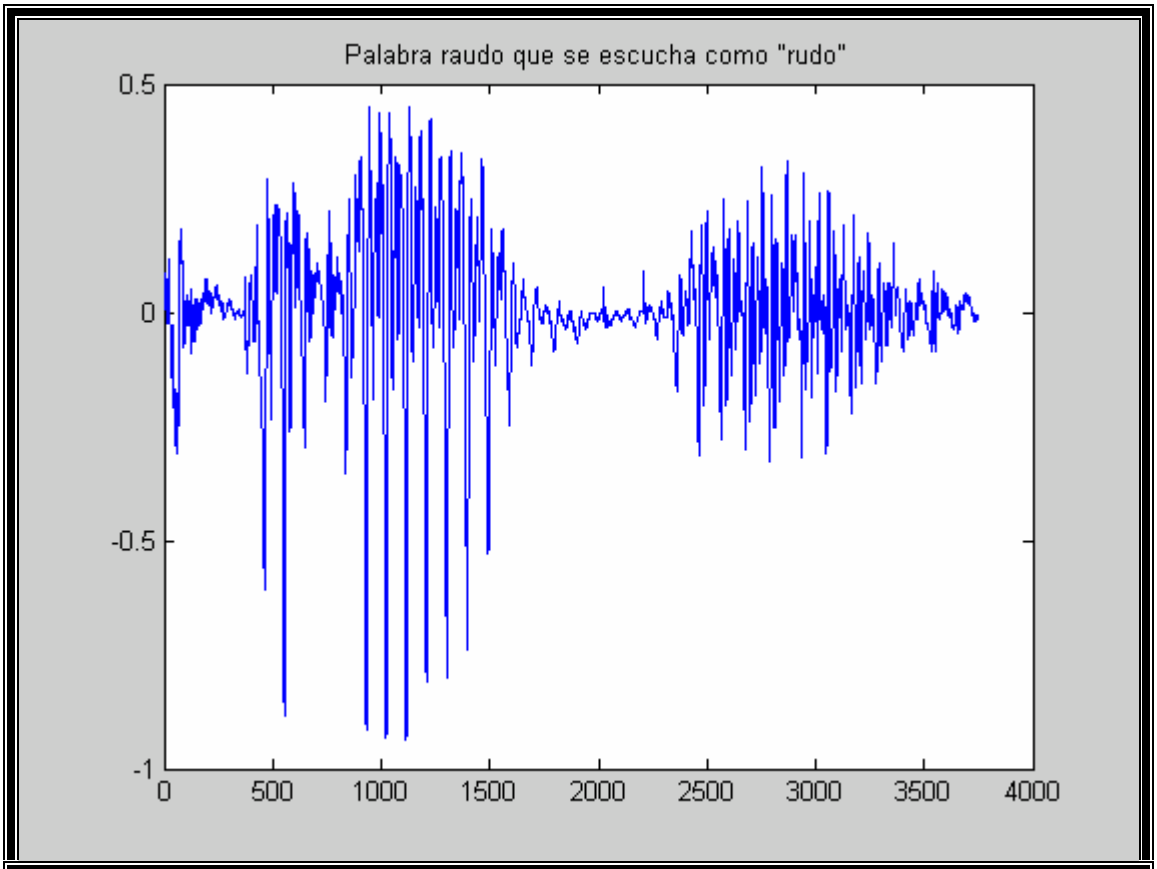

Espectograma de la palabra "Ingeniería"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
2
4
6
8
10
12
Time

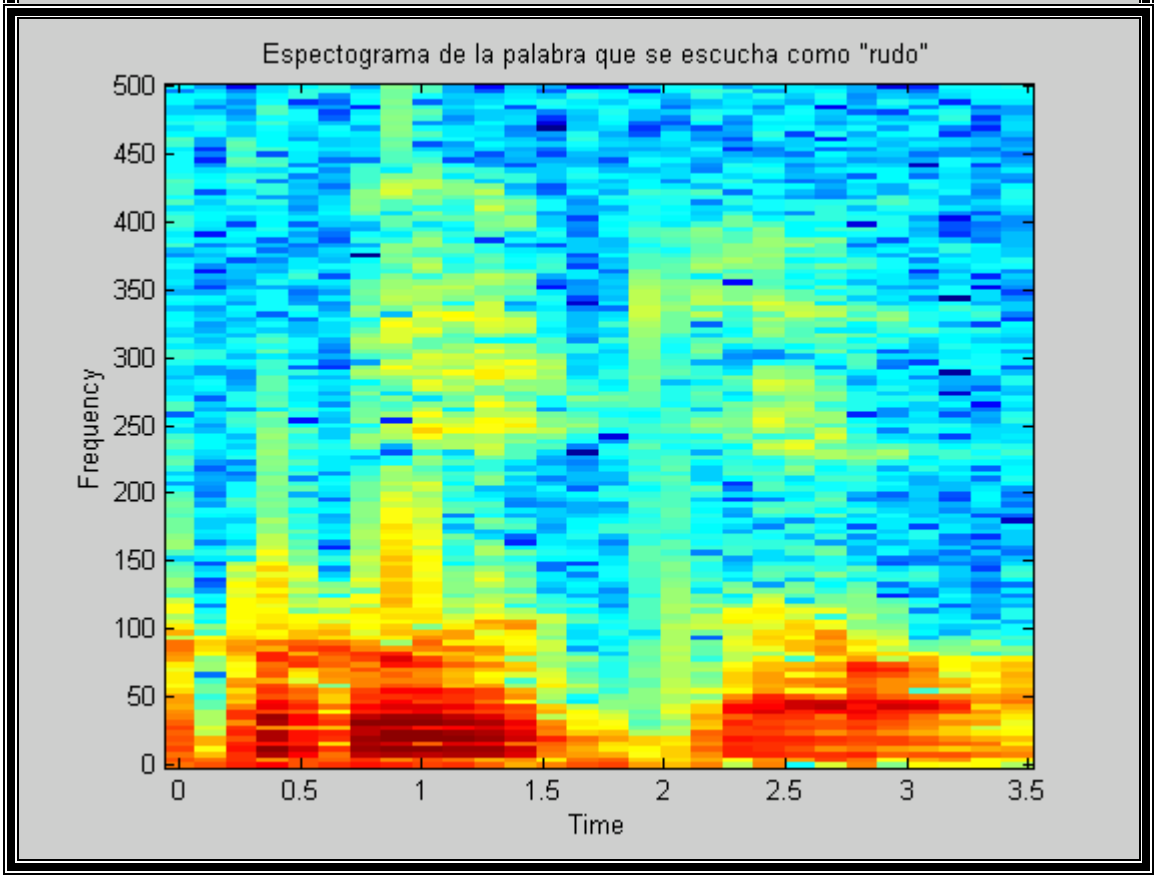

Palabra Ingeniería que se escucha como "ingenera"
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

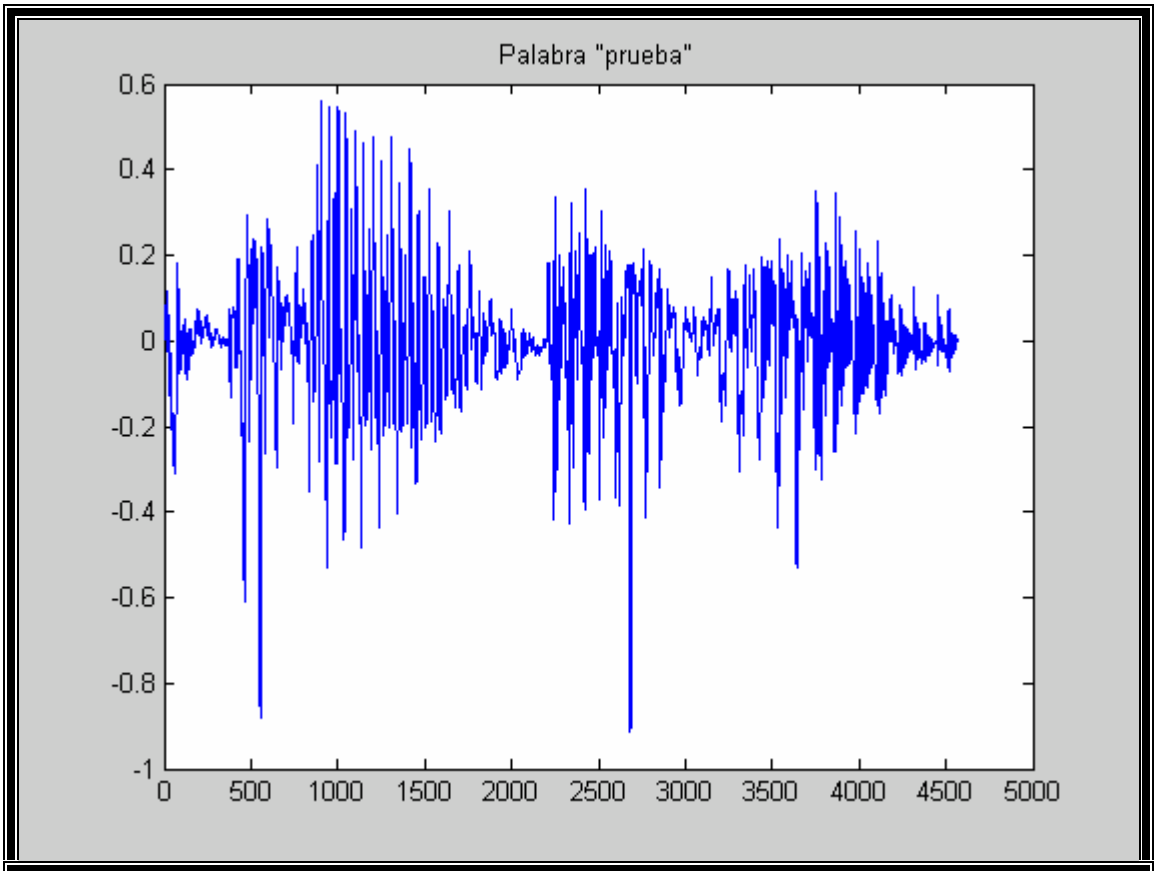

Espectograma de la palabra que se escucha como "ingenera"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
1
2
3
4
5
6
7
8
9
Time

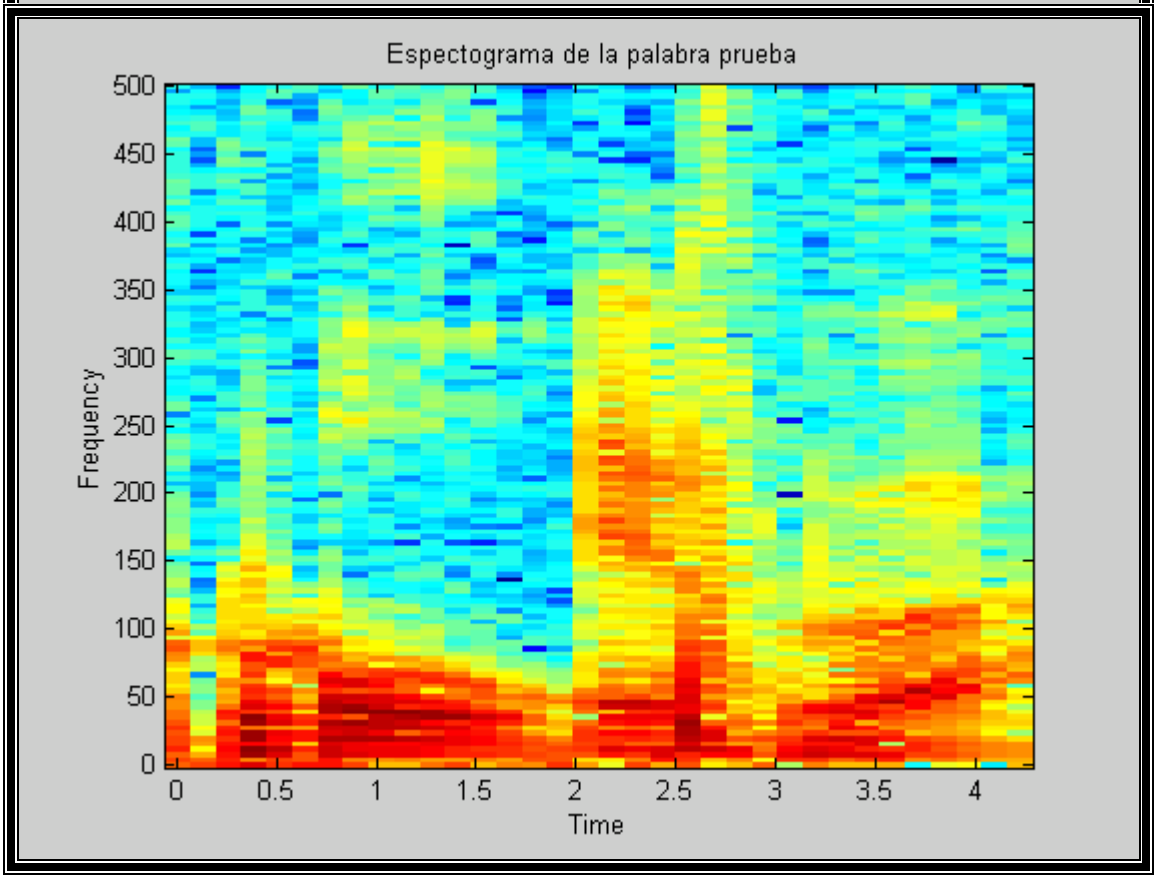

Palabra "raudo"
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
0
1000
2000
3000
4000
5000
6000

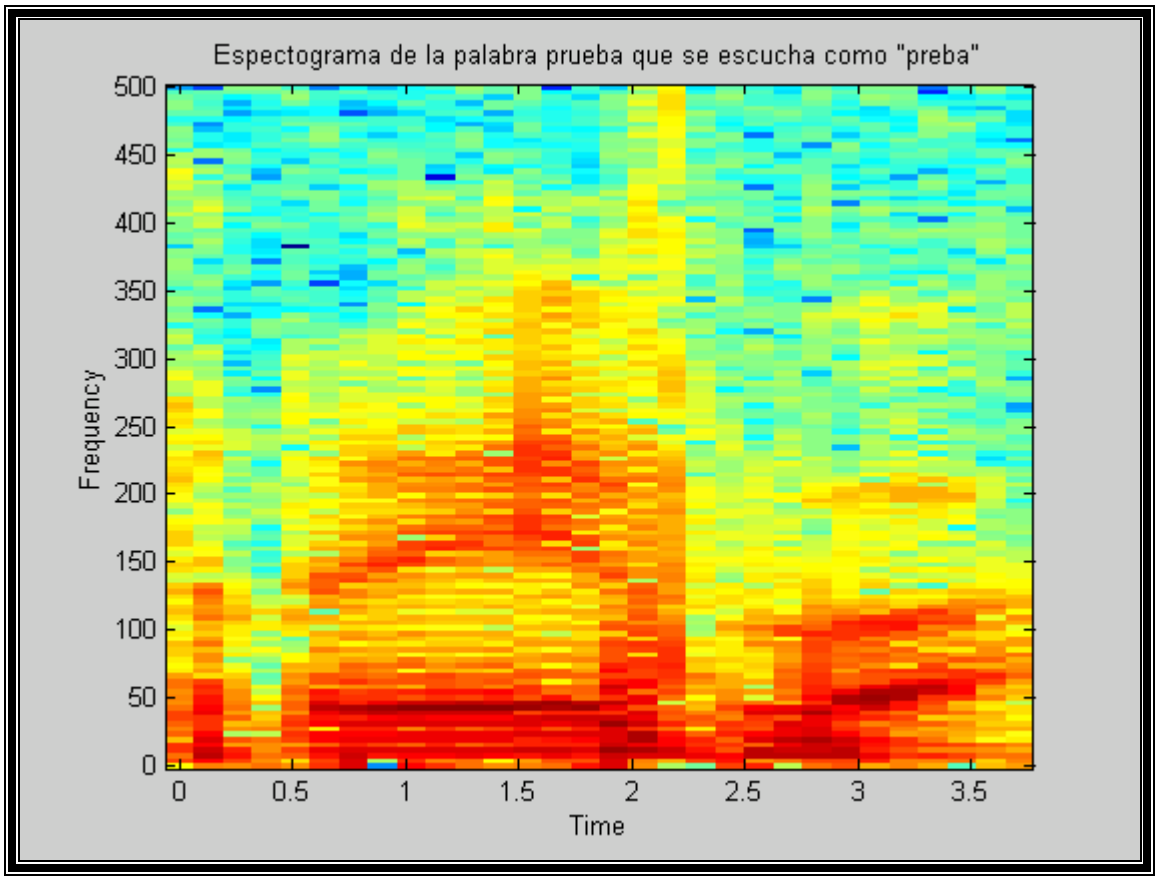

Espectograma de la palabra "raudo"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
1
2
3
4
5
Time


Palabra raudo que se escucha como "rudo"
0.5
0
-0.5
-1
0
500
1000
1500
2000
2500
3000
3500
4000


Espectograma de la palabra que se escucha como "rudo"
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5
Time


Palabra "prueba"
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000


Espectograma de la palabra prueba
Frequency
500
450
400
350
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5
4
Time

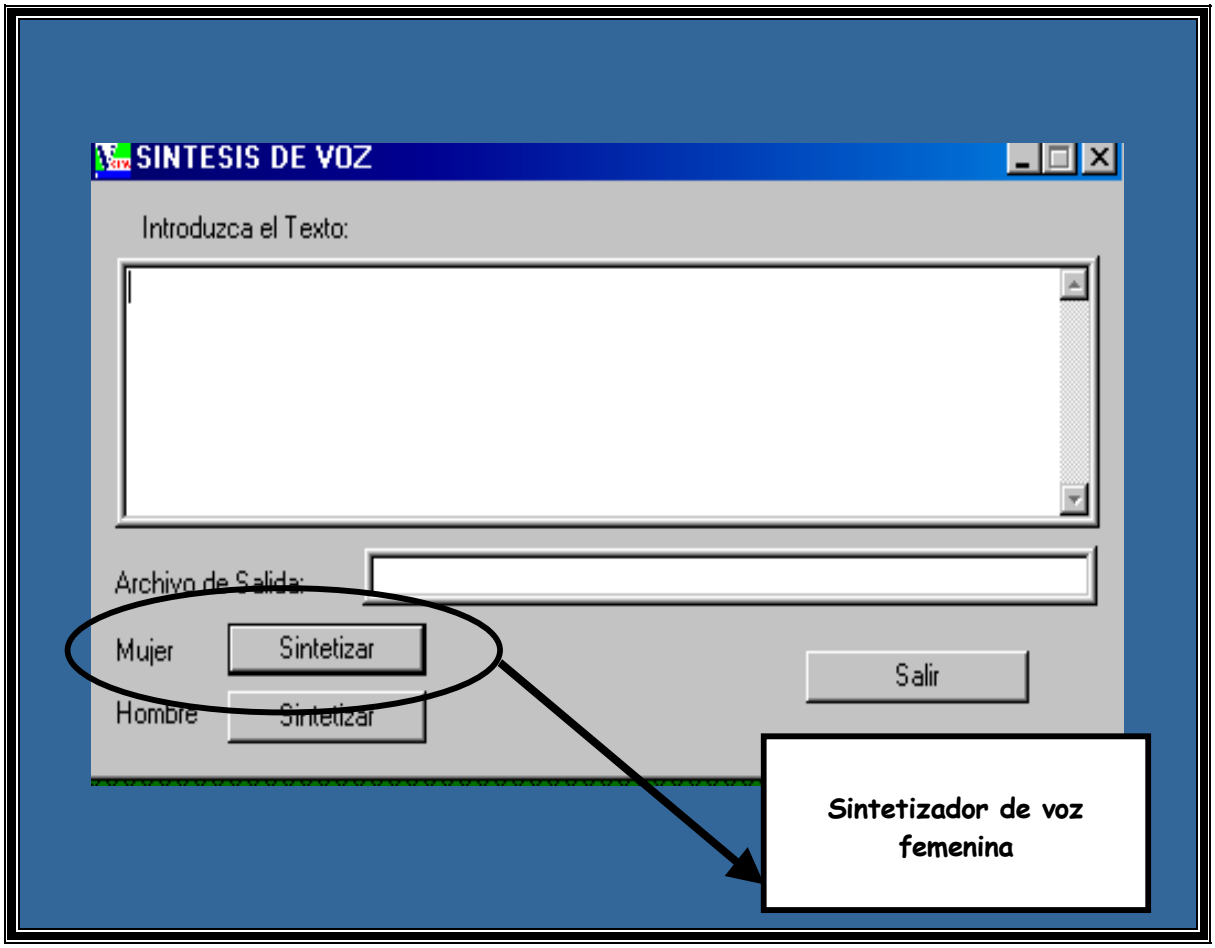

Espectograma de la palabra prueba que se escucha como "preba"
Frequency
Time
500
450
400
350
300
250
200
150
100
50
0
0
0.5
1
1.5
2
2.5
3
3.5

PRUEBA No.3
ADECUACIÓN DEL PROGRAMA PARA HACER REFERENCIA A VOZ FEMENINA

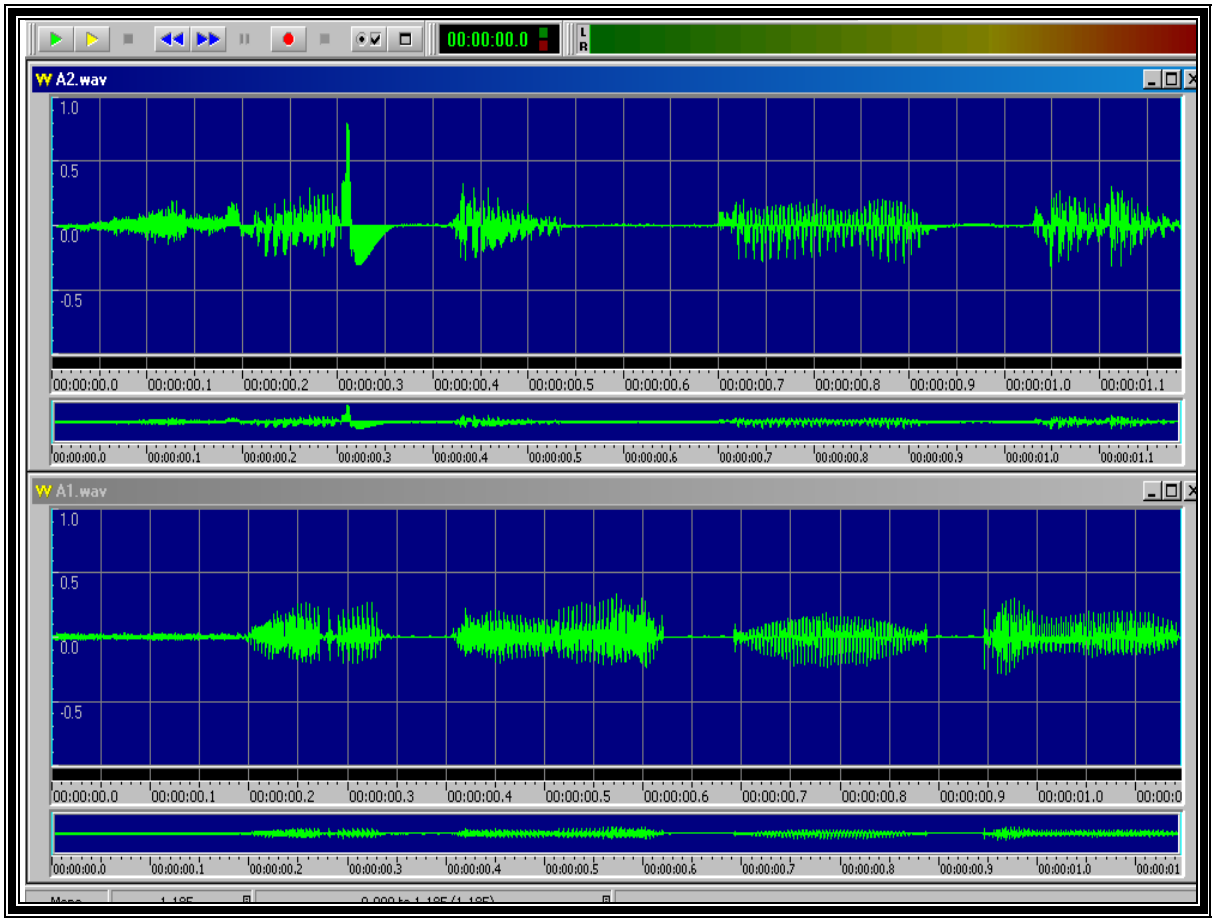
CAMBIOS AL PROGRAMA PRINCIPAL (MARCADOS EN AMARILLO)
ver anexo A

```
…
cadena= "data1\\"+ctemp+ctemp2+".pcm";
if (fsale.Open(cadena,CFile::modeRead ,&e1)) {
leido=fsale.Read(pbuf,30000);
if (!(vocal(l1) & !vocal(l2) & vocal(l3) ))
fentra.Write(pbuf,leido);
tamano=tamano+leido;
fsale.Close();}
else {
cadena="data1\\"+ctemp+"-.pcm";
if (fsale.Open(cadena,CFile::modeRead ,&e1)) {
leido=fsale.Read(pbuf,30000);
fentra.Write(pbuf,leido);
tamano=tamano+leido;
fsale.Close();}
cadena="data1\\-"+ctemp2+".pcm";
if (fsale.Open(cadena,CFile::modeRead ,&e1)) {
leido=fsale.Read(pbuf,30000);
fentra.Write(pbuf,leido);
tamano=tamano+leido;
fsale.Close();}  }  }

fentra.Seek(40,CFile::begin);
fentra.Write(&tamano,4);
tamano=tamano+40;
fentra.Seek(4,CFile::begin);
fentra.Write(&tamano,4);
fentra.Close();
cad1 = (CEdit*) GetDlgItem(IDC_EDIT2);
cad1->GetWindowText(cadena);
if (cadena=="") {cadena="default";}
cadena=cadena+".wav";
PlaySound(archson, NULL, SND_ASYNC | SND_FILENAME);


}
```

Nueva pantalla del programa



VOZ HOMBRE
Primera gráfica
VOZ MUJER
Segunda gráfica



PALABRA "ZAPATITO"	
ARCHIVO DE VOZ HOMBRE LONGITUD 1.18 Seg Archivo de sonido	ARCHIVO DE VOZ MUJER LONGITUD 1.14 Seg Archivo de sonido

5.3 Rediseño del sintetizador de voz por concatenación

Tomando como base el sintetizador realizado en el laboratorio de señales de voz, se rediseñó otro con técnica de síntesis por concatenación con el método PSOLA, teniendo como hipótesis la mejora de la calidad de la voz, la cual se analizará en el apartado 5.4.

Otras adecuaciones fueron la variación de la frecuencia fundamental que se puede seleccionar tanto en voz masculina como femenina.

El programa se realizó en C# dadas las ventajas de programación que éste tiene. **Ver anexo B**

A continuación de describe el funcionamiento del programa, Sintetizador de Voz

Como se mencionó anteriormente, este rediseño fue elaborado en base a un programa ya existente en C++ con el lenguaje de programación C#. Dado a esto a pesar de que la lógica es similar y los archivos de voz pregrabados (en formato PCM) sean los mismos, las funciones difieren en algunos casos por la diferencia del lenguaje aparte de la agregación de diversas características nuevas que no poseía el programa original.

El programa cuenta con un formulario y varias clases que en conjunto logran el cometido de sintetizar una frase escrita, crear un archivo de tipo WAV con dicha frase interpretada por el locutor predeterminado (las opciones disponibles son "hombre" y "mujer"), con la velocidad de pronunciación deseada (rápida, lenta o normal) y lo reproduce. Aunado a esto, después de tener una frase sintetizada el programa cuenta con la opción de visualizar la onda de sonido para tener una más gráfica de dicha frase.

A continuación se enlista un resumen del funcionamiento de cada clase del programa.
La interfaz de entrada y salida que tiene contacto directo con el usuario es un formulario (Form1) que se conforma de los medios visuales necesarios para introducir una frase y sintetizarla con las clases mencionadas con anterioridad.

El nombre de la clase principal es "CSintetizador", esta se encarga de la parte de síntesis del texto entrante. Esta clase para su creación recibe como parámetro la ruta de los archivos de voz que va a utilizar y la velocidad de reproducción.
El objeto de la clase "CSintetizador" se crea y utiliza con la acción "Click" de cualquiera de dos botones, uno que provee al objeto con la ruta de los archivos PCM que contienen difonemas pronunciados por un locutor femenino y el otro que lo provee con la ruta de los archivos PCM que contienen difonemas pronunciados por un locutor masculino. Este toma el texto escrito contenido en un área de texto y lo sintetiza con una velocidad de reproducción que depende de una etiqueta de opción múltiple (comboBox).

Posteriormente para sintetizar la frase se evoca la función "analizar" que recibe como parámetro una cadena de caracteres que contiene la frase en cuestión. Dicha función divide las palabras de los números, y las envía a diferentes funciones.

En el caso de las palabras, la función las manda a otras funciones que se encargan de evaluar su acentuación y de comparar si son palabras especiales que tengan otro significado (abreviaturas) y de ser así cambiar dicha abreviatura por una palabra completa que le corresponda. En el caso de los números a una función que los traduce de dígito a cadena de caracteres (ej. Cambia del dígito "1" a la cadena ya acentuada "úno"), de contener un punto entre números se encarga también de crear una frase de números con decimales.

Posterior a la separación, la función concatena todas las palabras ya acentuadas y los números traducidos a palabras acentuadas, manda a llamar a la función "creararchivo" que tiene como entrada la cadena creada en esta función, y por ultimo regresa dicha cadena como parámetro de salida.

La función de "creararchivo" se encarga de cambiar las palabras acentuadas a difonemas, de buscar los archivos PCM que contengan dichos difonemas, de filtrarlos, de crear el archivo WAV de salida y de reproducirlo.

Para todo esto se ayuda de un objeto que tiene como tipo una de las otras clases creadas en este programa para buscar los archivos PCM, recortar los extremos con vacío sonoro, crear la trama de cabecera del archivo WAV

tomando en cuenta la velocidad de reproducción y concatenar todo esto para crear un archivo WAV.

Para formar los difonemas a buscar, la función toma en cuenta tres casos diferentes, los cuales valora de manera jerárquica (tomando la unión de dos letras, un espacio con la primera letra o la segunda letra y un espacio) y posteriormente los manda a la parte de concatenación. La concatenación de los archivos PCM se ejecuta utilizando un ciclo repetitivo desde la función de "creararchivo" y otro en la clase "CWav". A la vez se utiliza un leve filtro para normalizar ligeramente la información encontrada en los archivos PCM.

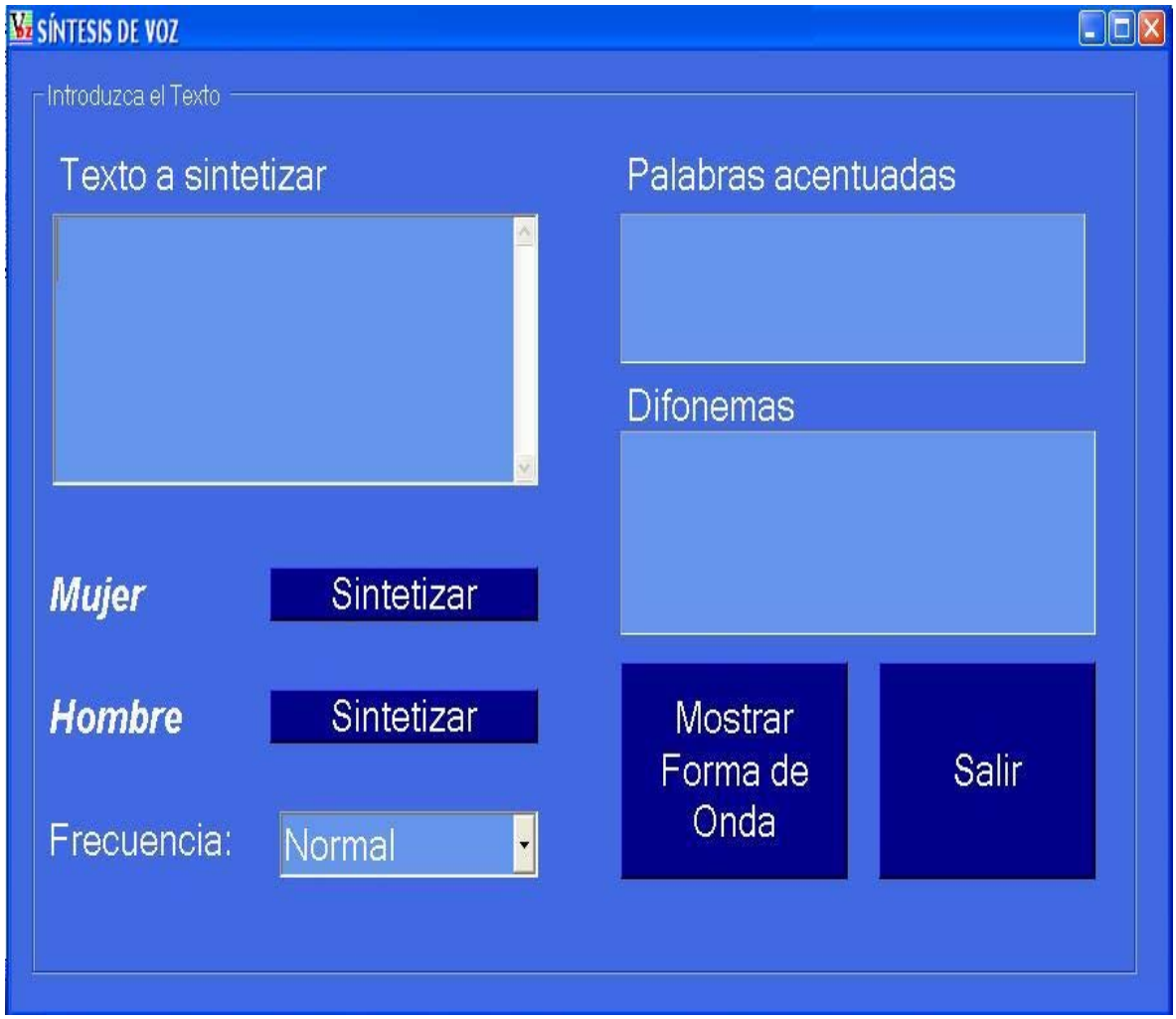
Al final esta función crea un objeto de tipo "Sound" (otra clase definida en este proyecto) que mediante una de sus funciones abre y reproduce finalmente el archivo WAV. La función termina regresando como parámetro de salida una cadena de caracteres que contiene los difonemas de la frase original obtenidos en esta función.

Posteriormente, se recopila la información de las palabras acentuadas y de los difonemas creados y se muestra en dos etiquetas en la ventana principal (Form1). Aunada a la sinterización y reproducción del archivo de sonido, el sintetizador cuenta con la opción de visualizar la forma de la onda sonora creada con un tercer botón. Este botón manda a llamar a un segundo formulario y le da como parámetro de entrada el nombre del archivo recién creado (en caso de que se presione el botón sin crear el archivo con anterioridad, el programa avisa sobre este hecho y no hace nada).

El nuevo formulario (Form2) en su evento "Form2_Paint", crea un objeto de tipo de otra clase definida en este proyecto que se encarga de leer el archivo de sonido WAV y de transformar la información contenida en este de manera que quede una secuencia de números que llevan la proporción de los decibeles a los que corresponden. Esta información se dibuja en las gráficas de la ventana y se crea la onda sonora correspondiente al archivo WAV que se recibió como dato de entrada, lo que ayuda a visualizar la frase analizada de manera gráfica.

Por ultimo, en la ventana inicial (Form1) se encuentra el botón de salida, el cual termina con la aplicación.

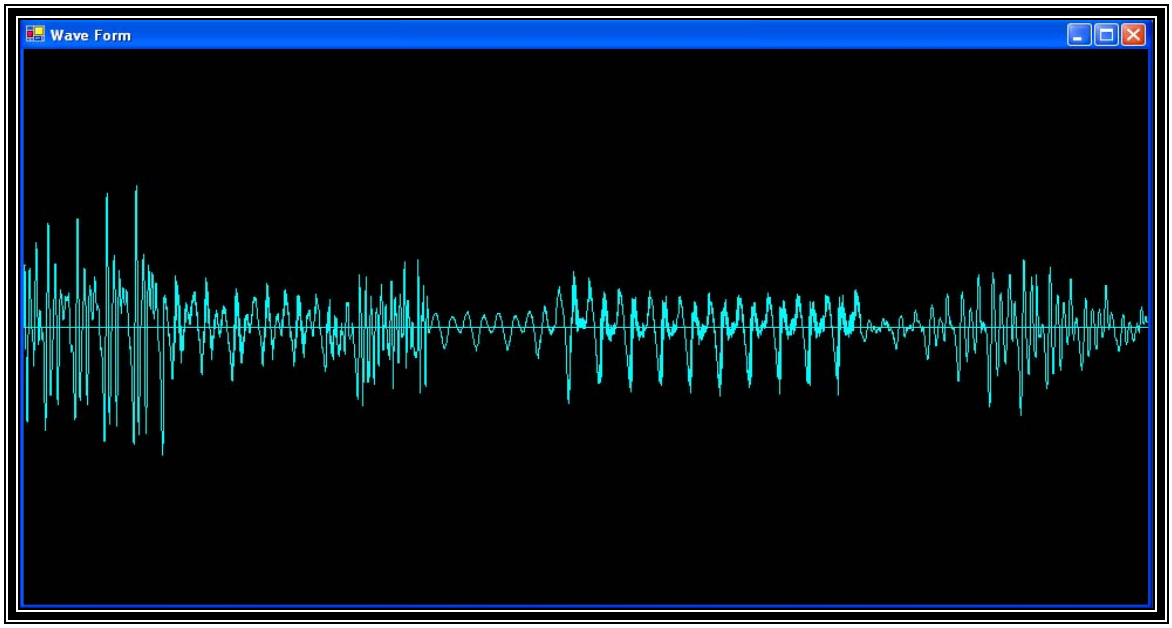
Ventana del Programa



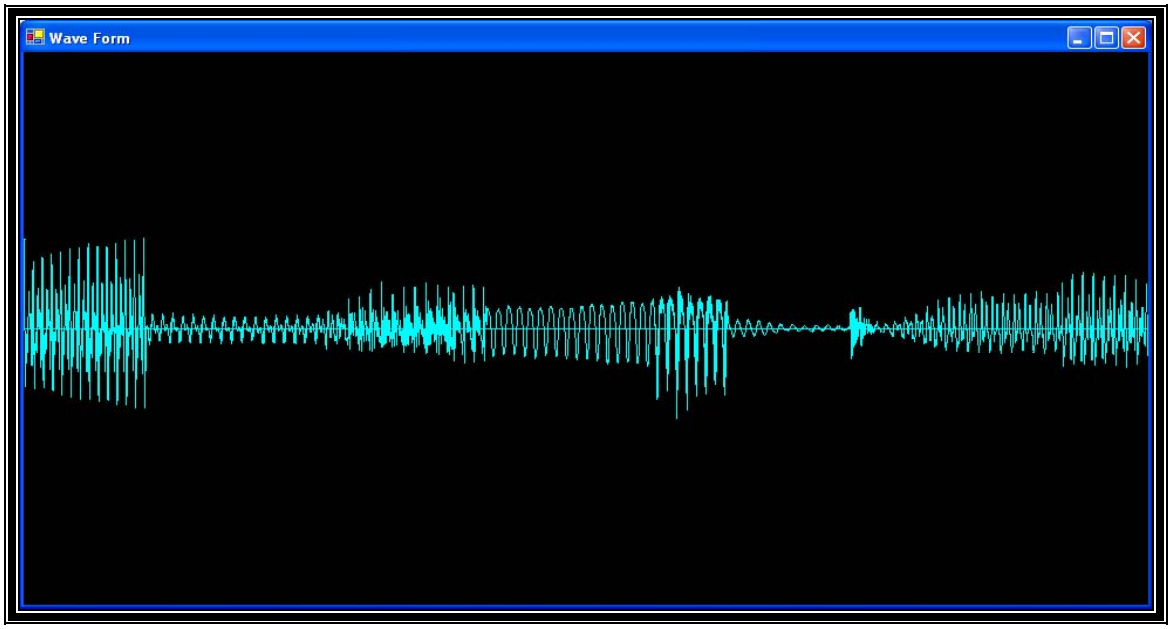
Ventana del Programa corriendo "hola amigo"

Ventana del Programa opciones de variación de frecuencia

Forma de Onda (Wave Form, en inglés) de "hola amigo" con voz de hombre



Forma de onda (Wave Form) de "hola amigo" con voz de mujer



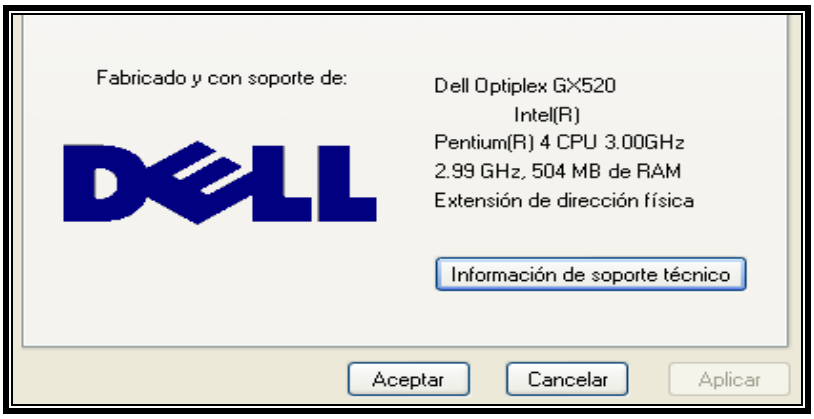
5.4 Evaluación de los sistemas de síntesis, "Pruebas MOS"

Como se mencionó en el apartado 4.2, los parámetros de diseño consideran como punto indispensable la evaluación del sistema de síntesis, para ello se toma como medidor un estándar conocido como pruebas MOS, por sus siglas en inglés Mean Opinion Store. Estas pruebas arrojan índices numéricos calculados como un promedio en un rango de 1 a 5, que están regidos bajo estándares preestablecidos, siendo el 1 la calidad más baja y 5 la más alta.
Estas pruebas están estandarizadas por la ITU-T, por sus siglas en inglés "The ITU Telecommunication Standardization Sector" y éstas a su vez se soportan por la ITU International Telecommunication Union. Hasta 1992, era conocida como CCITT, Comité Consultatif International Téléphonique et Télégraphique y que en inglés se conoció como International Telegraph and Telephone Consultative Comitte.
Los índices de calidad se calculan como un promedio aritmético que resulta de sumar todos los resultados individuales entre el número total de evaluadores; se califican un conjunto de pruebas subjetivas donde las personas que escuchan el audio dan un valor entre 1 (malo) y 5 (excelente) a las frases que se pongan a prueba.
Dado que son dos sintetizadores, se evaluaron bajo los mismos lineamientos; aunque los evaluadores no saben cual es el "programa original", para ellos sólo conocido como PROGRAMA 1 y "el rediseñado", mencionado como PROGRAMA 2. La tabla siguiente fue mostrada a cada evaluador para calificar el desempeño de los programas.

Mean Opinion Score (MOS)		
MOS	**CALIDAD**	**¿COMO SE ESCUCHA?**
5	**Excelente**	**Perceptible**
4	**Bueno**	**Perceptible pero no deteriorado**
3	**Regular**	**Ligeramente deteriorado**
2	**Pobre**	**deteriorado**
1	**Malo**	**Muy deteriorado**

Se consideraron 30 evaluadores escogidos al azar (estudiantes, profesores universitarios, amas de casa, secretarias, entre otros), cabe aclarar que estas personas desconocen completamente este tipo de trabajo.
Las pruebas se llevaron a cabo en dos aulas diferentes, una sesión se llevó a cabo en forma grupal (aprox. 15 estudiantes) y la otra sesión en forma

individual, mostrando en pantalla los programas y escribiendo las frases una por una con cada programa; corriéndose en un equipo de cómputo PC DELL de las siguientes características, con bocinas externas tipo "multimedia speaker system" con opción de audífonos.



Los puntos a evaluar son:

- Naturalidad
- Inteligibilidad del habla sintetizada
- Versatilidad del sistema: Reproducción completa de la frase, voz de hombre y mujer, variedad de frecuencia fundamental

Las 14 frases de prueba son:

1. **Hola amigo**
2. **Dónde vives**
3. **Por qué has tardado tanto**
4. **Cierra la puerta por favor**
5. **Guarda silencio**
6. **El día de hoy estoy cansado**
7. **Mi casa está cerca de la avenida 1000**
8. **Siéntate bien**
9. **Las computadoras sirven mucho**
10. **Revisa este escrito pronto**
11. **Localiza a las 50 personas**
12. **Tu trabajo 56.37 ha concluido**
13. **Los carros son rojos**
14. **El canta bien**

PROGRAMA #__	**CALIFICACIÓN 1 (MALO) A 5 (EXCELENTE)**

Parámetro → Frase ↓	NATURALIDAD	INTELIGIBILIDAD DEL HABLA SINTETIZADA	VERSATILIDAD DEL SISTEMA: REPRODUCCIÓN COMPLETA DE LA FRASE		
			Frase completa	Voz de hombre y mujer	Variedad de frecuencia fundamental
Hola amigo					
Dónde vives					
Por qué has tardado tanto					
Cierra la puerta por favor					
Guarda silencio					
El día de hoy estoy cansado					
Mi casa está cerca de la avenida 1000					
Siéntate bien					
Las computadoras sirven mucho					
Revisa este escrito pronto					
Localiza a las 50 personas					
Tu trabajo 56.37 ha concluido					
Los carros son rojos					
El canta bien					

5.5 Análisis y Resultados

Cada evaluador calificó las frases con cada uno de los tres parámetros solicitados y además lo realizó para cada uno de los programas.
Se realizó un promedio aritmético como se explicó anteriormente de cada parámetro y a continuación se muestran los resultados en las tablas 5.1, 5.2 y 5.3

Tabla 5.1a

Naturalidad Programa 1																													
Oyente	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Hola amigo	3	4	2	3.2	2.9	3	2.8	3	2	3	3.1	2	3	3	3	2	3.2	3	3.1	2	4	3	4.3	4	3	4	3	3.3	4	2.8
Donde vives	2	4	3	3.2	2.8	3	2.9	3.5	3	3	3.2	2	3	3	3	2	3.2	3	3	2	4	3	4	4	3	4	3	3.6	4	2.9
Por que has tardado tanto	3	4	3	3.2	2.8	3	2.9	3	3	3	3.8	2	3	3	3	2	3.1	3	3	2	4	3	4	4	3	4	3	3.2	4	3
Cierra la puerta por favor	3	4	3	3.2	2.7	3	2.9	3.3	2	3	3	2	3.1	2.5	3	2	3	3	3	2	4	3	4	4	3.1	4	3	3.7	4	3.1
Guarda silencio	5	4	2	4	2.9	3	3	3.4	4	3	3	2	3	3	3	2	3.2	3	3.3	2	4	3	4	4	3	4	3	3.7	4	3.1
El día de hoy estoy cansado	4	4	4	4	2.5	3	3	3.5	5	3	3	2	2.9	2.5	3	2	3	3	3	2	4	3	4	4	3	4	3	3.1	4	2.8
Mi casa esta cerca de la avenida mil	4	4	4	4	2.3	3	3.1	3.5	4	3	3.4	2	3	2.5	3	2	2.9	3	3	2	4	3	4	4	3.2	4	3	3	4	2.7
Siéntate bien	5	4	3	5	2.8	3	3.5	3.5	5	3	3.6	2	3	3	3	2	3	3	3	2	4	3	4.1	4	3	4	3	3.2	4	3
Las computadoras sirven mucho	3	4	2	3	2.5	3	3	3.5	3	3	3.9	2	2.9	3	3	2	2.9	3	3	2	4	3	4	4	3.1	4	3	3.3	4	3
Revisa este escrito pronto	3	4	4	3	2.3	3	3.2	3.4	2	3	3	2	3.7	2.5	3	2	3	3	3	2	4	3	4	4	3	4	3	3.5	4	2.9
Localiza a las 50 personas	4	4	3	3	2.2	2	2.8	2.5	4	3	2	2	2	2	3	2	2.5	3	3	2	4	3	4	4	2.5	4	3	3.1	4	1
Tu trabajo 56.37 ha concluido	4	4	3	3	2.2	2	2.5	2.5	4	3	2	2	2	2	3	2	2.5	3	3	2	4	3	4	4	2.5	4	3	3.1	4	1
Los carros son rojos	4	4	4	3	2.3	3	3.3	3.5	5	3	3.8	2	3	2	3	2	2.9	3	3.5	2	4	3	4.5	4	3	4	3	3.2	4	3
El canta bien	4	4	4	3	2.4	3	3	3.8	4	3	3.2	2	3	2.5	3	2	2.9	3	3.1	2	4	3	4	4	3	4	3	3.2	4	3

Tabla 5.1b

Naturalidad Programa 2																														
Oyente	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Hola amigo	3	4.5	3	3	4	3	3.9	4	3	3	3.5	3	3.5	4	3	2	3	3	3	2	4	3	4	4	3	4	3	3.4	4	3
Donde vives	2	4.5	3	3	4	3	3.7	4.2	3	3	3.5	3	3.5	4	3	2	3.5	3	3.1	2	4	3	4	4	3	4	3	3.6	4	3
Por que has tardado tanto	4	4.5	3	3.2	4	3	3.6	4.1	4	3	4	3	3.5	4.1	3	2	3	3	3	2	4	3	4	4	3	4	3	3.3	4	3.1
Cierra la puerta por favor	4	4.5	3	3	4	3	4	3.9	5	3	3.4	3	3.5	3	3	2	3	3	3	2	4	3	4	4	3.1	4	3	3.8	4	3
Guarda silencio	4	4.5	5	3	4	3	3.7	4.2	5	3	3.4	3	3.5	4	3	2	3	3	3	2	4	3	4	4	3	4	3	3.7	4	3
El día de hoy estoy cansado	4	4.5	4	3	4	3	3.2	4	5	3	3.6	3	3.5	3	3	2	3.2	3	3.3	2	4	3	4	4	3	4	3	3.1	4	3.2
Mi casa esta cerca de la avenida mil	5	4.5	4	3.1	4	3	3.2	4	5	3	4	3	3.5	3	3	2	3	3	3	2	4	3	4	4	3.2	4	3	3.1	4	3.1
Siéntate bien	5	4.5	5	2.8	4	3	3.7	4	4	3	4.1	3	3.5	3	3	2	3	3	3	2	4	3	4	4	3	4	3	3.1	4	3.3
Las computadoras sirven mucho	5	4.5	3	3	4	3	3.4	4.2	5	3	4	3	3.5	3	3	2	3	3	3.1	2	4	3	4	4	3.1	4	3	3.3	4	3.1
Revisa este escrito pronto	4	4.5	3	3	4	3	3.5	4.2	5	3	4	3	3.5	4.5	3	2	3.5	3	3	2	4	3	4	4	3	4	3	3.5	4	3.2
Localiza a las 50 personas	4	4.5	3	3.5	4	3	3.9	4.5	5	3	4	3	3.5	4	3	2	3.6	3	3	2	4	3	4	4	3	4	3	3.3	4	3
Tu trabajo 56.37 ha concluido	4	4.5	4	3.5	4	3	3.9	4.5	4	3	4	3	3.5	4.2	3	2	3.6	3	3.2	2	4	3	4	4	3	4	3	3.3	4	3
Los carros son rojos	5	4.5	4	4	4	3	4	4.2	5	3	4.5	3	3.5	4	3	2	5	3	5	2	4	3	5	4	3.9	5	3	3.9	4	5
El canta bien	5	4.5	4	4	4	3	4	4.2	5	3	4.1	3	3.5	4	3	2	3	3	3	2	4	3	4.1	4	3	4	3	3.2	4	4.1

Tabla 5.1c

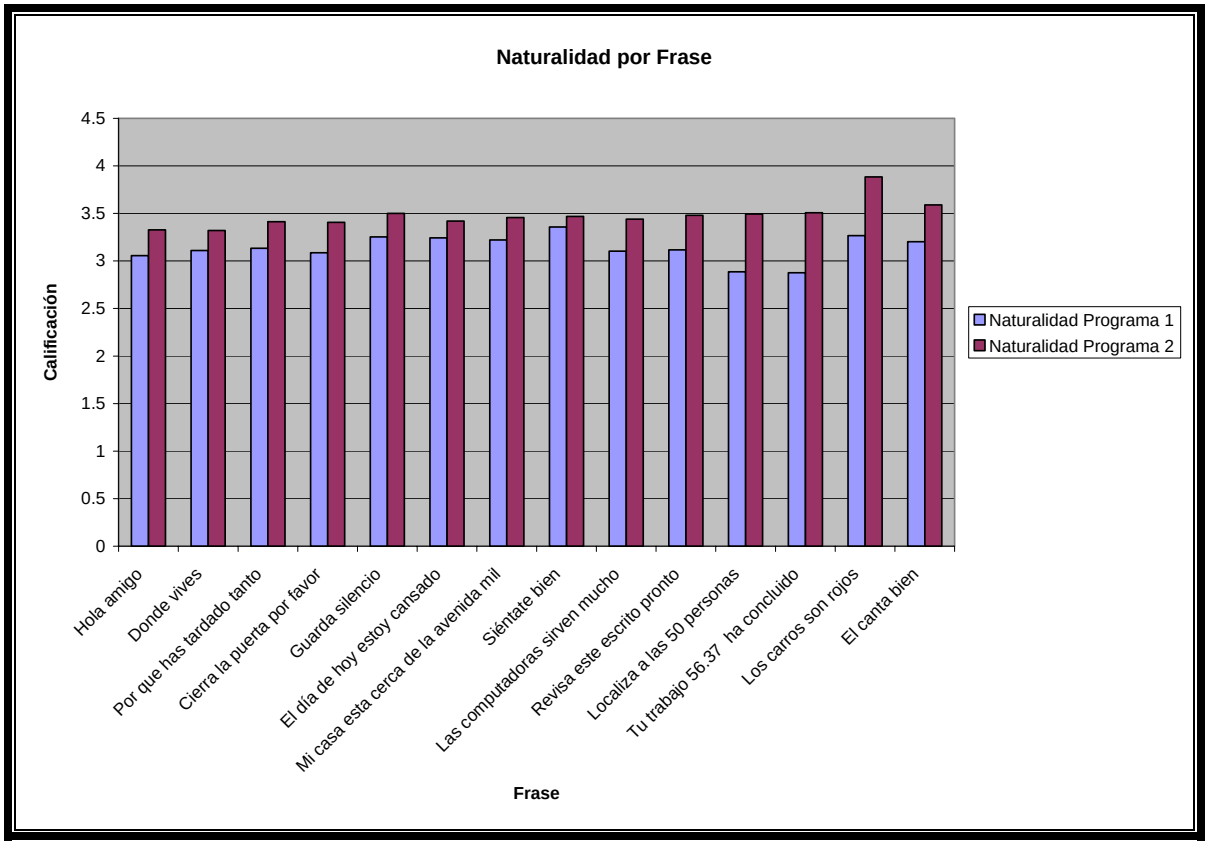


Tabla 5.2a

Inteligibilidad Programa 1																													
Oyente	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Hola amigo	5	4	4	4	5	4	4.5	5	4	4	5	4	5	4.2	5	3	4	3	3.1	4	4	5	4.3	4	3	4	4	3.9	5	4.1
Donde vives	5	4	5	4	5	4	4.5	5	5	4	5	4	5	4.2	5	3	4	3	3.2	4	4	5	4	4	3	4	4	3.9	5	4.2
Por que has tardado tanto	5	4	5	4	5	4	4.5	5	5	4	5	4	5	4.2	5	3	4	3	3	4	4	5	4	4	3	4	4	3.8	5	4
Cierra la puerta por favor	5	4	5	4	5	4	4.5	5	5	4	5	4	5	4.2	5	3	4	3	3	4	4	5	4	4	3.1	4	4	3.9	5	4
Guarda silencio	5	4	5	4	5	4	5	5	5	4	5	4	5	4.2	5	3	4	3	3	4	4	5	4	4	3	4	4	3.9	5	4
El día de hoy estoy cansado	3	4	4	4	5	4	4.5	5	4	4	5	4	5	4.2	5	3	4	3	3	4	4	5	4	4	3	4	4	3.8	5	4.1
Mi casa esta cerca de la avenida mil	4	4	4	4	5	4	4.5	5	5	4	5	4	5	4.2	5	3	4	3	3	4	4	5	4	4	3.2	4	4	3.7	5	4
Siéntate bien	4	4	5	4	5	4	5	5	5	4	5	4	5	4.2	5	3	4	3	3.1	4	4	5	4.1	4	3	4	4	3.8	5	4.2
Las computadoras sirven mucho	4	4	4	4	5	4	4.2	5	4	4	5	4	5	4.2	5	3	4	3	3.1	4	4	5	4	4	3.1	4	4	3.9	5	4
Revisa este escrito pronto	4	4	5	4	5	4	4.2	5	5	4	5	4	5	4.2	5	3	4	3	3	4	4	5	4	4	3	4	4	3.9	5	4
Localiza a las 50 personas	4	4	5	4	5	4	3.4	4.5	5	4	5	4	5	4.2	5	3	3.1	3	3	4	4	5	4	4	2.5	4	4	3.7	5	1
Tu trabajo 56.37 ha concluido	5	4	5	4	5	4	3.4	4.5	5	4	5	4	5	4.2	5	3	3.1	3	3.2	4	4	5	4	4	2.5	4	4	3.9	5	1
Los carros son rojos	4	4	5	4	5	4	4.5	5	5	4	5	4	5	4.2	5	3	4	3	3.2	4	4	5	4.5	4	3	4	4	3.9	5	4
El canta bien	4	4	4	4	5	4	5	5	4	4	5	4	5	4.2	5	3	4	3	3	4	4	5	4	4	3	4	4	3.8	5	4.1

Tabla 5.2b

Inteligibilidad Programa 2																													
Oyente	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Hola amigo	3	4.5	5	4	5	4	4.5	5	3	4	5	4.5	5	4.5	5	3	4	3	3	4	4	5	4	4	3	4	4	3.9	5	4.1
Donde vives	2	4.5	5	4	5	4	5	5	3	4	5	4.5	5	4.5	5	3	4	3	3	4	4	5	4	4	3	4	4	3.9	5	4.2
Por que has tardado tanto	4	4.5	4	4	5	4	5	5	4	4	5	4.5	5	4.2	5	3	4	3	3	4	4	5	4	4	3	4	4	3.8	5	4
Cierra la puerta por favor	4	4.5	5	4	5	4	5	5	5	4	5	4.5	5	4.5	5	3	4	3	3	4	4	5	4	4	3.1	4	4	3.9	5	4
Guarda silencio	4	4.5	5	4	5	4	4.8	5	5	4	5	4.5	5	4.6	5	3	4	3	3	4	4	5	4	4	3	4	4	3.9	5	4
El día de hoy estoy cansado	4	4.5	3	4	5	4	5	5	5	4	5	4.5	5	4.5	5	3	4	3	3	4	4	5	4	4	3	4	4	3.9	5	4
Mi casa esta cerca de la avenida mil	5	4.5	3	4	5	4	5	5	5	4	5	4.5	5	4.5	5	3	4	3	3.2	4	4	5	4	4	3.2	4	4	3.9	5	4.2
Siéntate bien	5	4.5	4	4	5	4	4.8	5	4	4	5	4.5	5	5	5	3	4	3	3	4	4	5	4	4	3	4	4	3.9	5	4.1
Las computadoras sirven mucho	5	4.5	4	4	5	4	5	5	5	4	5	4.5	5	4.5	5	3	4	3	3	4	4	5	4	4	3.1	4	4	3.9	5	4.3
Revisa este escrito pronto	4	4.5	4	4	5	4	5	5	5	4	5	4.5	5	4.5	5	3	4	3	3.1	4	4	5	4	4	3	4	4	3.7	5	4.1
Localiza a las 50 personas	4	4.5	4	4	5	4	4.9	5	4	4	5	4.5	5	4.5	5	3	4	3	3	4	4	5	4	4	3	4	4	3.7	5	4.2
Tu trabajo 56.37 ha concluido	4	4.5	4	4	5	4	5	5	5	4	5	4.5	5	4.5	5	3	4	3	3	4	4	5	4	4	3	4	4	4	5	4.2
Los carros son rojos	5	4.5	4	4	5	4	5	5	5	4	5	4.5	5	4.5	5	3	4	3	5	4	4	5	5	4	3.9	5	4	4.2	5	5
El canta bien	5	4.5	4	4	5	4	5	5	5	4	5	4.5	5	4.1	5	3	4	3	3	4	4	5	4.1	4	3	4	4	3.8	5	4.5

Tabla 5.2c

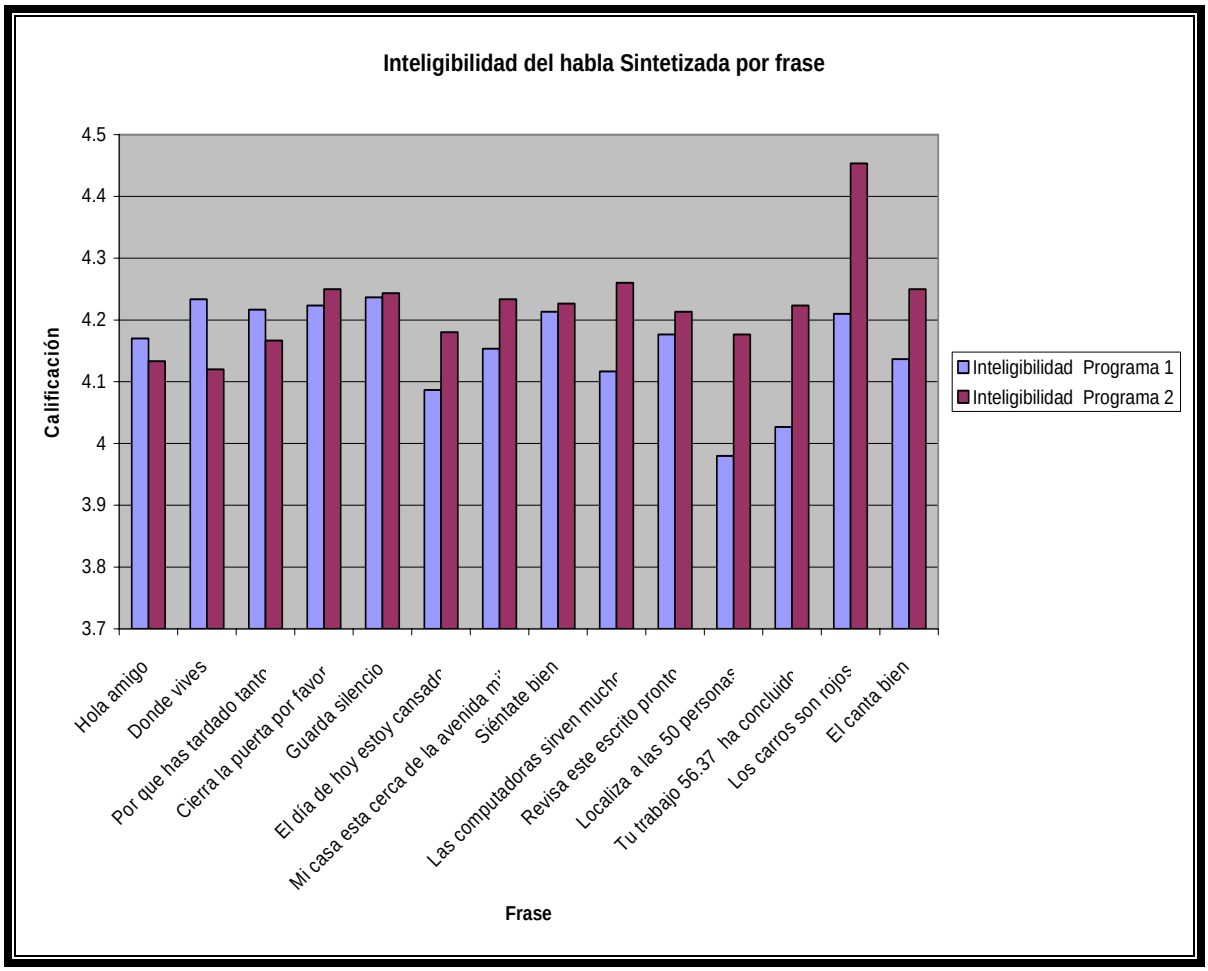


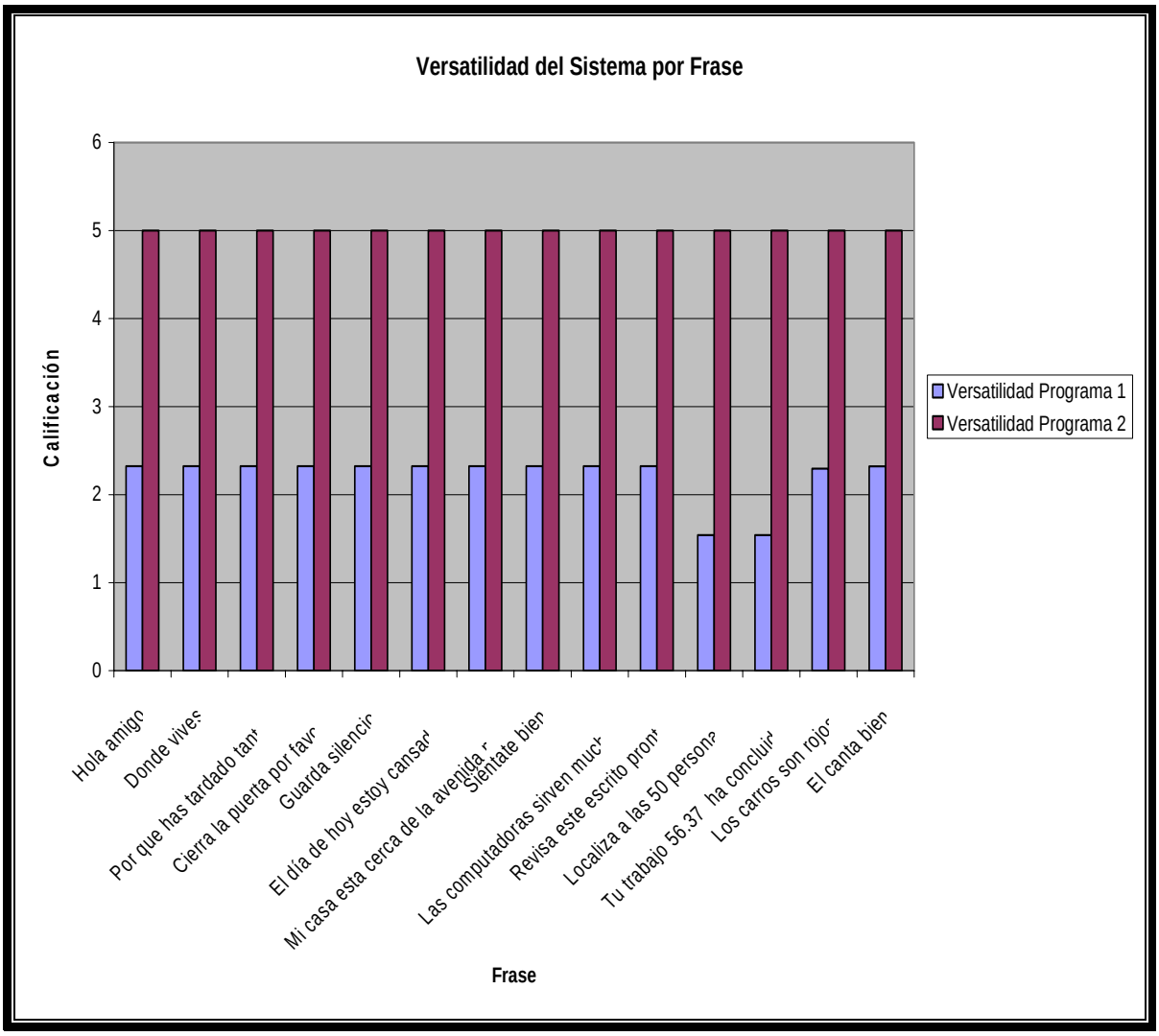
Tabla 5.3a

Versatilidad Programa 1																															
Oyente	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Hola amigo	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2	2.3	2.3
Donde vives	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2	2.3	2.3
Por que has tardado tanto	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2	2.3	2.3
Cierra la puerta por favor	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2	2.3	2.3
Guarda silencio	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2	2.3	2.3
El día de hoy estoy cansado	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2	2.3	2.3
Mi casa esta cerca de la avenida mil	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2	2.3	2.3
Siéntate bien	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2	2.3	2.3
Las computadoras sirven mucho	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2	2.3	2.3
Revisa este escrito pronto	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2	2.3	2.3
Localiza a las 50 personas	1	1	1	1.3	1.5	1	2.5	1.8	1	2.3	1	1	1.5	2.3	2.3	1	1	2.3	1	2.3	1.7	1	2.3	2.3	1	1	2.3	1.7	1.5	1
Tu trabajo 56.37 ha concluido	1	1	1	1.3	1.5	1	2.5	1.8	1	2.3	1	1	1.5	2.3	2.3	1	1	2.3	1	2.3	1.7	1	2.3	2.3	1	1	2.3	1.7	1.5	1
Los carros son rojos	2.3	2.3	2.3	2.3	1.5	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2	2.3	2.3
El canta bien	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2.3	2	2.3	2.3

Tabla 5.3b

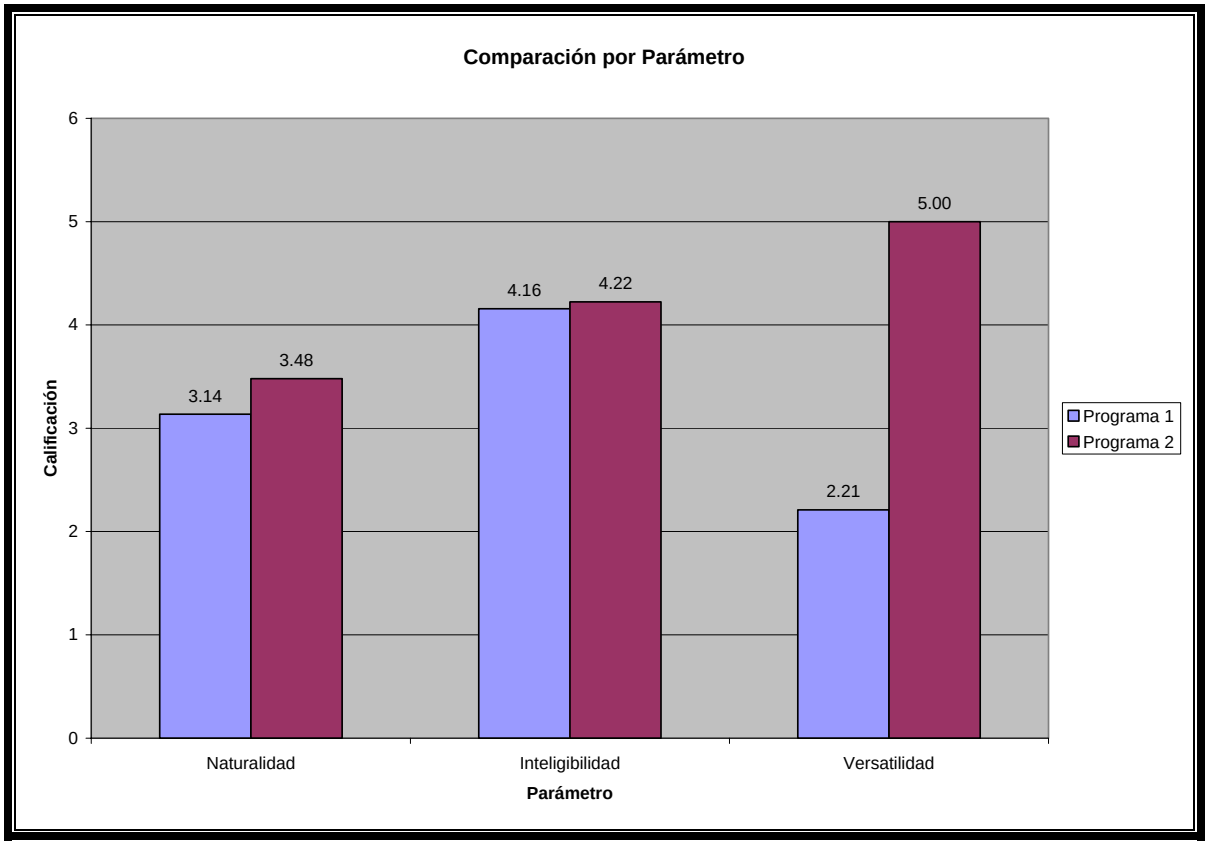
Versatilidad Programa 2																															
Oyente	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Hola amigo	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Donde vives	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Por que has tardado tanto	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Cierra la puerta por favor	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Guarda silencio	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
El día de hoy estoy cansado	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Mi casa esta cerca de la avenida mil	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Siéntate bien	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Las computadoras sirven mucho	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Revisa este escrito pronto	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Localiza a las 50 personas	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Tu trabajo 56.37 ha concluido	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Los carros son rojos	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
El canta bien	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5

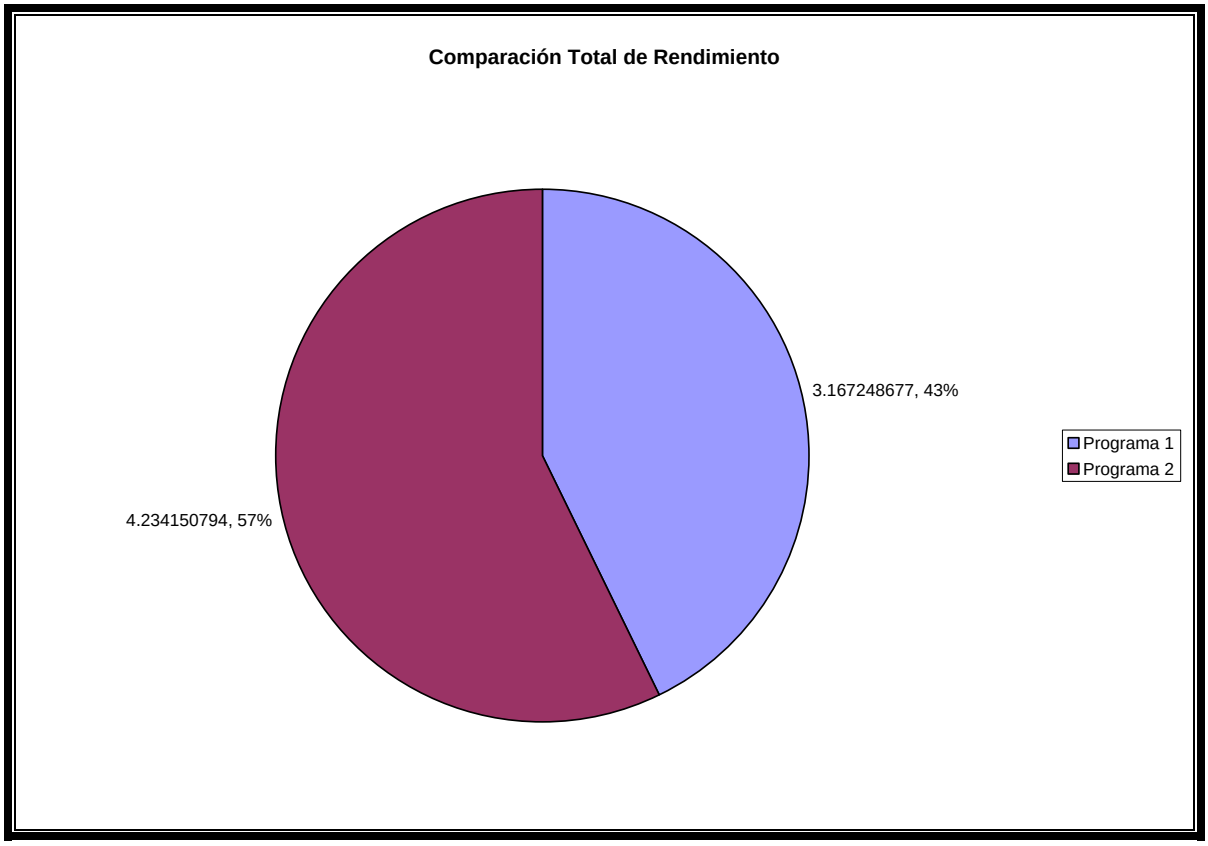
Tabla 5.3c



Obteniendo un promedio global por programa se tiene:

	NATURALIDAD	INTELIGIBILIDAD DEL HABLA SINTETIZADA	VERSATILIDAD DEL SISTEMA	PROMEDIO GLOBAL
Programa # 1	**3.14**	**4.16**	**2.21**	**3.16**
Programa # 2	**3.48**	**4.22**	**5.00**	**4.23**


Comparación por Parámetro
6
5
4
3
2
1
0
Calificación
3.14
3.48
4.16
4.22
2.21
5.00
Naturalidad
Inteligibilidad
Versatilidad
Parámetro
Programa 1
Programa 2


Comparación Total de Rendimiento
3.167248677, 43%
4.234150794, 57%
Programa 1
Programa 2

CONCLUSIONES

El avance tecnológico, sumado al abaratamiento de los equipos, ha llevado a la presencia de computadoras accesibles e indispensables en todos los ámbitos de nuestro quehacer diario. Dado el alto grado de conveniencia que esto implica, ha sido también creciente el avance científico para elaborar una forma de comunicación más natural con las máquinas, en el sentido de posibilitar una comunicación oral mejorada para que sea cada vez más utilizada.

La comunicación bidireccional con las máquinas exige la presencia, por un lado, de un módulo de decodificación de mensajes orales que llegan a las máquinas y por otro lado, de un módulo capaz de codificar y emitir mensajes comprendidos por los seres humanos. A lo largo del tiempo se han ido desarrollando técnicas de reconocimiento (decodificación) y síntesis (codificación) de voz, cuyo propósito es hacer posible esta bidireccionalidad.

Dentro de la parte de síntesis de voz, se han desarrollado los Conversores Texto-Voz, los cuales, primero realizan una traducción completa de los textos escritos a una representación lingüística del mensaje (procesamiento lingüístico prosódico); y posteriormente, de esta representación a una onda acústica (procesado acústico).

La implantación de los sistemas de síntesis de voz en aplicaciones reales ha sido posible gracias a los avances producidos en áreas de procesamiento digital de señales, pero también en el campo de los conocimientos lingüísticos, ya que estos avances exigen alcanzar un nivel alto de conocimientos fonéticos.

Tal como se mencionó al final del Capítulo 4, se tienen varias aplicaciones tales como, -sistema de lectura para ciegos, -sistema de habla para personas mudas, -ayuda a la enseñanza, -verificación de texto, -alarmas habladas, -instrucciones de montaje habladas. Es importante tener presente que estas aplicaciones pueden inclusive mejorar la calidad de vida de la personas como de sus tareas cotidianas.

Se puede concluir que después de estudiar las técnicas de síntesis, las características que deben tener los Conversores Texto-Voz son:

- Capacidad de ofrecer una CALIDAD elevada en los enunciados en cuanto a naturalidad, inteligibilidad y emotividad.
- Flexibilidad para producir cualquier mensaje representable en las comunicaciones ordinarias.
- Procesamiento relativamente simple y rápido de las formas de onda, ya que se tienen en la actualidad computadoras que realizan los algoritmos programados con cierta facilidad.

Los *sintetizadores por concatenación* requieren más memoria (por la necesidad de tener almacenada la colección de unidades), que los articulatorios y los de formantes. También son menos flexibles al no incorporar una colección de parámetros internos de control sobre los cuales actuar (por ejemplo, para variar el tipo de voz sintetizada).

Y las ventajas principales son la realización sencilla (al menos en una primera versión); y que son mejorables trabajando sobre el conjunto de unidades, sin necesidad de modificar el sintetizador propiamente dicho, tal como el rediseño en su segunda versión.

La técnica de PSOLA permite un control de la prosodia relativamente sencillo actuando sobre la representación de la señal en el dominio del tiempo. El hecho de que la señal se almacena y multiplica por la ventana supone un planteamiento original que aporta una sensible disminución de las necesidades de cómputo en tiempo real. La aplicación de este esquema para el control de prosodia, incluyendo acentuación, cantidad y entonación, parece una opción para un sistema de síntesis ya que la simplicidad de los cálculos que requiere, deriva fundamentalmente del hecho que se aplica a la representación temporal de la señal.

El método de síntesis que se utilizó en el segundo diseño algorítmico comparado al primero se puede calificar como bueno ya que el resultado de las pruebas MOS así lo indican pues se tiene un elevado grado de comprensión del texto escrito respecto al escrito en pantalla.

Cabe mencionar, que en el primer diseño se obtiene una *voz con aspecto artificial* y esto se debe al problema de coarticulación entre fonemas, que, si bien es parcialmente resuelto en el segundo diseño (PSOLA), con el tipo de

unidades empleadas, no se puede resolver sin un completo conjunto de reglas a aplicar en el proceso de síntesis.

Trabajos de investigación a futuro

Dentro de las principales áreas de desarrollo, se tienen aquellas donde se haga síntesis de voz tomando en cuenta diferentes características de la emotividad, la entonación, para que la señal de salida sea cada vez más natural a la voz humana.

Una mejora al segundo sintetizador diseñado sería incluir más alófonos por cada fonema en el conjunto de unidades, pero implicaría mayor consumo de memoria y tiempo, aunque como ya se mencionó, el avance en tecnología supondría esto como un menor problema.

Para mejorar el aspecto artifical de la voz, sería necesario recurrir a un método de síntesis por reglas o basado en algún tipo de aprendizaje inductivo, llevado a cabo por una computadora de mayor capacidad.

ANEXO A

Código en C++ del Sintetizador de Voz

```
// PruebaDlg.cpp : implementation file
//

#include "stdafx.h"
#include <mmsystem.h>
#include "Prueba.h"
#include "PruebaDlg.h"

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif


/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About



class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard generated virtual function
overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void
DoDataExchange(CDataExchange* pDX);    //
DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange*
pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CAboutDlg)
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
        //{{AFX_MSG_MAP(CAboutDlg)
                // No message handlers
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
////////////////////////////
// CPruebaDlg dialog

CPruebaDlg::CPruebaDlg(CWnd* pParent
/*=NULL*/)
        : CDialog(CPruebaDlg::IDD, pParent)
{
        //{{AFX_DATA_INIT(CPruebaDlg)
                // NOTE: the ClassWizard will
add member
initialization here
        //}}AFX_DATA_INIT
        // Note that LoadIcon does not require a
subsequent
DestroyIcon in Win32
        m_hIcon = AfxGetApp()-
>LoadIcon(IDR_MAINFRAME);
}

void CPruebaDlg::DoDataExchange(CDataExchange*
pDX)
{
        CDialog::DoDataExchange(pDX);
        //{{AFX_DATA_MAP(CPruebaDlg)
                // NOTE: the ClassWizard will
add DDX and DDV
calls here
        //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CPruebaDlg, CDialog)
        //{{AFX_MSG_MAP(CPruebaDlg)
        ON_WM_SYSCOMMAND()
        ON_WM_PAINT()
        ON_WM_QUERYDRAGICON()
        ON_BN_CLICKED(IDC_BUTTON1,
OnButton1)
        ON_BN_CLICKED(IDC_BUTTON2,
OnButton2)
        ON_BN_CLICKED(IDC_BUTTON3,
OnButton3)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

```
/////////////////////////////////////////////////////////////////////
/////////
/////////////////////
// CPruebaDlg message handlers

BOOL CPruebaDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system
menu.

	// IDM_ABOUTBOX must be in the
system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) ==
IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu =
GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;

	strAboutMenu.LoadString(IDS_ABOUTBO
X);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu-
>AppendMenu(MF_SEPARATOR);
			pSysMenu-
>AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The
framework does this automatically
	//  when the application's main window is
not a dialog
	SetIcon(m_hIcon, TRUE);
	// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set
small icon

	// TODO: Add extra initialization here

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

void CPruebaDlg::OnSysCommand(UINT nID,
LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID,
lParam);
	}
}

// If you add a minimize button to your dialog, you
will need the code below
//  to draw the icon.  For MFC applications using
the document/view model,
//  this is automatically done for you by the
framework.

void CPruebaDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device
context for painting


	SendMessage(WM_ICONERASEBKGND,
(WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon =
GetSystemMetrics(SM_CXICON);
		int cyIcon =
GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon +
1) / 2;
		int y = (rect.Height() - cyIcon +
1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to
display
while the user drags
//  the minimized window.
HCURSOR CPruebaDlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}


void CPruebaDlg::OnOK()
```

```
{
        // TODO: Add extra validation here

//      CDialog::OnOK();
}


/* Analisis de palabras */

int vocal(TCHAR a) {
        if (a=='a' | a=='á'  | a=='o' | a=='ó' |
a=='u'
| a=='ú' | a=='e' | a=='é' | a=='i' | a=='í')
    return(1);
        else
        return(0);}

int vdebil(TCHAR a) {
        if ( a=='e' | a=='é' | a=='i' | a=='í')
    return(1);
        else
        return(0);}

int acento(TCHAR a) {
                if (a=='á' | a=='é' | a=='í' |
a=='ó' | a=='ú')
    return(1);
        else
        return(0);}

int palabra_acentuada(CString palabra) {
int a,b;
TCHAR l1;
b=palabra.GetLength();
        b=b-1;
for (a=0;a<=b;a++) {
l1=palabra[a];
if (acento(l1)) return(1);
}
return(0);
}

int nsv(CString palabra) {
int b;
TCHAR l1;
b=palabra.GetLength();
b=b-1;
l1=palabra[b];
if(l1=='n' | l1=='s' | vocal(l1) ) return(1);
else return(0);
}


CString poner_acento(CString palabra,int lugar) {
LPTSTR p = palabra.GetBuffer( 50 );
if (palabra[lugar]=='a') p[lugar]='á';
else if (palabra[lugar]=='e') p[lugar]='é';
else if (palabra[lugar]=='i') p[lugar]='í';
else if (palabra[lugar]=='o') p[lugar]='ó';
else if (palabra[lugar]=='u') p[lugar]='ú';
palabra==p;
        return(p);
}

TCHAR nacc(TCHAR p) {
if (p=='á') p='a';
else if (p=='é') p='e';
else if (p=='í') p='i';
else if (p=='ó') p='o';
else if (p=='ú') p='u';
        return(p);
}


CString palabra(CString palabra) {
CString salida="";
int a,b,look,remember,found;
TCHAR p1,p2,l1,l2,l3;
b=palabra.GetLength();
b=b-1;

if (b==0) {
l1=palabra[0];
        switch(l1){
        case 'b':
                palabra="be";
        break;
        case 'c':
                palabra="ce";
        break;
        case 'd':
                palabra="de";
        break;
        case 'f':
                palabra="efe";
        break;
        case 'g':
                palabra="ge";
        break;
        case 'h':
                palabra="ache";
        break;
        case 'j':
                palabra="jota";
        break;
        case 'k':
                palabra="ka";
        break;
        case 'l':
                palabra="ele";
        break;
        case 'm':
                palabra="eme";
        break;
        case 'n':
                palabra="ene";
```

```
        break;
        case 'p':
                palabra="pe";
        break;
        case 'q':
                palabra="ku";
        break;
        case 'r':
                palabra="erre";
        break;
        case 's':
                palabra="ese";
        break;
        case 't':
                palabra="te";
        break;
                case 'v':
                palabra="ube";
        break;
                case 'w':
                palabra="dóble-u";
        break;
                case 'x':
                palabra="ekis";
        break;
                case 'y':
                palabra="i";
        break;
                case 'z':
                palabra="zeta";
        break;
        }
b=palabra.GetLength();
b=b-1;
}


for (a=0;a<=b;a++) {

l1=palabra[a];
if(a<b)   l2=palabra[a+1];
else l2=' ';
if(a<b-1) l3=palabra[a+2];
else l3=' ';
if(a>0)   p1=palabra[a-1];
else p1=' ';
switch (l1){
/*case 'h':
        l1='-';
        break;*/
case 'c':
   l1='k';
        if (l2=='h') {l1='C';a=a+1;}
        if (vdebil(l2)) {l1='s';}
   break;
case 's':
        if (l2=='h') { l1='S';a=a+1;}
break;
```

```
case 'l':
        if (l2=='l') {l1='L';a=a+1;}
break;
case 'r':
        if (l2=='r') {l1='R';a=a+1;}
if (!vocal(p1)) {l1='r';}
break;
case 'q':
        l1='k';
        if(l2=='u') {a=a+1;}
        break;
case 'v':
        l1='b';
        break;
case 'z':
        l1='s';
        break;
case 'y':
        if(!vocal(l2)) {l1='i';}
        break;
case 'g':
        if (l2=='e' | l2=='i') {l1='j';}
        if (l2=='u' & (vdebil(l3))) {a=a+1;}
        break;
case 'ü':
        l1='u';
        break;
case 'x':
        salida=salida+'k';
        l1='s';
   break;
case ' ':
        l1='-';
 break;}
if(l1!='h') salida = salida + l1; }

/* aqui hay que encontrar la vocal acentuda */
/* paso 1 -> si ya esta acentuada no hacer nada*/
if (!palabra_acentuada(salida)) {
/* paso 2 -> penultima silaba?? */
                found=0;
        if (nsv(salida)) {
                                for
(a=0;a<=b;a++) {
                        l2=0;
                        l3=0;
                        p1=0;
                        p2=0;
                        l1=salida[a];
                        if(a<b) l2=salida[a+1];
                        if(a<(b-1))
l3=salida[a+2];
                        if(a>0) p1=salida[a-1];
                        if(a>1) p2=salida[a-2];

                        if (vocal(l1) & !found) {
                                remember=a;
                                look=0;
```

```
                                        if((a<=(b-2)) &
(l2=='u' | l2=='i')) look=a+2;
                                        else if(a<b)
look=a+1;
                                        l1=0;
                                        while(look<=b &
(!vocal(salida[look]))) look=look+1;
                                        /*if (look>=0 &
look<=b) */
                                        l1=salida[look];
                                        if(look>=(b-1)
& vocal(l1)) found=1;
                                }

                        }



        }
/* paso 3 -> pues ultima silaba */
    if (!found) {
                    if (b==0 & vocal(salida[0]))
{remember=0;found=1;}
if(b>0){
        if (vocal(salida[b-1])) {remember=b-
1;found=1;}
        else if(b>1 & vocal(salida[b-2]) &
(!vocal(salida[b-1]) | salida[b-1]=='i' | salida[b-
1]=='u') & !vocal(salida[b])) {remember=b-
2;found=1;}
}

        }

if (found==1)
salida=poner_acento(salida,remember);
} /*en este parentesis termina funcion de
acentuación */


/*quikie para diptongos */
/*                                      for
(a=0;a<=b;a++) {
                                l2=0;
                                p1=0;
                                l1=salida[a];
                                if(a<(b-1))
l2=salida[a+1];
                                if(a>0) p1=salida[a-1];
                                if (vocal(l1) & vocal(l2)
& vocal(p1) & !acento(p1) & !acento(l2)) {

        salida=poner_acento(salida,a);}
                                else if (vocal(l1) &
!vocal(l2) & vocal(p1) & !acento(p1)) {

        salida=poner_acento(salida,a);} }
```

```
*/
salida=salida+"-";
return (salida);}



/* rutinas para números */
CString numero(CString numero) {
int a,b;
CString salida="";
TCHAR l1;
b=numero.GetLength();
b=b-1;
for (a=0;a<=b;a++) {

l1=numero[a];
        switch(l1) {
case '1':
                salida=salida+palabra("úno");
                        break;
case '2':
                salida=salida+palabra("dós");
                        break;
case '3':
                salida=salida+palabra("trés");
                        break;
case '4':
                salida=salida+palabra("cuátro");
                        break;
case '5':
                salida=salida+palabra("cínco");
                        break;
case '6':
                salida=salida+palabra("séis");
                        break;
case '7':
                salida=salida+palabra("siéte");
                        break;
case '8':
                salida=salida+palabra("ócho");
                        break;
case '9':
                salida=salida+palabra("nuéve");
                        break;
case '0':
                salida=salida+palabra("céro");
                break;
case '.':
                salida=salida+palabra("púnto");
                break;}

}
return(salida);
}



/* Rutina principal*/
```

```
void CPruebaDlg::OnButton1()
{
int i,j,k,tipo;
UINT leido,tamano;
TCHAR l1,l2,l3;
CFile fentra,fsale;
CFileException e1;
CEdit* cad1 = (CEdit*) GetDlgItem(IDC_EDIT1);
CString cadena="",ctemp="",ctemp2="",analisis="";
CString salida=" -";
CString archson;
char pbuf[30000];

/* Leer cadena & analisis preeliminar*/
SetWindowText("Sintesis de Voz --> Analizando");
cad1->GetWindowText(cadena);
if (cadena=="") {cadena="Bienvenidos al sistema de
síntesis de voz";}
cadena.MakeLower();
cadena=cadena+" ";
j=cadena.GetLength();

/* separar por palabras */
ctemp="";
tipo=0;
for (i=0;i<=j-1;i++) {
l1=cadena[i];
                        if(i<(j-1))
l2=cadena[i+1];
                        if(i<(j-1))
l3=cadena[i+2];
if (( (l1>='a' & l1<='z') | (l1>='á' & l1<='ú') |
l1=='ñ' | l1=='ü') & (tipo==1 | tipo==0))
{ctemp=ctemp+l1;tipo=1;}
else if ( (l1>='0' & l1<='9') & (tipo==0 | tipo==2)
)
{ctemp=ctemp+l1;tipo=2;
}
else if ((l1=='.' | l1==',') & ( l1>='0' & l1<='9') &
tipo==2)
{
ctemp=ctemp+l1;
}


else
{
if (tipo==1) salida=salida+palabra(ctemp);
if (tipo==2) salida=salida+numero(ctemp);
ctemp="";
tipo=0;
if (l1==',') salida=salida+"-0-";
else if (l1==';') salida=salida+"-1-";
else if (l1=='.') salida=salida+"-2-";
if((l1>='a' & l1<='z') | (l1>='á' & l1<='ú') |
l1=='ñ' | (l1>='1' & l1<='9') | l1=='ü'  )
i=i-1; } }
```

```
/* escribir header */
SetWindowText("->"+salida);
salida=salida+"-";
cad1 = (CEdit*) GetDlgItem(IDC_EDIT2);
cad1->GetWindowText(archson);
if (archson=="") {archson="default";}
archson=archson+".wav";

if
(!fsale.Open("data\\header.wav",CFile::modeRead
,&e1)) {}
if (!fentra.Open(archson,CFile::modeCreate |
CFile::modeWrite ,&e1)) {}
leido=fsale.Read(pbuf,44);
fsale.Close();
fentra.Write(pbuf,44);
j=salida.GetLength();
j=j-2;
cadena="";
tamano=0;
for (i=1;i<=j;i++) {
l1=salida[i];
l2=salida[i+1];
ctemp=l1;
ctemp2=l2;
switch (l1){
case 'C':
ctemp="ch";
break;
case 'L':
ctemp="y";
break;
case 'R':
ctemp="rr";
break;
case 'S':
ctemp="sh";
break;}
switch (l2){
case 'C':
ctemp2="ch";
break;
case 'L':
ctemp2="y";
break;
case 'R':
ctemp2="rr";
break;
case 'S':
ctemp2="sh";
break;}
cadena= "data\\"+ctemp+ctemp2+".pcm";
if (fsale.Open(cadena,CFile::modeRead ,&e1)) {
leido=fsale.Read(pbuf,30000);
if (!(vocal(l1) & !vocal(l2) & vocal(l3) ))
fentra.Write(pbuf,leido);
tamano=tamano+leido;
fsale.Close();}
```

```
else {
cadena="data\\"+ctemp+"-.pcm";
if (fsale.Open(cadena,CFile::modeRead ,&e1)) {
leido=fsale.Read(pbuf,30000);
fentra.Write(pbuf,leido);
tamano=tamano+leido;
fsale.Close();}
cadena="data\\-"+ctemp2+".pcm";
if (fsale.Open(cadena,CFile::modeRead ,&e1)) {
leido=fsale.Read(pbuf,30000);
fentra.Write(pbuf,leido);
tamano=tamano+leido;
fsale.Close();}  }  }

fentra.Seek(40,CFile::begin);
fentra.Write(&tamano,4);
tamano=tamano+40;
fentra.Seek(4,CFile::begin);
fentra.Write(&tamano,4);
fentra.Close();
cad1 = (CEdit*) GetDlgItem(IDC_EDIT2);
cad1->GetWindowText(cadena);
if (cadena=="") {cadena="default";}
cadena=cadena+".wav";
PlaySound(archson, NULL, SND_ASYNC |
SND_FILENAME);

 }

void CPruebaDlg::OnButton2()
{
        // TODO: Add your control notification
handler code here
        exit(0);
}



void CPruebaDlg::OnButton3()
{
int i,j,k,tipo;
UINT leido,tamano;
TCHAR l1,l2,l3;
CFile fentra,fsale;
CFileException e1;
CEdit* cad1 = (CEdit*) GetDlgItem(IDC_EDIT1);
CString cadena="",ctemp="",ctemp2="",analisis="";
CString salida=" -";
CString archson;
char pbuf[30000];

/* Leer cadena & analisis preeliminar*/
SetWindowText("Sintesis de Voz --> Analizando");
cad1->GetWindowText(cadena);
if (cadena=="") {cadena="Bienvenidos al sistema de
síntesis de voz";}
cadena.MakeLower();
cadena=cadena+" ";
```

```
j=cadena.GetLength();

/* separar por palabras */
ctemp="";
tipo=0;
for (i=0;i<=j-1;i++) {
l1=cadena[i];
                                if(i<(j-1))
l2=cadena[i+1];
                                if(i<(j-1))
l3=cadena[i+2];
if (( (l1>='a' & l1<='z') | (l1>='á' & l1<='ú') |
l1=='ñ' | l1=='ü') & (tipo==1 | tipo==0))
{ctemp=ctemp+l1;tipo=1;}
else if ( (l1>='0' & l1<='9') & (tipo==0 | tipo==2)
)
{ctemp=ctemp+l1;tipo=2;
}
else if ((l1=='.' | l1==',') & ( l1>='0' & l1<='9') &
tipo==2)
{
ctemp=ctemp+l1;
}


else
{
if (tipo==1) salida=salida+palabra(ctemp);
if (tipo==2) salida=salida+numero(ctemp);
ctemp="";
tipo=0;
if (l1==',') salida=salida+"-0-";
else if (l1==';') salida=salida+"-1-";
else if (l1=='.') salida=salida+"-2-";
if((l1>='a' & l1<='z') | (l1>='á' & l1<='ú') |
l1=='ñ' | (l1>='1' & l1<='9') | l1=='ü'  )
i=i-1; } }

/* escribir header */
SetWindowText("->"+salida);
salida=salida+"-";
cad1 = (CEdit*) GetDlgItem(IDC_EDIT2);
cad1->GetWindowText(archson);
if (archson=="") {archson="default";}
archson=archson+".wav";

if
(!fsale.Open("data\\header.wav",CFile::modeRead
,&e1)) {}
if (!fentra.Open(archson,CFile::modeCreate |
CFile::modeWrite ,&e1)) {}
leido=fsale.Read(pbuf,44);
fsale.Close();
fentra.Write(pbuf,44);
j=salida.GetLength();
j=j-2;
cadena="";
tamano=0;
```

```
for (i=1;i<=j;i++) {
l1=salida[i];
l2=salida[i+1];
ctemp=l1;
ctemp2=l2;
switch (l1){
case 'C':
ctemp="ch";
break;
case 'L':
ctemp="y";
break;
case 'R':
ctemp="rr";
break;
case 'S':
ctemp="sh";
break;}
switch (l2){
case 'C':
ctemp2="ch";
break;
case 'L':
ctemp2="y";
break;
case 'R':
ctemp2="rr";
break;
case 'S':
ctemp2="sh";
break;}
cadena= "data1\\"+ctemp+ctemp2+".pcm";
if (fsale.Open(cadena,CFile::modeRead ,&e1)) {
leido=fsale.Read(pbuf,30000);
if (!(vocal(l1) & !vocal(l2) & vocal(l3) ))
fentra.Write(pbuf,leido);
tamano=tamano+leido;
fsale.Close();}
else {
cadena="data1\\"+ctemp+"-.pcm";
if (fsale.Open(cadena,CFile::modeRead ,&e1)) {
leido=fsale.Read(pbuf,30000);
fentra.Write(pbuf,leido);
tamano=tamano+leido;
fsale.Close();}
cadena="data1\\-"+ctemp2+".pcm";
if (fsale.Open(cadena,CFile::modeRead ,&e1)) {
leido=fsale.Read(pbuf,30000);
fentra.Write(pbuf,leido);
tamano=tamano+leido;
fsale.Close();}  }  }
```

```
fentra.Seek(40,CFile::begin);
fentra.Write(&tamano,4);
tamano=tamano+40;
fentra.Seek(4,CFile::begin);
fentra.Write(&tamano,4);
fentra.Close();
cad1 = (CEdit*) GetDlgItem(IDC_EDIT2);
cad1->GetWindowText(cadena);
if (cadena=="") {cadena="default";}
cadena=cadena+".wav";
PlaySound(archson, NULL, SND_ASYNC |
SND_FILENAME);


}
```

ANEXO B

Código en C# del Sintetizador de Voz

Clase principal "CSintetizador"

```
using System;
using System.IO;
using System.Text;

using WAV;

namespace Sintetizador
{
        public class CSintetizador
        {
                private string dir;
                public string difonemas;
                public string viz="";
                public string viz2="";
                public int al;
                public int bl;

                public CSintetizador(string
path,int ar, int br)
                {
                        this.dir = path;
                        this.difonemas = "";
                        this.al=ar;
                        this.bl=br;
                }


                private bool vdebil(char a)
                {
                        if ( a=='e' | a=='é' |
a=='i' | a=='í')
                                return true;
                        else
                                return false;
                }

                private bool vocal(char a)
                {
                        if (a=='a' | a=='á'  |
a=='o' | a=='ó' | a=='u' | a=='ú' | a=='e' |
a=='é' | a=='i' | a=='í')
                                return true;
                        else
                                return false;
                }

                private bool acento(char a)
                {
                        if (a=='á' | a=='é' |
a=='í' | a=='ó' | a=='ú')
                                return true;
                        else
                                return false;
                }

                private bool nsv(string palabra)
                {
                        char l1 =
palabra[palabra.Length-1];

                        if(l1=='n' | l1=='s' |
vocal(l1) )
                                return true;
                        else
                                return false;
                }


                private string poner_acento(string
palabra, int lugar)
                {
                        string p0 =
palabra.Substring(0, lugar);
                        char p1 =
palabra[lugar];
                        string p2 =
palabra.Substring(lugar+1, palabra.Length-lugar-
1);

                        if (p1=='a') p1='á';
                        else if (p1=='e') p1='é';
                        else if (p1=='i') p1='í';
                        else if (p1=='o') p1='ó';
                        else if (p1=='u') p1='ú';

                        palabra = p0 + p1 + p2;

                        return palabra;
                }

                public string analizar(string
cadena)
                {

                        string ctemp = "", salida
= " -";
                        string ctemp2 = "";
                        int tipo = 0, n;
                        char l1;

                        if (cadena == "")
                                cadena =
"Bienvenidos al sistema de síntesis de voz";

                        cadena = cadena + " ";
                        cadena =
cadena.ToLower();

                        n = cadena.Length;
```

```
                        for (int i=0; i<n; i++)
                        {
                                l1 = cadena[i];

                                if (((l1>='a' &
l1<='z') | (l1>='á' & l1<='ú') | l1=='ñ' | l1=='ü')
& (tipo==1 | tipo==0))
                                {
                                        ctemp
= ctemp + l1;
                                        tipo =
1;
                                }
                                else if
((l1>='0' & l1<='9') & (tipo==0 | tipo==2))
                                {
                                        ctemp
= ctemp + l1;
                                        tipo =
2;
                                }
                                else if ((l1=='.'
| l1==',') & tipo==2)
                                {


        tipo=3;
                                }
                                else if
((l1>='0' & l1<='9')& (tipo==3 | tipo==4))
                                {

        ctemp2 = ctemp2+l1;
                                        tipo =
4;
                                }
                                else
                                {
                                        if
(tipo==1) salida = salida + palabra(ctemp)+"-";
                                        if
(tipo==2) salida = salida + numero(ctemp)+"-";
                                        if
(tipo==3) salida = salida + numero(ctemp)+"-púnto-
";
                                        if
(tipo==4) salida = salida + numero(ctemp)+"-púnto-
"+numero(ctemp2)+"-";
                                        if
(l1==',') salida = salida + "-0-";
                                        if
(l1==';') salida = salida + "-1-";
                                        if
(l1=='.') salida = salida + "-2-";
                                        ctemp
= "";
                                        tipo =
0;
                                }
                        }


                        creararchivo(salida);

                        return salida;
                }

                private string creararchivo(string
cadena)
                {
                        cadena+="-";
                        StreamReader fpcm;
                        char l1, l2 = ' ', l3 = '
';
                        string archson, path,
dif, ctemp, ctemp2, ctemp3;


                        CWAV wav = new
CWAV( al, bl);


                        for (int i=1;
i<cadena.Length-2; i++)
                        {
                                l1 = cadena[i];

        if(i<(cadena.Length-1))     l2 =
cadena[i+1];

        if(i<(cadena.Length-3))     l3 =
cadena[i+2];



                                ctemp =
l1.ToString();
                                ctemp2 =
l2.ToString();
                                ctemp3 =
l3.ToString();

                                switch (l1)
                                {
                                        case
'C':

        ctemp = "ch";

        break;
                                        case
'L':

        ctemp = "y";

        break;
```

```
                                        case
'R':

        ctemp = "rr";

        break;
                                        case
'S':

        ctemp = "sh";

        break;
                                }
                                switch (l2)
                                {
                                        case
'C':

        ctemp2 = "ch";

        break;
                                        case
'L':

        ctemp2 = "y";

        break;
                                        case
'R':

        ctemp2 = "rr";

        break;
                                        case
'S':

        ctemp2 = "sh";

        break;
                                }
                                switch (l3)
                                {
                                        case
'C':

        ctemp3 = "ch";

        break;
                                        case
'L':

        ctemp3 = "y";

        break;
                                        case
'R':

        ctemp3 = "rr";
```

```
        break;
                                        case
'S':

        ctemp3 = "sh";

        break;
                                }




                                try
                                {
                                        path =
dir + ctemp + ctemp2 + ".pcm";
                                        fpcm
= new StreamReader(path, Encoding.Default);
                                        if
(!((vocal(l1) & !vocal(l2) & vocal(l3))))
                                        {

        dif = fpcm.ReadToEnd();

        fpcm.Close();

        wav.addPCM(dif);

        this.difonemas += "*" + ctemp + ctemp2 +
"*";




                                        }
                                }
                                catch
                                {
                                        try
                                        {

        path = dir + ctemp + ".pcm";

        fpcm = new StreamReader(path,
Encoding.Default);

        dif = fpcm.ReadToEnd();

        fpcm.Close();

        wav.addPCM(dif);

        this.difonemas += "*" + ctemp + ctemp2 +
"*";
                                        }
```

```
                                        catch
                                        {


        try

        {

                path = dir + ctemp2 + ".pcm";

                fpcm = new StreamReader(path,
Encoding.Default);

                dif = fpcm.ReadToEnd();

                fpcm.Close();

                wav.addPCM(dif);

                this.difonemas += "*" + ctemp +
ctemp2 + "*";



        }

        catch

        {

                try

                {

                        path = dir + "1-" +
".pcm";

                        fpcm = new
StreamReader(path, Encoding.Default);

                        dif = fpcm.ReadToEnd();

                        fpcm.Close();

                        wav.addPCM(dif);

                        this.difonemas += "*" +
"1-" + "*";

                }

                catch{}

                }

                                        }


                                    }
```

```
                        }

                        archson = "..//..//..//"
+ cadena;
                        archson = archson +
".wav";

                        //Los filtros y
algoritmos de mejora de sonido

                        //Se carga la data en
un string



                        //Se pasa el string a un
arreglo cambiando los chars por ints
                        int[] arreglo1=new
int[wav.data.Length];
                        for (int
vi=0;vi<wav.data.Length;vi++)
                        {

        arreglo1[vi]=(int)wav.data[vi];
                        }


                        //Se crea un segundo
arreglo para que en este se apliquen los filtors
                        int[] arreglo2=new
int[wav.data.Length];
                        arreglo2[0]=arreglo1[0];

        arreglo2[wav.data.Length-
1]=arreglo1[wav.data.Length-1];
                        arreglo2[1]=arreglo1[1];

        arreglo2[wav.data.Length-
2]=arreglo1[wav.data.Length-2];


                        //Filtros

                        //números significativos
(son los cambiados)
                        for (int
va=2;va<wav.data.Length-2;va++)
                        {
                                if(
(arreglo2[va]>arreglo2[va-2]&
arreglo2[va]<arreglo2[va+2]))
                                {

        arreglo2[va]=arreglo1[va];
                                        va++;
                                }
                                else
```

```
                                        if(
(arreglo2[va]<arreglo2[va-2]&
arreglo2[va]>arreglo2[va+2]))
                                {

        arreglo2[va]=arreglo1[va];
                                        va++;
                                }
                                else
                                {

        arreglo2[va]=((arreglo1[va]+arreglo1[va-
2]+arreglo1[va+2])/3);
                                        va++;

                                }

                        }


                        //números no
significativos (se pasan igual)
                        for (int
va=3;va<wav.data.Length-3;va++)
                        {




        arreglo2[va]=arreglo1[va];
                                        va++;



                        }

                        //for (int
a=0;a<wav.data.Length;a++)
                        //{

        //viz+=arreglo2[a];
                        //}
                        //for (int
a=0;a<wav.data.Length;a++)
                        //{

        //viz2+=((int)wav.data[a]).ToString();
                        //}

                        //Se pasan las
modificaciones al wavfile en su data
                        int fi=wav.data.Length;
                        wav.data="";
                        for (int a=0;a<fi;a++)
                        {

        wav.data+=(char)arreglo2[a];
                        }
```

```
                        //creación de archivo


        wav.createWav(archson);


                        //lectura de archivo
para que suene en el momento
                        StreamReader aref;
                        aref= new
StreamReader(archson,System.Text.Encoding.Defa
ult);
                        string FILE_NAME =
archson;

        Sound.Play(FILE_NAME);
                        return archson;
//regresa los difonemas para que se vean en
pantalla
                }

                private bool
palabra_acentuada(string palabra)
                {
                        for (int i=0;
i<palabra.Length; i++)
                        {
                                if
(acento(palabra[i]))
                                        return
true;
                        }

                        return false;
                }

                private string acentuacion(string
palabra)
                {
                        string salida = palabra;
                        int n = palabra.Length,
look, remember = 0;
                        bool found;
                        char l1,l2;


                        if
(string.Compare(palabra, "en")==0)
                        {
                                palabra="én";
                                return salida;
                        }
                        if
(string.Compare(palabra, "es")==0)
                        {
                                palabra="és";
```

```
                                return salida;
                        }
                        if
(string.Compare(palabra, "an")==0)
                        {
                                palabra="án";
                                return salida;
                        }
                        if
(string.Compare(palabra, "as")==0)
                        {
                                palabra="ás";
                                return salida;
                        }
                        if
(string.Compare(palabra, "un")==0)
                        {
                                palabra="ún";
                                return salida;
                        }
                        if
(string.Compare(palabra, "us")==0)
                        {
                                palabra="ús";
                                return salida;
                        }
                        if
(string.Compare(palabra, "que")==0)
                        {
                                palabra="ke";
                                return salida;
                        }

                        if
(!palabra_acentuada(salida))
                        {
                                found = false;
                                if (nsv(salida))
                                {

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

        l1 = salida[i];

        l2 = (char)0;


        if(i<(n-1))             l2 =
salida[i+1];


        if (vocal(l1) & !found)

        {

                remember = i;

                look = 0;


                if((i<=(n-3)) & (l2=='u' | l2=='i'))

                        look = i+2;

                else

                        if(i<(n-1))

                                look = i+1;


                l1 = (char)0;


                while(look<=n &
(!vocal(salida[look])))

                        look++;


                l1 = salida[look];


                if(look>=(n-2) & vocal(l1))

                        found = true;

        }

                                        }
                                }

                                if (!found)
                                {
                                        if
(n==0 & vocal(salida[0]))
                                        {

        remember = 0;

        found = true;
                                        }
                                        if
(n>0)
                                        {

        if (vocal(salida[n-2]))

        {

                remember = n-2;

                found = true;
```

```
        }

        else

                if(n>2 & vocal(salida[n-3]) &
(!vocal(salida[n-2]) | salida[n-2]=='i' | salida[n-
2]=='u') & !vocal(salida[n-1]))

                {

                        remember = n-3;

                        found = true;

                }
                                        }
                                }

                                if (found)
                                        salida
= poner_acento(salida, remember);
                        }

                        return salida;

                }

                private string palabra(string
palabra)
                {
                        string salida = "";
                        char l1, l2, l3, p1;
                        int n = palabra.Length;

                        if (n == 1)
                        {
                                l1 =
palabra[0];
                                switch(l1)
                                {
                                        case
'b':

        palabra = "be";

        break;
                                        case
'c':

        palabra = "ce";

        break;
                                        case
'd':

        palabra = "de";

        break;
                                        case
'f':

        palabra = "efe";

        break;
                                        case
'g':

        palabra = "ge";

        break;
                                        case
'h':

        palabra = "ache";

        break;
                                        case
'j':

        palabra = "jota";

        break;
                                        case
'k':

        palabra = "ka";

        break;
                                        case
'l':

        palabra = "ele";

        break;
                                        case
'm':

        palabra = "eme";

        break;
                                        case
'n':

        palabra = "ene";

        break;
                                        case
'ñ':

        palabra = "eñe";

        break;
                                        case
'p':
```

```
        palabra = "pe";
        break;
                                        case
'q':
        palabra = "ku";
        break;
                                        case
'r':
        palabra = "erre";
        break;
                                        case
's':
        palabra = "ese";
        break;
                                        case
't':
        palabra = "te";
        break;
                                        case
'v':
        palabra = "ube";
        break;
                                        case
'w':
        palabra = "doble-u";
        break;
                                        case
'x':
        palabra = "ekis";
        break;
                                        case
'y':
        palabra = "i";
        break;
                                        case
'z':
        palabra = "zeta";
        break;
```

```
                                }
                        }

        if(string.Compare("av",palabra)==0)
        palabra="avenida";

        if(string.Compare("ing",palabra)==0)
        palabra="ingeniero";

        if(string.Compare("lic",palabra)==0)
        palabra="licenciado";

        if(string.Compare("arq",palabra)==0)
        palabra="arquitecto";

        if(string.Compare("fis",palabra)==0)
        palabra="físico";

        if(string.Compare("prof",palabra)==0)
        palabra="profesor";

        if(string.Compare("dr",palabra)==0)
        palabra="doctor";

        if(string.Compare("sr",palabra)==0)
        palabra="señor";

        if(string.Compare("sra",palabra)==0)
        palabra="señora";

        if(string.Compare("srita",palabra)==0)
        palabra="señorita";

        if(string.Compare("gral",palabra)==0)
        palabra="general";

        if(string.Compare("ud",palabra)==0)
        palabra="usted";

        if(string.Compare("uds",palabra)==0)
        palabra="ustedes";

        if(string.Compare("hno",palabra)==0)
        palabra="hermano";

        if(string.Compare("pdte",palabra)==0)
        palabra="presidente";

        if(string.Compare("sn",palabra)==0)
        palabra="san";

        if(string.Compare("km",palabra)==0)
        palabra="kilometro";

        if(string.Compare("cm",palabra)==0)
        palabra="centimetro";
```

```
if(string.Compare("kg",palabra)==0)
palabra="kilogramo";

if(string.Compare("mm",palabra)==0)
palabra="milímetro";

if(string.Compare("etc",palabra)==0)
palabra="etcétera";

if(string.Compare("atte",palabra)==0)
palabra="atentamente";

if(string.Compare("pág",palabra)==0)
palabra="página";

if(string.Compare("sig",palabra)==0)
palabra="siguiente";

if(string.Compare("vol",palabra)==0)
palabra="volumen";

if(string.Compare("cap",palabra)==0)
palabra="capítulo";

if(string.Compare("cda",palabra)==0)
palabra="cerrada";

if(string.Compare("dpto",palabra)==0)
palabra="departamento";

if(string.Compare("cto",palabra)==0)
palabra="circuito";

if(string.Compare("col",palabra)==0)
palabra="colonia";

if(string.Compare("admón",palabra)==0)
palabra="administración";

if(string.Compare("cía",palabra)==0)
palabra="compañia";

if(string.Compare("esq",palabra)==0)
palabra="esquina";

if(string.Compare("calz",palabra)==0)
palabra="calzada";

if(string.Compare("adj",palabra)==0)
palabra="adjetivo";

if(string.Compare("adv",palabra)==0)
palabra="adverbio";

if(string.Compare("sust",palabra)==0)
palabra="sustantivo";
```

```
if(string.Compare("conj",palabra)==0)
palabra="conjunción";

if(string.Compare("ntra",palabra)==0)
palabra="nuestra";

if(string.Compare("cel",palabra)==0)
palabra="celular";

if(string.Compare("tel",palabra)==0)
palabra="teléfono";

if(string.Compare("izq",palabra)==0)
palabra="izquierda";

if(string.Compare("cta",palabra)==0)
palabra="cuenta";

if(string.Compare("pd",palabra)==0)
palabra="posdata";

if(string.Compare("cte",palabra)==0)
palabra="corriente";

if(string.Compare("dls",palabra)==0)
palabra="dólares";

if(string.Compare("dl",palabra)==0)
palabra="dólar";

if(string.Compare("ej",palabra)==0)
palabra="ejemplo";

if(string.Compare("vs",palabra)==0)
palabra="contra";

if(string.Compare("edo",palabra)==0)
palabra="estado";

if(string.Compare("fig",palabra)==0)
palabra="figura";

if(string.Compare("mex",palabra)==0)
palabra="méxico";

if(string.Compare("org",palabra)==0)
palabra="organización";

if(string.Compare("mich",palabra)==0)
palabra="michoacán";

if(string.Compare("jal",palabra)==0)
palabra="jalisco";

if(string.Compare("mor",palabra)==0)
palabra="morelos";
```

```
        if(string.Compare("coah",palabra)==0)
        palabra="coahuila";

        if(string.Compare("mer",palabra)==0)
        palabra="merida";

        if(string.Compare("dur",palabra)==0)
        palabra="durango";

        if(string.Compare("feb",palabra)==0)
        palabra="febrero";

        if(string.Compare("abr",palabra)==0)
        palabra="abril";

        if(string.Compare("jun",palabra)==0)
        palabra="junio";

        if(string.Compare("jul",palabra)==0)
        palabra="julio";

        if(string.Compare("ago",palabra)==0)
        palabra="agosto";

        if(string.Compare("sept",palabra)==0)
        palabra="septiembre";

        if(string.Compare("oct",palabra)==0)
        palabra="octubre";

        if(string.Compare("nov",palabra)==0)
        palabra="noviembre";

        if(string.Compare("dic",palabra)==0)
        palabra="diciembre";

                        palabra =
acentuacion(palabra);

                        n = palabra.Length;
                        for (int i=0; i<n; i++)
                        {
                                l1 = palabra[i];
                                l2 = ' ';
                                l3 = ' ';
                                p1 = ' ';

                                if(i<(n-1))
                l2 = palabra[i+1];
                                if(i<(n-2))
                l3 = palabra[i+2];

                                switch (l1)
                                {
                                        case
'c':
```

```
                l1 = 'k';

                if (l2=='h')

                {

                        l1 = 'C';

                        i++;

                }

                if (vdebil(l2))

                {

                        l1 = 's';

                }

                break;
                                        case
's':

                if (l2=='h')

                {

                        l1 = 'S';

                        i++;

                }

                break;
                                        case
'l':

                if (l2=='l')

                {

                        l1 = 'L';

                        i++;

                }

                break;
                                        case
'r':

                if (l2=='r')

                {

                        l1='R';
```

```
                i++;
        }
        if (!vocal(p1))
        {
                l1 = 'r';
        }
        break;
                                        case
'q':
        l1 = 'k';
        if(l2=='u')
        {
                i++;
        }
        break;
                                        case
'v':
        l1 = 'b';
        break;
                                        case
'z':
        l1 = 's';
        break;
                                        case
'y':
        if(!vocal(l2))
        {
                l1 = 'i';
        }
        break;
                                        case
'g':
        if (l2=='e' | l2=='i')
        {
                l1='j';
        }
        if (l2=='u' & (vdebil(l3)))
        {
                i++;
        }
        break;
                                        case
'ü':
        l1 = 'u';
        break;
                                        case
'x':
        salida = salida + 'k';
        l1 = 's';
        break;
                                        case '
':
        l1 = '-';
        break;
                                }
                                if(l1!='h')
                                        salida
= salida + l1;
                        }

                        return salida;
                }
                private string numero(string
numero)
                {
                        int a,b;
                        string salida="";
                        char l1;
                        b=numero.Length;
                        int pos=b;
                        b=b-1;

                        while(pos>=0)
                        {
```

```
                                        for
(a=0;a<=b;a++)
                                        {

        l1=numero[a];

        if(pos==1)
        switch(l1)
        {
                case '1':

        salida=salida+palabra("úno");
                        break;
                case '2':

        salida=salida+palabra("dós");
                        break;
                case '3':

        salida=salida+palabra("trés");
                        break;
                case '4':

        salida=salida+palabra("cuátro");
                        break;
                case '5':

        salida=salida+palabra("cínco");
                        break;
                case '6':

        salida=salida+palabra("séis");
                        break;
                case '7':

        salida=salida+palabra("siéte");
                        break;
                case '8':

        salida=salida+palabra("ócho");
                        break;
                case '9':

        salida=salida+palabra("nuéve");
                        break;
                case '0':

        salida=salida+palabra("céro");
                        break;
                }
                                                else
if(pos==2)
        switch(l1)
        {
                case '1':
                switch(numero[a+1])
                {
                        case '1':

        salida=salida+palabra("ónce");
                                        pos=0;
                                        break;
                        case '2':

        salida=salida+palabra("dóce");
                                        pos=0;
```

```
                    break;
                case '3':
salida=salida+palabra("tréce");
                    pos=0;
                    break;
                case '4':
salida=salida+palabra("catórce");
                    pos=0;
                    break;
                case '5':
salida=salida+palabra("quínce");
                    pos=0;
                    break;
                case '0':
salida=salida+palabra("diéz");
                    pos=0;
                    break;
                default:
salida=salida+palabra("diéci");
                    break;
            }
                break;
            case '2':
            switch(numero[a+1])
            {
                case '0':
salida=salida+palabra("véinte");
                    pos=0;
                    break;
                default:
salida=salida+palabra("véinti");
                    break;
            }
                break;
            case '3':
            switch(numero[a+1])
            {
                case '0':
salida=salida+palabra("treinta");
                    pos=0;
                    break;
                default:
salida=salida+palabra("tréinta-í-");
                    break;
            }
                break;
            case '4':
            switch(numero[a+1])
            {
                case '0':
salida=salida+palabra("cuarénta");
                    pos=0;
```

```
                        break;
                    default:
salida=salida+palabra("cuarénta-í-");
                        break;
            }
                    break;
            case '5':
            switch(numero[a+1])
            {
                    case '0':
salida=salida+palabra("cíncuénta");
                        pos=0;
                        break;
                    default:
salida=salida+palabra("cíncuénta-í-");
                        break;
            }
                    break;
            case '6':
            switch(numero[a+1])
            {
                    case '0':
salida=salida+palabra("sesénta");
                        pos=0;
                        break;
                    default:
salida=salida+palabra("sesénta-í-");
                        break;
            }
                    break;
            case '7':
            switch(numero[a+1])
            {
                    case '0':
salida=salida+palabra("seténta");
                        pos=0;
                        break;
                    default:
salida=salida+palabra("seténta-í-");
                        break;
            }
                    break;
            case '8':
            switch(numero[a+1])
            {
                    case '0':
salida=salida+palabra("ochénta");
                        pos=0;
                        break;
                    default:
salida=salida+palabra("ochénta-í-");
```

```
                                        break;
                                }
                                        break;

                                case '9':
                                switch(numero[a+1])
                                {
                                        case '0':

                        salida=salida+palabra("novénta");
                                                pos=0;
                                                break;
                                        default:

                        salida=salida+palabra("novénta-í-");
                                                break;
                                }
                                        break;

                                case '.':

                        salida=salida+palabra("púnto");
                                        break;}

                        //**********pos 3*************
                        // para cientos 100,200,etc
                                                        else
if(pos==3)

                        switch(l1)
                        {

                                case '1':
                                switch(numero[a+1])
                                {
                                        case '0':
                                        switch(numero[a+2])
                                        {
                                                case '0':

                        salida=salida+palabra("cién");
                                                        pos=0;
                                                        break;
                                                default:

                        salida=salida+palabra("ciénto");
                                                        pos=2;
                                                        a++;
                                                        break;
                                        }
                                                break;

                                        default:

                        salida=salida+palabra("ciénto");
                                                break;
                        }
                                        break;

                                case '2':

                        salida=salida+palabra("dosciéntos");
                                switch(numero[a+1])
                                {

                                        case '0':
```

```
                switch(numero[a+2])
                {
                        case '0':
                                pos=0;
                                break;
                        default:
                                pos=2;
                                a++;
                                break;
                }
                        break;
        }
                break;
        case '3':
salida=salida+palabra("tresciéntos");
        switch(numero[a+1])
        {
                case '0':
                switch(numero[a+2])
                {
                        case '0':
                                pos=0;
                                break;
                        default:
                                pos=2;
                                a++;
                                break;
                }
                        break;
        }
                break;
        case '4':
salida=salida+palabra("cuatrociéntos");
        switch(numero[a+1])
        {
                case '0':
                switch(numero[a+2])
                {
                        case '0':
                                pos=0;
                                break;
                        default:
                                pos=2;
                                a++;
                                break;
                }
                        break;
        }
                break;
        case '5':
salida=salida+palabra("quiniéntos");
```

```
                switch(numero[a+1])
                {
                        case '0':
                        switch(numero[a+2])
                        {
                                case '0':
                                        pos=0;
                                        break;
                                default:
                                        pos=2;
                                        a++;
                                        break;
                        }
                                break;
                }
                        break;

                case '6':

        salida=salida+palabra("seisciéntos");
                switch(numero[a+1])
                {
                        case '0':
                        switch(numero[a+2])
                        {
                                case '0':
                                        pos=0;
```

```
                                        break;
                                default:
                                        pos=2;
                                        a++;
                                        break;
                        }
                                break;
                }
                        break;

                case '7':

        salida=salida+palabra("seteciéntos");
                switch(numero[a+1])
                {
                        case '0':
                        switch(numero[a+2])
                        {
                                case '0':
                                        pos=0;
                                        break;
                                default:
                                        pos=2;
                                        a++;
                                        break;
                        }
                                break;
                }
```

```
                    break;

            case '8':

salida=salida+palabra("ochociéntos");
            switch(numero[a+1])
            {

                    case '0':
                    switch(numero[a+2])
                    {
                            case '0':

                                    pos=0;
                                    break;
                            default:
                                    pos=2;
                                    a++;
                                    break;
                    }
                            break;

            }
                    break;

            case '9':

salida=salida+palabra("noveciéntos");
            switch(numero[a+1])
            {

                    case '0':
                    switch(numero[a+2])
                    {
                            case '0':

                                    pos=0;
                                    break;
                            default:
                                    pos=2;
                                    a++;
                                    break;
                    }
                            break;

            }
                    break;
            case '.':

salida=salida+palabra("púnto");
                    break;}

            //***********pos 4******************
            // para miles 1000,2000,etc
                                    else
if(pos==4)

            switch(l1)
            {

                    case '1':

            salida=salida+palabra("míl");
                    switch(numero[a+1])
                    {
```

```
                        case '0':
                        switch(numero[a+2])
                        {
                                case '0':
        switch(numero[a+3])
                                {
                                        case
'0':
        pos=0;
        break;
                                }
                                        break;
                        }
                                break;
                }
                        break;
                case '2':
        salida=salida+palabra("dosmíl");
                switch(numero[a+1])
                {
                        case '0':
                        switch(numero[a+2])
                        {
                                case '0':
        switch(numero[a+3])
                                {
```

```
                                        case
'0':
        pos=0;
        break;
                                }
                                        break;
                        }
                                break;
                case '3':
        salida=salida+palabra("tresmíl");
                switch(numero[a+1])
                {
                        case '0':
                        switch(numero[a+2])
                        {
                                case '0':
        switch(numero[a+3])
                                {
                                        case
'0':
        pos=0;
        break;
                                }
                                        break;
```

```
                        }
                                break;
                }
                        break;
                case '4':
        salida=salida+palabra("cuatromíl");
                switch(numero[a+1])
                {
                        case '0':
                        switch(numero[a+2])
                        {
                                case '0':
        switch(numero[a+3])
                                {
                                        case
'0':
        pos=0;
        break;
                                }
                                        break;
                        }
                                break;
                }
                        break;
                case '5':
```

```
        salida=salida+palabra("cincomíl");
                switch(numero[a+1])
                {
                        case '0':
                        switch(numero[a+2])
                        {
                                case '0':
        switch(numero[a+3])
                                {
                                        case
'0':
        pos=0;
        break;
                                }
                                        break;
                        }
                                break;
                }
                        break;
                case '6':
        salida=salida+palabra("seismíl");
                switch(numero[a+1])
                {
                        case '0':
                        switch(numero[a+2])
```

```
                        {
                                case '0':
        switch(numero[a+3])
                                {
                                        case
'0':
        pos=0;
        break;
                                }
                                        break;
                        }
                                break;
                }
                        break;
                case '7':
        salida=salida+palabra("seietemíl");
                switch(numero[a+1])
                {
                        case '0':
                        switch(numero[a+2])
                        {
                                case '0':
        switch(numero[a+3])
                                {
                                        case
'0':
        pos=0;
        break;
                                }
                                        break;
                        }
                                break;
                case '8':
        salida=salida+palabra("ochomíl");
                switch(numero[a+1])
                {
                        case '0':
                        switch(numero[a+2])
                        {
                                case '0':
        switch(numero[a+3])
                                {
                                        case
'0':
        pos=0;
        break;
                                }
                                        break;
                        }
```

```
                                        break;
                                }
                                        break;
                                case '9':
                        salida=salida+palabra("nuevemíl");
                                switch(numero[a+1])
                                {
                                        case '0':
                                        switch(numero[a+2])
                                        {
                                                case '0':
                        switch(numero[a+3])
                                                {
                                                        case
'0':
                        pos=0;
                        break;
                                                }
                                                        break;
                                        }
                                                case '1':
                                                switch(numero[a+3])
                                                {
                                                        case
'0':
                        pos=0;
                        break;
                                                }
                                                        break;
                                        }
                                                case '2':
                                                switch(numero[a+3])
                                                {
                                                        case
'0':
                        pos=0;
                        break;
                                                }
                                                        break;
                                        }
                                                case '3':
                                                switch(numero[a+3])
                                                {
                                                        case
'0':
                        pos=0;
                        break;
                                                }
                                                        break;
                                        }
                                                case '4':
                                                switch(numero[a+3])
                                                {
                                                        case
'0':
                        pos=0;
                        break;
                                                }
                                                        break;
                                        }
                                                case '5':
                                                switch(numero[a+3])
                                                {
                                                        case
'0':
                        pos=0;
                        break;
                                                }
                                                        break;
                                        }
                                                case '6':
                                                switch(numero[a+3])
                                                {
                                                        case
'0':
                        pos=0;
                        break;
                                                }
                                                        break;
                                        }
                                                case '7':
                                                switch(numero[a+3])
                                                {
                                                        case
'0':
                        pos=0;
                        break;
                                                }
                                                        break;
                                        }
                                                case '8':
                                                switch(numero[a+3])
                                                {
                                                        case
'0':
                        pos=0;
                        break;
                                                }
                                                        break;
                                        }
                                                case '9':
                                                switch(numero[a+3])
                                                {
                                                        case
'0':
                        pos=0;
                        break;
                                                }
                                                        break;
                                        }
                                }
                                        break;
                                case '1':
                                switch(numero[a+2])
                                {
                                        case '0':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '1':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '2':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '3':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '4':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '5':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '6':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '7':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '8':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '9':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                }
                                        break;
                                case '2':
                                switch(numero[a+2])
                                {
                                        case '0':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '1':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '2':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '3':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '4':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '5':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '6':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '7':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '8':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '9':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                }
                                        break;
                                case '3':
                                switch(numero[a+2])
                                {
                                        case '0':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '1':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '2':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '3':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '4':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '5':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '6':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '7':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '8':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '9':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                }
                                        break;
                                case '4':
                                switch(numero[a+2])
                                {
                                        case '0':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '1':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '2':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '3':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '4':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '5':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '6':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '7':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '8':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '9':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                }
                                        break;
                                case '5':
                                switch(numero[a+2])
                                {
                                        case '0':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '1':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '2':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '3':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '4':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '5':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '6':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '7':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '8':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '9':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                }
                                        break;
                                case '6':
                                switch(numero[a+2])
                                {
                                        case '0':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '1':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '2':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '3':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '4':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '5':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '6':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '7':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '8':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '9':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                }
                                        break;
                                case '7':
                                switch(numero[a+2])
                                {
                                        case '0':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '1':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '2':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '3':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '4':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '5':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '6':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '7':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '8':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '9':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                }
                                        break;
                                case '8':
                                switch(numero[a+2])
                                {
                                        case '0':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '1':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '2':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '3':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '4':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '5':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '6':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '7':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '8':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '9':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                }
                                        break;
                                case '9':
                                switch(numero[a+2])
                                {
                                        case '0':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '1':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '2':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '3':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '4':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '5':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '6':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '7':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '8':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                        case '9':
                                switch(numero[a+3])
                                {
                                        case '0':
                        pos=0;
                        break;
                                        }
                                                break;
                                }
                                        break;
                        }
                        salida=salida+palabra("púnto");
```

```
                        break;}
                                pos--;
                                }
                        }
                        return(salida);
                }
        }
}
```

Clase para el manejo de wav files "CWAV.cs"

```
using System;
using System.IO;
using System.Text;

namespace WAV
{
        public class CWAV
        {
                struct Header
                {
                        public string ChunkID;
                        public string ChunkSize;
                        public string Format;
                        public string
Subchunk1ID;
                        public string
Subchunk1Size;
                        public string
AudioFormat;
                        public string
NumChannels;
                        public string
SampleRate;
                        public string ByteRate;
                        public string BlockAlign;
                        public string
BitsperSample;
                        public string
Subchunk2Id;
                        public string
Subchunk2Size;
                }

                private Header _header;
                public string data;
                private string data2;
                private int tamano;
                public int al;
                public int bl;

                public CWAV( int ai, int bi)
                {
                        int al=ai;
                        int bl=bi;
                        this.tamano = 0;
```

```
                        this.BasicHeader(al,bl);
                        this.data="";
                        this.data2="";

                }

                public void BasicHeader(int c, int
d)
                {
                        _header.ChunkID =
"RIFF";
                        _header.ChunkSize = ""
+ (char)36 + (char)0 + (char)0 + (char)0;
                        _header.Format =
"WAVE";
                        _header.Subchunk1ID =
"fmt ";
                        _header.Subchunk1Size
= "" + (char)16 + (char)0 + (char)0 + (char)0;
                        _header.AudioFormat =
"" + (char)1 + (char)0;
                        _header.NumChannels =
"" + (char)1 + (char)0;
                        _header.SampleRate =
"" + (char)17 + (char)c + (char)0 + (char)0;
                        _header.ByteRate = ""
+ (char)34 + (char)d + (char)0 + (char)0;
                        _header.BlockAlign = ""
+ (char)2 + (char)0;
                        _header.BitsperSample
= "" + (char)16 + (char)0;
                        _header.Subchunk2Id =
"data";
                        _header.Subchunk2Size
= "" + (char)0 + (char)0 + (char)0 + (char)0;
                }

                public string header
                {
                        get
                        {

        _header.ChunkSize = inttohex(this.tamano
+ 36);

        _header.Subchunk2Size =
inttohex(this.tamano);


                                return (
        string.Concat(_header.ChunkID,
_header.ChunkSize, _header.Format,

                _header.Subchunk1ID,
_header.Subchunk1Size, _header.AudioFormat,

                _header.NumChannels,
_header.SampleRate, _header.ByteRate,

                _header.BlockAlign,
_header.BitsperSample, _header.Subchunk2Id,

                _header.Subchunk2Size)
        );
                        }
                }

                private string inttohex(int size)
                {
                        int hex1, hex2, hex3,
hex4;

                        hex1 = size%256;size -
= hex1; size /= 256;
                        hex2 = size%256;size -
= hex2; size /= 256;
                        hex3 = size%256;size -
= hex3; size /= 256;
                        hex4 = size%256;

                        return ("" + (char)hex1
+ (char)hex2 + (char)hex3 + (char)hex4);
                }

                public void addPCM(string dif)
                {
                        byte[][] help1=null;
                        byte[][] help2=null;
                        byte[][] help3=null;
                        string daf="";
                        string dof="";
                        string duf="";
                        for (int l=0;l<500;l++)
                        {
                                duf+=dif[l];
                        }

                        for (int a=500;
a<dif.Length-500;a++)
                        {
                                int u=a-1;
                                dof+=dif[a];
                        }


                        data2="";
                        for (int h=0;h<500;h++)
                        {

        data2+=dif[dif.Length-500+h];
                        }

                        this.tamano +=
dof.Length;
                        this.data += dof;
                }
```

```
                public bool createWav(string
archson)
                {
                        try
                        {
                                StreamWriter
fwav = new StreamWriter(archson, false,
Encoding.Default);

        fwav.Write(this.header + this.data);
                                fwav.Close();
                                return true;
                        }
                        catch
                        {
                                return false;
                        }
                }
        }
}
```

Clases “Sound” y “Helpers” para escuchar wav files

```
using System;

namespace Sintetizador
{
        /// <summary>
        /// Descripción breve de Class1.
        /// </summary>
        public class Sound
        {
                public static void Play( string
strFileName )
                {
                        Helpers.PlaySound(
strFileName, IntPtr.Zero,
Helpers.PlaySoundFlags.SND_FILENAME |
Helpers.PlaySoundFlags.SND_ASYNC );
                }
        }

}

using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;
using System.Runtime.InteropServices;


namespace Sintetizador
{
        /// <summary>
        /// Descripción breve de Helpers.
        /// </summary>
        internal class Helpers
        {
                [Flags]
                        public enum
PlaySoundFlags : int
                {
                        SND_SYNC = 0x0000,
/* play synchronously (default) */
                        SND_ASYNC = 0x0001,
/* play asynchronously */
                        SND_NODEFAULT =
0x0002,  /* silence (!default) if sound not found */
                        SND_MEMORY =
0x0004,  /* pszSound points to a memory file */
                        SND_LOOP = 0x0008,
/* loop the sound until next sndPlaySound */
                        SND_NOSTOP =
0x0010,  /* don't stop any currently playing sound
*/
                        SND_NOWAIT =
0x00002000, /* don't wait if the driver is busy */
                        SND_ALIAS =
0x00010000, /* name is a registry alias */
                        SND_ALIAS_ID =
0x00110000, /* alias is a predefined ID */
                        SND_FILENAME =
0x00020000, /* name is file name */
                        SND_RESOURCE =
0x00040004  /* name is resource name or atom */
                }

                [DllImport("WinMM.dll")]
                public static extern bool
PlaySound( string szSound, IntPtr hMod,
PlaySoundFlags flags );
        }

}
```

Clase del formulario principal de la aplicación de Síntesis de Voz “Form1.cs”

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.IO;
using System.Text;
using Sintetizador;

namespace Sintetizador_de_Voz
{
        public class Form1 :
System.Windows.Forms.Form
```

```
        {
                /// <summary>
                /// Variable del diseñador
requerida.
                /// </summary>
                private
System.ComponentModel.Container components =
null;

                private System.Drawing.Pen pen;
                private
System.Windows.Forms.Button button4;
                private
System.Windows.Forms.Label label4;
                private
System.Windows.Forms.Label label3;
                private
System.Windows.Forms.Button button3;
                private
System.Windows.Forms.Button button2;
                private
System.Windows.Forms.Button button1;
                private
System.Windows.Forms.Label label2;
                private
System.Windows.Forms.Label label1;
                private
System.Windows.Forms.TextBox textBox1;
                private
System.Windows.Forms.ComboBox comboBox1;
                private
System.Windows.Forms.Label label5;
                private
System.Windows.Forms.Label label6;
                private
System.Windows.Forms.Label label7;
                private
System.Windows.Forms.Label label8;
                private
System.Windows.Forms.GroupBox groupBox1;



                public Form1()
                {
                        //
                        // Necesario para
admitir el Diseñador de Windows Forms
                        //
                        InitializeComponent();

        this.comboBox1.Items.Add("Slow");

        this.comboBox1.Items.Add("Normal");

        this.comboBox1.Items.Add("Fast");
                        //
                        // TODO: agregar
código de constructor después de llamar a
InitializeComponent
                        //
                }

                /// <summary>
                /// Limpiar los recursos que se
estén utilizando.
                /// </summary>
                protected override void Dispose(
bool disposing )
                {
                        if( disposing )
                        {
                                if (components
!= null)
                                {

        components.Dispose();
                                }
                        }
                        base.Dispose( disposing
);
                }

                #region Código generado por el
Diseñador de Windows Forms
                /// <summary>
                /// Método necesario para
admitir el Diseñador. No se puede modificar
                /// el contenido del método con
el editor de código.
                /// </summary>
                private void InitializeComponent()
                {

        System.Resources.ResourceManager
resources = new
System.Resources.ResourceManager(typeof(Form1))
;
                        this.button4 = new
System.Windows.Forms.Button();
                        this.label4 = new
System.Windows.Forms.Label();
                        this.label3 = new
System.Windows.Forms.Label();
                        this.button3 = new
System.Windows.Forms.Button();
                        this.button2 = new
System.Windows.Forms.Button();
                        this.button1 = new
System.Windows.Forms.Button();
                        this.label2 = new
System.Windows.Forms.Label();
                        this.label1 = new
System.Windows.Forms.Label();
```

```
            this.textBox1 = new
System.Windows.Forms.TextBox();
            this.groupBox1 = new
System.Windows.Forms.GroupBox();
            this.comboBox1 = new
System.Windows.Forms.ComboBox();
            this.label5 = new
System.Windows.Forms.Label();
            this.label6 = new
System.Windows.Forms.Label();
            this.label7 = new
System.Windows.Forms.Label();
            this.label8 = new
System.Windows.Forms.Label();

        this.groupBox1.SuspendLayout();
            this.SuspendLayout();
            //
            // button4
            //
            this.button4.BackColor =
System.Drawing.Color.DarkBlue;
            this.button4.Font = new
System.Drawing.Font("Microsoft Sans Serif",
20.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.button4.ForeColor =
System.Drawing.Color.White;
            this.button4.Location =
new System.Drawing.Point(456, 344);
            this.button4.Name =
"button4";
            this.button4.Size = new
System.Drawing.Size(176, 128);
            this.button4.TabIndex =
8;
            this.button4.Text =
"Mostrar Waveform";
            this.button4.Click +=
new System.EventHandler(this.button4_Click);
            //
            // label4
            //
            this.label4.BackColor =
System.Drawing.Color.CornflowerBlue;
            this.label4.BorderStyle
= System.Windows.Forms.BorderStyle.Fixed3D;
            this.label4.Font = new
System.Drawing.Font("Microsoft Sans Serif",
20.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.label4.ForeColor =
System.Drawing.Color.White;
            this.label4.Location =
new System.Drawing.Point(456, 208);
            this.label4.Name =
"label4";
            this.label4.Size = new
System.Drawing.Size(368, 120);
            this.label4.TabIndex =
7;
            //
            // label3
            //
            this.label3.BackColor =
System.Drawing.Color.CornflowerBlue;
            this.label3.BorderStyle
= System.Windows.Forms.BorderStyle.Fixed3D;
            this.label3.Font = new
System.Drawing.Font("Microsoft Sans Serif",
20.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.label3.ForeColor =
System.Drawing.Color.White;
            this.label3.Location =
new System.Drawing.Point(456, 80);
            this.label3.Name =
"label3";
            this.label3.Size = new
System.Drawing.Size(360, 88);
            this.label3.TabIndex =
6;
            //
            // button3
            //
            this.button3.BackColor =
System.Drawing.Color.DarkBlue;
            this.button3.Font = new
System.Drawing.Font("Microsoft Sans Serif",
20.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
            this.button3.ForeColor =
System.Drawing.Color.White;
            this.button3.Location =
new System.Drawing.Point(656, 344);
            this.button3.Name =
"button3";
            this.button3.Size = new
System.Drawing.Size(168, 128);
            this.button3.TabIndex =
5;
            this.button3.Text =
"Salir";
            this.button3.Click +=
new System.EventHandler(this.button3_Click);
            //
            // button2
            //
            this.button2.BackColor =
System.Drawing.Color.DarkBlue;
```

```
                this.button2.Font = new
System.Drawing.Font("Microsoft Sans Serif",
20.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
                this.button2.ForeColor =
System.Drawing.Color.White;
                this.button2.Location =
new System.Drawing.Point(184, 360);
                this.button2.Name =
"button2";
                this.button2.Size = new
System.Drawing.Size(208, 32);
                this.button2.TabIndex =
4;
                this.button2.Text =
"Sintetizar";
                this.button2.Click +=
new System.EventHandler(this.button2_Click);
                //
                // button1
                //
                this.button1.BackColor =
System.Drawing.Color.DarkBlue;
                this.button1.Font = new
System.Drawing.Font("Microsoft Sans Serif",
20.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
                this.button1.ForeColor =
System.Drawing.Color.White;
                this.button1.Location =
new System.Drawing.Point(184, 288);
                this.button1.Name =
"button1";
                this.button1.Size = new
System.Drawing.Size(208, 32);
                this.button1.TabIndex =
3;
                this.button1.Text =
"Sintetizar";
                this.button1.Click +=
new System.EventHandler(this.button1_Click);
                //
                // label2
                //
                this.label2.BackColor =
System.Drawing.Color.RoyalBlue;
                this.label2.Font = new
System.Drawing.Font("Arial", 20.25F,
((System.Drawing.FontStyle)((System.Drawing.Font
Style.Bold | System.Drawing.FontStyle.Italic))),
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
                this.label2.ForeColor =
System.Drawing.SystemColors.ControlLightLight;
                this.label2.Location =
new System.Drawing.Point(8, 360);
```

```
                this.label2.Name =
"label2";
                this.label2.Size = new
System.Drawing.Size(168, 32);
                this.label2.TabIndex =
2;
                this.label2.Text =
"Hombre";
                //
                // label1
                //
                this.label1.BackColor =
System.Drawing.Color.RoyalBlue;
                this.label1.Font = new
System.Drawing.Font("Arial", 20.25F,
((System.Drawing.FontStyle)((System.Drawing.Font
Style.Bold | System.Drawing.FontStyle.Italic))),
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
                this.label1.ForeColor =
System.Drawing.SystemColors.ControlLightLight;
                this.label1.Location =
new System.Drawing.Point(8, 288);
                this.label1.Name =
"label1";
                this.label1.Size = new
System.Drawing.Size(152, 32);
                this.label1.TabIndex =
1;
                this.label1.Text =
"Mujer";
                //
                // textBox1
                //
                this.textBox1.BackColor
= System.Drawing.Color.CornflowerBlue;
                this.textBox1.Font =
new System.Drawing.Font("Microsoft Sans Serif",
20.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
                this.textBox1.ForeColor
= System.Drawing.Color.White;
                this.textBox1.Location =
new System.Drawing.Point(16, 80);
                this.textBox1.Multiline
= true;
                this.textBox1.Name =
"textBox1";
                this.textBox1.ScrollBars
= System.Windows.Forms.ScrollBars.Vertical;
                this.textBox1.Size =
new System.Drawing.Size(376, 184);
                this.textBox1.TabIndex
= 0;
                this.textBox1.Text =
"";
                //
```

```
                        // groupBox1
                        //

        this.groupBox1.BackColor =
System.Drawing.Color.RoyalBlue;

        this.groupBox1.Controls.Add(this.label8);

        this.groupBox1.Controls.Add(this.label7);

        this.groupBox1.Controls.Add(this.label6);

        this.groupBox1.Controls.Add(this.comboBo
x1);

        this.groupBox1.Controls.Add(this.label4);

        this.groupBox1.Controls.Add(this.label3);

        this.groupBox1.Controls.Add(this.textBox1
);

        this.groupBox1.Controls.Add(this.button4)
;

        this.groupBox1.Controls.Add(this.label1);

        this.groupBox1.Controls.Add(this.label2);

        this.groupBox1.Controls.Add(this.button1)
;

        this.groupBox1.Controls.Add(this.button2)
;

        this.groupBox1.Controls.Add(this.button3)
;

        this.groupBox1.Controls.Add(this.label5);
                        this.groupBox1.Font =
new System.Drawing.Font("Microsoft Sans Serif",
12F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));

        this.groupBox1.ForeColor =
System.Drawing.SystemColors.ControlLightLight;
                        this.groupBox1.Location
= new System.Drawing.Point(16, 16);
                        this.groupBox1.Name =
"groupBox1";
                        this.groupBox1.Size =
new System.Drawing.Size(856, 528);

        this.groupBox1.TabIndex = 1;
                        this.groupBox1.TabStop
= false;
                        this.groupBox1.Text =
"Introduzca el Texto";
                        //
                        // comboBox1
                        //

        this.comboBox1.BackColor =
System.Drawing.Color.CornflowerBlue;
                        this.comboBox1.Font =
new System.Drawing.Font("Microsoft Sans Serif",
20.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));

        this.comboBox1.ForeColor =
System.Drawing.Color.White;
                        this.comboBox1.Location
= new System.Drawing.Point(192, 432);
                        this.comboBox1.Name =
"comboBox1";
                        this.comboBox1.Size =
new System.Drawing.Size(200, 39);

        this.comboBox1.TabIndex = 9;
                        this.comboBox1.Text =
"Normal";
                        //
                        // label5
                        //
                        this.label5.BackColor =
System.Drawing.Color.RoyalBlue;
                        this.label5.Font = new
System.Drawing.Font("Microsoft Sans Serif",
20.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
                        this.label5.ForeColor =
System.Drawing.SystemColors.ControlLightLight;
                        this.label5.Location =
new System.Drawing.Point(8, 432);
                        this.label5.Name =
"label5";
                        this.label5.Size = new
System.Drawing.Size(216, 32);
                        this.label5.TabIndex =
3;
                        this.label5.Text =
"Sample Rate:";
                        //
                        // label6
                        //
                        this.label6.Font = new
System.Drawing.Font("Microsoft Sans Serif",
20.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
                        this.label6.Location =
new System.Drawing.Point(16, 40);
```

```
                        this.label6.Name =
"label6";
                        this.label6.Size = new
System.Drawing.Size(240, 32);
                        this.label6.TabIndex =
10;
                        this.label6.Text =
"Texto a sintetizar";
                        //
                        // label7
                        //
                        this.label7.Font = new
System.Drawing.Font("Microsoft Sans Serif",
20.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
                        this.label7.Location =
new System.Drawing.Point(456, 40);
                        this.label7.Name =
"label7";
                        this.label7.Size = new
System.Drawing.Size(312, 32);
                        this.label7.TabIndex =
11;
                        this.label7.Text =
"Palabras acentuadas";
                        //
                        // label8
                        //
                        this.label8.Font = new
System.Drawing.Font("Microsoft Sans Serif",
20.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point,
((System.Byte)(0)));
                        this.label8.Location =
new System.Drawing.Point(456, 176);
                        this.label8.Name =
"label8";
                        this.label8.Size = new
System.Drawing.Size(312, 32);
                        this.label8.TabIndex =
12;
                        this.label8.Text =
"Difonemas";
                        //
                        // Form1
                        //
                        this.AutoScaleBaseSize
= new System.Drawing.Size(5, 13);
                        this.BackColor =
System.Drawing.Color.RoyalBlue;
                        this.ClientSize = new
System.Drawing.Size(896, 566);

                this.Controls.Add(this.groupBox1);
                        this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this
.Icon")));
                        this.Name = "Form1";
                        this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterSc
reen;
                        this.Text = "SÍNTESIS
DE VOZ";
                        this.Load += new
System.EventHandler(this.Form1_Load);

        this.groupBox1.ResumeLayout(false);

        this.ResumeLayout(false);

                }
                #endregion

                /// <summary>
                /// Punto de entrada principal de
la aplicación.
                /// </summary>
                [STAThread]
                static void Main()
                {
                        Application.Run(new
Form1());
                }

                private void button1_Click(object
sender, System.EventArgs e)
                {
                        string alfa="";
                        string beta="";
                        int al=43;
                        int bl=86;
                        if
(string.Compare("Slow",this.comboBox1.Text)==0)
                        {
                                al=32;
                                bl=64;
                        }
                        else
                        {
                                if
(string.Compare("Fast",this.comboBox1.Text)==0)
                                {
                                        al=49;
                                        bl=98;
                                }
                                else
                                {
                                        al=43;
                                        bl=86;
                                }

                        }
                        CSintetizador
sintetizador = new
CSintetizador("..//..//..//data1/", al, bl);
```

```
                        label3.Text =
sintetizador.analizar(this.textBox1.Text);
                        label4.Text =
sintetizador.difonemas;
                        alfa=sintetizador.viz;
                        beta=sintetizador.viz2;

        //label3.Text=alfa.Length.ToString();

                        //for (int
hu=0;hu<alfa.Length;hu++)
                        //{
                        //
        label3.Text+=alfa[hu];
                        //}

        //label4.Text=beta.Length.ToString();
                        //for (int
hi=0;hi<beta.Length;hi++)
                        //{
                        //
        label4.Text+=beta[hi];
                        //}



                }

                private void button2_Click(object
sender, System.EventArgs e)
                {
                        int al=43;
                        int bl=86;
                        if
(string.Compare("Slow",this.comboBox1.Text)==0)
                        {
                                al=32;
                                bl=64;
                        }
                        else
                        {
                                if
(string.Compare("Fast",this.comboBox1.Text)==0)
                                {
                                        al=49;
                                        bl=98;
                                }
                                else
                                {
                                        al=43;
                                        bl=86;
                                }

                        }

                        CSintetizador
sintetizador = new
CSintetizador("..//..//..//data/", al, bl);
                        label3.Text =
sintetizador.analizar(this.textBox1.Text);
                        label4.Text =
sintetizador.difonemas;



                }

                private void button3_Click(object
sender, System.EventArgs e)
                {
                        Application.Exit();
                }



                private void Form1_Load(object
sender, System.EventArgs e)
                {


                }



                private void button4_Click(object
sender, System.EventArgs e)
                {
                        pen = new
Pen(Color.Aqua, 1);



                        if(label3.Text=="")
                        {

        MessageBox.Show("No existe ningún wav a
analizar");

                        }
                        else
                        {
                        Form f=new
Form2(this.label3.Text);
                        f.Show();

                        Refresh();



                }

                }
```

```
        }
}
```

Clase para el dibujo de la Wave Form “WaveFile”

```
using System;
using System.IO;
using System.Drawing;
using System.Windows.Forms;

using System.Diagnostics;

namespace Sintetizador
{
        /// <summary>
        /// Summary description for WaveFile.
        /// </summary>
        public class WaveFile
        {
                /// <summary>
                /// The Riff header is 12 bytes
long
                /// </summary>
                class Riff
                {
                        public Riff()
                        {
                                m_RiffID =
new byte[ 4 ];
                                m_RiffFormat
= new byte[ 4 ];
                        }

                        public void ReadRiff(
FileStream inFS )
                        {
                                inFS.Read(
m_RiffID, 0, 4 );

                                Debug.Assert(
m_RiffID[0] == 82, "Riff ID Not Valid" );

                                BinaryReader
binRead = new BinaryReader( inFS );

                                m_RiffSize =
binRead.ReadUInt32( );

                                inFS.Read(
m_RiffFormat, 0, 4 );
                        }

                        public byte[] RiffID
                        {
                                get { return
m_RiffID; }
                        }

                        public uint RiffSize
                        {
                                get { return (
m_RiffSize ); }
                        }

                        public byte[] RiffFormat
                        {
                                get { return
m_RiffFormat; }
                        }

                        private byte[]
                m_RiffID;
                        private uint
                m_RiffSize;
                        private byte[]
                m_RiffFormat;
                }

                /// <summary>
                /// The Format header is 24
bytes long
                /// </summary>
                class Fmt
                {
                        public Fmt()
                        {
                                m_FmtID =
new byte[ 4 ];
                        }

                        public void ReadFmt(
FileStream inFS )
                        {
                                inFS.Read(
m_FmtID, 0, 4 );

                                Debug.Assert(
m_FmtID[0] == 102, "Format ID Not Valid" );

                                BinaryReader
binRead = new BinaryReader( inFS );

                                m_FmtSize =
binRead.ReadUInt32( );
                                m_FmtTag =
binRead.ReadUInt16( );
                                m_Channels =
binRead.ReadUInt16( );

        m_SamplesPerSec = binRead.ReadUInt32(
);
```

```
        m_AverageBytesPerSec =
binRead.ReadUInt32( );
                                m_BlockAlign =
binRead.ReadUInt16( );

        m_BitsPerSample = binRead.ReadUInt16(
);

                                // This
accounts for the variable format header size
                                // 12 bytes of
Riff Header, 4 bytes for FormatId, 4 bytes for
FormatSize & the Actual size of the Format
Header
                                inFS.Seek(
m_FmtSize + 20, System.IO.SeekOrigin.Begin );
                        }

                        public byte[] FmtID
                        {
                                get { return
m_FmtID; }
                        }

                        public uint FmtSize
                        {
                                get { return
m_FmtSize; }
                        }

                        public ushort FmtTag
                        {
                                get { return
m_FmtTag; }
                        }

                        public ushort Channels
                        {
                                get { return
m_Channels; }
                        }

                        public uint
SamplesPerSec
                        {
                                get { return
m_SamplesPerSec; }
                        }

                        public uint
AverageBytesPerSec
                        {
                                get { return
m_AverageBytesPerSec; }
                        }

                        public ushort BlockAlign
                        {
                                get { return
m_BlockAlign; }
                        }

                        public ushort
BitsPerSample
                        {
                                get { return
m_BitsPerSample; }
                        }

                        private byte[]
                m_FmtID;
                        private uint
                m_FmtSize;
                        private ushort
                m_FmtTag;
                        private ushort
                m_Channels;
                        private uint
                m_SamplesPerSec;
                        private uint
                m_AverageBytesPerSec;
                        private ushort
                m_BlockAlign;
                        private ushort
                m_BitsPerSample;
                }

                /// <summary>
                /// The Data block is 8 bytes +
???? long
                /// </summary>
                class Data
                {
                        public Data()
                        {
                                m_DataID =
new byte[ 4 ];
                        }

                        public void ReadData(
FileStream inFS )
                        {
                                //inFS.Seek(
36, System.IO.SeekOrigin.Begin );

                                inFS.Read(
m_DataID, 0, 4 );

                                Debug.Assert(
m_DataID[0] == 100, "Data ID Not Valid" );

                                BinaryReader
binRead = new BinaryReader( inFS );
```

```
                                m_DataSize =
binRead.ReadUInt32( );

                                m_Data = new
Int16[ m_DataSize ];

                                inFS.Seek( 40,
System.IO.SeekOrigin.Begin );

                                m_NumSamples
= (int) ( m_DataSize / 2 );

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

        m_Data[ i ] = binRead.ReadInt16( );
                                }
                        }

                        public byte[] DataID
                        {
                                get { return
m_DataID; }
                        }

                        public uint DataSize
                        {
                                get { return
m_DataSize; }
                        }

                        public Int16 this[ int
pos ]
                        {
                                get { return
m_Data[ pos ]; }
                        }

                        public int NumSamples
                        {
                                get { return
m_NumSamples; }
                        }

                        private byte[]
                m_DataID;
                        private uint
                m_DataSize;
                        private Int16[]
                m_Data;
                        private int
                        m_NumSamples;
                }

                public WaveFile( String inFilepath
)
                {
                        m_Filepath = inFilepath;
                        m_FileInfo = new
FileInfo( inFilepath );
                        m_FileStream =
m_FileInfo.OpenRead( );

                        m_Riff = new Riff( );
                        m_Fmt = new Fmt( );
                        m_Data = new Data( );
                }

                public void Read( )
                {
                        m_Riff.ReadRiff(
m_FileStream );
                        m_Fmt.ReadFmt(
m_FileStream );
                        m_Data.ReadData(
m_FileStream );
                }

                public void Draw( Graphics grfx,
Pen pen )
                {


                        if ( m_PageScale ==
0.0f )
                                m_PageScale =
(grfx.VisibleClipBounds.Size.Width /
m_Data.NumSamples);

        grfx.InterpolationMode=System.Drawing.D
rawing2D.InterpolationMode.High;
                        grfx.PageScale =
m_PageScale;

                        RectangleF visBounds =
grfx.VisibleClipBounds;

                        grfx.DrawLine( pen, 4,
visBounds.Size.Height / 2, visBounds.Size.Width-
4, visBounds.Size.Height / 2 );


        grfx.TranslateTransform( 0,
visBounds.Size.Height );
                        grfx.ScaleTransform( 1,
-1 );

                        Draw16Bit( grfx, pen,
visBounds );
                }

                void Draw16Bit( Graphics grfx,
Pen pen, RectangleF visBounds )
                {
```

```
                        short val = m_Data[ 0
];

                        int prevX = 4;
                        int prevY = (int) (( (val
+ 32768) * visBounds.Height ) / 65536 );


                        for ( int i = 0; i <
m_Data.NumSamples-4; i++ )
                        {
                                val = m_Data[ i
];

                                int scaledVal =
(int) (( (val + 32768) * visBounds.Height ) / 65536
);

                                grfx.DrawLine(
pen, prevX, prevY, i, scaledVal );

                                prevX = i;
                                prevY =
scaledVal;

                                if (
m_Fmt.Channels == 2 )
                                        i++;

                        }
                }

                public void ZoomIn( )
                {
                        m_PageScale /= 2;
                }

                public void ZoomOut( )
                {
                        m_PageScale *= 2;
                }

                private string
        m_Filepath;
                private FileInfo
        m_FileInfo;
                private FileStream
        m_FileStream;

                private Riff
        m_Riff;
                private Fmt
                m_Fmt;
                private Data
        m_Data;

                private float
        m_PageScale = 0.0f;
```

```
        }
}
```

Clase del formulario secundario para el dibujo del Wave Form “Form2.cs”

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using Sintetizador;

namespace Sintetizador_de_Voz
{
        /// <summary>
        /// Descripción breve de Form2.
        /// </summary>
        public class Form2 :
System.Windows.Forms.Form
        {
                /// <summary>
                /// Variable del diseñador
requerida.
                /// </summary>
                private
System.ComponentModel.Container components =
null;
                public string caden="";
                public Form2(string cadena)
                {
                        //
                        // Necesario para
admitir el Diseñador de Windows Forms
                        //
                        this.caden=cadena;
                        InitializeComponent();

                        //
                        // TODO: agregar
código de constructor después de llamar a
InitializeComponent
                        //
                }

                /// <summary>
                /// Limpiar los recursos que se
estén utilizando.
                /// </summary>
                protected override void Dispose(
bool disposing )
                {
                        if( disposing )
                        {
                                if(components
!= null)
                                {
```

```
            components.Dispose();
                                    }
                        }
                        base.Dispose( disposing
);
            }

            #region Código generado por el
Diseñador de Windows Forms
            /// <summary>
            /// Método necesario para
admitir el Diseñador. No se puede modificar
            /// el contenido del método con
el editor de código.
            /// </summary>
            private void InitializeComponent()
            {
                        //
                        // Form2
                        //
                        this.AutoScale = false;
                        this.AutoScaleBaseSize
= new System.Drawing.Size(5, 13);
                        this.BackColor =
System.Drawing.Color.Black;
                        this.ClientSize = new
System.Drawing.Size(960, 518);
                        this.Name = "Form2";
                        this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterSc
reen;
                        this.Text = "Wave
Form";
                        this.Load += new
System.EventHandler(this.Form2_Load);
                        this.Paint += new
System.Windows.Forms.PaintEventHandler(this.For
m2_Paint);

            }
            #endregion

            private void Form2_Paint(object
sender, System.Windows.Forms.PaintEventArgs e)
            {
                        Pen pen = new
Pen(System.Drawing.Color.Aqua);
                        WaveFile wave= new
WaveFile("..//..//..//"+ caden +"-.wav");
                        wave.Read( );
                        wave.Draw( e.Graphics,
pen );


            }
      }
}
```

BIBLIOGRAFÍA

[1] CHRIS ROWDEN: Speech Processing, Essex Series in Telecommunications and information systems, cap 1, 2 y3, Mc Graw Hill, 1991.

[2] F.A. WESTALL, S.F.A. Ip: Digital signal Processing in Telecommunications, cap 1, 10 y 11, Gran Bretaña, Chapman & Hall, B.T. Telecommunications, 1993.

[3] PANOS E. PAPAMICHALIS: Practical Approaches to Speech Coding, cap. 1 y 4, New Jersey, Prentice Hall, 1987.

[4] BURRUS, C.S. & PARKS, T.W.: DFT/FFT and Convolution, E.U.A., John Wiley and Sons, 1985.

[5] PARKS T.W. & BURRUS, C.S.: Digital Filter Design, E.U.A., John Wiley and Sons, 1987.

[6] A. W. BLACK & P. TAYLOR. CHATR: **A** generic Speech Synthesis System. In *Proceedings of COLING-94, vo*lume II, pages 983-986, Kyoto, Japan, 1994.

[7] AVILA RAUL: La lengua y los hablantes: área y lenguaje y comunicación, Ed. Trillas, México, 1981.

[8] AVILA RAUL: La lengua y los hablantes, Ed. Trillas, México 1999.

[9] N. CAMPBELL & A. BLACK. Prosody and the Selection of Source Units for Concatenative Synthesis. In J. van Santen, R. Sproat, J. Olive, and J. Hirshberg, editors, *Progress **in** speech synthesis.* Springer Veriag, 1995.

[10] L. RABINER & B. JUANG: Fundamentals *of Speech* Recognition. Prentice Hall, 1993.

[11] Y. SAGISAKA, N. KAIKI & N. LWAHASHI: ATR - V-TALK Speech Synthesis System. In Proceed*ings of ICSLP 92,* vol. 1, pp 483-486, 1992.

[12] H. VALBRET, E. MONLINES, AND J. TUBACH. Voice Transformation

using PSOLA technique. Speech Communications, 11:175-187, 1992.

[13] ABE, M.: Speaking Styles: Statistical Analysis and Synthesis by a Text-to-Speech System, in van SANTEN, J.P.H. - SPROAT, R.W.- OLIVE, J.P.- HIRSCHBERG. J. (Eds.) *Progress in Speech Synthesis*. New York: Springer. pp. 495-510., 1997.

[14] ALLEN, J. "Synthesis of Speech from Unrestricted Text", *Proceedings of the IEEE* 64,4: 433-442; 1976.

[15] RABINER (Eds.) *Speech Synthesis*. Stroudsburg, Penn.: Dowden, Hutchinson & Ross Inc. pp. 416-28; 1973.

[16] ATAL, B.S.- MILLER, L.J.- KENT, R.D. (Eds.) *Papers in Speech Communication: Speech Processing*. New York: Acoustical Society of America. pp. 3-12. 1991.

[17] ALLEN, J. "Overview of Text-to-Speech Systems", in FURUI, S.- SONDHI, M. (Eds.) *Advances in Speech Signal Processing*. New York, 1992.

[18] ALLEN, J.- HUNNICUTT, M.S.- KLATT, D.H. (with R.C. ARMSTRONG and D. PISONI) (1987) *From Text to Speech: The MITalk System*. Cambridge: Cambridge University Press (Cambridge Studies in Speech Science and Communication).

[19] LEMMETTY SAMI, “Review of Speech Synthesis Technology”, Abstract of the Master´s Thesis, Helsinski, 1999.

[20] FERNANDO DEL RIO AVILA, Diseño de un sitetizador de voz por difonemas, tesis de licenciatura, 2003.

[21] FERNANDO DEL RIO AVILA, Diseño de un sitetizador de voz en español usando el método TD- PSOLA, tesis de maestría, 2005.

[22] http://www.text2speech.com/

[23] http://www.natvox.es/tecnologia_sintesis.html

[24] http://www.cs.bris.ac.uk/

[25] http://www.crl.research.digital.com/

[26] http://www.tik.ee.ethz.ch/

[27] http://liceu.uab.es/~joaquim/speech_technology/tecnol_parla/synthesis/units/unitats_sintesi.html

[28] http://dihana.cps.unizar.es/investigacion/voz/ctv.html

[29] http://www-gth.die.upm.es/~juancho/pfcs/GMS/cap6.pdf

[30]http://www.dcc.uchile.cl/~mhormaza/memoria/Pres_alumnos_27_09_2002.pdf